The Web Browser as a Tool
A Programmatic Approach to Graphic Design on the Web

Francis Benoit

A Thesis submitted to the Faculty of Graduate Studies
in partial fulfillment of the requirements for the Degree of Master of Design

Graduate Program in Design
York University
Toronto, Ontario
May 2017

ABSTRACT

In recent years, the web browser's rendering capabilities have grown considerably. However, it remains a window through which design is seen rather than being used as a tool. This thesis seeks to develop a programmatic method that questions the web browser's original role as a display and redefines it by investigating its alternative role as a tool in the graphic design process. Through exploratory work, this research demonstrates that the web browser can be a fertile space for visual experimentation. This thesis demonstrates that graphic designers can benefit from a more pragmatic and logical approach to creation and invites them to adopt a process similar to a programmer's process using the web browser as a tool.

ACKNOWLEDGEMENTS

I would like to thank my supervisor David Gelb and my advisor Angela Norwood for their support and insight throughout my thesis work. I truly couldn't have asked for a better team. Your insight and encouragements were keys to my success; sometimes you probably had better words to explain what I wanted to do than myself. I thank you greatly to believe in my potential and to push me out of my comfort zone. You made our meetings always pertinent and fun.

This whole adventure could not have been possible without the unshakable and unconditional love and support of my family, especially my parents Hélène Tremblay and Pierre Benoit, and my cousin Eric Tremblay. Perhaps you still don't fully understand what I have been doing for the last two years; hopefully, this paper should help. Merci pour tout.

I wish to thanks my old friends in Québec that kept me grounded and encouraged me, and new ones that I've made in Toronto. You all made this adventure easier on a daily basis and inspired me to continue onwards.

I was lucky enough to share the studio with great minds and would like to express my gratitude to Julia Grzeskowiak, Justine Rudnicki, and Nadine Arseneault. We shared critiques, discussions, but foremost great moments together and I am sure that our paths will cross again.

Finally, I couldn't thank enough Philippe Jean, who revealed to be one of the greatest friends I ever had. You convinced me to move to Toronto and to take on the challenge of the MDes. You inspired me, guided me and made me discover new horizons. I also wouldn't have achieved this thesis without the help of one of my oldest friend Philippe Caron, who taught me so much since we know each other, and even more during the last two years. My friends, I am now in your debt.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Personal Background

The 1990s were a decade of fast technological change and progress, which saw the inauguration of the Internet. The graphic design industry, too, saw rapid technological changes, and the Internet offered designers contradictions, endless possibilities. Daily, the Internet exposes designers to never-ending, tremendously fast advancements happening everywhere in the world, and this whirlwind of opportunities can feel like a constant bombardment.

This was certainly my feeling during my graphic design studies. Before considering graduate studies in design, I studied graphic design and fine arts, and I had little interest in web design. However, I found it hard to apply the creative process that I learned at school in my professional practice; I felt that the way I learned web design was not helping me to become a better graphic designer. These circumstances led me to graduate studies in graphic design, where I hoped to explore other types of creative processes that felt more relevant to design in the Internet age. While I was reluctant to choose the web browser as my thesis topic, challenging my creative process by integrating the web browser into a method made web design more appealing to me. The primary motivation of this thesis was thus to broaden my horizons and get out of my comfort zone.

This project was grounded in the designer Karl Gerstner's principles of programmes, and inspired by the work of designer and educator John Caserta who explored visual forms using the web browser as a space for graphic experimentation and production. This investigation allowed me to understand a tool that I have never experimented with and to immerse myself in a context of rapid changes and contradictions. It filled a gap in my practice and helped me position myself within a constellation of other graphic design practices.

Background of the Web Browser

Since the introduction of the Mosaic browser in 1993, the web browser has been the primary means by which users see and interact with content on the Internet. The web browser was created to broaden access to the Internet while by managing complex protocols in the background only those with the technical knowledge could understand. The development of its interface was profoundly influenced by the *desktop metaphor* (Baecker, 1995), which helped people understand the digital environment by analogizing it to an office workspace (using metaphors like folders, files, and pages), offering them an intuitive experience of the web. Like other computer software, the web browser depends on a graphical user interface (GUI) for user interaction. The GUI is an integral part of the web browser, providing functionality and context.

As the web browser's structure evolved, it increasingly enabled users to create content, leading to the development in the early 2000s of the Web 2.0 (O'Reilly, 2005). Web 2.0, whose content is generated by users through their browsers, is, as Tim O'Reilly explains, the result of "cumulative changes in the way web pages are used" (2005). Some of the most popular Web 2.0 platforms are *Wikipedia*, *Reddit*, and *Craigslist*. The web browser evolved to allow users to develop web content, changing from a static screen that simply displayed information to a "transport mechanism through which interactivity happens" (DiNucci, 1999). Web browsers are thus tools that empower users to interact with content and with each other.

Today's web designers often situate their work between the structure and the content of the web browser. For the purposes of this study, the browser's *structure* is defined as its information hierarchy: how it organizes information (visually accessible or not) in web pages and in programming scripts. The *content* is defined as the information displayed by the web browser and the visual elements present on the web page. Some designers experiment with this liminal space: they are not making visuals *for* the browser, but *within* the browser, which becomes a tool for visual experimentations. In fact, their experiments have become "the object of design, rather than […] the source of design" (van der Beek, 2012). These design experiments are the product of the *interaction* between the structure and the content, as designers can now access, reclaim, and reshape both elements to make graphic design. Despite the growing enthusiasm over these new experiments, no methodology specific to the creative process in the web browser has been laid out.

The development of the web browser has been marked by a tension between developers and graphic designers. Developers focus on data transparency and on making information clear and structured, while designers argue that information needs layout and visual consistency to be understood (Bolter, 2005) and that the web browser can be an interesting space for visual experimentations. Thus, the web browser and the Internet can be considered either as a "representational medium or more about having an experience" (Moss, 2008). However, the developers' vision does not exclude the designers' vision. Content must be clear and easy to understand, but it can be presented in a visually compelling manner. The web browser has detached itself from other software, developing into a unique tool and space for graphic design (Bolter, 2005). The evolution of the web browser has encouraged designers to further explore its capabilities by designing new functionality and tools as well as investigating new ways of creating directly within it.

Altering a new media object into a tool for creation is not a new idea. Graphic designers can look at the practice of artist Nam June Paik, who experimented with new media formats, calling their original purposes into question. He used televisions, traditionally designed as broadcast platforms, as interactive canvases in many of his sculptures (see Figure 1).

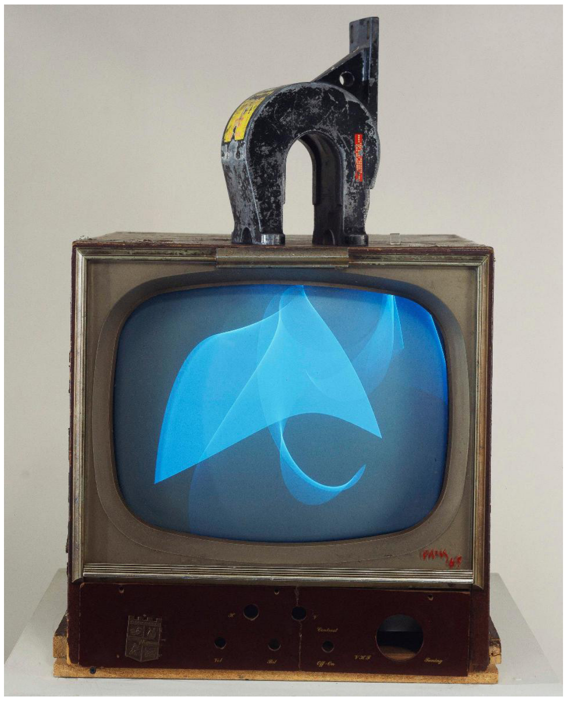


Figure 1: Nam June Paik, Magnet TV, 1965.

The television became an active participant in Paik's creative process rather than a passive object that simply projected his creative output. Like Paik's work, experiments with the web browser explore space and time through a medium that is made to entertain a unilateral relationship with the viewer or user (Hanhardt, 1982). Nam June Paik's process can inspire graphic designers to use the web browser in novel ways because it seeks to "break down the barrier between the viewer and the artwork" (Hanhardt, 1982); the web browser can similarly break down barriers between designers and design artefacts on the web, allowing users to interact with the design objects. The web browser, like Nam June Paik's televisions, can transcend its original role—simple display—through explorations that re-envision it as a tool for graphic design.

Background of the Programmatic Approach

The 1960s was a time of rapid change and tool development for graphic design. Although computers had not yet changed the way designers worked, Karl Gerstner, a typographer and graphic designer, approached his design process with a strong, computer-program-like logic (Gredinger, 2007). To assess his creative process, Gerstner pioneered a purely logical and pragmatic method called the *programmatic approach*. This design approach, which resembled machine and computer information processing, left no room for intuition or subjective decisions; Gerstner defined all parameters and variables in play before executing his designs. This logical process could take various forms, from tables to a set of written rules to shape outcomes. Each parameter represented a specific element of his design, and each parameter's respective value guided the production of the work. For example, Gerstner was able to draw the different weights of the Berthold font by following the programme that defined (1) the width of the letter, and (2) the stroke of the letter (see Figure 2). He could sketch any weight or variation depending on the values applied. Much of his known work was made based on tables that described the rules of creation (see Figure 3). All of his decisions were driven by a predetermined process and led to a significant number of variations of the same object. Gerstner was also able to translate his programmatic process to other forms of creation such as literature, photography, and architecture.

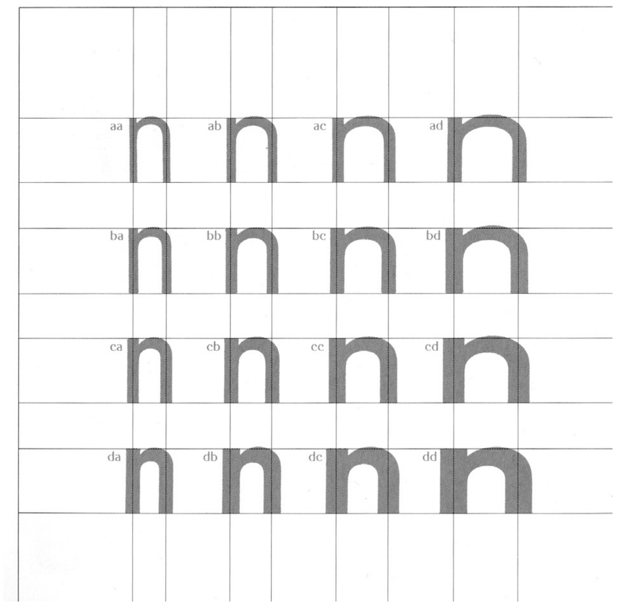
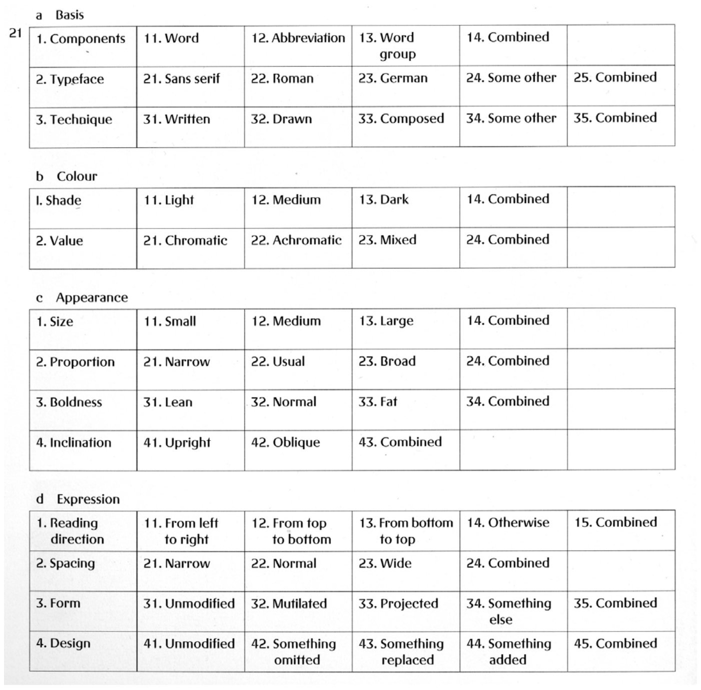


Figure 2: Karl Gerstner, Berthold font family, 1964.



