DHQ: Digital Humanities Quarterly
2017
Volume 11 Number 3

Digital Oulipo: Programming Potential Literature

Natalie Berkman <nberkman_at_princeton_dot_edu>, Princeton University

Abstract

The formally constrained work of the Oulipo (l'Ouvroir de Littérature Potentielle, loosely translated as Workshop of Potential Literature) lends itself particularly well to digital studies, which was quickly recognized by the members of the group. To facilitate its goal of avoiding chance in its literary production, the group was naturally drawn to the determinism of computers, where true chance is simply impossible. In its early years, therefore, the group used algorithmic procedures as a starting point for various texts and also attempted to program these texts on actual computers, creating some of the first electronic literature and embarking on proto-digital humanities work as early as the 1960s and 1970s, later abandoning these efforts and relegating all subsequent activity to a subsidiary group.

To understand the Oulipo's forays into computer science and more importantly, why they abandoned them, I designed and carried out one of the inaugural projects of the Princeton Center for Digital Humanities. The goal was twofold: first, through exploratory programming, I intended to create interactive, digital annexes to accompany my doctoral dissertation; more importantly, I hoped that by attempting to reproduce the Oulipo's own algorithmic efforts, I would gain similar insights into the nature of "Potential Literature" and be able to understand why the group abandoned such efforts after the 1970s.

This article describes the content, development, and results of my project. For each of my three Python-based annexes, I offer a historical survey of the Oulipian text or procedure discussed within and the Oulipo's own proto-digital humanities experiments; then, I will talk about my own experiences as a coder-researcher, what learning Python has brought to my project, and how my exploratory programming offered me a new kind of critical reflection. Establishing these annexes forced me to learn to code, a type of work that does not only produce digital texts, but also helped me to reflect on the notion of chance in a more nuanced way. Finally, coding has allowed me to better understand the Oulipian mentality concerning this sort of digital experimentation.

Introduction

The Ouvroir de Litterature Potentielle (OuLiPo, loosely translated into English as the Workshop for Potential Literature), an experimental writing group founded in Paris in 1960, attempts to apply mathematical procedures to literature, inventing procedures (known as *constraints*) to follow during the composition of a text. One of the main goals of this strategy is to reduce the role of chance, and it is therefore unsurprising that computers were one of the first items on its early agenda. Computers are utterly incompatible with the notion of chance, and in theory should have been a perfect, and rather timely solution. Indeed, the founding of the Oulipo coincided with a critical stage in the development of computers, and the group took full advantage thereof, programming their texts and procedures through partnerships with Bull Computers and the Centre Pompidou. Early computing put one in much closer contact with the basic fabric of coding and the members quickly learned that writing code requires one to divide a problem into its simplest, logical components, very much like the elementary procedures they were inventing in the group's formative years. However, by the 1970s, the group abandoned such efforts, relegating all future algorithmic experimentation to a subsidiary. 1

In recent years, the field of digital humanities has been gaining popularity. While far from nascent (the origins can be dated back to as early as the 1940s), the discipline has failed to define itself, often preferring a broad scope that situates 2

its activity at the intersection of computing, technology, and humanities scholarship. In theory, this sort of vague definition could be seen as beneficial, encompassing a wide range of tools and techniques that can be applied to humanistic work. In practice, while a great variety of scholarship has been and is currently being undertaken that claims to use digital humanities practices, a majority of this work seems to fall under the categories of textual encoding, the creation of digital archives, and the use of ready-made tools to run some type of analysis on a digital text or visualize data from it. The fact that digital humanists do not seem to prioritize learning to program has been lamented on several occasions by Nick Montfort, both in his own treatise on exploratory programming for humanistic inquiry [Montfort 2016] and also in *A New Companion to Digital Humanities* [Montfort 2015]. While not widespread, a knowledge of computing languages and algorithmic procedures can indeed be productive in humanistic research, as the Oulipo demonstrates through its aforementioned computer efforts as well as through its non-algorithmic productions, which given the highly intentional compositional methods, lend themselves well to computational methods of inquiry. The Oulipo is not only a particularly good example of raw material to be analyzed using digital methods, but is also present in many canonical digital humanities criticisms, one of the best examples being Stephen Ramsay's *Reading Machines* [Ramsay 2011].

