Hans Baldung Grien
schichte
NEU! Modulhandbücher
Von der Leinwand zum Bildschirm. Bewegtbilder in künstlerischen Kontexten
WiSe 19/20
FORSCHUNG
LEHRE
KU
Weitere Veranstaltungen
KREIS
Modulstruktur
Spots auf Hans Baldung Grien
FAKULTÄT
Sitemap
KUNST UND KULTUR
Mehr...
So finden Sie uns
12.2019–11.2022
ner
VERANSTALTUNGEN
gebiet Kunstgeschichte
erstraße 7, Geb. 20.40, EG,
r. 20 D-76131 Karlsruhe
+49 721 608 42191 E-Mail:

BROWSER ART: NAVIGATING WITH STYLE

INGE HINTERWALDNER, DANIELA HÖNIGSBERG

ABSTRACT | From the mid-1990s, works of browser art emerged as a particular variant of net art or speculative software, developing alternatives to conventional internet browsers and rendering the content of web pages in a different way. Since we see them as productive image-machines, we consider both the phenomenology of their audio-visual output and their operational code level. We use an example to show why it is helpful, or even necessary, to take a closer look at programmed art works – and thus paradoxically to adopt a 'distant reading'. There are many ways of creating a seemingly identical surface effect, though this may have been caused by completely different artistic gestures. Where should we look for the evidence? We agree with Florian Cramer that the code is a crucial site for the inventive engagement of their producers and is instructive in this regard. Furthermore, we are interested in capturing the dynamics of a piece of software during its execution. For this purpose, art history's vast methodological tradition is to be enriched with software visualization tools, while the latter are endowed with art history related reasoning. Lev Manovich, who developed the 'multi-scale view' for film, along with Shane Denson and Andreas Jahn-Sudmann, who transferred it to games, paved the intellectual path for our approach.

KEYWORDS | digital art, browser application, visual thinking, interface, methodology

Software art and its (contested) visuality

Given that software art or programmed applications are pieces of written code, there are legitimate arguments for vetoing the prioritization of their phenomenological level. The literature and art history scholar Florian Cramer, for example, draws attention to the fact that one cannot arrive at an adequate appreciation of software art if one – following the tradition of Romantic philosophy – privileges aisthesis (perception) over poiesis (construction), thereby limiting the concept of art to whatever can be touched, seen, and heard.[1]

Cramer sharpened his understanding of software art – which for him is characterized by an anti-visuality – by defining as its antithesis something he called 'neo-Pythagorean digital kitsch'. It was apologists of a 'post-optical age' or of a rigorous primacy of program code over its 'secondary effects' on the monitor that prompted the objection of artist Johannes Auer (aka Frieder Rusmann), who coined the neologism 'binary idealism' for it.[2] The pendulum thus swung back again, demanding once more a turn towards something sensually perceptible. This swing back implies the statement that in the digital world the image (Bild) receives little attention – which may be surprising given the fact that everything on the screen (Bildschirm) can be seen as an image due to its pixel matrix (Bildpunktmatrix).

Admittedly – to concur with the art historian Margarete Pratschke – there still lurks the possibility that "one of the most viewed images worldwide, if not the most viewed image of the present-day, [...] is hardly perceived as such. The image so often overlooked because of its instrumental character is the so-called graphical user interface. This refers to the image or image system specified by the computer's operating system, usually displayed on the screen as soon as a computer is booted up and through which users interact with the computer."[3]

It follows from this understanding that infrastructural image systems turn "any interaction on the computer into work on an image."[4] In his plea against neglecting the visual level, Auer refers to the artist couple JODI (Dirk Paesmans and Joan Hemskerk), who have been regarded for decades as classics in their field for their programmed interventions on the internet and for their game modifications. JODI explained: "Media art is always on the surface. You have to get people very quickly."[5] Admittedly, this proposition should be taken with a pinch of salt in the case of this artist duo.

Browser Art as Image-Machines

Let us consider both aisthesis and poiesis. We speak here of 'image-machines' in procedural applications such as interactive simulations, internet browsers, or computer games. This is because the output on the screen or through the loudspeaker is generated by intervening mechanisms that 'execute' and 'compose'. These applications are generative as well as operative, and thus do not merely represent something pre-existent. This is the reason why even commercial browsers differ in how they interpret web content and render it slightly differently. However, they all remain committed to the page metaphor, and to an output of the web page according to the presumed intensions of the web developers and web designers.

We call what browsers – as information-changing image-machines – produce as output, procedurally composed, iconic artifacts. The term 'artifact' does not refer to something object-like but instead emphasizes its status of having been created in the sense of 'arte factum'. Furthermore, the browser's access to the web servers and the retrieval and processing of the data of the accessed web pages are central components of these composites.

From the mid-1990s, works of art browsers emerged as a particular variant of net art or speculative software. Whether in their functionality or in their visuality, art browsers develop alternatives to conventional browsers and replace static presentations either "with alternative, readability-resolving static presentations [...] or with dynamic, transitory presentations."[6] In the static or dynamic output, the 'objets trouvés' from the internet are each characteristically arranged and staged. In this way, they set different emphases, often offering new functionalities, and encouraging us to rethink existing categorizations. This whole output complex of browser programs with their algorithmic substructure can be discussed in terms of a (multimodal) pictoriality. How the programs sample this in each case on the fly has not yet been adequately captured or understood.

Theory of the Digital Image

If we take Cramer's announcement seriously that in programmed works the code level is the main site of their producers' engagement, one could in a first step pay attention to how theorizing on digital imagery has conceptually addressed this domain. Are there ideas in the literature on how to imagine the 'subcutaneous' levels of programmed works? Can we find statements about how to think productively about the relationship between the code, the program environment, and the output from a humanities perspective?

Or do the text analyses reveal that the technology-intensive side of software-based works remains a black box? Around the turn of the millennium – in parallel with the advent of artistic browsers – fundamental discussions on the digital image emerged. In the following we will take a brief look at one of these positions to see how far it can take us in our investigation:

If the concepts of surface – interface – subface of the computer graphics pioneer and computer scientist Frieder Nake had to be roughly assigned to what has been outlined so far, the preceding statements would speak of software