21

a Basis

1. Components	11. Word	12. Abbreviation	13. Word group	14. Combined	
2. Typeface	21. Sans serif	22. Roman	23. German	24. Some other	25. Combined
3. Technique	31. Written	32. Drawn	33. Composed	34. Some other	35. Combined

b Colour

l. Shade	11. Light	12. Medium	13. Dark	14. Combined	
2. Value	21. Chromatic	22. Achromatic	23. Mixed	24. Combined	

c Appearance

1. Size	11. Small	12. Medium	13. Large	14. Combined	
2. Proportion	21. Narrow	22. Usual	23. Broad	24. Combined	
3. Boldness	31. Lean	32. Normal	33. Fat	34. Combined	
4. Inclination	41. Upright	42. Oblique	43. Combined		

d Expression

1. Reading direction	11. From left to right	12. From top to bottom	13. From bottom to top	14. Otherwise	15. Combined
2. Spacing	21. Narrow	22. Normal	23. Wide	24. Combined	
3. Form	31. Unmodified	32. Mutilated	33. Projected	34. Something else	35. Combined
4. Design	41. Unmodified	42. Something omitted	43. Something replaced	44. Something added	45. Combined

Figure 3: Karl Gerstner set up programmes similar to this one to guide his process.

As computers evolved and became part of graphic design processes, design tools such as code editors and prototyping applications adopted a mode of processing known as “if-then,” or conditional, logic: if this happens, then do that. Others have used this computational logic to theorize design processes similar to Gerstner’s. For example, Andrew Blauvelt coined the term “if-then approach” (2011) to define this particular design approach, and he argues that this process, mostly used by programmers, can also be beneficial to graphic designers. His conditional approach to creation leaves little room for subjective decisions and intuition because it is based on logical but open-ended paths.

Blauvelt’s “if-then” approach is quite different from most designers’ processes, which are driven by concepts and problem-solving; this more typical design thinking is exemplified by Nigel Cross’s design process. Cross created the analysis, synthesis and evaluation process, which is still one of the most commonly used design processes (1984). The notable differences between Blauvelt’s “if-then” approach and the Cross-type concept-driven graphic design process may explain the tension between developers and designers during the development of the web browser. While programmatic and concept-driven processes offer different visions and often complete themselves in the creative process (Bolter, 2005), each approach has something to offer the other.

The “if-then” approach has been adopted in other fields, including music. Composer John Cage’s experimental music pieces, which are driven by randomness and chance operations, were created using a programmatic approach like that of Gerstner. Cage used logical operations to compose his music pieces instead of “operating according to [his] likes and dislikes” (Kostelanetz, 2003). Cage’s programmatic, open-ended rules allowed him to embrace randomness within a given framework. A similar programmatic approach to exploring visual forms with the web browser might require designers to give up control of outcomes to bypass the logic of the browser in search of randomness (Reas, 2016). The web browser processes data in a particular way as it generates web pages and renders visuals, but designing a predefined process prevents designers from creating visuals based on their preferences or intuitions and may unveil unexpected outcomes.

Karl Gerstner's work thus offers a model for a programmatic approach to contemporary graphic design practice using recent technologies such as the web browser. Gerstner's methods model ways to work with dynamic visual forms in the context of the web browser as a space for creation. In this thesis, I examine the web browser as a tool for an exploratory approach to design driven by a programmatic method. My thesis seeks *to break from the expected design outcomes on the web by using a programmatic method that considers the web browser as a tool in the creative process.*

RESEARCH STATEMENT

In the light of the technological advancements, this thesis will question the web browser's original role of display and it will investigate the role the browser can play as a tool in the graphic design process. This research aims to develop a new programmatic method to use the browser as a tool for exploratory work. As such, it will attempt to describe how a new programmatic method that considers the web browser as a tool in the creative process can be used to break from the expected design outcome on the web. This thesis examines how a programmatic method can be used to take on visual exploration in the context of the web browser, an environment that is fully responsive to the graphic designer's decisions and interactions.

The exploratory practice used in this study adopts a distinct perspective on creation, focusing not on output but on method. Three objectives have been defined to evaluate the exploratory work; these may not all be entirely fulfilled in all projects, but each is achieved to different degrees in every project. First, the project needs *to question the form and content of the web through visual exploration*. The project must be able to challenge the structure, the legibility, and the integrity of form and content of the web to attain this objective. Second, the project must *develop an alternate method that will generate an æsthetic, unique to the web browser, that breaks from design practices on the web*. This objective is not about studying the practices on the web; instead, it proposes a design method that generates a strong æsthetic. This method should emphasize a process that orients design decisions toward open-ended outcomes for the web browser. Third, the process should *increase the number of variations on a design object through a programmatic method*. This final objective strives to take advantage of the generative nature of the web browser. Generating more variations of the same object allows designers to better understand their project, revealing underlying patterns that can drive visual exploration toward unforeseen paths.

These three objectives cannot be achieved without a consistent shift in how designers think about the process of graphic design. The process coined by Cross (1984) is linear: the designer first analyses the projects, then produces a concept-driven design, then finishes by evaluating the final result. The logical, generative nature of the web browser offers an opportunity to break from Cross's approach and look at the design process from a different perspective.

Dorothy Sayers' process is more suitable for programmatic methods using the web browser. Sayers divides the design process into three parts: the idea, the implementation, and the interaction (1970). The idea is the conceptualization of the project—a mental vision of the finished project, complete with objectives, a plan to proceed, and hypotheses about its impacts. The second phase, the implementation of the idea, encompasses the production and execution of the process. The designer puts the idea into action and iterates it, exploring the possibilities within the framework of the method. Translated to a programmatic approach, this process will be in constant iteration to adjust the outcomes to the objectives of the project. It is at this stage that Sayers' process diverges from the conventional graphic design process elaborated by Cross. Sayers' last phase, the interaction, is defined by the creation of variations, which are the product of the interaction between the object developed in the implementation phase and the designer. In this phase, the designer navigates and interacts with different elements of the web pages using the web browser, programmatically generating outcomes. The exploratory practice of this thesis observes the results of the interaction between the designer and the programme. The process is stable, but the outcomes vary depending on the interaction in the context of the web browser.

This research seeks to provide an alternate method of assessing graphic design projects while shedding light on the web browser as an emerging tool. Although programmatic methods are nothing new to the field, the method investigated in this study can introduce an alternative graphic design process to that described by Cross. This process, which embraces the specific context of the web browser as a programmatic and generative tool for graphic design, breaks from Cross's linear process to focus on the interaction between the process and the designer and finds its force through iteration.

EMERGENCE OF PROGRAMMATIC PRACTICES

Conceptual Art

Programmatic methods first emerged in the arts at around the same time that computers were developed to do calculations and complex tasks for scientific purposes. One of the pioneers in using programmatic methods to create visuals was the artist Sol Lewitt, who used prompts and instruction to guide the production of artworks. As a conceptual artist, Lewitt believed that "the initial idea is paramount and that it must be fully understood by the artist before a work is carried out" (Legg, 1978). In other words, the artist's idea and intent are more important than the outcomes. Anyone could execute the process; the artwork would express the same idea in different variations (see Figure 4). Lewitt offers an early conceptualization of the *programme*, defining it as a set of directives that produce various outcomes sharing the same idea, goal, and intention. Lewitt's particular method was to join written prompts with visual execution, thus "establishing a dialogue between two different kinds of symbols" (Legg, 1978). Lewitt designed procedures to make works of art. Just like Lewitt's instructions, computer programming languages act as procedural guidelines that translate the ideas of the writer (programmer), enabling them to be executed by the computer or by a third party that is part of the computer (such as the web browser). Writing a programme to create visual exploration in the web browser thus borrows from Lewitt's instructions for making artworks.

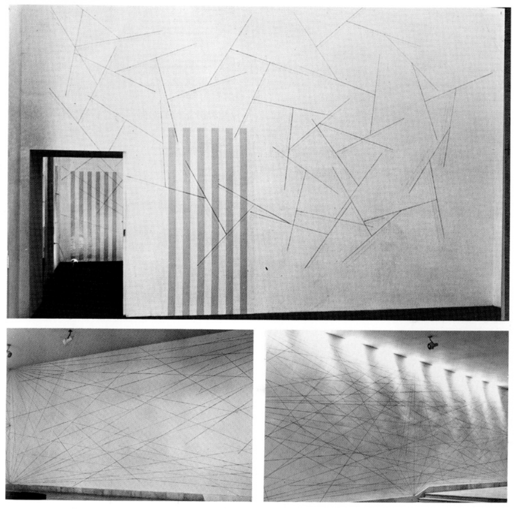


Figure 4: The work done at the Lisson Gallery in London centred on the idea of using the terms TO, TOWARD, and THROUGH with reference points and lines of the architectural setting (Legg, 1978).

More recent artists, including Gerhard Richter, have integrated computers into processes similar to Lewitt's. Richter used a method based on chance operations to realize the series *4900 Colours* (see Figure 5).