3 However, the goal of this article is not to give a critical overview of the use of Oulipo studies for digital humanities. It is rather to demonstrate how I was able to use exploratory programming to understand the Oulipo's forays into computer science and more importantly, why the group abandoned such initiatives. To this end, I designed and carried out one of the inaugural projects of the Princeton Center for Digital Humanities under the guidance of Cliff Wulfman[1] as my technical lead. My project had two main goals: first, through exploratory programming, I intended to create interactive, digital annexes to accompany my doctoral dissertation; more importantly, I hoped that by attempting to reproduce the Oulipo's own algorithmic efforts, I would gain similar insights into the nature of "Potential Literature" and be able to understand why the group abandoned such efforts. My original intention was to create five digital annexes (of which I only completed three that would constitute the final product)[2], each one to accompany a chapter in my dissertation. The purpose and development of these annexes were intertwined: my preliminary research on the Oulipo's computer efforts enriched my own understanding of the task at hand; by learning to program, I was afforded a rare glimpse into the mindset of the early Oulipo; finally, by manipulating the finished product, my reader would better understand the underlying mathematical principles at work in these texts.

4 This article describes the content, development, and results of my project. For each of my three Python-based annexes, I offer a historical survey of the Oulipian text or procedure discussed within and the Oulipo's own proto-digital humanities experiments; then, I will talk about my own experiences as a coder-researcher, what learning Python has brought to my project, and how my exploratory programming offered me a new kind of critical reflection. Establishing these annexes forced me to learn to code, a type of work that does not only produce digital texts, but also helped me to reflect on the notion of chance in a more nuanced way. Finally, coding has allowed me to better understand the Oulipian mentality concerning this sort of digital experimentation. Carrying out a digital humanities project on the Oulipo — itself a quasi-academic research group that both analyzes and creates — resulted in a hybrid experiment with multiple, varied results and insights, which helped me both understand the history and development of Oulipian aesthetics as well as understand their texts better. While unconventional given the nature of digital humanities work today, the results of my research demonstrate a productive use of programming as both a creative and analytic exercise that can undoubtedly prove fruitful in future digital humanities work.

Part I: Combinatorial Poetry:

Cent mille milliards de poèmes

5 Published in 1961 by Oulipian cofounder Raymond Queneau and followed by a critical essay by his fellow cofounder François Le Lionnais, the *Cent mille milliards de poèmes* (*A Hundred Thousand Billion Poems*) is generally considered to be the first "Oulipian" text and therefore serves as an illustration of the group's initial goals and influences, one of the main ones being computer science.

6 In the preface, unconventionally called a *mode d'emploi* or user's manual, Queneau explains how the text was *written*, but discourages the reader by claiming that the volume is impossible to *read* (it would take a very determined reader

190,258,751 years to read all the poems, even devoting 24 hours a day, 7 days a week [CMMP, 2]). To produce his machine, Queneau wrote 10 “base” sonnets, each of which is constrained by not only the rules of this famous fixed form (14 verses, two quatrains and two tercets, a fixed rhyme scheme, the poetic meter of the French alexandrine, etc.), but also by three additional rules [CMMP, 1]:

1. Each rhyme in any of the original sonnets must rhyme with the corresponding verse of any of the other sonnets. Queneau imposes the additional constraint that the rhymes should not be too banal, rare, or unique.
2. Each sonnet must have a “theme and continuity,” however the sonnets produced by the system do not have that same “charm.”
3. Finally, the grammatical structure had to permit the substitution of any individual verse with the corresponding verse from any of the other nine sonnets.