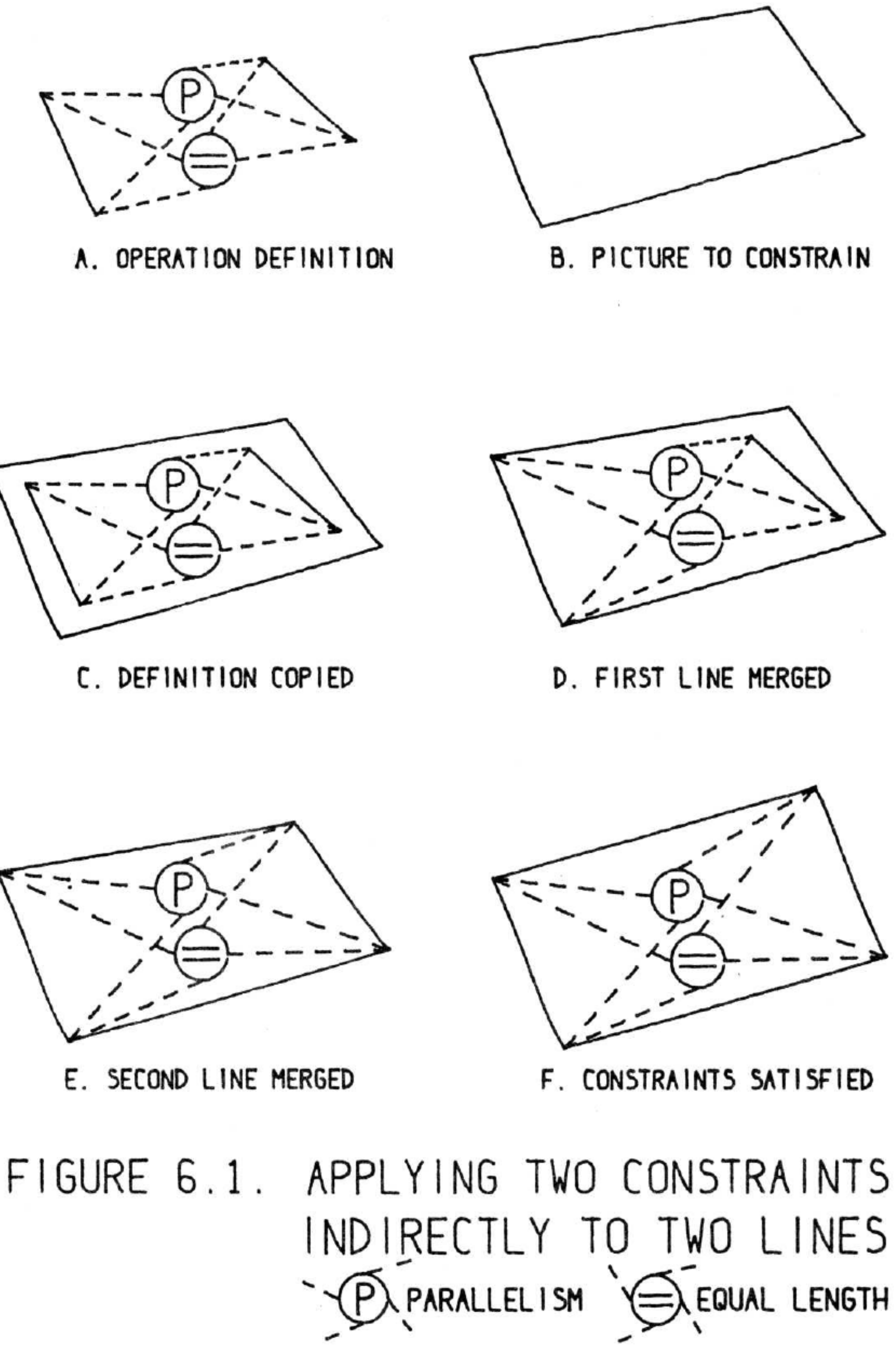


Figure 1. Ivan Sutherland; Sketch depicting the application of a condition composed of two constraints (parallel, equal length). Thus the rectangle becomes a parallelogram; 1963. Ivan Sutherland; Sketchpad: A man-machine graphical communication system, PhD thesis, MIT, Cambridge 1963, fig. 6.1, p. 94. cit. in: Nake, Frieder. "Das doppelte Bild." Bildwelten des Wissens. Kunsthistorisches Jahrbuch für Bildkritik, vol. 3, no. 2 (2005): 40–50, here: 45.

art as (primarily) subface, media art as (primarily) surface. Of course, this is not mutually exclusive. According to Nake, as soon as the image became algorithmic, it doubled, in that in addition to the surface it now simultaneously possessed a “subface inwardness”. He explained this with the help of an illustration found in the seminal work by Ivan Sutherland (fig. 1).

“The image as a digital image has become first and foremost *algorithmic*: It now also possesses a subface/ subfacial inwardness or is surface and subface at the same time. Both – this is decisive – are objectively present. The surface of the digital image is *visible*, while the subface is editable. The surface exists for the user, the subface for the processor. To the subface belongs solely that which exists as data structure and algorithm.”[7]

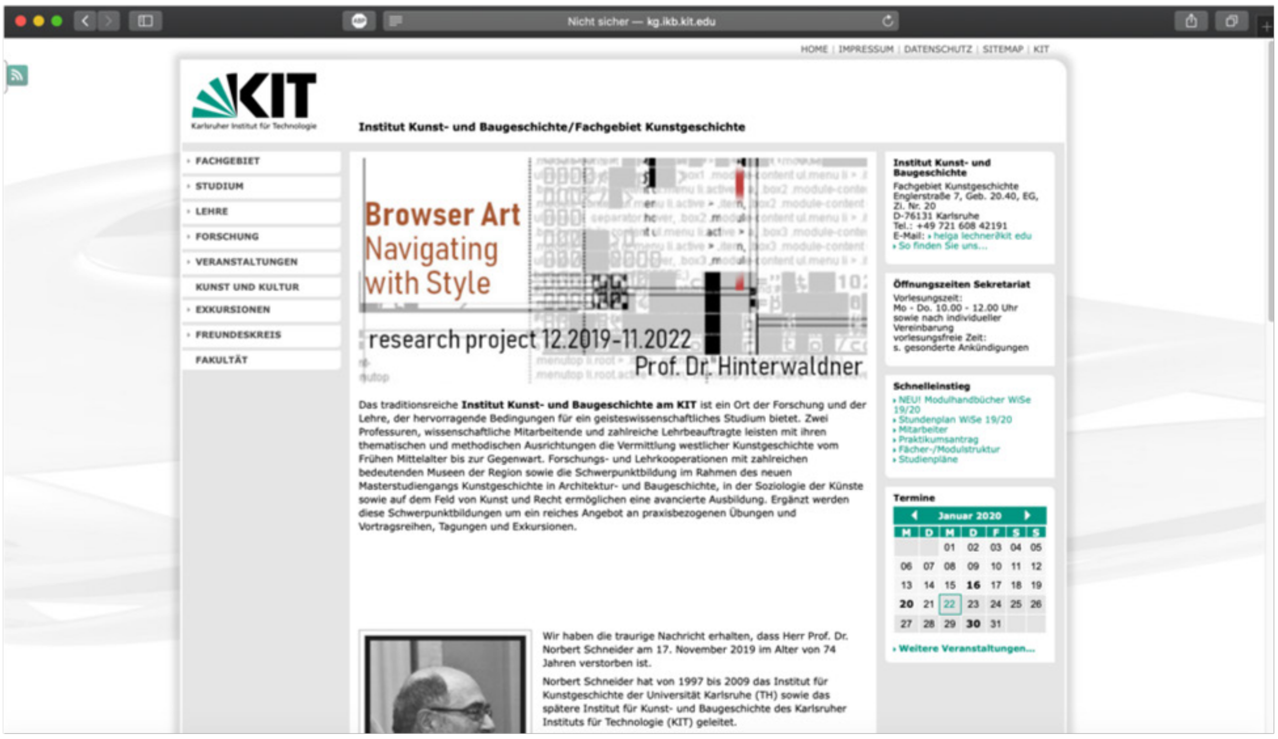
Comparing two web browsers on a phenomenological level – the surface

Using examples from the body of art works we are investigating we will now illustrate how far we can get in an art historical analysis of artistic web browsers without including the source code or Nake’s idea of the subface in a formal description of the artwork. To this end a basic comparison of a web page opened in Apple’s Safari browser 13.0.1 and the same page opened in the Riot browser by the artist Mark Napier from 2005, both opened on the same day on January 22, 2020, seems to be a reasonable first step. The page accessed by both browsers is the former landing page of the art history website at the Karlsruhe Institute of Technology (fig. 2).

The Safari browser interface is roughly composed of a menu bar at the top of the window, including the URL input field, a scroll bar on the right side of the window, and a scalable display area, which allows for the appearance of a further scroll bar at the bottom of the web page. It is not to be expected that a more detailed description of Safari’s interface would lead us to any meaningful insights since in this case the browser interface is not from the same time period and thus cannot tell us much about the similarities or differences to the Riot browser’s interface regarding conventional browser interfaces of the time. Furthermore, while the interface is part of our investigation, it will not be the prime concern of this text. It is the visual output of the webpage being accessed that we are going to focus our attention on for the moment. This visual output is shown in the display area of the web browser. The way it is, or rather was, displayed in the Safari browser is supposedly the way the website developers and authors of the page intended it to be seen. The page shows a clear structure, which roughly consists of a pictorial header or banner, a menu bar on the left, a column with information boxes on the right and a scrollable section in between that contains both text and images. The first element of the middle section is a slider that changes the displayed picture every few seconds. The other parts of the page are, apart from their partial scrollability, static. There is a discernable color scheme with a neutral black for most of the text and a shade of turquoise for selected graphical elements and hyperlinks. The hyperlinks are therefore embedded in the design and structure of the page.

Even when compared to the much more recent Apple browser, Mark Napier’s Riot browser is very similar in its interface. Again we find a menu bar at the top of the window, scroll bars on the right and conditionally at the bottom, and a scalable display field. The menu bar contains a label on the left with the name of the browser, a button for bookmarks, the input field for URLs with the accompanying label ‘address’ and a Windows logo (1992-2006) on the right. The visual output of the page, however, differs considerably from that of the Safari browser. The structure of the page is completely resolved, and images, text and links are subject to layout rules that obviously differ from those followed by the conventional browser.

At the upper left edge of the display area three URLs are listed horizontally in bright green. The first of them is the URL we just typed in, while the two others are the URLs of webpages entered by previous users of the browser. This aspect is already knowledge gained through the artistic narration accompanying the work, and not something that can be directly discerned from the image on the screen. However, there are clear indications that the origin of the content we find is not solely the accessed webpage. We see a graphic of a skull, links and text that seem to be Japanese, and a picture of a cat. None of these elements can be found on the page we accessed. What can be found on our art history homepage are the pictures that are used for the slider. But in contrast to the depiction in the Safari browser we can recognize at least three of these pictures included in the composition, temporally compressed into the same layer, into the same image. It is challenging to describe the occurring elements in a systematic way, because pictures, text and links are organized in an overlapping manner and seem to be concentrated in the


Nicht sicher — kg.ikb.kit.edu
HOME | IMPRESSUM | DATENSCHUTZ | SITEMAP | KIT
KIT
Karlsruher Institut für Technologie

Institut Kunst- und Baugeschichte/Fachgebiet Kunstgeschichte

- FACHGEBIET
- STUDIUM
- LEHRE
- FORSCHUNG
- VERANSTALTUNGEN
- KUNST UND KULTUR
- EXKURSIONEN
- FREUNDESKREIS
- FAKULTÄT

Browser Art
Navigating
with Style
research project 12.2019-11.2022
Prof. Dr. Hinterwaldner

Das traditionsreiche **Institut Kunst- und Baugeschichte am KIT** ist ein Ort der Forschung und der Lehre, der hervorragende Bedingungen für ein geisteswissenschaftliches Studium bietet. Zwei Professuren, wissenschaftliche Mitarbeitende und zahlreiche Lehrbeauftragte leisten mit ihren thematischen und methodischen Ausrichtungen die Vermittlung westlicher Kunstgeschichte vom Frühen Mittelalter bis zur Gegenwart. Forschungs- und Lehrkooperationen mit zahlreichen bedeutenden Museen der Region sowie die Schwerpunktbildung im Rahmen des neuen Masterstudiengangs Kunstgeschichte in Architektur- und Baugeschichte, in der Soziologie der Künste sowie auf dem Feld von Kunst und Recht ermöglichen eine avancierte Ausbildung. Ergänzt werden diese Schwerpunktbildungen um ein reiches Angebot an praxisbezogenen Übungen und Vortragsreihen, Tagungen und Exkursionen.

