DRS 2018 Design Research Society 2018 Catalyst

University of Limerick
25th–28th June 2018

# Reinventing Graphic Design Software by Bridging the Gap Between Graphical User Interfaces and Programming

MAUDET Nolwenn

The University of Tokyo
nolwenn.maudet@gmail.com
doi: 10.21606/drs.2018.611

Graphic Design Software Applications radically transformed the practice and the industry of graphic design. However, they barely evolved since their introduction, leading designers to question their ubiquity. In this paper, we explore this mismatch by analysing digital design tools through two lenses. We first investigate digital design tools from a "lineage" perspective: how they reproduced the pre-existing design tools and practices. We then use two familiar examples: the colour picker and the alignment and distribution commands to explore the vision of design that they promote. We reveal how these tools assume that designers already have in mind a desired outcome and thus introduce a mismatch with current designers' practices. To bridge this gap, we propose "graphical substrates", interactive and visual tools that combine the strengths of both programming and graphical user interfaces. We analyse how several recent research design tools embed this approach and we propose two principles: tweaking and creation from example to foster their adoption by designers.

*design tools; graphic design; graphical user interfaces*

## 1 Introduction

Graphic Design Software Applications revolutionised the graphic design process as soon as they were introduced in personal computers. Under the name of desktop publishing, they greatly facilitated and optimised the different steps of the graphic design and production process. Designers could finally access and interact with real time visualisation of their work. Before the digitalisation of the printing industry, graphic design was an entire industry with many different and complementary professions (typesetters, paste-up artists, photomechanical technicians...) coexisting with complex machinery to operate (Briar, 2017). The profound transformation led by the adoption of graphic design software applications first drew a lot of criticism from established designers (Armstrong, 2016) but they were rapidly adopted by the industry. More than 25 years after, we saw the democratisation of internet and the wide adoption of mobile phones. Design practice accompanied this movement and many novel design disciplines appeared, including interaction design and user

experience design. Yet, contrary to design practice, the digital design software landscape mostly did not change. Some of the same design applications that were introduced in the 1990's are still being used by graphic designers almost 30 years later.

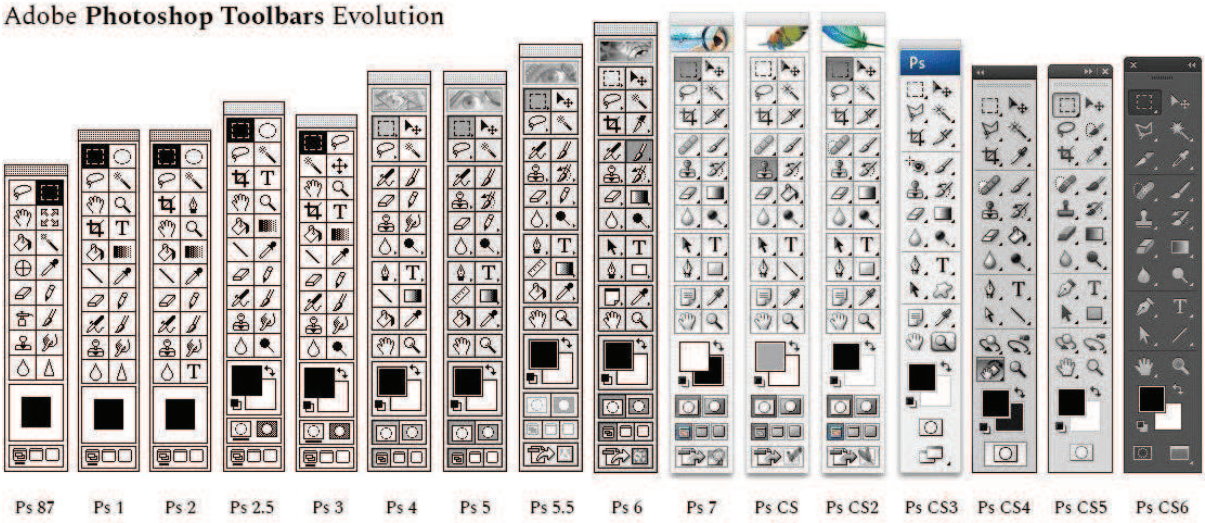


*Figure 1 - Comparison of Adobe Photoshop Toolbars since 1987. Note how little they have changed.*

For example, if we look at the toolbars from Adobe Photoshop, one of the most iconic design software application, over the years we can see that they provide the same tools since their origin and only added few new ones (Figure 1). Following McGrenere's analysis of mainstream software, we could describe their evolution as a form of "software bloat" (McGrenere, 2000). Does this mean that designer's tools are a solved problem? Two different elements demonstrate that design software remains an open question.

First, the importance of design tools is particularly striking when we consider the reasons behind design birth. Design birth is generally traced back to the industrialisation of Britain in the 19th century. For design pioneer William Morris and the British Arts and Crafts movement, the emerging industrialised mass production meant a uniformisation of the resulting products, as well as a degradation in product quality (Morris, 1884). In response to this trend, they advocated for a tighter connection between design, craft and production. Before the era of industrialisation and the separation of people and the means of production, craftsmen could create their own tools. As we can see in Figure 2, they were ingenious in adapting their tools to one's hand size and handedness, or to achieve particular effects. Morris sought to preserve this tradition. A few decades later, the pioneer Bauhaus design school encouraged its students to embrace machines and explore their potential. Designers were to appropriate industrial processes to create high quality products (Papanek, 1972). Thus, one of the first goals of designers was to reappropriate production means and to fusion design and production. Following this line of thought, separating the question of design and design tools is impossible. Design Software is an open issue because part of a designer's work ethos is to choose and question their tools.