With 10 sonnets of 14 lines each, adhering to these three simple rules, Queneau’s system is capable of producing 10^{14} “potential” poems. The use of the word *potential* is a double entendre — *potentiel* in French is the adjectival form of the word *puissance*, or exponential power. 7

Before reaching the poems, the reader is faced with an epigraph by Alan Turing: “Only a machine can read a sonnet written by another machine” [CMMP].[3] This approximate translation comes from Turing’s famous *Times* interview: “...I do not see why it [a computer] should not enter any one of the fields normally covered by the human intellect, and eventually compete on equal terms. I do not think you can even draw the line about sonnets, though the comparison is perhaps a little bit unfair because a sonnet written by a machine will be better appreciated by another machine” [Hodges 2012, 420]. 8

Note the use of the passive voice in the original Turing citation. Queneau (who understood English too well to make such a blatant translation error) makes the machine the subject of his epigraph. The reader of his volume, he claims while ventriloquizing Turing, is a computer. The physical conception of the volume does indeed support this proposition: rather than taking pleasure in reading an exponential number of sonnets (most of which are an incoherent jumble of verses from prewritten sonnets, devoid of a theme and written by no one), the reader must manipulate the verses of Queneau’s original poems, cut into strips to produce a functional book-machine hybrid. The Oulipo quickly turned to actual machines to reimagine this foundational text, literally programming a combinatorial poetry collection that already owed much to computers. 9

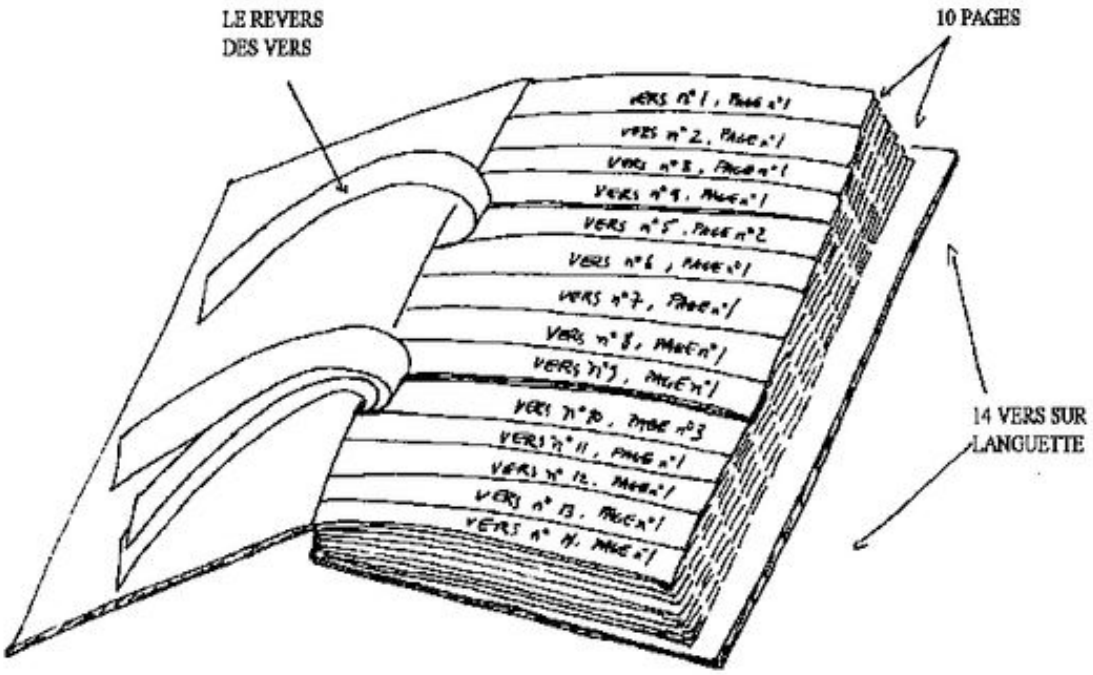

Figure 1. Image depicting how the volume functions. Source: http://openset.nl/blog/wp-content/uploads/2012/08/Prevert.jpg

Oulipian Computer experiments

Before designing my own digital editions, I needed to take into consideration what the Oulipo had already attempted. The *Cent mille milliards de poèmes*, in addition to being the first Oulipian text, was also the one that had been most programmed by the group — three times, in fact, in the 1960s, 1970s, and finally in 1999.[4] Evaluating these attempts, contextualizing them given the historical capabilities of computers at the time, and understanding how the group perception of this programming challenge changed with each new attempt would not only enrich my dissertation, but would also help me in designing my digital annex. 10