Wir haben die traurige Nachricht erhalten, dass Herr Prof. Dr. Norbert Schneider am 17. November 2019 im Alter von 74 Jahren verstorben ist.

Norbert Schneider hat von 1997 bis 2009 das Institut für Kunstgeschichte der Universität Karlsruhe (TH) sowie das spätere Institut für Kunst- und Baugeschichte des Karlsruher Instituts für Technologie (KIT) geleitet.

Institut Kunst- und Baugeschichte
Fachgebiet Kunstgeschichte
Englerstraße 7, Geb. 20.40, EG,
Zi. Nr. 20
D-76131 Karlsruhe
Tel.: +49 721 608 42191
E-Mail: › helga lechner@kit edu
› So finden Sie uns...

Öffnungszeiten Sekretariat
Vorlesungszeit:
Mo - Do. 10.00 - 12.00 Uhr
sowie nach individueller Vereinbarung
vorlesungsfreie Zeit:
s. gesonderte Ankündigungen

Schnelleinstieg
› NEU! Modulhandbücher WiSe 19/20
› Stundenplan WiSe 19/20
› Mitarbeiter
› Praktikumsantrag
› Fächer-/Modulstruktur
› Studienpläne

Termine

Januar 2020

M	D	M	D	F	S	S
		01	02	03	04	05
06	07	08	09	10	11	12
13	14	15	**16**	17	18	19
20	21	22	23	24	25	26
27	28	29	**30**	31		

› Weitere Veranstaltungen...

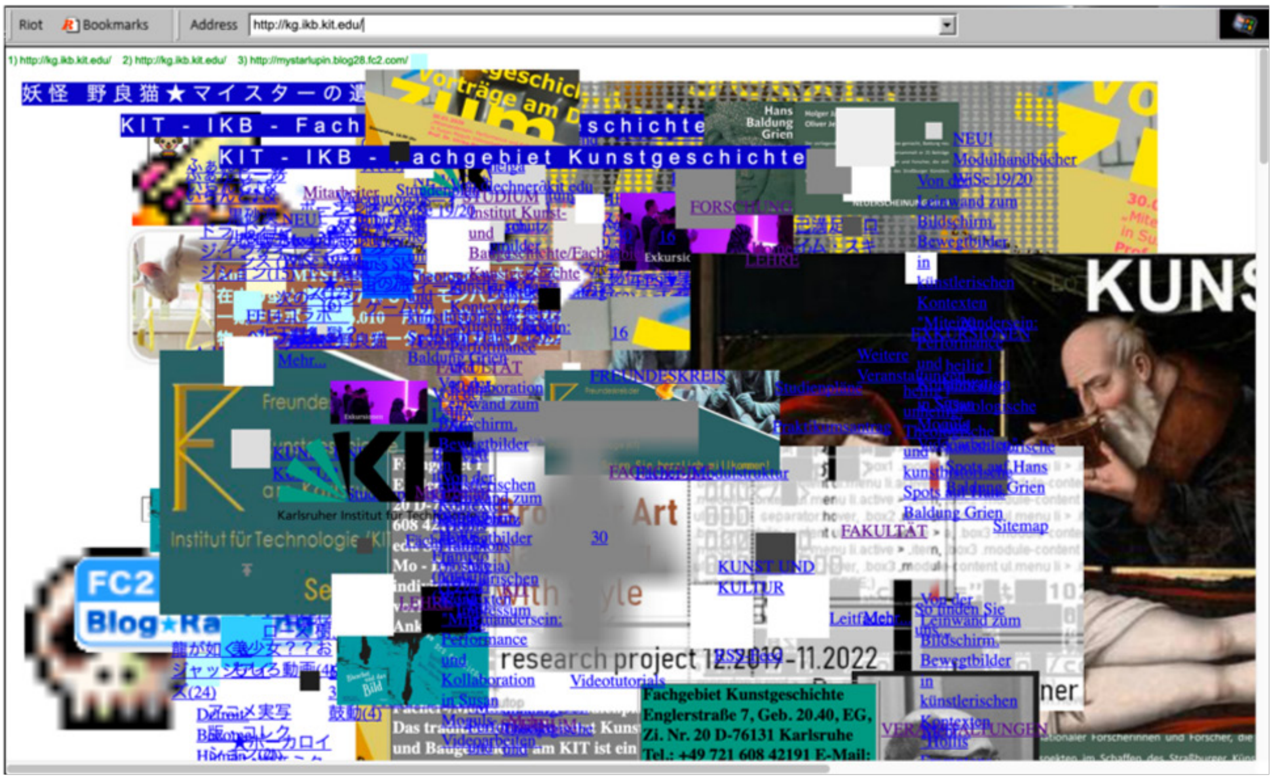

Riot
Bookmarks
Address
http://kg.ikb.kit.edu/
1) http://kg.ikb.kit.edu/ 2) http://kg.ikb.kit.edu/ 3) http://mystarlupin.blog28.fc2.com/
KIT - IKB - Fachgebiet Kunstgeschichte
research project 12.2019-11.2022
Fachgebiet Kunstgeschichte
Englerstraße 7, Geb. 20.40, EG,
Zi. Nr. 20 D-76131 Karlsruhe
Tel.: +49 721 608 42191 E-Mail:
KUNST

Figure 2. above: Apple; Safari Browser 13.01.; 2019; accessed page: http://kg.ikb.kit.edu/; screenshot (January 22, 2020), below: Mark Napier; Riot Browser; 2005; http://www.potatoland.org/riot/; ac-cessed page: http://kg.ikb.kit.edu/; screenshot (January 22, 2020).

middle section of the page displayed in the browser window. And although the page remains scrollable, most of the emerging composition can be seen in the section of the browser window first displayed. All of the intended design for the color scheme and text fonts of the webpage has been discarded. The functionality of the links on the page is retained, but instead of the original turquoise they are displayed in a generic way: underlined and in a blue font.

The resulting image created by re-composing the page's content and collaging it with two other pages detracts from our ability both to recognize the boundaries of the three pages and to access the information included in the intended content of any of them. As this short formal analysis shows, an examination of the visuality produced by the browser already contains a multitude of insights and possibilities for further investigations. Is it therefore possible to answer the questions we as art historians would like to ask about the artwork? These images – one can be sure – are more than they show or at first would have us believe. Some aspects become explicitly evident during user interaction, but other aspects remain hidden within the black box and implicit(ly effective) as long as we do not care about the level of algorithms and codes. Can we describe and compare the work not only on its motivic and iconic but also on the technical-procedural, geometric, ideological and reception-oriented levels? Which parameters play a role when it comes to determining stylistic differences of programmed application? Can we actually recognize a particular artistic style? And if so, where can it be found? In the code? In the resulting image? In the concept underlying it or the interaction it enables? Should we expect information about the variety of interpretations – cast in multimodal outputs – of what a certain artistic intervention is supposed to be and how it should be carried out? Ideas about propagated website norms, user-friendliness, the (re-)education of the user's habits, etc. anchor a specific understanding of the user in each browser.

For this purpose, it seems worthwhile not only to examine the screen outputs, but also to grasp the code in its conceptual dimension. One consequence could be to use computer-assisted imaging in programmed (both artistic and scientific) works to pave the way for analytical access to all levels of design. Without thereby preaching a new positivism, we therefore advocate an expansion of the comparison zone into the creative depth, methodological breadth, and disciplinary diversity within the framework of image-centered approaches.