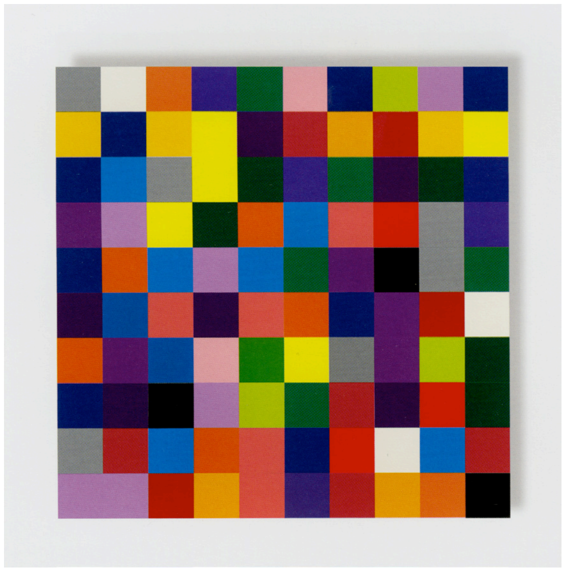
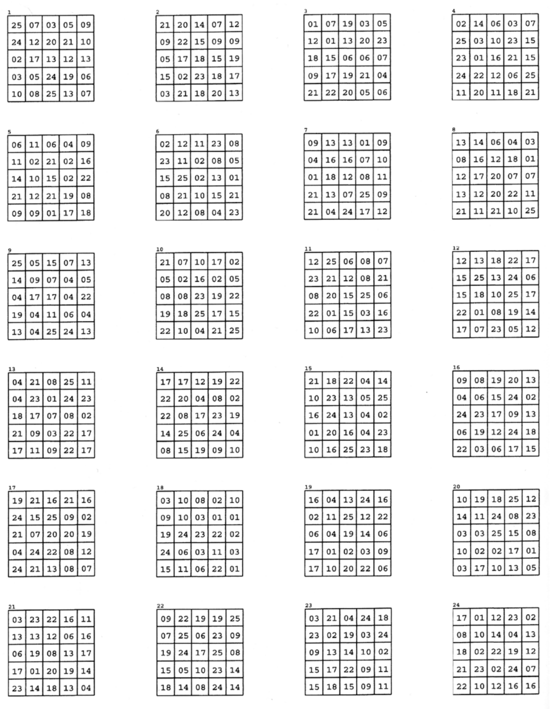


Figure 5: One of Richter's 49 paintings. Each painting consists of four panels, and each panel is made up of 25 coloured squares that can be reorganized in 11 variations.

Richter's work has been described as an "analogue of operational processes" rather than a visual experience (Buchloh, 2008). The series *4900 Colours* focuses on the representation of his process. Like Sol Lewitt's work, it emphasizes method over product. The compositions in Richter's series were determined by a chance operation: he rolled dice to decide the location of the coloured panels.[1] By using algorithms to generate art (see Figure 6), Richter raised many questions regarding the mechanisation of art, since "computers execute instructions without conscience or discernment, without intuition of will" (Pelzer, 2008). Richter's approach thus challenges the more intuitive and subjective processes used by graphic designers.

1 Rolling a dice is an algorithmic operation because it involves calculations and manipulation with the result of the dice.



1

25	07	03	05	09
24	12	20	21	10
02	17	13	12	13
03	05	24	19	06
10	08	25	13	07

2

21	20	14	07	12
09	22	15	09	09
05	17	18	15	19
15	02	23	18	17
03	21	18	20	13

3

01	07	19	03	05
12	01	13	20	23
18	15	06	06	07
09	17	19	21	04
21	22	20	05	06

4

02	14	06	03	07
25	03	10	23	15
23	01	16	21	15
24	22	12	06	25
11	20	11	18	21

5

06	11	06	04	09
11	02	21	02	16
14	10	15	02	22
21	12	21	19	08
09	09	01	17	18

6

02	12	11	23	08
23	11	02	08	05
15	25	02	13	01
08	21	10	15	21
20	12	08	04	23

7

09	13	13	01	09
04	16	16	07	10
01	18	12	08	11
21	13	07	25	09
21	04	24	17	12

8

13	14	06	04	03
08	16	12	18	01
12	17	20	07	07
13	12	20	22	11
21	11	21	10	25

9

25	05	15	07	13
14	09	07	04	05
04	17	17	04	22
19	04	11	06	04
13	04	25	24	13

10

21	07	10	17	02
05	02	16	02	05
08	08	23	19	22
19	18	25	17	15
22	10	04	21	25

11

12	25	06	08	07
23	21	12	08	21
08	20	15	25	06
22	01	15	03	16
10	06	17	13	23

12

12	13	18	22	17
15	25	13	24	06
15	18	10	25	17
22	01	08	19	14
17	07	23	05	12

13

04	21	08	25	11
04	23	01	24	23
18	17	07	08	02
21	09	03	22	17
17	11	09	22	17

14

17	17	12	19	22
22	20	04	08	02
22	08	17	23	19
14	25	06	24	04
08	15	19	09	10

15

21	18	22	04	14
10	23	13	05	25
16	24	13	04	02
01	20	16	04	23
10	16	25	23	18

16

09	08	19	20	13
04	06	15	24	02
24	23	17	09	13
06	19	12	24	18
22	03	06	17	15

17

19	21	16	21	16
24	15	25	09	02
21	07	20	20	19
04	24	22	08	12
24	21	13	08	07

18

03	10	08	02	10
09	10	03	01	01
19	24	23	22	02
24	06	03	11	03
15	11	06	22	01

19

16	04	13	24	16
02	11	25	12	22
06	04	19	14	06
17	01	02	03	09
17	10	20	22	06

20

10	19	18	25	12
14	11	24	08	23
03	03	25	15	08
10	02	02	17	01
03	17	10	13	05

21

03	23	22	16	11
13	13	12	06	16
06	19	08	13	17
17	01	20	19	14
23	14	18	13	04

22

09	22	19	19	25
07	25	06	23	09
19	24	17	25	08
15	05	10	23	14
18	14	08	24	14

23

03	21	04	24	18
23	02	19	03	24
09	13	14	10	02
15	17	22	09	11
15	18	15	09	11

24

17	01	12	23	02
08	10	14	04	13
18	02	22	19	12
21	23	02	24	07
22	10	12	16	16

Figure 6: The computer generates random numbers for the 25 different colours for *4900 Colours.*

Does Richter's artwork demonstrate that everything can be automated, be reduced to pure logic and calculations? The critical question is not about automation as a technique, but about whether the work generated by algorithms is successful. Computers cannot make this judgment; only humans can. Using computers as part of the artistic or design process does not imply that art can be created purely mechanically. Computers are only a tool. They are integral to Richter's process, as picking by computer "goes faster than picking places randomly" by hand (Pelzer, 2008); Richter's algorithms act as objective guides to the artist's hand. Similarly, a programme in the context of visual explorations within the web browser would act as an objective guide to the designer's hand; the programme's parameters, like algorithms, provide a value with which visuals can be created.

Conceptual artists have introduced programmatic methods and logical ways to create visuals, paving the way for graphic designers to use similar methods. They have used algorithms, rules and chance operations to create artworks that can be generated more quickly and more variously while challenging the importance placed on the artist's subjectivity and intuition. These methods become increasingly relevant as computers and software improve their rendering capabilities. In this study, the work of Sol Lewitt and

Gerhard Richter provides principles of programmatic methods that can be used in graphic design. Lewitt intended that "the plan would design the work" (Lewitt, 1967), and he showed that it is possible to create compelling multiple visuals using conceptually driven instruction, as long as each prompt reflects the concept and the intentions of the artist. Put differently, Lewitt used algorithms to carry out his intentions; in contrast, Gerhard Richter used algorithms to actually design and create his artworks. Each artist's practice raises questions about the nature of the creative process, its product, and the role of the artist—all questions that are equally relevant to the practice of graphic design.

Conditional Design

One of the most notable groups practicing design using programmatic processes is the Conditional Design group formed by designers Luna Maurer, Edo Paulus, Jonathan Puckey, and Roel Wouters. In their manifesto, they justify their logical approach to creation as a way to adapt the creative process to a world "characterized by speed and constant change" (Maurer et al., 2013). They believe that the creative process must reflect the complexity of a world driven by data, showing both its advantages and its limitations. Conditional Design reflects a method rather than a chosen medium of practice.

Like the work of Lewitt and Richter, Conditional Design emphasises the process and its elaboration rather than the final result. The principal characteristics of this process are *time*, *relationship*, and *change*. There are three principles that guide this approach: (1) the process is more important than the product; (2) the method is guided by logic rather than intuition; and (3) the process must embrace the context. These principles are translated into instructions and directives that pragmatically and logically drive outcomes. The instructions are made to be interpreted, and the personal perspective of each individual thus shapes their results. To demonstrate the process of Conditional Design, the group developed a series of workshops; the outcomes of the workshops vary by person and are influenced by other participants in the workshop (see Figure 7).

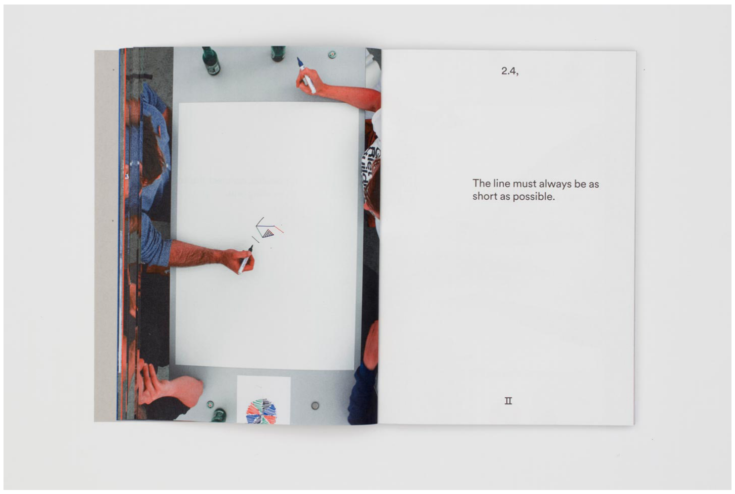


Figure 7: Participants follow the directives, but they are also influenced by the actions of other participants and by future directions.

Conditional Design offers an excellent introduction to programmatic methods for the graphic design field; it also draws interesting parallels between Karl Gerstner's programmatic method and programming languages for the web, as both of them are structures that convey concepts. Conditional Design is a logical process that handles concepts only, unlike Richter's algorithms, which generated numbers and values for the artist to manipulate.

As these programmatic approaches demonstrate, a designer's concepts can be translated into the parameters of a programme. In the context of the web browser, concepts must be broken down and translated into parameters that are compatible with the browser, which does its own form of translation using programming languages and visual rendering. Therefore, the parameters of the programme must interact with the web browser's visual elements to generate visuals.

Programming Languages as Generative Design

The programming languages used in the web are vehicles for instructions, prompts, and commands. There are many programming languages used for different purposes: for example, some lay out structure (HTML), others dictate style and visual elements (CSS, Scriptographer), and still others are made to build systems (Ruby, Python). These languages all act as written expressions of the algorithms that guide the web browser's outputs (Reas, 2010). But unlike the written instructions of Conditional

Design workshops or of Sol Lewitt, programming languages have only one possible interpretation, and they must be written without errors for the computer to execute the instructions as intended.

The web browser processes programming languages in a responsive and dynamic way. Every time a web page is loaded[2], the instructions are read and translated by the web browser. The web page that the user sees is only a single permutation of the instructions that the programming languages gave to the web browser. The variability of the programme allows the web browser to generate multiple configurations, creating new variations of the same object. The web browser thus executes the web page's instructions in an *iterative* fashion—it repeats its execution of the instructions multiple times, potentially producing multiple different outcomes. Put another way, the web browser privileges generative methods, and the refinement of the programme itself to obtain a more sophisticated result out of the web browser. Programmes act similarly to *generative design*, which is defined as a process that seeks "to produce an art of seriality, which would allow for permutation and variability within a given rule set" (Blauvelt, 2013). The web browser's rule set comes from the programming languages that generate web pages and visual forms. Like the artists' interpretations of the written rules in Conditional Design, the outcomes of the interaction between the browser and the programme are the product of permutation and variability, as defined in a set of decisions by the programmer.