After deciding at their second meeting to seek out computers to pursue various types of analytic work, the founding members of the Oulipo came into contact with Dmitri Starynkevitch, a French-born Russian immigrant, who was working for Bull Computers at the time. He began Oulipian computer work on the CAB 500 computer that operated with the PAF language (Programmation Automatique des Formules) [Starynkevitch article, 26]. Collaboration with Starynkevitch brought the Oulipo into direct contact with computing, but also enabled Starynkevitch to apply the highly mathematical programming language to creative endeavors, allowing both sides to gain insight into the potential of this new technology. 11

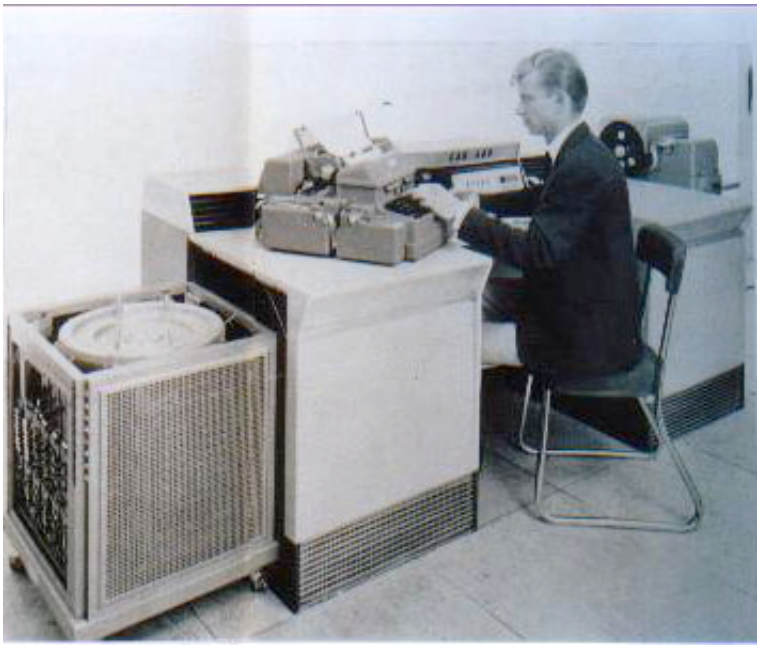

Figure 2. Image of a CAB 500 computer. Source: http://www.histoireinform.com/Histoire/+infos2/CAB_500.jpg

12 In August 1961, Starynkevitch sent Queneau excerpts from the *Cent mille milliards de poèmes* produced on the computer. The reaction of the Oulipians was less enthusiastic than might have been expected: “We would like M. Starynkevitch to detail the method he used: we hope that the choice of the verses was not left to chance” [Bens 2005, 79].[5] This response is key to understanding Oulipian aesthetics and what the early group wanted to gain from this computer experiment. While there is a certain amount of chance in the production of the poems, this chance lies entirely with the reader, who is autonomous in producing what he or she wishes to read. While Queneau’s epigraph indicates that the reader is a sort of machine, Starynkevitch’s literal machine is too random for the group’s aesthetics.

13 After their collaboration with Starynkevitch, the Oulipo had one final group effort in the 1970s, a project known as the *Atelier de Recherches et Techniques Avancées* (A.R.T.A.) at the Centre Pompidou, which Paul Fournel detailed first in a conference entitled *Écrivain-ordinateur* in June 1977 and subsequently in the *Atlas de Littérature Potentielle* of 1981. Directed by Paul Braffort (a computer scientist and member of the second generation of Oulipians coopted after the group’s founding), the goal of this project was to create a foundation for “a possible agreement between computer science and literary creation” [Oulipo Atlas, 298].[6] They worked primarily with preexisting texts that were already combinatorial or algorithmic in nature. Their intent was to demonstrate that with such texts, computers can assist in the reading process.

14 The first experimental text was Queneau’s *Cent mille milliards de poèmes*, of which Fournel admits: ““The printed collection is very nicely conceived but the manipulation of the strips on which each verse is printed is sometimes delicate” [Oulipo Atlas, 299].[7] The physical design of the collection is visually compelling and provides a more intuitive understanding of the combinatorics that regulate the volume. However, certain manipulations of the set of poems are cumbersome, for instance reading Queneau’s original ten sonnets. In order for the computer to facilitate the reading process, the A.R.T.A. program makes a selection from Queneau’s original ten sonnets based on the length of the reader’s name and the time it takes the reader to type it. This generates a “magic number” of fourteen digits, each of which determines from which of the original ten sonnets the corresponding verse is taken [Oulipo Atlas, 299].