The second, and probably more important reason is an emerging reappropriation movement coming from designers themselves. The iconic Processing programming language and environment, launched in 2001, was among the very first tool that sought "to introduce visual designers and artists to computational design" (Reas, 2007). For designers, programming offers a whole new range of dynamic capabilities that traditional software applications do not provide yet (Reas, 2010). These pioneer initiatives nurtured a new generation of designers who started creating design software, usually for their own needs. In a 2012 essay commissioned by the magazine *Graphisme en France*, Reas and McWilliams asked several designers who program their own tools: "How does writing your own software affect your design process and also the visual qualities of the final work?" (Reas &

McWilliams, 2012, p.26) They found that some ideas were prevalent across respondents. First, designers explained that writing custom software gives them more control over the resulting artefact. The second is that new tools bring novel creative opportunities: “Experienced designers know that off-the-shelf, general software applications obscure the potential of software as a medium for expression and communication. Writing custom, unique tools with software opens new potentials for creative authorship” (Reas & McWilliams, 2012, p.27).

*Figure 2 - Some of the many trowels that can be seen at the "Maison de l’outil et de la pensée ouvrière” in Troyes, France. Note how very similar they look, yet how uniquely different each one of them is.*

At the same time, several designers started to question the lack of interest and diversity in design software through their writings. According to designer and design critic David Reinfurt: “Function sets, software paradigms, and user scenarios are mapped out for each software project to ensure the widest possible usability, resulting in an averaged tool which skips the highs, lows, errors, and quirks.” (Reinfult, 2012, p.6). In his thesis “digital tools and graphic design”, graphic designer Kevin Donnot wonders “Why couldn’t we accept that tools influence us and that we could choose them depending on their impact? Shouldn’t we ask ourselves which tool is most appropriate before mechanically resorting to our usual software?” (Donnot, 2011). This recent interest started bringing design software in the spotlight (Leray & Vilayphiou, 2011) and shows the existing mismatch between designers’ practices and their current digital applications.

With this paper, we want to start qualifying the mismatch between graphic designers’ practices and current digital graphic design tools, as well as proposing principles for novel design tools that could mitigate this mismatch. In the first part of this paper, we analyse the “technical lineage” between the early examples of digital tools and the pre-existing processes and tools they were derived from. We then more specifically analyse two design tools, the colour picker and the alignment and distribution commands to understand the vision of design they embed and its limitations in designer’s practices. Through this analysis, we reveal some of the myths about the design process that underlie these tools. In the second part of this paper, we build on a recent wave of digital design tools and introduce the notion of “graphical substrates”, interactive visual objects that bridge the gap between traditional graphical user interfaces and programming. We briefly show how recent design tools started implementing such interactive objects and we propose two design guidelines to further enhance graphical substrates and facilitate their appropriation by designers.

## *1.1* A working definition of digital design tool

Defining design tools might be an endless endeavour, because the intricate architecture of software tends to blend different levels of granularity. In this paper, we focus on the main design tools produced by the industry and still in use today, even though the earliest tools created for designers were developed by researchers. Within this scope, we define design tool as individual tools within

design applications such as Adobe Illustrator and InDesign. Concretely, we call “design tools” individual panels and commands such as colour pickers, alignment commands, levels panel, filters, etc. We otherwise use the term design software application to refer to design applications such as Adobe Illustrator, Photoshop and InDesign that include a wide variety of design tools. Even with such a definition, some design tools can be quite complex and include several interactive components such as the levels panel in Adobe Photoshop, while others, such as the rectangle selection, are much simpler. However, because they are all accessible at the same level in tool palettes and in menus, we can posit for now that they were granted the same level of importance by tool creators.

## 2 A technical lineage approach to understanding digital design tools

According to Simondon, philosopher of technology, a study of technology should not approach technology from an individual perspective. Instead, each technical object belongs to what he calls “a technical lineage” and cannot be fully understood outside of it (Simondon, 1958). As Masure showed, to establish their economic success and wide adoption, design software publishers needed to pursue an apparent continuity with existing environment and techniques (Masure, 2014). This lineage approach can be an interesting first step to understand digital graphic design tools. Taking a few examples of functionalities, we can show that at least some functionalities of the first commercial graphic design applications were derived from pre-existing practices and techniques. Contrary to the very first digital design software applications, such as Sketchpad (Sutherland, 1963), that were designed by computer scientists who had little access to how graphic designers worked, the first commercial design applications were created by people who were in close relationship with graphic design. To give two examples, Aldus’ founder, Paul Brainerd, had himself been an editor for a small journal while the wife of John Warnock, co-founder of Adobe, was a graphic designer. They therefore had a close understanding of the concrete practices and tools used by graphic designers before the digital era.