Why the surface inspection is not enough – the subface

Let us use an example and a small practical experiment to demonstrate why a surface inspection is not enough. This text you are reading is written in Word, a word processing program by Microsoft. Pasting or typing a few lines of text into a document of a text editor gives us a wide range of options for editing that text. Our goal is to make part of the text invisible. Although it probably does not exhaust the functionality of the program, the three most obvious solutions seem to be: a) Select the part of the text to be withdrawn from view and set the font color to the active background color. b) Obscure the text part with a shape of the same color as the background color. And c) delete the text part and optionally replace it with blank spaces. Even though very different actions were performed, and if, in the third solution, we choose to replace the text with blanks, the visual result of all three options is the same and cannot be distinguished by a purely visual inspection. The visual output of a web browser created in an artistic context is visually determined in the same way and the executed processes may be indistinguishable in the visual result.

This is not quite as trivial as the short example in Word might suggest. Several artistic browsers withdraw the visibility of the text and in some cases other visual elements on the web pages, in a very similar manner (fig. 2). Rafaël Rozendaal's and Jonas Lund's Text Free Browsing is a Chrome extension dating from 2013 that erases the text of the accessed pages. Structurally, the pages are unchanged, so that the areas where the text would be remain blank. An ostensibly identical result is produced by the plug-ins Wordless Web by Ji Lee in 2012 and The Deletionist by Nick Montfort, Amaranth Borsuk and Jesper Juul from 2013. The text is purged from the webpage in all three cases, though texts embedded in graphics are still displayed and The Deletionist selects a number of text fragments that will continue to appear in the output. Another example is the simulated browser Internet Implorer from 2000 by Rolux aka Sebastian Lütgert. In this case, both text and images are extracted, but the structure of the website still remains intact. Pages opened in the simulated browser Boxplorer by Andy Deck from 2002 are also liberated from their structural design, so that not only are images and text removed from the website, but also any visual information that could indicate the intended content of the page. The historical visual result of this artistic intervention is primarily a composition of individually nested color fields framed in black. When used with more recent web pages, its structure is highlighted in the same way; the dominant color is white because today's way of designing web pages has changed. The same process of the browser now delivers results lacking colors and unintentionally highlights the structuredness more strongly. If it is now to be assumed that deleting content is an artistic act that differs from the artistic act of concealing, hiding, covering up or undifferentiating it, then it becomes relevant in the analysis to go beyond the mere observation and usage of a web- and computer-based work.

A web browser is a computer program and is therefore based on source code. As the code is largely determined in its interpretation, the easiest way to gain knowledge about the application would be to read the source code. This is a reasonable method used especially in the investigation of smaller applications and with a slightly different focus of inquiry. Two interconnected aspects of our investigation make this approach problematic: First, we focus explicitly on the processes of the artistic web browsers that lead to the perceivable outputs. Although the mere reading of the source code allows for various ensuing approaches, using this method to record the dynamic processes in the program flow is challenging. Second, web browsers are potentially extensive and complex programs. They act as an interface to the World Wide Web and are therefore closely linked to and integrated with the server's hardware and software systems. And so the processes that lead to the output through the browser are not exclusively determined in the web browser's source code.

This also becomes evident when we look at the schematic structure (fig. 3) of a conventional web browser, showing a diversity in infrastructure components. All these components are recorded in source code, which can potentially lead to a relatively high number of lines of code (LOC) to read, especially if all these components are implemented in the artistic projects. A higher number of LOCs does not necessarily lead to higher complexity, but it does make access by reading the code more difficult and it will significantly increase the amount of time involved. This ultimately makes the task of reading and identifying the program's flow and processes more complex. That is an obstacle that should not be underestimated, since we do not intend to limit our examination to a few exemplary artistic browsers but want to include the widest possible selection of works in our analysis. Even if we were both competent and willing to interpret each code in terms of critical hermeneutics, as called for by Mark Marino, representative of critical code studies, there is a good chance that this selection would render the enterprise unmanageable with a close reading approach. Such a 'close reading' can only be done with the help of a community and crowd sourcing.

Methodological Scuba Diving

We see the expansion of the zone of analysis in what academic texts often call 'behind' or 'beneath'. Talk of 'layering' or 'depth' ultimately touches on topological issues. 'Deep tech' refers to the coupling of scientific research and application – and we would add: design. Therefore, we can expect the resulting multimodal imagery to include data, models, parameter settings and so on. For this reason, we understand these programmed or program-based phenomena as 'deep images'. It is therefore mandatory to explore their depths. The most common view is the double-sidedness presented by Nake; besides the surface there is something like an 'underneath'. Do layering and depth represent a problematic way of thinking about our computer systems? In media theory, 'depth' suggests the underlying, the background, the more important, etc.,[8] and thereby implies that the surface is only secondary. Would there be alternative approaches like weaving/mesh[9] or rhizome (after Deleuze/Guattari) as conceptual lenses through which to examine our corpus? Each theory filter highlights something different. We pursue the goal of developing a visualization for image-machines-in-action in order to gain leverage for criticism and comparison at the level of code execution. So it is necessary to consider what basic topological structure to use as a basis. What topological structure do theorists suggest when it comes to characterizing the digital image? A lively discourse on this was developing at about the same time as the phenomena we are investigating. This historical dimension to the discourse should not pose an issue here. In order to take a step forward, we will proceed on several parallel methodical tracks.

A first path is pursued via theory building, so that the existing conceptions of digital imagery can also be incorporated in a reflected way. Several of the relevant texts describe their conceptions of the digital image vividly and metaphorically, so that the attempt of visualizing them in a 'reverse engineering' manner is obvious. A second path is application-centered and leads to an analysis of the individual source codes and their execution. This whole

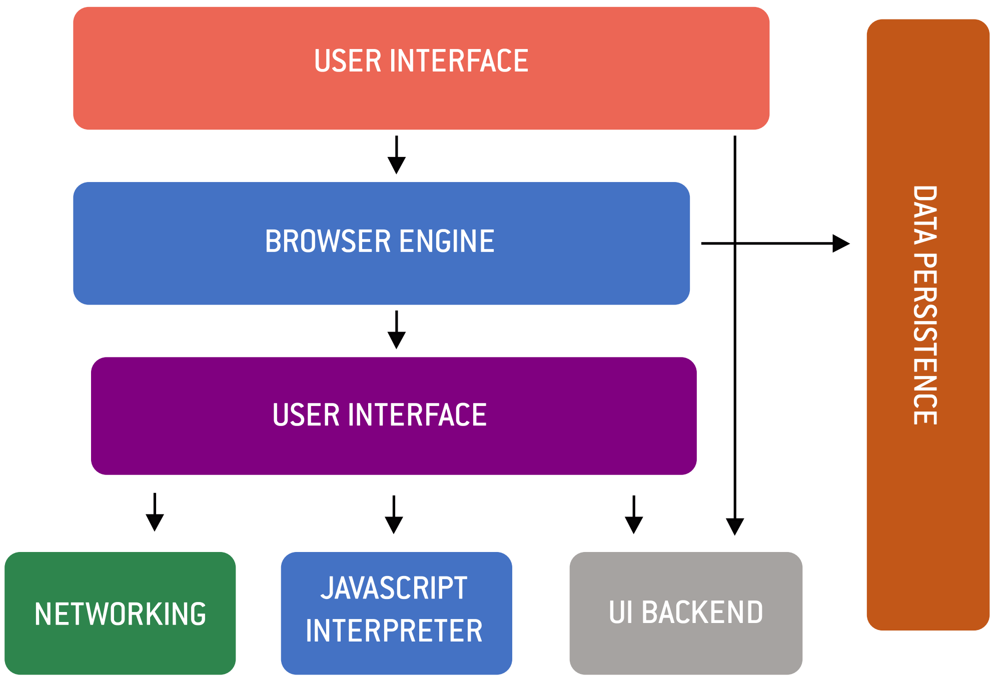


Figure 3. Tali Garsiel; "The schematic structure of a conventional web browser"; tali-garsiel.com/Projects/howbrowserswork1.htm. Refers to Alan Grosskurth; A Reference Architecture for Web Browsers, accessed January 23, 2020. https://grosskurth.ca/papers/browser-refarch.pdf.

area – software visualization – has been a flourishing applied research field since the 1970s.[10] Visualizing the code and the implemented algorithms is often revealing even for specialists. However, this task also brings difficulties with it. Computer scientists help themselves by developing visualizations that show the program structure or the algorithmic performance of the program under investigation in a different way than through lines of text. Many different visualization approaches and implicit topological models can already be distinguished here, so that on this meta-level also image criticism could offer additional insights. An image-critical analysis of visualizations using software visualization tools is, of course, still pending. Nor are we aware of any humanities case studies that have really attempted to get to the bottom of the production of such a complex interactive real-time image-machine.[11]

Can these visualization types and visualization sources (from theory humanities and software visualization computer science) be related to each other? Is it desirable, usable, viable to do so? If one holds these two kinds of visualizations up against each other as meta-images, friction and incommensurabilities are to be expected, which can have a retroactive effect on our theory building.