15 This method responds to the Oulipo’s earlier question about Starynkevitch’s program regarding the method of choosing the verses. However, this new program is still surprisingly restrictive. Mark Wolff acknowledges that this program has more “potential” in the Oulipian sense than the one produced with the help of Starynkevitch: “Such a program has no potential in the Oulipian sense because random numbers produce aleatory effects. The original algorithm preserves an

active role for the user, even if that role requires the minimal engagement of typing one's name in order to sustain the creative process" [Wolff 2007, 6]. However, with the physical edition, the reader has the freedom to read in different ways. This program, on the other hand, only produces a single poem using a method that is never explicitly stated. While a reader could type his/her name multiple times to produce new poetry to read, this is a fundamentally less interesting task than manipulating the physical volume, where the reader has more autonomy. Herein lies one of the great difficulties that I would soon face in carrying out my project: since the active selection of poems is what creates the illusion that the reader becomes the author, having a computer replace the manipulation of verses deprives the reader of the pleasure and creativity he or she would experience with the physical volume.

My CMMP experiment

Given my understanding of the text and the Oulipo's multiple attempts to program it, I began my project with the *Cent mille milliards de poèmes* annex. Its status as the first Oulipian text did not enter into this decision, but rather I hoped that this text would be simple enough to serve as a practical introduction to programming in Python. My goal of creating a digital edition of this text was not original. In addition to the Oulipian editions I detailed above, there are other excellent online editions[8] of the *Cent mille milliards de poèmes*. My aim was twofold: first, to use this text as a practical introduction to computer programming, allowing me to better understand the Oulipo's struggle with this text and its computer legacy; second, to use the creative enterprise of designing new methods to produce pseudo-random sonnets in order to grapple with the Oulipian notion of chance myself. 16

To transform Queneau's work, I first had to visualize the original 10 sonnets in computer science terms, as an *array* of *arrays*. In computer science, an *array* is a type of data structure consisting of a collection of elements, each identified by at least one index or key. A sonnet, by definition, is already an array of fourteen verses that can each be attributed a numbered index. In order to make Queneau's sonnets intelligible to a computer program, I needed to rewrite them as an *array* of *arrays*: ten sonnets of fourteen lines each. Once I had this data structure, I had to learn to write programs that would pick a single verse from one of the original sonnets, building upon that to design a program that would generate a pseudo-random sonnet from a 14-character key (the digits of this key would each be between 0 and 9, indicating which of the original 10 sonnets to take the verse from). Finally came the creative part: I had to decide how to produce *pseudo-random* sonnets in a way that would remain faithful to the Oulipian notion of *chance*. In the end, I came up with three different ways[9]: 17

1. **Age:** The reader inputs his/her birthday and the program uses the digits in the birthday to generate the key. The calculations I subject the digits of the birthday to in order to produce the key are invisible to the reader, and have nothing to do with the reader's age. My next goal is therefore to tweak it to calculate exactly how old the reader is at the time the poem is generated so that every minute the reader generates a poem, that poem is different from the previous.
2. **Location:** The reader is directed to a website which gives him/her a latitude and longitude value which he/she then inputs into my program. These numbers are then manipulated to create the 14-digit code to generate a sonnet. In this way, the reader can generate sonnets from not only his or her current location, but any location. The implications of this type of pseudo-random poem are certainly in the spirit of the Oulipo, but once again, the actual calculations to which I subject the data to generate a 14-digit string seems too divorced from the method to be considered truly Oulipian.
3. **UUID:** A UUID is generated from the user's computer, which will produce a new sonnet each time (given that the particular UUID that was enough characters and only digits) is dependent on the time.