Perhaps the programming language that best demonstrates the generative nature of the web browser is *P5.js*, a Javascript library[3] used for visual and creative sketches. From a design standpoint, *P5.js* is an interesting introduction to coding and to programmatic processes that "relate software to principles of visual forms, motion and interaction" of graphic design (Reas, 2014). The artist Rafaël Rozendaal uses *P5.js* to demonstrate the wide possibilities of visual exploration with the browser. His practice, which focuses on making infinite copies of his ideas, explores the concept of eternity (see Figure 8).

2 A web page is defined as a document suitable for the World Wide Web. Web browsers coordinate various resources, such as style sheets, images, and scripts, to present a web page.
3 Javascript libraries are a set of predefined functions—pre-written scripts that users can drop into their own applications without doing much additional coding.

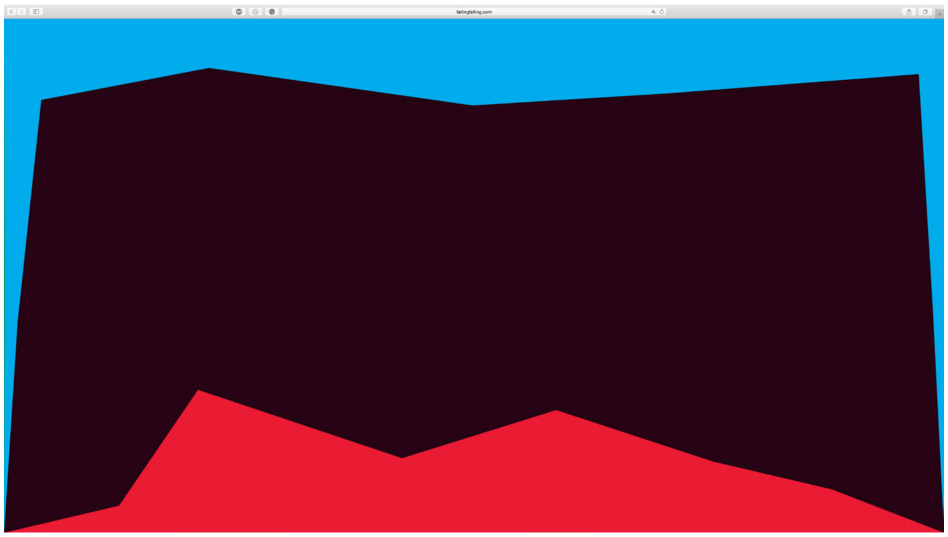


Figure 8: Rafaël Rozendaal, *Fallingfalling.com*, 2011. *Fallingfalling* expresses the concept of eternity by generating a never-ending flow of different variations of the same sketch.

Rozendaal's practice can be placed in conversation with the work of Sol Lewitt, which demands that the artist's process be interpreted by a third party. Just as Lewitt is interested in the way that his instructions for artwork can be interpreted by others, Rozendaal is focused on the interpretation that the web browser makes of his process. Both seek alternate and multiple interpretations of a process they have designed.

In this research, the Javascript *P5.js* library is investigated as a programming language that empowers the web browser as a tool while introducing a process for visual experimentation. This programming language can be read as instructions; integrated within the web browser, the script functions can visually highlight design decisions and their effects instantly. *P5.js* was selected because it is simple to learn, it emphasises visual forms, and it depicts the procedural "if-then" approach well.

The Web Browser as a Tool

For/With/In the Browser is a one-off project led by designer and educator John Caserta that promotes the web browser as a tool for generating graphic design rather than simply displaying it (2014); the web browser's rendering capabilities are powerful enough to generate a wide range of design explorations while providing instant feedback and many unexpected outcomes. Caserta's group uses visual explorations to investigate the limitations of algorithms and automation on the process, the known methods of design on the web, the current user experience of the web browser, and the notion of "responsive design" beyond the web browser (see Figure 9).

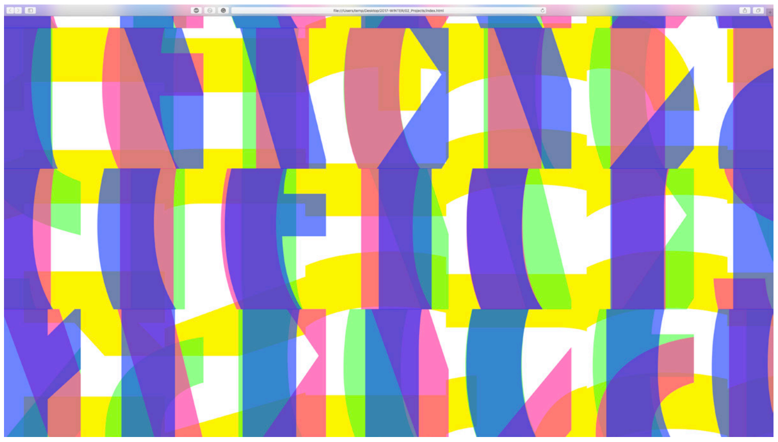


Figure 9: John Caserta, *Typographic Pattern*, 2014. Visuals are generated, styling the structural elements of a web page to generate patterns.

Unlike the programmatic methods used by the conceptual artists previously discussed, Caserta's group used a process based on intuition and iteration in the *For/With/In* project. The product of their inquiries was tested repeatedly in response to their subjective design decisions to produce a more sophisticated result. In addition to their visual explorations, the group also interviewed designers such as Andrew LeClair, who uses the web browser to build "a system instead of just building a single composition" (Rees, 2014) for a design work. Although most of the Caserta group's explorations are not executed using programmatic or logical methods, their investigation demonstrates that the web browser is intrinsically generative and can produce multiple design outcomes. My research benefits greatly from the *For/With/In* project because it actively uses the web browser to generate visual experiments that can be flexible to a wide variety of practices and experiments in graphic design.

PROGRAMME AS METHOD FOR THE WEB BROWSER

Programmatic design methods, which were introduced for the first time into graphic design in the 1960s by Karl Gerstner, can bridge the gap between the way designers think and the way computer software programmers think. In his five essays in *Designing Programmes*, Gerstner demonstrated that the creative process can be logical and pragmatic; Gerstner's programmes may familiarize designers with programmatic thinking, because they need to break down design thinking into individual components that represent concepts or graphic design elements. Andrew Blauvelt has argued that the processes of today's graphic designers tend to reflect a form of thinking similar to programmers' (Blauvelt, 2011).

Programmes are adjustable, modulable, designed processes: systems of instructions and prompts that define a framework through which concepts are expressed. To understand programmatic thinking, designers must apply procedural literacy, a concept defined by its originator, Michael Mateas, as "the ability to read and write processes, to engage procedural representation and æsthetics" (Mateas, 2005). Procedural literacy implies that programming or setting up instruction is not merely a technical task, but an act of communication. A programme's instructions may express concepts that are revealed only when the programme is executed in a computer (in this instance, by the web browser). The programmatic view of the creative process, introduced into the field of design by Gerstner, can help us to better understand how the web browser's "possibilities multiply as the [designer's] choices call forth different visual or textual responses" (Blauvelt, 2011).

The Experience of the Web Browser

The web browser provides fertile ground for programmatic methods and is thus a unique tool for designers. Like other design tools, it is transparent and flexible, and it can be hidden or revealed like any other medium (Bolter, 2005) by the way it displays visual elements. But the web browser has traits that other design tools do not. Designers need to be aware of how users experience the web browser to fully understand its potential as a design tool.

The web browser is highly responsive to design decisions, constantly reevaluating the relationship between the platform on which it displays (computer screen, mobile phone, wearable technology, etc.) and the version and brand of the web browser. The nature and fluidity of this constantly changing relationship changes the outcomes; a set of results for a given interaction between the programme, the platform, and the browser may be possible only under a particular combination of those elements. The web browser is also responsive to interactions with the designer, and it thus reflects the changes in the creative design process: it uses programmatic processes to generate web pages from various components, translating them into visual form and content. In this way, the web browser has outgrown its original role as a display and has become an active medium in the design process, serving as a rich working environment in which the interaction of the designer and the process generates outcomes. These designer/process interactions introduce new possibilities that the web browser must consider in the processing of data.

Programme as Method

The methodological framework for this thesis is inspired by the rich and nuanced background of programmatic methods described earlier, drawn from the work of conceptual artists, the current practices in graphic design, and most importantly the work of Karl Gerstner. Gerstner defines a programme as a method for "invent[ing] rules of arrangement" (Gerstner, 2007) that lead to a group of outcomes; energy is dedicated to the design of the process rather than toward a particular finished product. The design of a programme is precise and organized, so that the designer knows which parameters and variables are in play but does not know the outcome. By elaborating a very precise process, the designer can focus on the relations between parameters and outcomes, and they can adapt "each point in the process in order to modify the design outcome, including the very rules which have been used to generate the outcome" (Herdt, 2016). Through the process, designers will come to make some informed predictions by reflecting on the iterations, and they must begin to think in terms of how values in each parameter fluctuate to create different visuals.

Programmes connect programmatic thinking with design thinking by using a selection of parameters and variables to represent an idea or a means to achieve the idea. Parameters or variables can be understood as pieces of the puzzle that can fit in different situations and at different times in the process; their position in the process influences how the web browser interprets them and may result in different outcomes. Therefore, the choice of parameters is as important as the parameter's position in the programme; a concept may be broken down into parameters in multiple ways, allowing for multiple outcomes.

Chance and Randomness

To design a programme is to work toward the production of a defined (if potentially large) set of possibilities. In the context of this research, programmes serve as a method by which the web browser generates visual outcomes; said otherwise, they provide the web browser with a framework for outcome generation. Each interpretation of the programme by the browser may be novel, but it is predictable, because all outcomes remain within the framework of the programme. How much actual control of the process, then, do designers have?

To respond to this question, one must not "think of technology and [design] as contradicting influences pulling the design outcome in different directions" (Herdt, 2016); one must instead see them as working together. The designer defines the parameters, variables and values of the programme, and then intentionally allows the browser to exceed his or her intent in search of unexpected results. Therefore, this method attempts to guide the logic of the web browser in order to generate unexpected results, changing the web browser from a display to an active tool. In fact, programmes, which are one of the most stable, logical and pragmatic processes for graphic design, deliver unstable and unexpected design outcomes in the context of the web browser. This is because the web browser processes the programme: as a web page is generated, a new variation of that same page is interpreted using the lens of the programme. When the processes of the designers and of the web browser collide, truly unexpected results are produced. This method allows designers to approach creation in a methodical way, since the web browser produces a large number of visuals in a short time.

EXPLORATORY PRACTICE

The exploratory practice described in this section of the thesis aims to develop a methodology for using the web browser as a graphic design tool that produces unexpected results in the context of the web: first, to develop a body of work that emphasises logical and pragmatic approaches to graphic design; second, to promote the web browser as a tool offering a fresh perspective on creation; and third, to explore the possibilities the web browser offers for graphic experimentation to benefit graphic designers' practice.

The visual content used for these experiments is a set of frequently visited, popular websites, including *Wikipedia* and the *New York Times*, but these sites are used only as placeholders—material with which to display the methods. This content could be replaced with any website. To drive the experiments, three objectives that focus on specific characteristics of creation were set. The three objectives may be fulfilled at different levels in each project. The first objective is *to question the form and content of the web through visual explorations*. Graphic designers must juggle form and content in order to display information accurately and comprehensively in a visually compelling manner that is appropriate to the project. Web developers, who are often technically minded, are more concerned with information and data transparency, while artists see in the web the opportunity to explore and experiment with a new medium (Bolter, 2005). Graphic designers tend to practice between these two positions. The first objective seeks to treat both positions as equal in order to challenge them and to reveal assumptions that should be questioned.