Visualizing software dynamics for art history

We only know of a handful of projects utilizing advanced computer-based methods for analyzing programmed, digitally generated image worlds for humanistic ends. Most of them are to be found in game studies. Shane Denson and Andreas Jahn-Sudmann pursued the goal of comparing all existing game modifications of Super Mario Bros (originally 1983).[12] By 2014 they had collected 206 such game mods. What would be the best way of dealing with so many versions, each of which would require several hours of gaming, not to mention the need for studying the code and further para texts like the 'read me' files? Denson decided to go for a 'distant reading' at the code level. He needed a panoramic view of the 'subface'. This diagram (fig. 4) lists the different modifications vertically. The horizontal bars to the right of the titles symbolize the respective code. The orange areas mark paragraphs that were altered in comparison to the original from 1983. This helps at least to estimate the extent of the changes and to indicate where they can be found in the code. Code analysis could develop into a new auxiliary science for art history.

The term 'multi-scale view' was coined by the media and film scholar Lev Manovich. In 2007 he proposed a method termed 'Cultural Analytics' that offers a so-called 'multi-scale view' of big data pools. 'Multi-scale view' means providing a variety of tools for visualizing an artifact or a collection of artifacts on different levels of description, thus allowing for a 'distant reading'. In our view Denson and Jahn-Sudmann have begun to expand the 'multi-scale view' that Manovich had developed for the 'surface' to include the levels of programming and modelling. Code analysis and various forensic software could develop into a new auxiliary science for art history too.

This can be transmitted to browsers if it turns out the codes are sufficiently 'connatural' (related). One hypothesis to be tested is that artists have not reprogrammed all aspects of a browser but instead have adopted large portions of the code from conventional browsers that remained identical in the new application. Should this hypothesis be confirmed, the precondition for a machine-assisted comparison would be met and it would be possible to apply a visualization method such as that used by Denson and Jahn-Sudmann. Our project could thus show where artistic intervention takes place and identify those processes a web browser performs that are of intrinsic interest to artists. For it can be assumed that not all components of the web browser are equally interesting for the artistic intervention. This is also indicated by the artistic development of browser extensions (called add-ons, plug-ins or extensions, depending on the browser). These extensions modify or supplement individual functions of a conventional browser, but use the existing functions of the web browser as their basic technology.

One example is the artistic Chrome extension Text Free Browsing, which is mentioned in its general functions above. If we take a look at the complete code of the Chrome extension (fig. 5) we see there are only a few lines. This suggests that only very few lines of code are needed to generate far-reaching changes in the output of the web page. If we take a closer look at the individual sections of the code, we see that the whole section shown in figure 5a does nothing more than define the design and function of the button for the extension in the Chrome browser's interface. So only a few lines of code affect the difference in the output of the page when compared to the output of a conventional browser. If these lines of code were embedded in a browser of a similar size, such as the Chrome browser, as might be the case in the artworks that comprise a complete application, then our analysis faces the simple problem of finding these lines of code.

If a visualization technique could show these deviating sections in the source codes, it could be an extremely helpful tool. However, the application of such a tool is bound to a number of conditions, formulated above as a hypothesis. These conditions are that the artistic browsers are not only similar in some of their basic functions, but also that this functional equality is reflected in the structure of the source code. Such equality can only be presumed if common code segments are actually being used. Whether this is the case, and whether this approximation will be possible, is yet to be determined in our further investigation of the artworks.

However, if it is indeed possible to employ this tool, it will only provide an overview of the sections of the code that require a more detailed analysis. The color-marked bars in the Gantt chart visualize the sections in which the modified program differs from the original program. They do not, however, tell us anything about *what* happens in these modified areas or *when* in the program flow they come into effect, since the arrangement of the source code does not allow any conclusions to be drawn about the dynamic execution of the program. This visualization would thus only be a first step in the analysis of the source code with the objective of making visible the processes that lead to the perceptible output of the artistic browsers.

The software visualization offers many approaches for different queries concerning the program, individual program parts or program functions. Would it then be conceivable for our project to visually track the data requested from the server to the accessed website through the processes of the program? After all, we want to know how the reception and processing of the same data package can lead to displays that differ significantly from those of conventional browsers. Would it be possible to imagine the path of this data package through the program as being equivalent to a conveyor belt that is carrying a packet through a factory? Where maybe at the first station it is cut open and its contents are distributed to branching conveyor belts? And where at a subsequent station parts of the content are cut out, pasted over or rearranged and then at another station perhaps reunited with other contents of the package to arrive at the end of the assembly line as the output of the respective artistic browser?

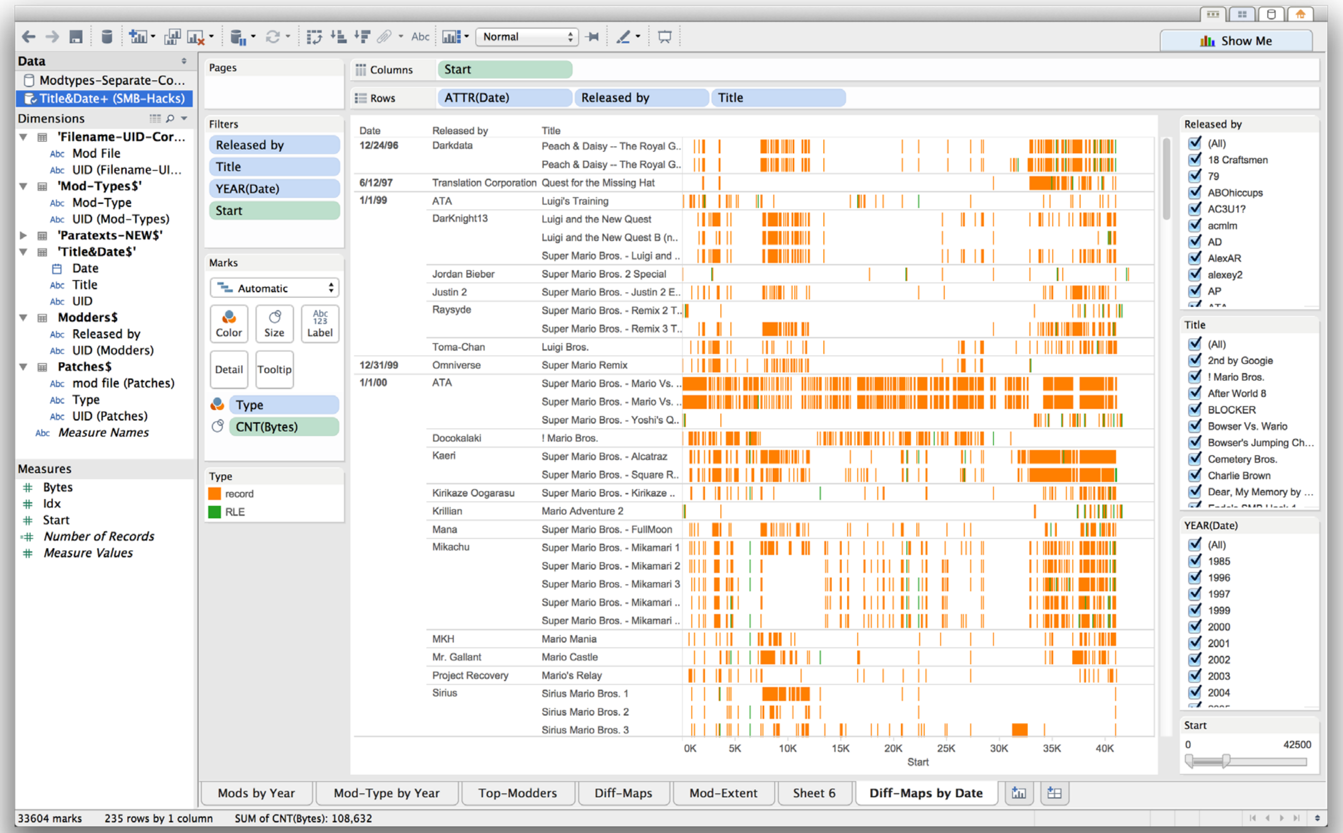


Figure 4. Shane Denson. "Super Mario Bros modifications as seen through Diffmaps by Modder"; 2014; screenshot. Shane Denson; "Visualizing Digital Seriality, Or: All Your Mods Are Belong to Us!". In Kairos. A Journal of Rhetoric, Technology, and Pedagogy, vol. 22, no. 1 (2017), fig. 11, accessed January 22, 2021. http://kairos.technorhetoric.net/22.1/topoi/denson/index.html.

Flowcharts are a common way of visualizing software in its structure and processes. For example, in figure 6 we see the control flow diagram of a rendering engine. The diagram also visualizes – although perhaps less dramatically than with our conveyor belt analogy – the path of the data through the rendering engine. However, it does not illustrate what specifically happens at each individual point. In order to be able to visualize the information about what happens at the individual steps of this process without losing the overview of the process flow in its entirety, it is necessary to increase the dimensions of the representation. The obvious additions would be those of another spatial and a temporal dimension. But the diagram can also be enhanced by further dimensions like form, color, or size of the data object and its contents. One could imagine that the idea of an assembly line operating in an image-producing factory might no longer seem that far-fetched at this point.