Ultimately, my annex is less than satisfying. Not only does the reader of my edition still have no freedom, but he/she is not even allowed into the process. The only person who had any fun in this is me. That said, perhaps *programming* this programmatic text is one viable way to read it. Queneau claims at the outset that his little poetry volume is utterly impossible to read in a human lifetime, but also claims that *only* a computer can appreciate poems generated in this way. Clearly, to have any chance at "reading" the volume, one must read it as a machine. 18

Through programming, I have gained a more thorough understanding of this text and its implications, but I also 19

understand how Queneau's choice of extremely specific subject matter for each of his original poems is the key factor in the hilarity of the potential poems. Each verse of Queneau's original poems is essentially *tagged* with vocabulary linking it to its specific subject. A verse from the first poem that is imbued with an Argentinian vocabulary, for instance, cannot possibly be interchanged with a verse about the leaning tower of Pisa without producing a destabilizing effect. And while certain combinations of these verses may indeed be funnier than others or produce unexpected effects, it is precisely the content of the original poems (that is to say, what would be invisible to a computer reading the text) that produces the surrealist effect of reading the potential poems. While my methods for generating pseudo-random poetry given Queneau's originals may not be satisfying, I can use my program to analyze the text further. I have even adapted my program, replacing Queneau's poems with Shakespearean sonnets that all have more or less the same theme. The resulting combinations are far less jarring than Queneau's original, precisely due to the lack of specific, tagging vocabulary.

In a similar vein, this method can be adapted to interpret and analyze all combinatorial poetry. Poetry, as the Oulipo was well aware, by its fixed forms and propensity for patterns, can be inherently combinatorial. The Oulipo was not the first in noting this property — the Grands Rhétoriqueurs, for instance, were writing similarly expandable poetry as early as the fifteenth century (Jean Molinet's *Sept rondeaux sur un rondeau* [Molinet 1999] is an excellent example). Furthermore, the Oulipo's work has inspired countless other examples of combinatorial poetry generators that can be adapted as I have and subjected to statistical analysis. Computers can help a digital humanist when faced with a volume of combinatorial poetry of which the time to read exceeds a human lifetime, allowing for a mathematical type of "distant reading" that is better equipped to explicate such computer-inspired poetry. 20

Part II: Arborescent Narratives

Un conte à votre façon

Queneau's *Un conte à votre façon* (*A Tale in Your Own Way*) is one of the earliest examples of the choose-your-own-adventure genre, with the added distinction that it was also inspired by computers: "This text is inspired by the presentation of instructions destined for computers or rather programmed teaching. This is an analogous structure to 'tree literature' proposed by F. Le Lionnais at the 79th meeting [of the Oulipo]" [CAVF, 277].[10] The importance of the *tree structure* here owes to mathematics, and more specifically to the field of *graph theory* that was being developed at that time by fellow Oulipian, Claude Berge. Queneau's tale can, incidentally, be represented as a graph. 21

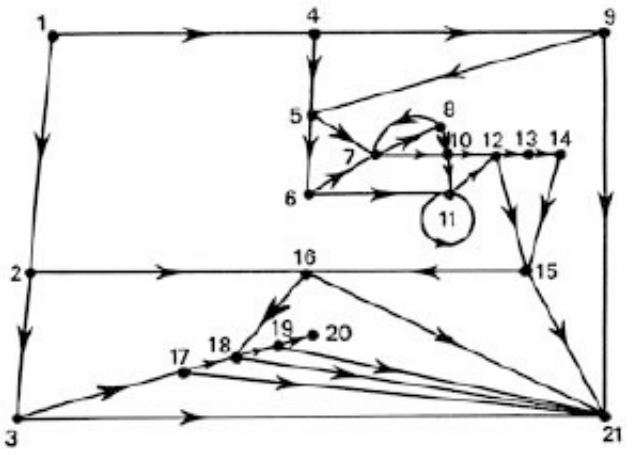

Figure 1 – Graphe du «Conte à votre façon»

Figure 3. Source: http://www.infolipo.org/ambroise/cours/immediat/images/queneau_cavf.pdf

While the graph of this text can be represented in an infinite number of ways, this particular one (created by Queneau) is particularly aesthetic considering the choices he makes: the nodes 1, 2, 3, 4, 9, and 21 frame the rest, as these

determine only expository elements, but not the action of the story; node 20 and not its double, 21 (both of which terminate the program) seems to be Queneau's preferred ending judging from its place within the graph, and not outside the story; in short, Queneau's artistic choices speak volumes about Queneau's own interpretation of his text.