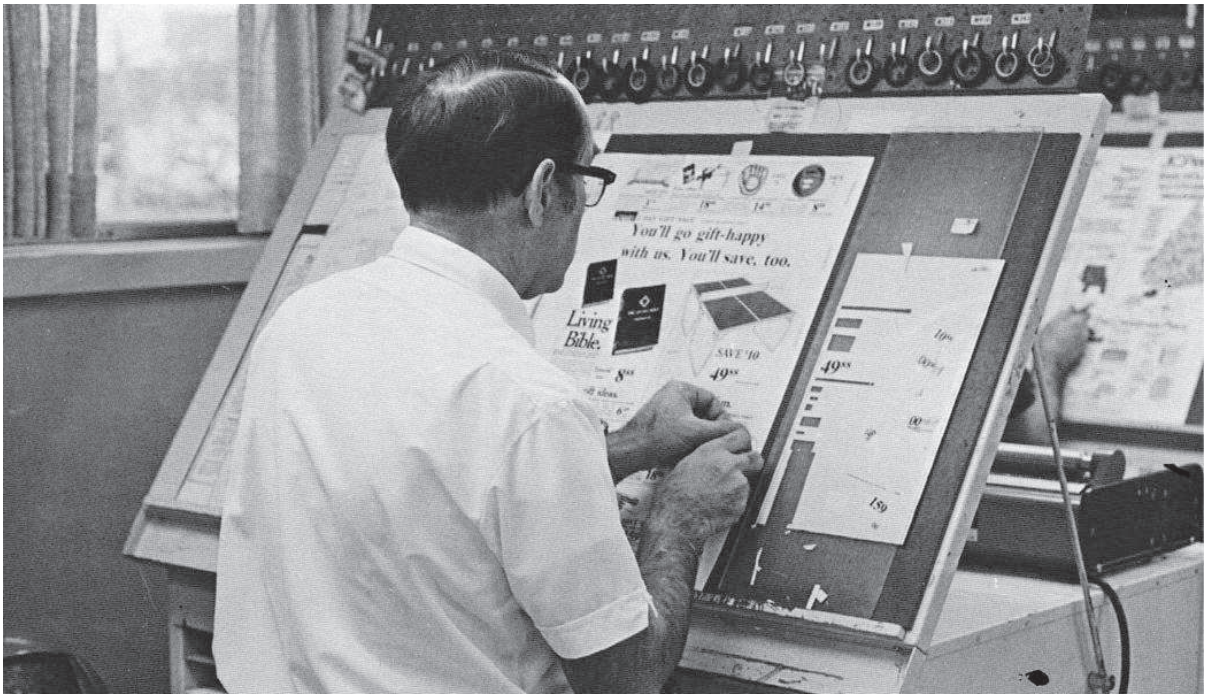


*Figure 3 - Prior to using design software applications, designers used to create layouts through “paste-up”, cutting and pasting different content elements onto a blank page. Image from Graphic Means: A History of Graphic Design Production.*

In 1985, Aldus released Page Maker, a Macintosh-dedicated software application for desktop publishing. This piece of software was specifically created to supplant traditional technologies and fit within the existing printing industry practices. As its creator explains, “*most of the page maker interface and dialogs and the way it works, the basic functions came from my experience of having*

*done paste up myself with a razor blade*" (Paul Brainerd, in Levit, 2017). Indeed, Page Maker, soon followed by its successors, Quark XPress and Adobe InDesign, introduced to graphic designers the possibility to freely drag and drop text and image onto the page, which is what they were used to do when creating layout through paste-up. Moreover, the way desktop publishing software applications handle text is also a reminiscence of the paste-up process that was prevalent in the industry at that time: First, designers would send the text to a phototypesetter to generate the whole text using the right font at the right size. They would receive single columns scrolls that they would then cut and paste onto the page. In design software applications too (and in contrast to text editors such as Microsoft Word), the text is received as one infinite scroll, it is disconnected from its containers. Designers can then compose, cut and adjust the containers onto the white page. The key difference being that the text continuously flows in the containers, and can freely be rearranged and recombined, offering greater exploration possibilities to designers.
A second example of the influence of the traditional process over desktop publishing can be found in its way of handling page format. In design software applications too, when creating a project, designers must first select pages' dimensions as well as margins. These parameters are then fixed and are not supposed to be modified. This echoes the traditional paste-up process in which the designer first chose a page format and established page margins. This page became the canvas onto which she could experiment with text and image layouts. Yet, in desktop publishing, the choice to first set page sizes and margins is not dictated by a technical constraint, rather, it simply reproduces a pre-existing process. Finally, when presenting their software, PageMakers' developers explained: "it was designed with the industry in mind, in other words it does half-tones, ligatures, kerning, all the words that the typesetting industry has been familiar with." (Paul Brainerd, in Computer Chronicles, 1986"). With these different examples, we can observe that desktop publishing first and foremost developed functionalities that matched previously existing ones in the industry. In fact, because their goal was to fit within existing workflow and to be easily adopted by designers, they tried to mimic the existing process.
The lineage approach can help us understand the design of some functionalities proposed in graphic design software applications. As we have seen, the environment behind the emergence of graphic design software applications led to the reproduction of some pre-existing constraints. However, this approach is limited when we look at individual design tools and try to understand their design. In fact, tools reproducing pre-existing processes coexisted with other functionalities that did not have direct equivalent in pre-existing processes, such as the undo command, the colour picker and the alignment and distribution commands.