In an effort to produce further results for our analysis, it would be desirable to be able to compare the artistic browsers with each other. However, the visualization in which the data moves through a set environment would make such a comparison problematic. Can we assume that all process steps can be found in every browser and that those common points only differ in the operations performed on the data package? Is it not the difference in the program environment of each individual artistic browser through which the data package moves that distinguishes the browsers and the visual output they produce? As we have seen in the example of the Text Free Browsing extension, the fundamental differences in the source code of each browser are possibly only very small. In a visualization that requires a certain degree of abstraction to model the basic processes of a web browser, these differences might be hardly perceivable or displayable at all. The development of possibilities that make both the macro- and micro-variability in the individual art projects visible and comprehensible and yet which still establish comparability is another task that we will address in the further course of our project.

Conclusion

In the early 2000s the digital image is often described as something different to more familiar phenomena: Nake starts from a drawing that is seen, and attributes the innovation that comes with the digital image to an incorporated splinter of intelligence.[13] The literary scholar Mark B. N. Hansen refers to Gilles Deleuze. He agrees with the latter on setting the cinematographic image or time-image as a starting point and discusses the conditions of the continuation of the tendencies already laid out in the cinema.[14] Media scientist Wolfgang Hagen starts with photography and presents a probability-only scenario via quantum mechanical principles.[15] In his text "Das digitale Bild gibt es nicht" (The digital image does not exist), media theorist Claus Pias focuses on the conversion pipeline and the opposition of visibility and digitality.[16] The image theorist and performance artist Ingrid Hoelzl bids farewell to dichotomies of all kinds and activates an economic transaction model, pairing this with New Materialism approaches to an intra-image and a networked image with egalitarian agents of various kinds.[17]

For the sake of brevity, we cannot go into the individual results of these analyses here. They have the following in common: they all speak of the digital image as a prototypical unity/multiplicity/processuality, etc.; thus, they are not concerned with portraying a specific application.

For a better comparability of these positions, we aim conversely at visualizing the lines of argumentation of the individual texts. In the text and discourse analysis we were able to extract and sketch the subcutaneously implied topological and symbolic clues related to the respective understanding of the digital image. In fact, we would literally have to animate every text in order to witness the argumentation step by step and to filter out which topological ideas appear at what point and where they remain indeterminate.

Why do we need these pictorial textual analyses? They serve to set the conceptual-topological course for the most productive transfer of understanding to a particular programmed work. According to our questions, productivity is measured in terms of the clarification of those elements which are phenomenologically decisive for the differences. It is still unclear whether (or how) the representations resulting from the theoretical analyses of digital images *in general* are transferable to sketches of concrete programmed works.

It is confusing to attempt to highlight the components of a *specific* web browser without visual aids. In contrast, figure 3 depicts the essential components of a generic web browser. Even in this simple schema, each of these squares is a black box in itself. The diagram clearly conveys the composite nature of a web browser. There are some obvious shortcomings: First, it does not tell us anything about the individual browser. Second, this sketch is not developed in accordance with our main interests and does not support art historical analyses of artistic web browsers. Third, it does not explain in any way how the elements are actually connected via the code and the program, or how they work together dynamically.

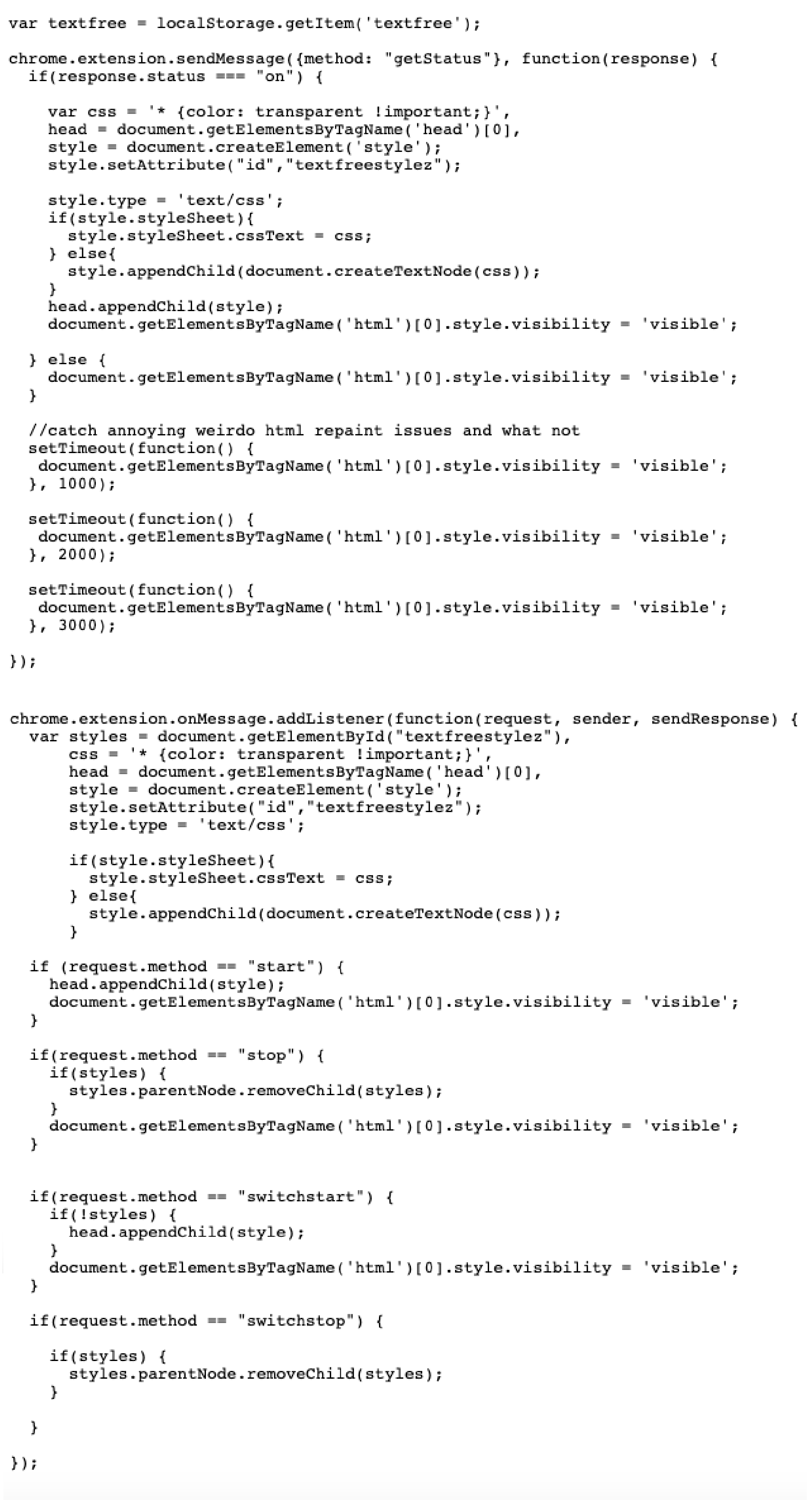
All these elements play a role. But to grasp their coherence in actu, it is necessary to find a way to inspect them in their interlocking performance. Geoff Cox, Alex McLean, and Adrian Ward argue that – analogously to poetry – the aesthetic value of code lies in its execution, performance, or presentation, and not simply in its written form. Nor in the analyzed written form that distinguishes subsections (generative aesthetics analyzes processes anyway).