The second objective is *to develop an alternate method that will generate an æsthetic unique to the web browser*. To attain this objective, the method must open up the process, orienting the design decisions toward open-ended, interpretable guidelines for the web browser. The web browser uses a specific logic to render visuals, but when it interacts with the programme, the browser's logic may be altered, allowing a unique æsthetic to emerge. The programme seeks to generate visually compelling variations of the same design object and develop them toward a more refined set of solutions.

The third objective is *to increase the number of variations of a design object using a programmatic method*. This last objective aims to produce an applied benefit for graphic designers who use the web browser alongside the programmatic method. The web browser's rendering capabilities allow the programme to create all of its possible outcomes in a very short time.

Re-Shape

In this project, I *explored the message and content of web pages through their deconstruction and restructuration*. A programme was elaborated to play with the content and forms of information and news web pages such as *Wikipedia*, the *New York Times*, and *Wired Magazine* (see Figure 10), while preserving visual cues about the context of the websites. The programme deconstructs and reorganizes information already present on the page to provide a new meaning or reading, while preserving familiar elements that give context. By following the general requirements of the programme, the web browser generated unexpected compositions and interesting visuals,. Even though I had an idea about what elements would appear on the page, the type and the image were styled in unexpected ways.

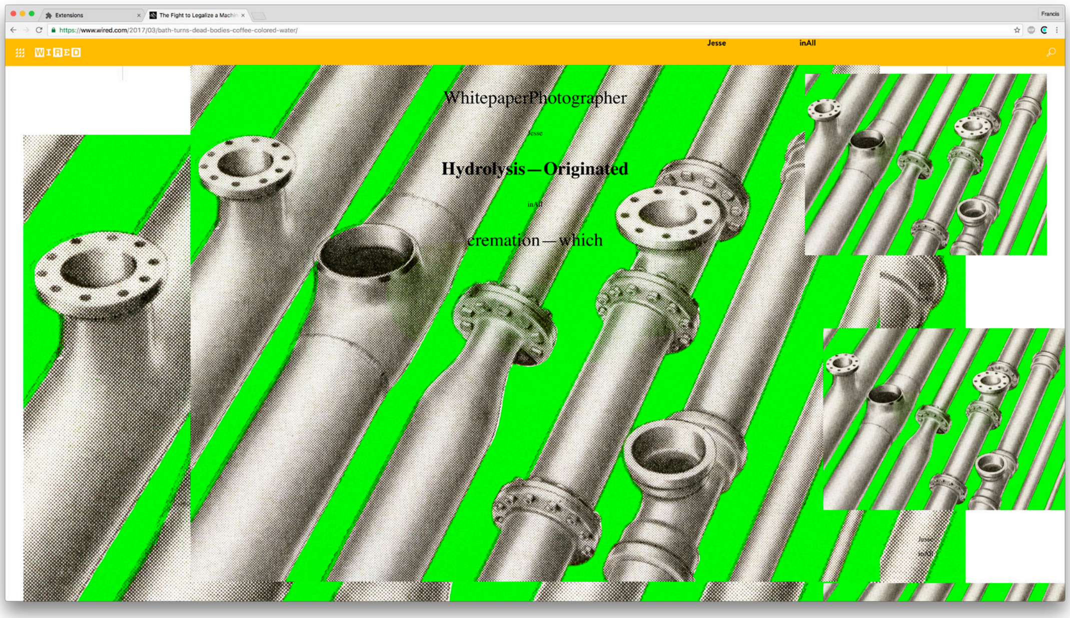


Figure 10.1: Francis Benoit, *Re-Shape: Wired Magazine*, 2017

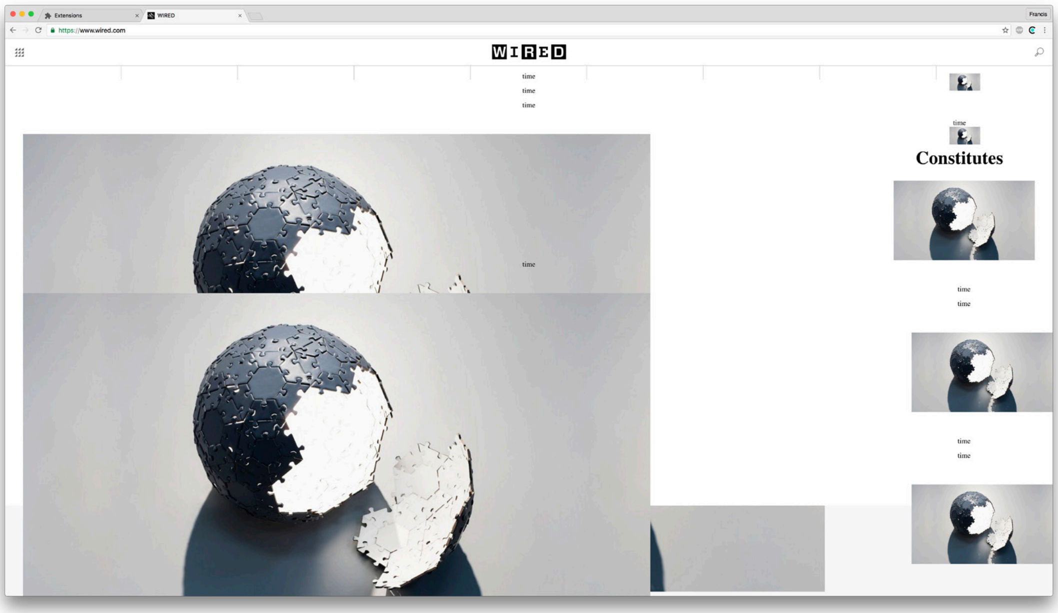
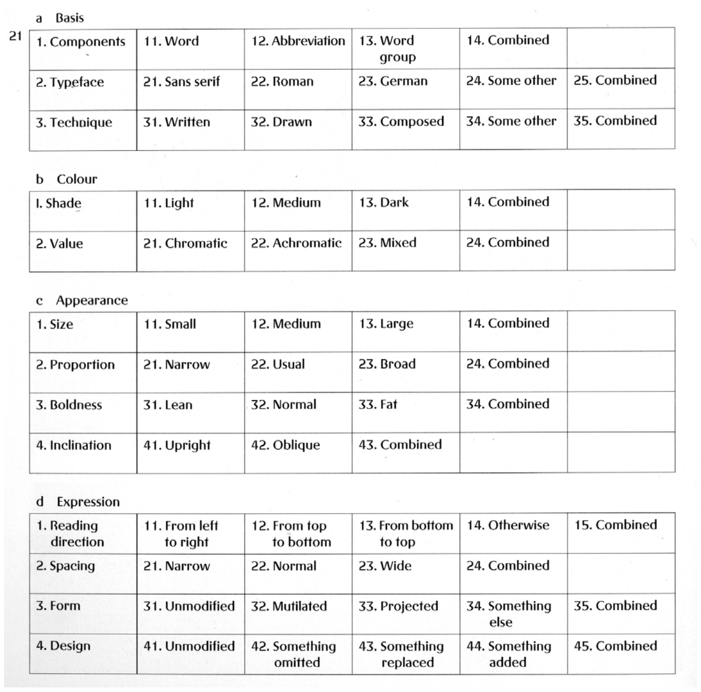


Figure 10.2: Francis Benoit, *Re-Shape: Wired Magazine*, 2017.

The programme used in this investigation was designed to manage textual elements, images, and forms while keeping visual cues about the context, distorting the original message of the web page, but keeping the source recognizable to the reader. Inspired by the *morphological box* of Karl Gerstner (see Table 1), the programme is a table in which parameters and values are set, allowing the designer to alter which elements are in play. At first, the parameter values are intentionally broad and rough (see Table 2). These values and parameters are iteratively refined, changed, or removed to improve the design of the variations. Every tweak and adjustment made by the designer is a step toward better understanding and toward mastery of the process and its outcomes.


a Basis

21

1. Components	11. Word	12. Abbreviation	13. Word group	14. Combined	
2. Typeface	21. Sans serif	22. Roman	23. German	24. Some other	25. Combined
3. Technique	31. Written	32. Drawn	33. Composed	34. Some other	35. Combined

b Colour

I. Shade	11. Light	12. Medium	13. Dark	14. Combined	
2. Value	21. Chromatic	22. Achromatic	23. Mixed	24. Combined	

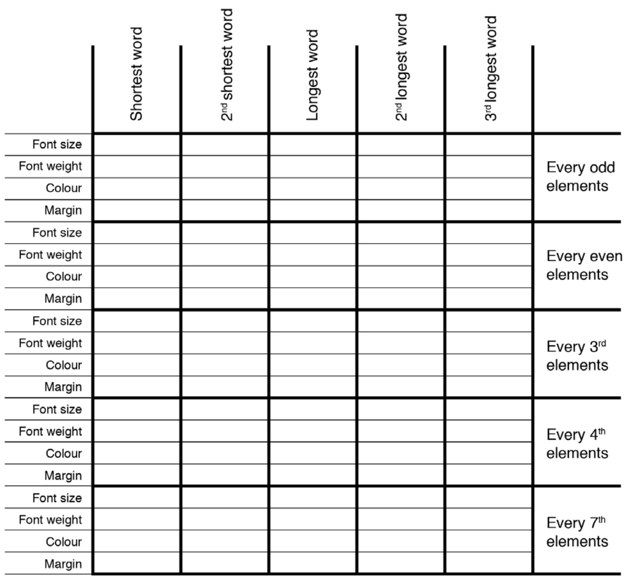
c Appearance

1. Size	11. Small	12. Medium	13. Large	14. Combined	
2. Proportion	21. Narrow	22. Usual	23. Broad	24. Combined	
3. Boldness	31. Lean	32. Normal	33. Fat	34. Combined	
4. Inclination	41. Upright	42. Oblique	43. Combined		

d Expression

1. Reading direction	11. From left to right	12. From top to bottom	13. From bottom to top	14. Otherwise	15. Combined
2. Spacing	21. Narrow	22. Normal	23. Wide	24. Combined	
3. Form	31. Unmodified	32. Mutilated	33. Projected	34. Something else	35. Combined
4. Design	41. Unmodified	42. Something omitted	43. Something replaced	44. Something added	45. Combined

Table 1: Karl Gerstner, *Morphological Box*, 1964.



	Shortest word	2nd shortest word	Longest word	2nd longest word	3rd longest word	
Font size						Every odd elements
Font weight						
Colour						
Margin						
Font size						Every even elements
Font weight						
Colour						
Margin						
Font size						Every 3rd elements
Font weight						
Colour						
Margin						
Font size						Every 4th elements
Font weight						
Colour						
Margin						
Font size						Every 7th elements
Font weight						
Colour						
Margin						

Table 2: Francis Benoit, *Re-Shape programme*, 2017.