## 3 Design myths behind design tools

To understand the current mismatch between designers' practices and digital graphic design tools, the work of Suchman can provide a helpful approach. According to the anthropologist, "Every human tool relies on, and materialises, some underlying conception of the activity that it is designed to support" (Suchman, 2007, p.31). By carefully observing individual digital design tools and how designers use them on a daily basis, we can analyse the perception of the design process that they embody. In this section, we focus on the two aforementioned design tools: the colour picker and the alignment and distribution command and we analyse the underlying conception of the design process that they embed in their design. We chose these two tools for several reasons. First, they don't directly mimic pre-existing mechanisms but they feature two different and pervasive mechanisms: selection and command. In his analysis of design applications such as Adobe Photoshop, Manovich, showed that selection mechanisms are pervasive in current design applications and he correlates this with the development of a remix aesthetic (Manovich, 2001). Moreover, they are among the oldest digital design tools and, above all, their design did not evolve since their first introduction in design software applications (Jalal, Maudet & Mackay, 2015) (Ciolfi, Maudet, Mackay & Beaudouin-Lafon, 2016).

### *3.1 The Design process as a Hylomorphic process*

Since its origin, the colour picker presents three common features: “a visual representation of a specified colour model, the organisation of displayable colours into a three-dimensional colour space, and controls to change parameter values within that space” (Jalal et al., 2015, p.1). Its design significantly differs from the traditional colour mixing process used by painters or the colour charts used in industry. Designers now potentially have access to every possible colour. The colour picker focuses on selecting a specific individual colour from all the possible colours. The design brief behind the tool could be summarised as: “given that a designer wants to select a specific colour, help her achieve this goal in the fewest steps possible”. This brief assumes that designers already have a clear idea of the colour they want to select and the colour picker simply displays them in a comprehensive manner to facilitate its retrieval. The second example, the alignment and distribution commands also don’t have direct equivalent in pre-digital graphic design. There was no dedicated tool but using rulers and tracing lines on paper to verify alignment. In computers, alignment and distribution can be executed through a set of twelve commands, six for alignment and 6 for distribution. Designers can align graphical element using their centre or their bounding boxes’ vertices as reference points. The designer first selects the elements and then presses the command to have them aligned following one of the six possibilities. This alignment is not permanent but is computed ad hoc by the system when the designer presses the command button. Here again, the command approach focuses on a single and specific action.

However, several recent studies showed that these two design tools have limitations. In our study of colour manipulation with designers and artists, we showed that designers have a wide variety of colour manipulation strategies that don’t follow a simple selection process. Designers rarely used the colour picker directly (Jalal et al., 2015). In contrast, designers and artists created many different strategies, using diverse tools to manipulate colours beyond colour selection. For example, we showed how designers focus on the notion of palette while the colour picker only lets designers select individual colours in the context of their surrounding colours in the colour space. Similarly, in a study with 12 users of graphical authoring applications, we showed that designers find alignment and distribution commands confusing (Ciolfi et al., 2016). Moreover, commands do not support designers’ practices: designers often resort to creating graphical elements and use them as “spacing objects”. By focusing on the immediate action, commands omit the fact that alignment and distribution take place within a much larger process of layout composition.

From these two examples, we can see that both the colour picker and the alignment and distribution commands conform with the vision that design is what anthropologist Ingold calls a “hylomorphic” process: they posit that designers already have in mind the outcome they want to achieve (Ingold, 2013). In a commercial for the ground-breaking Adobe Illustrator 88, the narrator explains that this software application is “a revolution based on new tools, tools that free the imagination and eliminate drudgery” (Illustrator 88). Behind this assumption lies the idea that tools impose restriction on an otherwise boundless creativity. This idea also implies that the act of creativity and tool use are separated phenomena. According to this idea, design tools should allow designers to reach their preconceived outcome with the least effort, without getting in the way. Both command and selection mechanisms are extremely efficient when it comes to attaining specific goals with preconceived and definite outcomes: either choosing an element within a defined set of possibilities or applying a specific action to selected elements.

This conception of design as a hylomorphic process leads to the conclusion that design tools are necessary obstacles on the way of the designer’s creativity. This vision is echoed in a 1987 Adobe Illustrator 88 commercial in which the narrator explains that traditional graphic design tools “take considerable skill to use, and even in the hands of a pro, take too much time, time that could be used to design and create” (Illustrator88). To overcome these limitations, Illustrator 88 is advertised as easy to learn and more efficient than traditional tools. As New Media professor Olia Lialina argued, the message from Adobe in their advertisement campaign is that the best kind of design requires designers to forget about their tools, so that they can focus on the core of their work: being

creative (Lialina, 2012). The logic behind this assertion is that, ideally, the creative process should be decoupled from the tools. Because digital design tools were envisioned as obstacle on the way of the design process, they were designed by putting an emphasis on their user-friendliness and efficiency. Thus, the transparency, or the "invisibilisation" to put it in other words, of design tools should in fact become the ultimate goal for tool creators.