But in order to really appreciate the generative code and to grasp what is going on, we must perceive it – "we need to 'sense' the code".[18] Otherwise we would have to admit that we have only a restricted view. Consequently, scholars face the danger of a limited ability to criticize as soon as they methodically separate the code from its actions or the actions resulting from it. The written code – as revealing as the analysis of it may be, because this writing is also individually fabricated and idiosyncratically colored in many ways, despite its rigid grammar and orthography – does not yet reveal the poetic and functional qualities of its performance. In the words of Cox and colleagues: "Code is a notation of an internal structure that the computer is executing, expressing ideas, logic, and decisions that operate as an extension of the author's intentions. The written form is merely a computer-readable notation of logic, and is a representation of this process. Yet the written code is not what the computer really executes, since interpreting and compiling and linking takes place on many levels. Code is only really understandable within the context of its overall structure – this is what makes it like a language (be it source code or machine code, or even raw bytes)."[19]

```
var on = 0;
chrome.browserAction.onClicked.addListener(function(tab) {
  if(on === 0) {

    chrome.browserAction.setIcon({path: 'text-no.png'});
    chrome.browserAction.setTitle({title: "Stop Text Free Browsing"});
    on = 1;
    localStorage.setItem("textfree", "on");

    chrome.tabs.getSelected(null,function(tab) {
      chrome.tabs.sendMessage(tab.id,{method:"start"}, function(response){
      });
    });

  } else {

    on = 0;
    localStorage.setItem("textfree", "off");
    chrome.browserAction.setIcon({path: 'text-yes.png'});
    chrome.browserAction.setTitle({title: "Start Text Free Browsing"});


    chrome.tabs.getSelected(null,function(tab) {
      chrome.tabs.sendMessage(tab.id,{method:"stop"}, function(response){
      });
    });


  }
});

chrome.extension.onMessage.addListener(function(request, sender, sendResponse) {
    if (request.method == "getStatus")
      sendResponse({status: localStorage['textfree']});
    else
      sendResponse({}); // snub them.
});


chrome.tabs.onActivated.addListener(function(info) {
  var tabID = info.tabId;
  if(localStorage.getItem("textfree") === "on") {
    chrome.tabs.sendMessage(tabID, {method:"switchstart"}, function(response){
    });

  } else {
    chrome.tabs.sendMessage(tabID, {method:"switchstop"}, function(response){
    });

  }
});

//get all tabs and insert check.js
chrome.tabs.query({}, function(tabs) {
  for (var i = 0; i < tabs.length; i++) {
    var tabid = tabs[i].id;
    chrome.tabs.executeScript(tabid, {
        file: "check.js"
    });
  }
});
```

```
html {visibility: hidden;}
```

```
html {visibility: visible !important;}
```



```
var textfree = localStorage.getItem('textfree');

chrome.extension.sendMessage({method: "getStatus"}, function(response) {
  if(response.status === "on") {

    var css = '* {color: transparent !important;}',
    head = document.getElementsByTagName('head')[0],
    style = document.createElement('style');
    style.setAttribute("id","textfreestylez");

    style.type = 'text/css';
    if(style.styleSheet){
      style.styleSheet.cssText = css;
    } else{
      style.appendChild(document.createTextNode(css));
    }
    head.appendChild(style);
    document.getElementsByTagName('html')[0].style.visibility = 'visible';

  } else {
    document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }

  //catch annoying weirdo html repaint issues and what not
  setTimeout(function() {
   document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }, 1000);

  setTimeout(function() {
   document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }, 2000);

  setTimeout(function() {
   document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }, 3000);

});


chrome.extension.onMessage.addListener(function(request, sender, sendResponse) {
  var styles = document.getElementById("textfreestylez"),
      css = '* {color: transparent !important;}',
      head = document.getElementsByTagName('head')[0],
      style = document.createElement('style');
      style.setAttribute("id","textfreestylez");
      style.type = 'text/css';

      if(style.styleSheet){
        style.styleSheet.cssText = css;
      } else{
        style.appendChild(document.createTextNode(css));
      }

  if (request.method == "start") {
    head.appendChild(style);
    document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }

  if(request.method == "stop") {
    if(styles) {
      styles.parentNode.removeChild(styles);
    }
    document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }


  if(request.method == "switchstart") {
    if(!styles) {
      head.appendChild(style);
    }
    document.getElementsByTagName('html')[0].style.visibility = 'visible';
  }

  if(request.method == "switchstop") {

    if(styles) {
      styles.parentNode.removeChild(styles);
    }

  }

});
```

Figure 5. Rafaël Rozendaal, and Jonas Lund; Source code of the Chrome extension Text Free Browsing; 2013; screenshot; back-ground.js (p. 50, above), hiddens.html and shows.css (p. 50, below), check.js (p. 51).

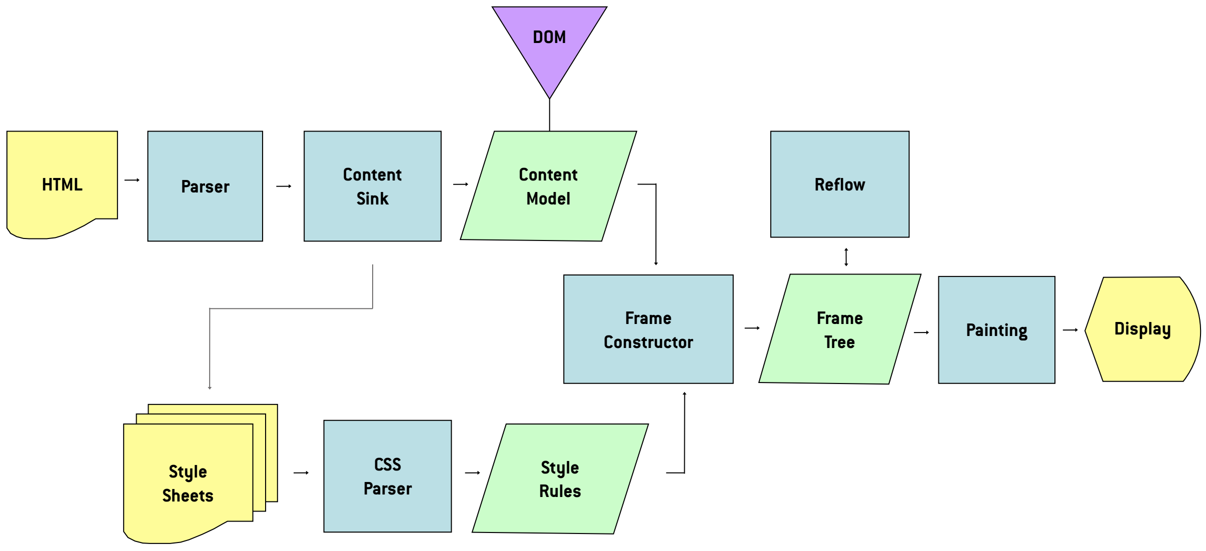


Figure 6. Tali Garsiel; Flowchart of the Mozilla's Gecko rendering engine main flow; 2011; Garsiel, Tali, and Paul Irish. "How Browsers Work: Behind the scenes of modern web browsers." Last modified August 05, 2011. https://www.html5rocks.com/en/tutorials/internals/how-browserswork/. Refers to Chris Waterson. "Mozilla's Gecko rendering engine main flow." (2002), accessed January 28, 2020. https://www.mozilla.org/de/newlayout/doc/gecko-overview.htm.

It follows that the subface must also be multiplied, and processes must be focused. A scaling problem is foreseeable. Finding a way to visualize the scanning of the decision tree of program structures of artistic internet browsers in actu involves witnessing these processes running in secret and comparing them on the level of their procedures and functional mechanisms. Time-based portraits of the respective effective mode of operation allow for observing how the synthesizing of internet content proceeds in a browser-specific way.

We therefore wish to extend the territory of analysis so that it encompasses the perceivable audio-visual screen output through to sequences of program mechanics. This could ultimately be framed as a re-phenomenologizing of generative aesthetics. For the investigation, this ultimately means we need a tool, i.e., a close reading using meta-images. We see that there is still work to be done in visualizing generative aesthetics in procedural applications, we still see a task ahead – and with browsers that means not only surfing but also diving.

NOTES

1 Florian Cramer, "Concepts, Notations, Software, Art." *Netzliteratur.net* (March 23, 2002), accessed December 14, 2002, https://www.netzliteratur.net/cramer/concepts_notations_software_art.html.

2 Johannes Auer, "Screaming Screen und binärer Idealismus/Screaming Screen and Binary Idealism," in *p0es1s. Ästhetik digitaler Poesie / The aesthetics of digital poetry*, ed. Friedrich W. Block, Christiane Heibach, and Karin Wenz (Ostfildern-Ruit: Hatje Cantz, 2004), 277–282.

3 Margarete Pratschke, "Die grafische Benutzeroberfläche als Bild. Zur Rezeption von Rudolf Arnheim und Ernst Gombrich in der Computer Science der 1970er Jahre," *kritische berichte* 37, no. 4 (2009): 54. Translated by the authors.

4 Pratschke, "Die grafische Benutzeroberfläche als Bild. 54.

5 Dirk Paesmans, interview by Tilman Baumgärtel, October 6, 1997. See also Inge Hinterwaldner, "Programmierte Operativität und operative Bildlichkeit," in *Die Kunst der Systemik*, ed. Mikuláš Roman, Sibylle Moser, and Karin S. Wozonig, (Münster: Lit Verlag, 2013), 77–108.

6 Thomas Dreher, "NetArt: Links (alphabetisch)," IASL online, http://iasl.uni-muenchen.de/links/NALink.html, accessed December 15, 2020. Dreher refers to: Nina Kahnwald, "Kunstbrowser. Neue Strategien der Inszenierung von Informationsstrukturen." Magisterthesis, Freie Universität Berlin, 2002. Print version: Kahnwald, Nina. *Netzkunst als Medienkritik. Neue Strategien der Inszenierung von Informationsstrukturen* (Munich: kopaed 2006).

7 Frieder Nake, "Das doppelte Bild," *Bildwelten des Wissens. Kunsthistorisches Jahrbuch für Bildkritik* 3, no. 2 (2005): 47.