It is important that the chosen parameters are not self-contained; simply put, if the value of a parameter is not within the defined framework, the web browser should interpret it in a way that suits the browser's logic rather than the logic dictated by the programme. For example, image placement is not defined in this programme, therefore the web browser sets their position in relation to the decisions made in the programme beforehand (see Appendix A). The process is logical and pragmatic, yet imperfect in its application. The imperfections within the programme prevent it from being entirely predictable, allowing it to generate unexpected outcomes, which can be as creative and interesting as anticipated ones. In fact, the surprise factor of some of the outcomes encouraged me to iterate the programme and modify it to follow paths unveiled by unexpected results (see Figure 11). The designers control only the programme, and they thus experience its limitations; the logic of the web browser, which is out of the designer's control, also takes part in generating the visual outcomes. The web browser thus alters the outcomes of programmes to generate design—perhaps a difficult notion for some designers to accept.

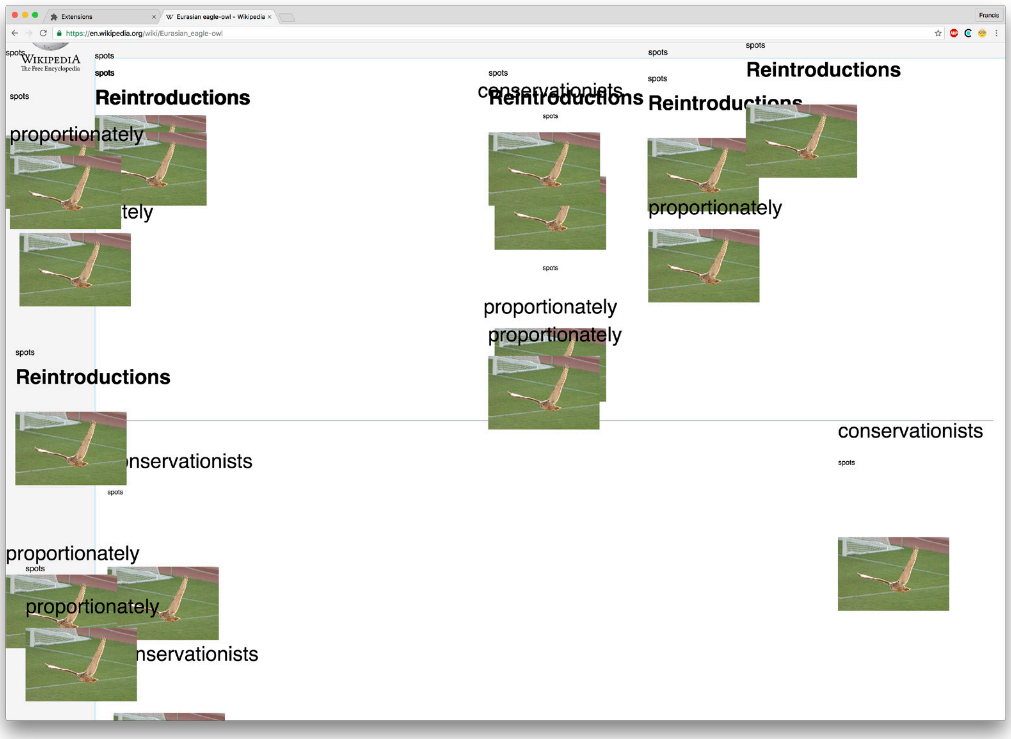
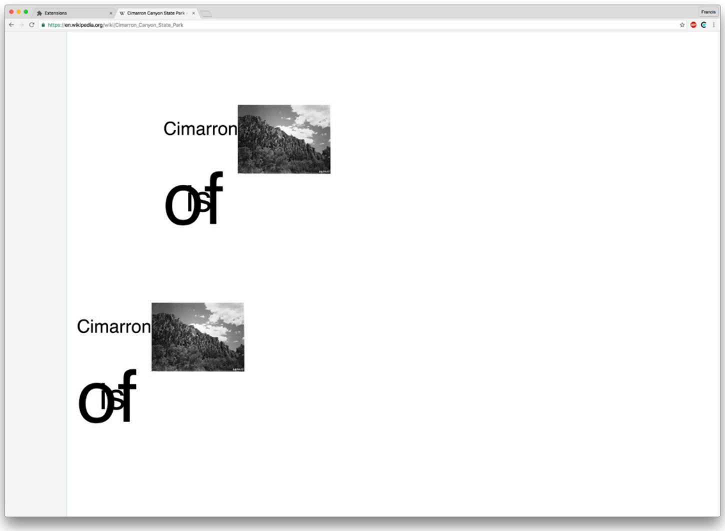
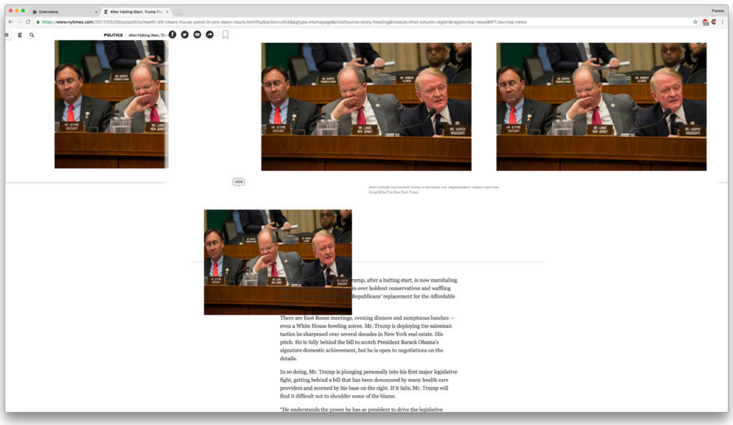


Figure 11: Francis Benoit, *Re-Shape: Eurasian Eagle-Owl*, 2017. The image was not anticipated to display multiple times. Further explorations took note of this element to push the project forward.

This project demonstrated that the limitations of a programmatic method can also be its strengths: the logic of the designed programme collides with the logic of the web browser, producing unexpected outcomes. In this project, the outcomes are not judged by their compositional or formal qualities, but by how much they disrupted the source web pages. The designer must decide whether the variations are successful in creating a new meaning out of the initial web pages (see Figure 12).





Doug Mills/The New York Times

rump, after a halting start, is now marshaling in over holdout conservatives and waffling Republicans' replacement for the Affordable

There are East Room meetings, evening dinners and sumptuous lunches — even a White House bowling soiree. Mr. Trump is deploying the salesman tactics he sharpened over several decades in New York real estate. His pitch: He is fully behind the bill to scotch President Barack Obama's signature domestic achievement, but he is open to negotiations on the details.

In so doing, Mr. Trump is plunging personally into his first major legislative fight, getting behind a bill that has been denounced by many health care providers and scorned by his base on the right. If it fails, Mr. Trump will find it difficult not to shoulder some of the blame.

"He understands the power he has as president to drive the legislative

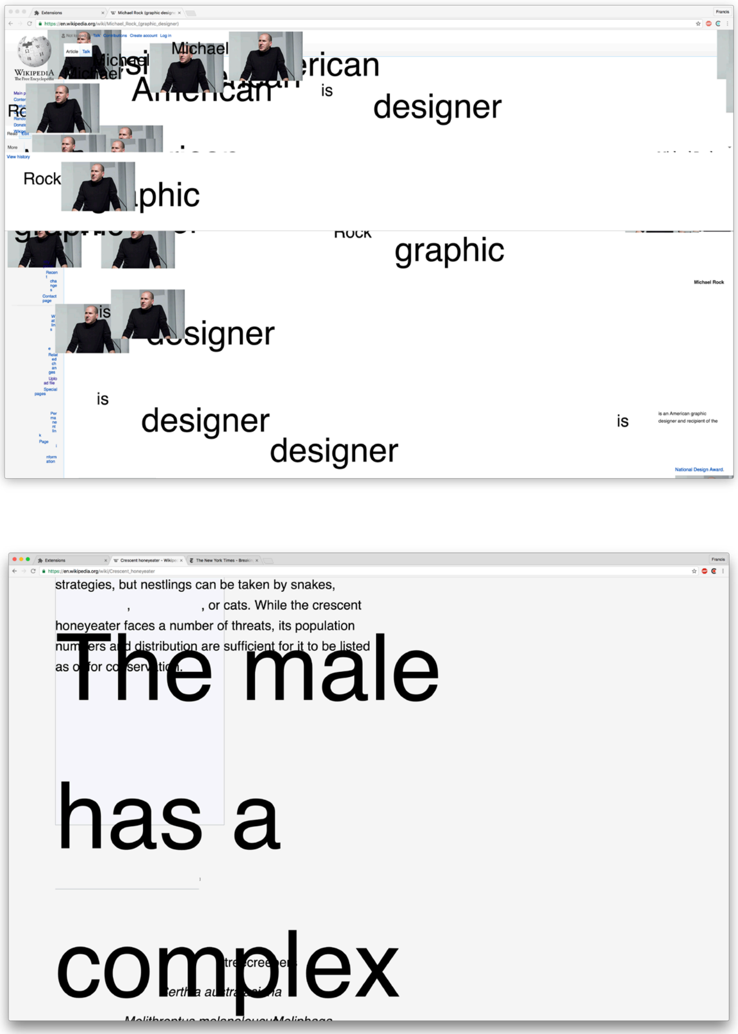


Figure 12: Francis Benoit, *Re-Shape: Wikipedia*, 2017.

Cascade

For the project titled "Cascade," I used a programme that focuses only on textual elements to explore the æsthetics that the web browser can generate from predetermined content and forms. This project aimed to develop an æsthetic-driven programme that concentrated only on the characteristics of specific elements.

This programme works differently than the "Re-Shape" programme described in the previous section. I call the Cascade programme a *relational timeline* (see Figure 13). The line represents the textual elements, and the position of the arrow on the line defines the values of all parameters. All the parameters are linked to each other; when one parameter fluctuates, the effect ripples across the programme. This programme provides a framework in which the web browser acts on nuances and intricacies, because all parameters are altering characteristics of the same element. For textual elements, many parameters, such as size, opacity, kerning, and colour, can be modified (see Appendix B). The unexpected variations then appear in details and intricacies, not in drastically different compositions. This project revealed that programmes can adapt to different scopes of exploration. Once parameters have been set, the designer can work with the web browser to generate a strong æsthetic that could be developed only through iterations and refinements of specific elements.

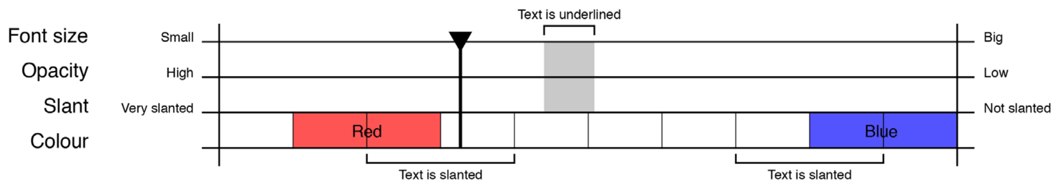
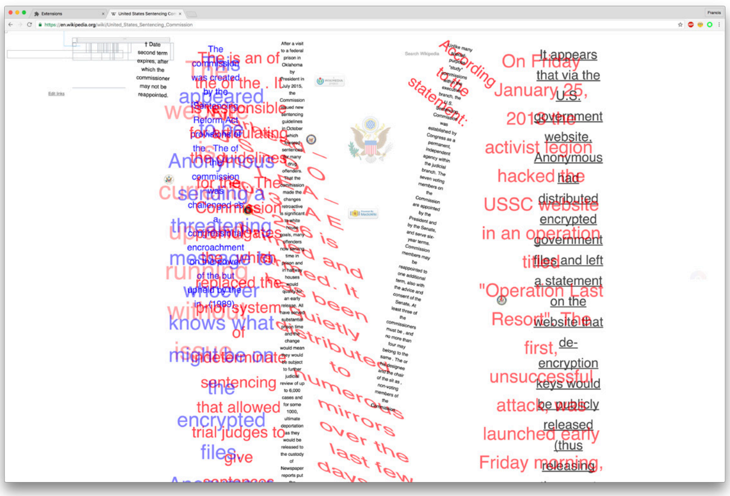


Figure 13: Francis Benoit, *Relational Timeline*, 2017.