### *3.2 Limitation of the transparent design tool myth*

The principle of transparency is not exclusive to design tools. Instead, it represents one of the core value behind the development of personal computing. As early as the Xerox Star, the first commercial Graphical User Interface system, user interfaces were designed to be as transparent to users and easy to learn as possible (Bolter & Gromala, 2003). While these values were very productive and can still be considered ideals of design in many contexts, they faced some criticism very early on. When it comes to learnability, Lucy Suchman, in her account of users' encounter with an "easy-to-use" photocopier in 1984, demonstrated that self-explanatory digital artefacts are a designers' fantasy and that despite their sophistication, interfaces will always require an "active sense-making" from the user and that it "[…] called into question the viability of marketing the machine as "self-explanatory or self-evidently easy to use" (Suchman, 2007).
Moreover, while these values might be worth pursuing in a strictly productive or in leisure-oriented software applications, they may not best support creative design work. Contrary to traditional work, designers face wicked problems (Rittel & Webber, 1973) that cannot be solved by following a prescribed series of steps that can then be optimised. About the notion of ease of use in graphic design software applications, Masure shows how new versions of Adobe Photoshop add functionalities that automate part of the design process, for example, automatically replacing objects on a photograph with a generated background in Adobe Photoshop CS5. He argues that this type of functionalities is meant to simplify the work of the designer by automating it. In doing so, Masure argues that "the semi-automatic functionalities orient the image towards a state that is socially and culturally accepted" (Masure, 2014). By focusing on the final outcome, current design tools neglect the intermediary steps in the design process and the relationship designers need to establish with their tools. This approach may for example limit exploration, one of the defining aspect of design work (Gaver, 2000).

### *3.3 The instrumental turn of design tools*

Against this ideal of transparency and efficiency, we can observe what we can call "an instrumental turn" in the perception of designers and other creative professional's relationship with their tools. In a structured observation conducted with 12 graphic designers, Jalal showed that designers preferred to use general purpose Graphical User Interface (GUI) tools rather than the more novel and specific design tools, because they felt less in control with the latter (Jalal, 2016). Early on, from a set of observations with creative professionals, Schön demonstrated how designers approach problems as unique cases and focus on the peculiarities of the situation at hand. They don't propose or look for standard solutions. Instead, Schön argues that designers perform a conversation with the material of their design and that any action will have effects beyond what they had imagined. In Schön's terms, "[the designer]'s materials are continually talking back to him, causing him to apprehend unanticipated problems and potentials" (Schön, 1983). More recently, Dalsgaard further explores the pragmatist perspective to consider tools in design as "instruments of inquiry" (Dalsgaard, 2017). He argues that tools also affect our perception and understanding of the world and help us explore and make sense of it. Furthermore, he argues that "repeated use of a computer is likely to alter the way you think about and engage in the writing process through the changes it effects on seemingly functional levels". The perception of digital design tools as instrument is also developed by Bertelsen et al. Originally proposed in the context of musical creation, they introduce the notion of *instrumentness* as a "quality of human-computer interaction" (Bertelsen et al., 2007). They propose to consider creative software as instrument in the musical sense, to be able to move away from the ideals of transparency and usability. They explain that "the software is comparable to a musical

instrument since the software becomes the object of [the composer] attention and something he explores, tweaks, observes, and challenges in a continuous shift of focus between the sounding output and the instrument". They argue that the notion of *instrumentness* can be adapted beyond music creation and be relevant to describe designers' relationship with their digital tools.

## 4 Graphical Substrates: towards a novel type of design tools

To acquire new possibilities and enhance their control over the design, graphic designers currently need to turn to programming. In our interviews with 12 graphic designers, we showed how five of them used programming to create projects that they could not have created using traditional graphic design software applications (Maudet et al., 2017). While these designers needed to spend time establishing their program, they then were able, for example, to easily produce hundreds of posters in one night, or to explore radical layout modifications in a second. The aforementioned principles, transparency, efficiency and user-friendliness, deeply integrated into current Graphical User Interface-based design applications, may partly be responsible for designers increasing interest for programming languages such as Processing or max/MSP. Programming does not focus on specific and production-oriented tasks, but rather, they offer new languages through which designers can think and work in new ways. More than producing one final artefact, programming lets designers setting up a process that can then be executed and modified.

While there is no doubt that learning to program can be extremely valuable for designers, programming cannot easily replace GUI-based design tools. A paper is not enough to thoroughly investigate the differences between programming and visual interfaces and how they impact creative work, but there are a few elements that can help us understand that we need to bridge the gap between the two approaches. First, it is still hard for designers to learn how to program (Ko, Myers & Aung, 2004) as programming may force designers to think in a different way. In the context of interaction design, we studied how designer and developer represent interaction in their own way (Maudet, Leiva, Beaudouin-Lafon & Mackay, 2017). We observed how they envision interaction from different perspectives. While visual software applications can predispose designers to focus on visual examples that describe specific moments of an interaction, programming forces developers to provide a complete and abstracted representation of the same interaction. Similarly, in a lab study, park showed that designers and developers describe differently interaction behaviours, stating that "designer's experience with tools like Photoshop and PowerPoint influences their natural expression of behaviors" (Park, Myers & Ko, 2008). Therefore, visual and textual representations provide different benefits. In his visual essay about "climbing the ladder of abstraction" (Victor, 2011), Victor shows how concrete, visual and symbolic representations might complement each other, providing different ways of seeing, interacting and understanding the same phenomenon.