8 Cf. Boris Groys, *Unter Verdacht. Eine Phänomenologie der Medien* (Munich: Carl Hanser, 2000).

[9] Cf. Birgit Schneider, *Textiles Prozessieren. Eine Mediengeschichte der Lochkartenweberei* (Zurich: diaphanes, 2007).

[10] Cf. John T. Stasko et al., eds. *Software Visualization* (Cambridge, London: MIT Press, 1998). Stephan Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software* (Berlin, Heidelberg: Springer, 2007).

[11] In corresponding discussions, the view repeatedly comes to light that such a positivistic approach, which also looks at small details, does not justify the effort, as marginal results are to be expected. However, it is important to know that the code explicitly and effectively implements policies and biases regarding the (role of the) users.

[12] Shane Denson, "Visualizing Digital Seriality," in *Duke Visualization Friday Forum, Durham, Duke University*, January 16, 2015, accessed April 8, 2016, https://compsci.capture.duke.edu/Panopto/Pages/Viewer.aspx?id=e4a5e2b8-1bdd-4ad5-b734-be4bc23baf2e.

[13] Frieder Nake, "The Display as a Looking-Glass: Zu Ivan E. Sutherlands früher Vision der grafischen Datenverarbeitung," in *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*, ed. Hans Dieter Hellige (Berlin, Heidelberg: Springer, 2004), 339–365.

[14] Mark B. N. Hansen, "Cinema Beyond Cybernetics, or How to Frame the Digital Image," *Configurations* 10, no. 1 (2002): 51–90.

[15] Wolfgang Hagen, "Computer-Bild-Welten," in *Vortrag Kunsthalle Vaduz*, 2001, accessed December 5, 2020, https://docplayer.org/65985762-Wolfgang-hagen-computer-bild-welten.html. Wolfgang Hagen, "Es gibt kein 'digitales Bild'. Eine medienepistemologische Anmerkung," in Licht und Leitung, ed. Lorenz Engell, Bernhard Siegert, and Joseph Vogl (Weimar: Bauhaus-Universität, 2002), 103–112.

[16] Claus Pias, "Das digitale Bild gibt es nicht – Über das (Nicht-)Wissen der Bilder und die informatische Illusion," *zeitenblicke* 2, no. 1 (2003), accessed May 1, 2007, http://www.zeitenblicke.historicum.net/2003/01/pias/index.html.

[17] Ingrid Hoelzl, "Image-Transaction," in *Parallax, Vol. 25, Nr. 4: Networked Liminalities*, ed. Grant Bollmer and Yigit Soncul (November 2019) and in *Retracing Political Dimensions: Strategies in Contemporary New Media Art*, ed. Oliver Grau, and Inge Hinterwaldner (Berlin: De Gruyter, 2020), 19–33.

[18] Geoff Cox, Alex McLean, and Adrian Ward, "The Aesthetics of Generative Code", in *International Conference on Generative Art*, 2000, accessed November 3, 2017, url: http://web.archive.org/web/20100227185447/ http://generative.net/papers/aesthetics/ (archived).

[19] Cox, McLean, and Ward, "The Aesthetics of Generative Code."

BIBLIOGRAPHY

Auer, Johannes. "Screaming Screen und binärer Idealismus/Screeming Screen and Binary Idealism." In *p0es1s. Ästhetik digitaler Poesie / The aesthetics of digital poetry*, edited by Friedrich W. Block, Christiane Heibach, and Karin Wenz, 277–282. Ostfildern-Ruit: Hatje Cantz, 2004.

Cramer, Florian. "Concepts, Notations, Software, Art." *Netzliteratur.net*, Last modified March 23, 2002. https://www.netzliteratur.net/cramer/concepts_notations_software_art.html.

Cox, Geoff, Alex McLean, and Adrian Ward. "The Aesthetics of Generative Code." In *International Conference on Generative Art*, 2000, accessed November 3, 2017. http://web.archive.org/web/20100227185447/http:/generative.net/papers/aesthetics/ (archived) or https://www.academia.edu/10519146/The_Aesthetics_of_Generative_Code.

Denson, Shane. "Visualizing Digital Seriality," filmed January 16, 2015 at *Duke Visualization Friday Forum*, Durham, video, 50:56, http://compsci.capture.duke.edu/Panopto/Pages/Viewer.aspx?id=e4a5e2b8-1bdd-4ad5-b734-be4bc23baf2e.

Diehl, Stephan. *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Berlin/Heidelberg: Springer, 2007.

Dreher, Thomas. "NetArt: Links (alphabetisch)." In *IASL online*. Accessed December 15, 2020. http://iasl.uni-muenchen.de/links/NALink.html.

Groys, Boris. *Unter Verdacht: eine Phänomenologie der Medien*. Munich: Carl Hanser, 2000.

Hagen, Wolfgang. "Computer-Bild-Welten." Last modified 2001. https://docplayer.org/65985762-Wolfgang-hagen-computer-bild-welten.html.

Hagen, Wolfgang. "Es gibt kein 'digitales Bild'. Eine medienepistemologische Anmerkung." In *Licht und Leitung*, edited by Lorenz Engell, Bernhard Siegert, and Joseph Vogl, 103–112. Weimar: Bauhaus-Universität, 2002.

Hansen, Mark B. N. "Cinema Beyond Cybernetics, or How to Frame the Digital Image." *Configurations* 10, no. 1 (Winter 2002): 51–90.

Hinterwaldner, Inge. "Programmierte Operativität und operative Bildlichkeit." In *Die Kunst der Systemik*, edited by Roman Mikuláš, Sibylle Moser, and Karin S. Wozonig, 77–108. Münster: Lit Verlag, 2013.

Hoelzl, Ingrid. "Image-Transaction." In *Retracing Political Dimensions: Strategies in Contemporary New Media Art*, edited by Oliver Grau and Inge Hinterwaldner, 19–33. Berlin: De Gruyter, 2020.

Kahnwald, Nina. "Kunstbrowser. Neue Strategien der Inszenierung von Informationsstrukturen." Magisterthesis, Freie Universität Berlin, 2002. Print version: Kahnwald, Nina. *Netzkunst als Medienkritik. Neue Strategien der Inszenierung von Informationsstrukturen*. Munich: kopaed, 2006.

Nake, Frieder. "The Display as a Looking-Glass: Zu Ivan E. Sutherlands früher Vision der grafischen Datenverarbeitung." In *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*, edited by Hans Dieter Hellige, 339–365. Berlin/Heidelberg: Springer, 2004.

Nake, Frieder. "Das doppelte Bild." *Bildwelten des Wissens. Kunsthistorisches Jahrbuch für Bildkritik* 3, no. 2 (2005): 40–50.

Paesmans, Dirk. "Interview with Jodi." Interview by Tilman Baumgärtel. *Telepolis*, October 6, 1997. https://www.heise.de/tp/features/Interview-with-Jodi-3446080.html.

Pias, Claus. "Das digitale Bild gibt es nicht – Über das (Nicht-)Wissen der Bilder und die informatische Illusion." *zeitenblicke* 2, no. 1 (May 2003). http://www.zeitenblicke.historicum.net/2003/01/pias/index.html.

Pratschke, Margarete. "Die grafische Benutzeroberflächen als Bild. Zur Rezeption von Rudolf Arnheim und Ernst Gombrich in der Computer Science der 1970er Jahre." *kritische berichte* 37, no. 4 (2009): 54–63.

Schneider, Birgit. *Textiles Prozessieren. Eine Mediengeschichte der Lochkartenweberei*. Zurich: diaphanes, 2007.

Stasko, John T., John B. Dominigue, Marc H. Brown, and Blaine A. Price, eds. *Software Visualization*. Cambridge, London: MIT Press, 1998.

INGE HINTERWALDNER received her PhD in art history from the University of Basel with a thesis on interactive computer simulations (The Systemic Image, MIT Press 2017). Fellowships and grants allowed her to pursue her research at MECS in Lueneburg, Duke University, and MIT. From 2016-2018 she was Professor for Modern and Contemporary Art at the Humboldt University in Berlin. Since 2018 she has held a professorship for art history at the Karlsruhe Institute of Technology, Germany. Her research focuses on computer-based art and architecture, model theory, art & technology since the 19th century. Currently she is writing on Fluid Form Conceptions. She has co-edited volumes addressing medical and scientific visualizations as composites (2006), the relation between imaging and modelling (2011, 2017), disposable images (2016), and current political dimensions in contemporary art (2021).

Correspondence e-mail: inge.hinterwaldner@kit.edu

DANIELA HÖNIGSBERG is an academic assistant at the Karlsruhe Institute of Technology in the Department of Art History. Since 2014, she has been working on her PhD at Technical University of Berlin on the topic of software as artistic material. In 2015, she worked as a research assistant at the TU Berlin for six months preparing the proposal for the project Authorship 2.0. Her research focuses on computer-based art of the 1990s and early 2000s, intertwinements of arts and sciences, and the theory of classical modernism and early postwar modernism.

Correspondence e-mail: daniela.hoenigsberg@kit.edu