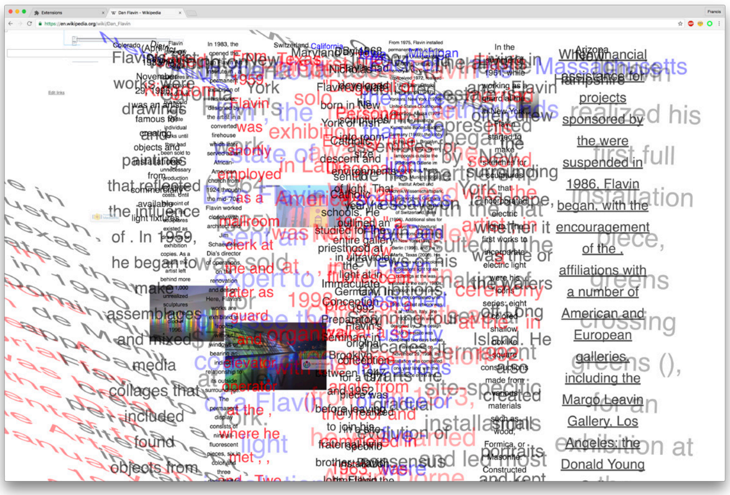
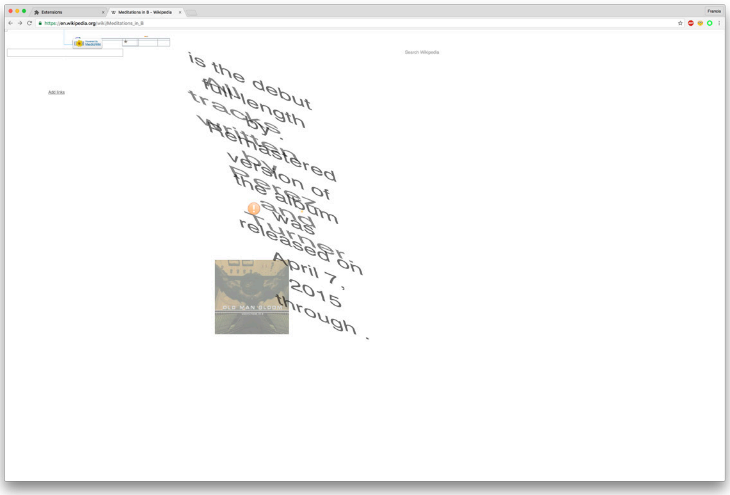
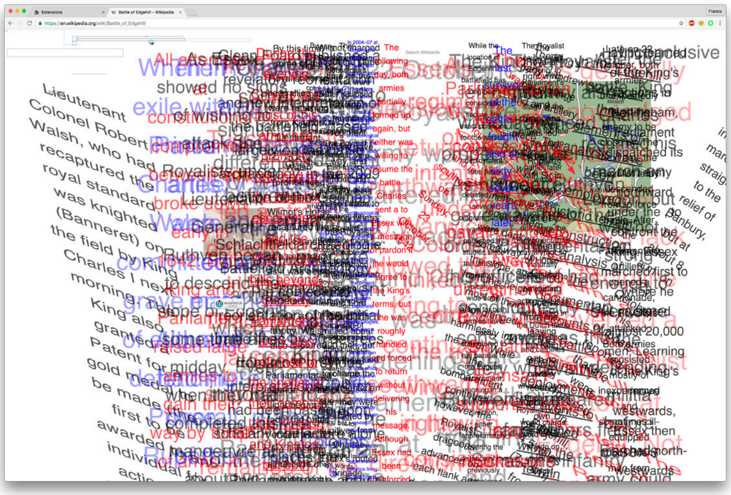




Figure 14.1: Francis Benoit, *Cascade: US Sentencing Commission*, 2017.
Figure 14.2: Francis Benoit, *Cascade: Dan Flavin*, 2017.
Figure 14.3: Francis Benoit, *Cascade: Meditation in B*, 2017.
Figure 14.4: Francis Benoit, *Cascade: Battle of Edgehill*, 2017.

Break and Play

In this project, I explored the generative nature of the web browser and sought to increase the number of variations of a design object produced in a small amount of time. This project was designed to generate multiple variations, allowing for the evaluation of a design artefact's possibilities. The programme was implemented in a *Bootstrap* framework, which is widely used by designers to lay out basic structures: layout, type, functionality, etc. This project thus demonstrates that programmes need not stand alone, but can be integrated into designers' pre-established processes.

The design of the programme is based on a tree structure, where every decision leads to its respective set of possibilities. The programme plays only with the position of the letters of the composition. The words *Break*, *and*, *play* can be placed anywhere in the layout of the page. However, the letters of each word follow a logic; words can be formed in different ways, but still be legible (see Figure 15).

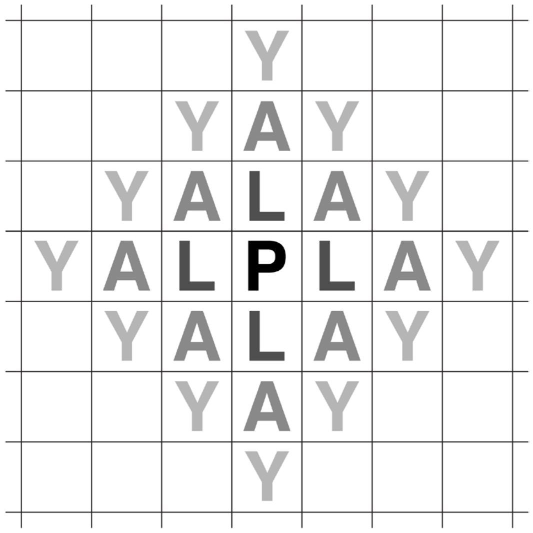


Figure 15: The word *play* can be placed in any configuration that respects this diagram. Each tone of grey represents a level of positioning.

The placement of the letters may, on some occasions, be in conflict with the grid in which the words are displayed, causing the words to be incomplete or broken. Through repetitions, the internal flaws of the programme are exposed by the web browser. These flaws were discovered to be an unexpected variable that needed to be included in the process. A flaw can appear in any composition and became an interesting part of it (see Appendix C).

As a way to grasp the extent of the possibilities generated by the web browser, the variations were printed out. Printing visuals of the web is unusual in the graphic design process, but printing the variations externalizes the outcomes and fixes them in time. It also demonstrates that the web browser can be a production tool, like other software used by graphic designers, rather than a display mechanism.

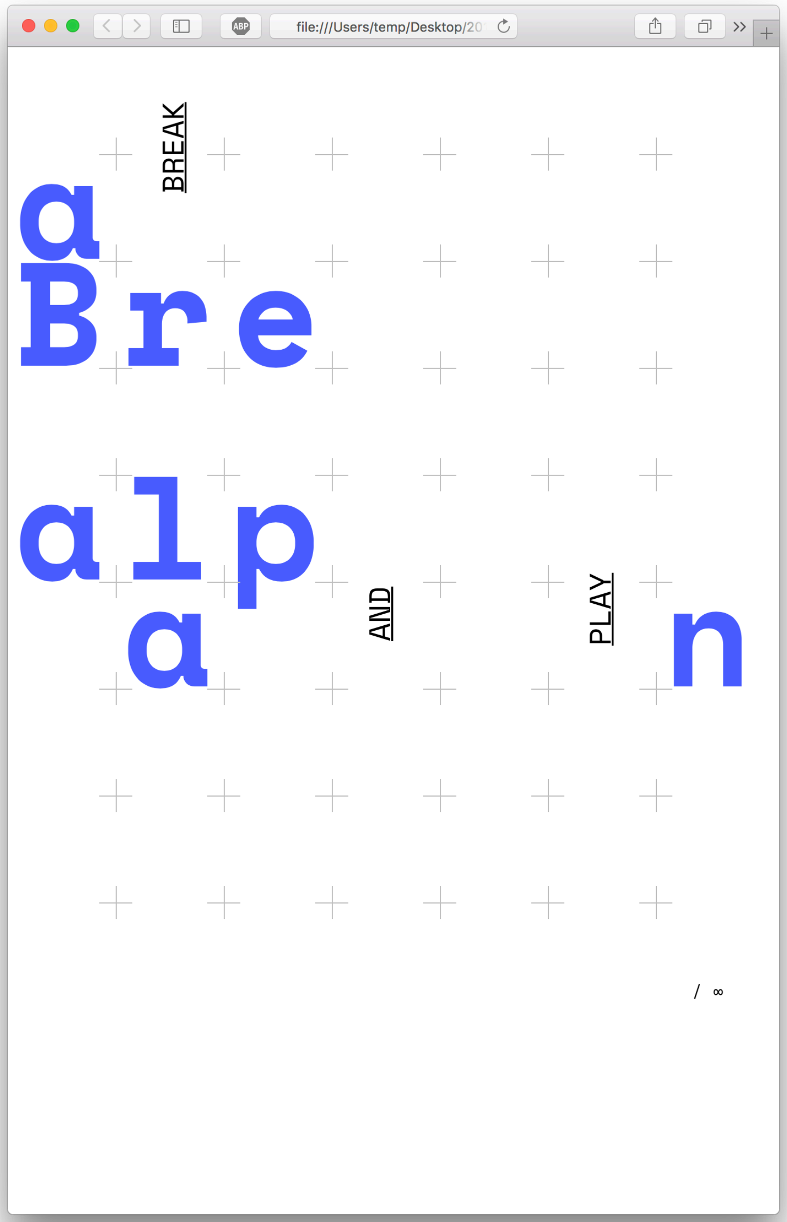
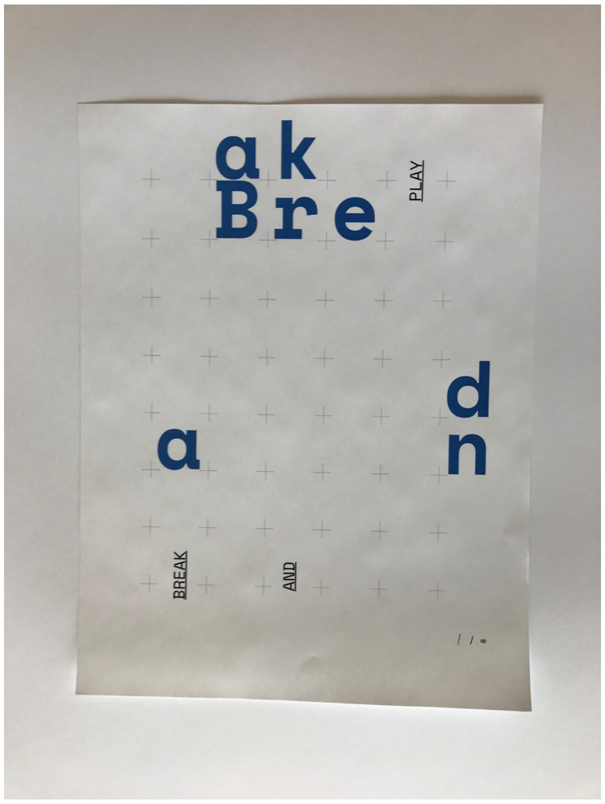


Figure 16.1: Francis Benoit, *Break and Play*, 2017. A composition in the web browser.