Today, programming and Graphical User Interfaces are generally two mutually exclusive sets of tools. We can consider them as two opposite bounds of a large range of possible design tools. Some researchers and tool creators proposed a few models to bridge the gap. For example, departing from the strictly text-based representation of code, visual programming seeks to give a visual representation to code (Myers, 1986). Visual programming tries to simultaneously preserve the range of capabilities offered by programming while enhancing it through visual representations. On the other hand of the spectrum, graphic designers work with visual content. Current GUI-based design tools generally let designers manipulate content through direct manipulation and in the context of their final outcome. This characteristic makes them very flexible and easily appropriable by designers (Jalal et al., 2016). The power of direct manipulation (Shneiderman, 1981) originally led to the wide acceptance of digital design tools and greatly facilitated graphic designers' work. As graphic designer and critic Ellen Lupton recalls about the introduction of graphic design software applications: "being able to directly manipulate type, photography, colour, and being able to see it in real time, as you are working, that's what it's all about, that's the revolution" (Briar, 2017). In his paper about instrumental interaction, Beaudouin-Lafon proposes the notion of degree of indirectness to qualify different types of tools: a small temporal and spatial offset means that the

action is performed closely to the object (Beaudouin-Lafon, 2000). Resize handles are a good example of such tools. While GUIs can have very little indirection, textual programming is generally further away from the object it is manipulating, both temporally and spatially.

### *4.1 Graphical Substrates*

To fill in the gap between GUI-based design tools and programming, we need to invent novel types of tools. We argue that we need to preserve the qualities that GUIs can provide while enhancing them with more computational power. Grounding our proposal in the idea of *instrumentness*, we introduce the notion of graphical substrates to qualify a new wave of graphic design tools. Graphical substrates are interactive graphical objects that embed behaviours and interact with content elements. In the last part of this paper, we use the notion of graphical substrate to analyse how a new generation of prospective design tools supports designers' practices in novel ways. We provide examples of tools that embed design substrates but also identify two principles that can guide tool creators in making design substrates more effective in supporting designers' practices.

Graphical Substrates are interactive visual tools that represent relationships between graphical elements. By reifying these relationships, e.g., turning them into interactive objects (Beaudouin-Lafon & Mackay, 2000), they scaffold designers' exploration phase. The notion of substrate was first introduced by Garcia et al. who coined the term in the context of musical creation (Garcia, 2012). They proposed and designed substrates, a set of different types of musical scores that give structure and relationships to musical data. We then brought this notion in the graphic design context by observing how designers establish what they call principles, rules and constraints to guide their layout creation in digital applications (Maudet et al., 2017). They share a common characteristic: they define and guide the layout, but rarely appear in the final result.

For example, the concept of alignment can be reified into an object that embodies the alignment behaviour. Ciolfi et al. provide a recent example with StickyLines (Ciolfi et al., 2016), an interactive guideline that automatically aligns and distributes the objects that are attached to it. As a visual object, StickyLines not only provides interaction mechanisms that follow direct manipulation principles, but also embodies behaviours and rules, giving designers new possibilities for testing their ideas. In Object Oriented Drawing (Xia et al., 2015), the authors propose a graphical authoring application in which they reify attributes into cards. As they are turned into interactive objects, these properties can be moved, cloned, linked, and freely associated with several graphical elements. Another example is Histomages (Chevalier, Dragicevic & Hurter, 2012), a tool that allows users to edit images' colours by modifying a histogram of the coloured pixels within the image. A histogram is a *spatial rearrangement* of the image's pixels. The coloured pixels are grouped depending on the value of the color channel that is visualized. Therefore, it becomes very easy to select and manipulate related colors independently of where they appear on the original picture. Designers can select and change subsets of colours, such as turning the sky from shades of blue to shades of orange. Finally, Kitty, a sketch-based tool for creating animated illustrations, reifies parameters (for emission and oscillation textures for examples) into bubbles that can be linked to produce functional relationships among the graphical elements of an illustration (Kazi, 2014). These relationships can then be activated by the illustration viewers through drag gestures. For example, putting an egg into a pan provokes the emission of water drops.

Because they are interactive and persistent objects, graphical substrates can easily be modified. Design Substrates are particularly powerful when they embody rules and relationships that are automatically applied to content. This automation gives designers a much greater scale of exploration because if they decide to modify their substrates, they will be able to observe the results on all the content. For example, changing one colour card applies its result to all the graphical elements it was linked to. Analysing existing examples of graphical substrates led us to observe some of their current limitations. We propose two design principles to further develop Graphical Substrates and reinforce their adequacy with designers' practices.