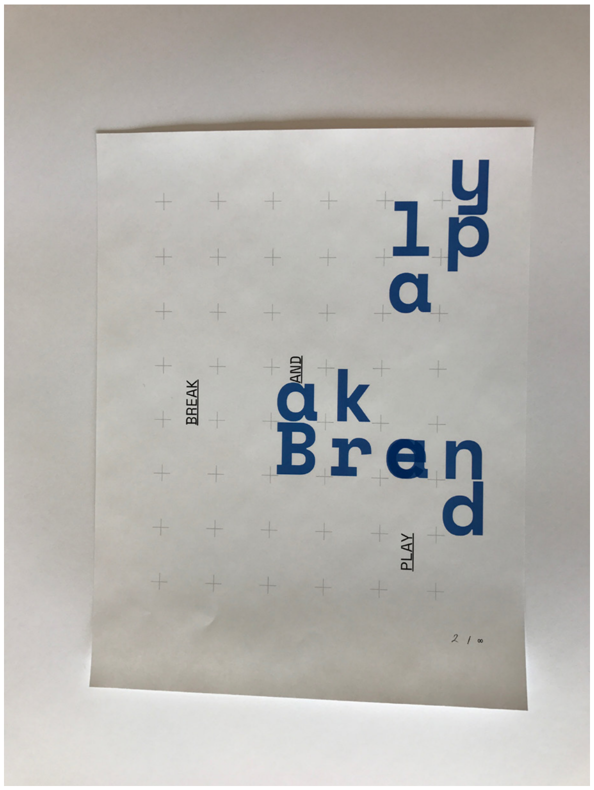


Figure 16.2: Francis Benoit, *1/infinite*, 2017.
Figure 16.3: Francis Benoit, *2/infinite*, 2017.

CONCLUSION

This thesis describes how a new programmatic method that considered the web browser as a tool in the creative process can be used to break from the expected design outcome on the web. My investigation revealed that the elaboration of a structured, logical, and pragmatic process to guide the production of design objects could help designers to focus on their projects' objectives. The web browser used to introduce this method can provide graphic designers with an alternate approach to the design process and open them up to a different mode of thinking. The programmatic approach encourages logical thinking while opening the door to unexpected outcomes. The work of Karl Gerstner (2007) and that of John Caserta (2014) reinforce this fact, and the work of Andrew Blauvelt affirms that modern graphic designers' process is becoming more similar to the "if-then" approach of the programmer (Blauvelt, 2011). Therefore, the web browser and the programmatic approach together contribute to the practice of graphic design by inviting us to reimagine the design process in the context of the web. In the exploratory practice, I defined three objectives that drove projects. All these objectives challenged different aspects of my process and my technical skills, and demonstrated that it was possible to design using the web browser as a tool while engaging in procedural literacy.

First, the programmatic method was found to be an excellent match with the web browser, elevating it from a simple browser of information to a tool for generating graphic design. The web browser stands apart from other tools because it works as a translator between programming languages and visual forms, enabling designers to create using parameters and variables, rather than with a mouse or with a pen and tablet. The web browser is uniquely and intrinsically responsive to every decision made by the designer. This emphasises an iterative process because the designer is in constant dialogue between his decisions and the response of the browser.

Second, I had hypothesised that the programmatic method would provide a focused output for a proposed project. It was revealed to be more powerful than that: with the addition of the web browser, the programmatic method generated variations of the same design object within a specified framework. It also provided unexpected results that opened unintended paths. The web browser's specific logic for generating visuals is often in conflict with the logic of the process. Although the method defines variables and parameters, the web browser may interpret them in a different way to fit its logic (Reas, 2016). This conflict can be limiting, but unexpected outcomes can also emerge from this confrontation.

Third, programmes are well defined and concise while providing a wide range of possibilities within their frameworks, and their scope can vary depending on the purpose of the inquiry. This flexibility allows designers to explore and iterate their visual experiments at a very granular level. In my exploratory practice, I noticed that even at the most advanced iterations of my programme, unexpected and interesting directions continued to emerge, motivating me to keep refining the process. The designer must decide when to take a step back and define which variations are successful; designers must have an organized process for managing all the possible outcomes so that the designer can understand each parameter's impact.

As a designer, I understood that graphic design on the web was not only a purely applied form of design without much space for experimentation. In fact, this investigation proved that the context of the web offers opportunities for exploration, and that this exploration could produce outcomes that I did not envision. I think that the experiments in this investigation can inspire graphic designers to experiment more with the web browser as a tool and to consider new approaches to their creative process.

The primary goal of this research was to break from expected outcomes on the web by using a programmatic method alongside the web browser as a design tool, and to demonstrate that programmatic methods can introduce an alternate way of thinking about the creative process. In short, this thesis demonstrated several things: designers should be aware of how parameters interact with each other in programmes so as to gain a deeper understanding of the way changes in parameters make their experimentation more novel and powerful; designers can break from expected design outcomes of the web by using a programmatic method that brings new structure to their creative process while generating a wide variety of outcomes; and designers should rethink the use of the web browser as a responsive design tool that reflects design decisions in real time.

In the end, this thesis was a way to investigate my own design process and my practice as a graphic designer. This research is not only an inquiry into design, but also an inquiry into myself as a graphic designer. It questioned the way I thought about graphic design and broadened my horizons about unfamiliar processes, especially the value of processes that are purely logical and pragmatic. Understanding alternative processes can help designers position themselves in a constellation of other practitioners, learning from them and becoming better, more complete graphic designers. I hope that my research encourages a greater curiosity about the programmatic process and the web browser as a tool in the current context of graphic design, and that it demonstrates that the web is also a space for exploration and experimentation.

BIBLIOGRAPHY

Baecker, Ronald M. 1995. *Readings in Human-Computer Interaction: Toward the Year 2000*. 2nd ed. San Francisco: Morgan Kaufmann Publishers.

Beek, Sanne van der. 2012. "From Representation to Rhizome: Open Design from a Relational Perspective," *The Design Journal* 15, no. 4: 423–42.

Blauvelt, Andrew. 2013. "Ghost in the Machine: Distributing Subjectivity." In *Conditional Design Workbook*, iii–xi. Amsterdam, The Netherlands: Valiz.

Blauvelt, Andrew. 2011. "Toward Relational Design." *The Design Observer*, March 8.

Bolter, J. David, and Diane Gromala. 2005. *Windows and Mirrors: Interaction Design, Digital Art, and the Myth of Transparency*. Cambridge, MA : MIT Press.

Buchloh, Benjamin H. D. 2008. "The Diagram and the Colour Chip: Gerhard Richter's 4900 Colours." In *Gerhard Richter 4900 Colours*, 61–71. Ostfildern: Hatje Cantz Verlag.

Caserta, John, ed. 2014. *For/With/In the Browser: Graphic Design For, With, and in the Browser*. Providence, RI: The Design Office.

Cross, Nigel. 1984. *Developments in Design Methodology*. Chichester: Wiley.

DiNucci, Darcy. 1999. "Fragmented Future." *Print* 53, no. 4: 32, 221–222.

Gerstner, Karl. 2007. *Designing Programmes: Instead of Solutions for Problems Programmes for Solution*. 3rd ed. Baden: Lars Müller Publishers.

Gredinger, Paul. 2007. "Pro-Programmatic" in *Designing Programmes: Instead of Solutions for Problems Programmes for Solution*. 3rd ed. Baden: Lars Müller Publishers.

Hanhardt, John G., and Whitney Museum of American Art. 1982. *Nam June Paik*. New York: Whitney Museum of American Art, in association with W.W. Norton. Published in conjunction with the exhibition of the same name, shown at the Whitney Museum of American Art.

Herdt, Tanja. 2016. "Generative Design and the Problem of Anticipation." In *The Intrinsic Logic of Design*, 118–27. Zurich: Niggli/bnb media GmbH.

Kostelanetz, Richard. 2003. *Conversing with Cage*. 2nd ed. New York: Routledge, 2003.

Legg, Alicia. 1978. "Introduction" in *Sol Lewitt: the Museum of Modern Art*. New York: Museum of Modern Art. Published in conjuction with the exhibition of the same name, shown at the Museum of Modern Art.

Lewitt, Sol. 1967. "Paragraphs on Conceptual Art." *Artforum*, June: 79–83.

Mateas, Michael. 2005. “Procedural Literacy: Educating the New Media Practitioner.” *On the Horizon* 13, no. 2: 101–11.

Maurer, Luna, Edo Paulus, Jonathan Puckey, and Roel Wouters. 2013. “Conditional Design Manifesto.” In *Conditional Design Workbook*, ii. Amsterdam: Valiz.

Moss, Ceci. 2008. “Interview with Harm van Den Dorpel on Club Internet’s ‘Free Fall.’” *Rhizome*, October 17. http://rhizome.org/editorial/2008/oct/17/interview-with-harm-van-den-dorpel-on-club-interne/.

O’Reilly, Tim. 2005. “What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software.” *O’Reilly*, September 30. http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

Pelzer, Birgit. 2008. “The Asymptote of Chance.” In *Gerhard Richter 4900 Colours*, 117–129. Ostfildern: Hatje Cantz Verlag, 2008.

Reas, Casey. 2016. “A Rude and Undigested Mass.” *Holo* 1, no. 2 (September): 85–87.

Reas, Casey, Chandler McWilliams, and LUST. 2010. *Form+Code in Design, Art, and Architecture*. New York: Princeton Architectural Press.

Reas, Casey, and Ben Fry. 2014. *Processing: A Programming Handbook for Visual Designers and Artists*. 2nd ed. Cambridge, MA: MIT Press.

Rees, Christina. 2014. “Andrew LeClair Talks with Christina Rees.” In *For/With/In the Browser: Graphic Design For, With, and in the Browser*. Providence, RI: The Design Office.

Sayers, Dorothy L. 1970. *The Mind of the Maker*. Westport, CT: Greenwood Press.

APPENDICES

Appendix A

The integral of the source code of the *Re-Shape* project can be consulted, downloaded and executed on the GitHub repository: https://github.com/francisbenoit/Re-Shape. The main snippet of the project is in the file named “style-script.js”.

Appendix B

The source code of the *Cascade* project can be consulted, downloaded and executed on the GitHub repository: https://github.com/francisbenoit/Cascade.The main snippet of the project is in the file named "style-script.js".

Appendix C

The source code of the *Break and Play* project can be consulted, downloaded and executed on the GitHub repository: https://github.com/francisbenoit/Break-and-Play. The main snippet of the project is in the file named "breakandplay.js".