### *4.2 Tweaking*

Current graphical substrates are still very binary in their application. In practice, however, designers need to take into account exceptions. In our studies of designers' practices, we found numerous examples of this need. When manipulating colours, designers often sample existing ones, but they then manually adjust the resulting colour (Jalal, 2015); when aligning and distributing graphical elements, designers usually tweak individual objects to account for mismatches between objects' perceived visual weight and reference points (Ciolfi et al., 2016); when structuring layout, designers establish structures but very often need to break their own rules to take into account extreme cases (Maudet et al., 2017). Revealing and reifying relationships or constraints into interactive objects can be a powerful mechanism for designing design tools. However, in current software, existing structuring mechanisms tend to be rigid and binary: either graphical elements fully obey the structure they belong to, or there is no structure at all. When creating design tools, tool designers should take into account the flexibility of their substrates. Enforcing rigid rules greatly undermines substrates' usability and designers might end up resorting to a more manual process even when there is an existing mechanism. For example, StickyLines integrates two different mechanisms for designers to tweak individual objects' alignment: tweaks and bounding boxes. To create a tweak, designers can reposition objects, but the object remains logically attached to the guideline. This offset, called a tweak, is recorded and displayed as a purple line. They are first-class objects that can be edited, copied onto other objects, and deleted. Designers can also modify the bounding box of an object in order to finely control its placement on a guideline. Bounding boxes can be copied onto other objects, replacing their current one.

### *4.3 Creation from example*

Allowing designers to manipulate and interact with Design Substrates can be an interesting perspective for future design tools, but to make substrates truly useful, we need to address the question of their creation. As we have seen already, with automation comes a greater risk of loosing creative freedom. In the context of weaving, Luther Hooper mentioned that "with each stage of mechanical improvement of the loom, as moreover is the case with all machine in varying degrees, the weaver's freedom and his or her control of the conception of their work is reduced" (Fetro, 2017). If all graphical substrates provide fixed and predetermined behaviours, their appropriability will be limited. When possible, structures should be reifiable from examples, i.e., design tools should let designers extract relationships and rules from existing examples. In doing so, they provide a way for designers to first explore different variations and then to apply principles to all the content. Creating the substrates thus becomes part of the design process itself. To continue on with the Styckylines example, the system lets designers create guidelines based on existing shapes by creating "a ghost", a guideline that takes the shape of an existing object. In Palette Explorer, a colour tool based on interviews with designers (Jalal et al., 2015), they can create a sample palette and can then modify this original palette as a whole on one of the colour axes (hue, saturation or contrast), retaining its original harmony by keeping the other axes fixed.

## 5 Conclusion

In this paper, we proposed an analysis of digital graphic design tools to better understand the current mismatch between designers and their tools. We first showed how design tools followed a lineage approach in their design, providing functionalities that mimic pre-existing. We also analysed in more detail two specific design tools: the colour picker and the alignment and distribution commands and we revealed how their design supports a vision of design as a hylomorphic process. This conception of design led to designing design tools with values such as transparency and efficiency. However, design research shows how the "instrumentness" of design tools more appropriately supports current designers' practices. To resolve this mismatch, designers currently turn to programming but we argue that we can combine both the strength of graphical user interface and programming. We call this novel type of design tools graphical substrates and illustrate

it with several examples in recent design tool research. We argue that integrating mechanisms for creating graphical substrates from examples and tweaking them would further extend their appropriability by designers. Beyond design tools, this paper questions the underlying emphasis on invisibility, efficiency and user-friendliness in tool-design.

**Acknowledgments:** I would like to thank the reviewers who provided valuable feedback on the first version of this paper. Defining the broader notion of substrates is a collective effort in the ex)situ team. I would like to thank my colleagues, Ghita Jalal, Marianela Ciolfi Felice, Germán Leiva and Philip Tchernavskij, with whom I collaborated on the projects described in this paper. I would also like to thank my PhD advisors, Michel Beaudouin-Lafon and Wendy Mackay who profoundly influenced this work.

## 6 References

Adobe. (1988). Adobe Illustrator 88 (Promotional Instruction Video).

Armstrong, H. (2016). *Digital Design Theory. Readings from the Field*. Princeton Architectural Press.

Beaudouin-Lafon, M. (2000). Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 446-453). ACM.

Beaudouin-Lafon, M., & Mackay, W. E. (2000). Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. Proceedings of the Working Conference on Advanced Visual Interfaces, 102–109. https://doi.org/10.1145/345513.345267

Bertelsen, O. W., Breinbjerg, M., & Pold, S. (2007). Instrumentness for Creativity Mediation, Materiality & Metonymy. In Proceedings of the 6th ACM SIGCHI Conference on Creativity &Amp; Cognition (pp. 233–242). New York, NY, USA: ACM. https://doi.org/10.1145/1254960.1254992

Bolter, J. D., & Gromala, D. (2003). Windows and mirrors: Interaction design, digital art, and the myth of transparency. MIT press.

Briar, L. (2017). *Graphic Means, a History of Graphic Design Production*.

Chevalier, F., Dragicevic, P., & Hurter, C. (2012). Histomages: fully synchronized views for image editing. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (pp. 281-286). ACM.

Ciolfi Felice, M., Maudet, N., Mackay, W. E., & Beaudouin-Lafon, M. (2016). Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (pp. 133–144).

Dalsgaard, P. (2017). Instruments of inquiry: Understanding the nature and role of tools in design. International Journal of Design, 11(1).

Donnot, K. (2011). Outil Numérique et Design Graphique. École des Beaux-Arts de Rennes.

Fetro, S. (2017). Working with Digital Machines. In Back-Office (pp. 86–97). B42 and Fork.

Garcia, J., Tsandilas, T., Agon, C., & Mackay, W. (2012). Interactive Paper Substrates to Support Musical Creation. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1825–1828). New York, NY, USA: ACM. https://doi.org/10.1145/2207676.2208316

Gaver, B., & Martin, H. (2000). Alternatives: exploring information appliances through conceptual design proposals. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 209–216).

Ingold, T. (2013). *Making: Anthropology, Archaeology, Art and Architecture*. Routledge.

Jalal, G., Maudet, N., & Mackay, W. E. (2015). Color Portraits: From Color Picking to Interacting with Color. In Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems (Vol. 1, pp. 4207–4216). Retrieved from http://dx.doi.org/10.1145/2702123.2702173

Kazi, R. H., Chevalier, F., Grossman, T., & Fitzmaurice, G. (2014). Kitty: Sketching Dynamic and Interactive Illustrations. In Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14 (pp. 395–405). New York, New York, USA: ACM Press. https://doi.org/10.1145/2642918.2647375

Ko, A. J., Myers, B. A., & Aung, H. H. (2004). Six learning barriers in end-user programming systems. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on* (pp. 199-206). IEEE.

Leray, A., & Vilayphiou, S. (2011). Considering your tools. Libre Graphics Research Unit. Retrieved from http://reader.lgru.net/pages/about/

Levit, B. (2017). Graphic Means: A History of Graphic Design.

Lialina, O. (2012). Turing Complete User. Considering Your Tools. Retrieved from http://contemporary-home-computing.org/turing-complete-user/
Llach, D. C. (2015). Builders of the Vision: Software and the Imagination of Design. Routledge.
Manovich, L. (2001). The language of new media. Cambridge Mass.: MIT Press.
Masure, A. (2014). Le design des programmes, des façons de faire du numérique. Paris 1, Sorbonne.
Maudet, N., Jalal, G., Tchernavskij, P., Beaudouin-Lafon, M., & Mackay, W. E. (2017). Beyond Grids: Interactive Graphical Substrates to Structure Digital Layout. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 5053–5064). New York, NY, USA: ACM. https://doi.org/10.1145/3025453.3025718
Maudet, N., Leiva, G., Beaudouin-Lafon, M., & Mackay, W. E. (2017, February). Design Breakdowns: Designer-Developer Gaps in Representing and Interpreting Interactive Systems. In CSCW 2017-ACM Conference on Computer Supported Cooperative Work and Social Computing (pp. 630-641).
McGrenere, J. (2000). "Bloat": The Objective and Subject Dimensions. In CHI '00 Extended Abstracts on Human Factors in Computing Systems (pp. 337–338). New York, NY, USA: ACM. https://doi.org/10.1145/633292.633495
Moholy-Nagy, L. (1973). Painting Photography Film. MIT Press.
Morris, W. (1884). Art and Socialism. In Political Writings of William Morris. A. L. Morton.
Myers, B. A. (1986). Visual programming, programming by example, and program visualization: a taxonomy. In ACM SIGCHI Bulletin (Vol. 17, pp. 59–66).
Papanek, V. J. (1972). Design for the real world: human ecology and social change. Thames and Hudson London.
Park, S. Y., Myers, B., & Ko, A. J. (2008). Designers' natural descriptions of interactive behaviors. In *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on* (pp. 185-188). IEEE.
Reas, C., Fry, B., & Maeda, J. (2007). Processing: A Programming Handbook for Visual Designers and Artists. The MIT Press.
Reas, C., & McWilliams, C. (2012). Progammer avec Erik van Blokland, Catalogtree, Amanda Cox, Nicholas Felton, FIELD, LUST, Boris Müller, onformative, Jonathan Puckey, Sosolimited & Trafik. Graphisme En France, Code<>outils<>design.
Reas, C., & McWilliams, C. (2010). Form+code in design, art, and architecture (1st ed.). Princeton Architectural Press New York.
Reinfurt, D. (2007). Making do and getting by. In : Kyes, Zak ; Owens, Mark. Forms of Inquiry : The Architecture of Critical Graphic Design.
Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a general theory of planning. Policy Sciences, 4(2), 155–169.
Schön, D. A. (1984). The reflective practitioner: How professionals think in action (Vol. 5126). Basic books.
Shneiderman, B. (1981). Direct Manipulation: A Step Beyond Programming Languages. In Proceedings of the Joint Conference on Easier and More Productive Use of Computer Systems. (Part - II): Human Interface and the User Interface - Volume 1981 (p. 143--). New York, NY, USA: ACM. https://doi.org/10.1145/800276.810991
Simondon, G. (1958). Du mode d'existence des objets techniques. Méot.
Suchman, L. A. (2007). Human-Machine Reconfiguration, Plans and Situated Action, 2nd Edition. Cambridge University Press.
Sutherland, I. (1963). *SKETCHPAD-a man-machine graphical interface* (Doctoral dissertation, PhD thesis, MIT)
Victor, B. (2011). Up and Down the Ladder of Abstraction. Self-Published. Retrieved from http://worrydream.com/LadderOfAbstraction/
Xia, H., Araujo, B., Grossman, T., & Wigdor, D. (2016). Object-oriented drawing. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 4610–4621).

About the Author:

**Nolwenn Maudet** is an interaction designer and design researcher working as a post-doc at the University of Tokyo. Her key research interests are in studying designers' practices and how we can design digital tools that better support them