22 Queneau's story initially gives this reader a choice between the story of three little peas, three big skinny beanpoles, or three average mediocre bushes. The choices are all binary, and mostly stark oppositions. For instance, either the reader chooses to read the tale of the three peas or he or she does not. Should the reader prefer not to read this first option, he or she will find that the two alternatives offer meager results. Refusing all three terminates the program almost immediately. If the reader chooses the proper beginning and advances in the tale, the first few choices of the story allow the reader to have a say in descriptive aspects of the story — whether or not the peas dream, what color gloves they wear as they sleep in their pod, and whether or not they roll around on a highway first. In many cases, pointless alternatives are either offered to the reader in an effort to convince him or her of his or her autonomy (the alternative descriptions) or as unsatisfying dead ends (the false beginnings). Regardless of the path the reader takes, there is only one "real" story. There are even two choices that are not choices at all: node 13 asks the reader if he or she wishes to know since when the one little pea has been analyzing dreams, but the alternative takes the reader to the same node, since the answer will not be provided in any case [CAVF, 279]. The following node similarly promises the analysis of the dream, but the alternative proposition affirms that: "If not, go to 15 anyway, because you won't see anything" [CAVF, 279].[11]

23 In short, this *conte* is not at all *à notre façon*. The reader must read a certain subsection of the story as Queneau wrote it, and his/her involvement is superficial. While the story and its various itineraries can be visualized as a flow chart, the text leaves very little freedom to the reader. What I have identified as potential "glitches" could alternatively describe the true nature of this programmatic text. The reader is a computer who, once committed to participation in this program, has no real freedom. The true pleasure comes from the realization that the system, while possible to program on a computer, is fundamentally incompatible with such a design. The illusion of freedom is the defining feature of the text, problematizing the nature of the genre itself.

Oulipo computer experiments

24 Starynkevitch never had the opportunity to program *Un conte à votre façon*, as he worked with the Oulipo several years before its genesis. However, through the A.R.T.A. project at Pompidou, the Oulipo produced a computer program of Queneau's flowchart text programmed in APL (A Programming Language) by Dominique Bourguet. The goal was simple: take a text that was designed with computers in mind and literally put it on a computer in order to facilitate reading. Bourguet's program functioned as follows: "First, the computer 'dialogues' with the reader by proposing the different choices, then edits and cleans up the chosen text without the questions. The pleasure of playing and the pleasure of reading are therefore combined" [Oulipo Atlas, 299].[12]

25 The computer is reductive in terms of reader interaction. While Paul Braffort published one such *conte* produced using this machine in an essay entitled *Prose et Combinatoire* [Oulipo Atlas, 306–318], he conveniently selects the option allowing the peas to dream and the machine has carefully edited out any confusion. Unfortunately, their original program is no longer available, but I would have liked to see how Bourguet chose to "clean up" the discrepancies in the text itself once the reader's tale is produced. Indeed, the genius of Queneau's text lies in the *simultaneous* display of all possible paths, which can then be represented graphically. Should the reader read linearly rather than follow the proposed algorithm, he would note the contradictory nature of much of the short tale. By allowing the reader to see only one step at a time, the mechanics of the system are masked and the reader remains unaware of the mathematical potential of the whole story.

My Program

26 At this point in the project, the *Cent mille milliards de poèmes* annex had taught me some programming basics, but Cliff suggested that I try to find a way to become better initiated with programming. I worked independently through *Learn Python the Hard Way*[13], a free, online, practical introduction to autonomous programming that I highly recommend.

Beginning with basic information about how to use the terminal, followed by a practical introduction to programming, the lessons gradually built upon one another to allow the reader to undertake his/her own projects by the end of the book.

27 At the end of the program, the author had an example where the user could make a program that allowed a reader to move through a “Map” with a pre-established number of scenes. For my first semi-autonomous programming, I adapted that program to make my own interactive edition of *Un conte à votre façon*. Aside from a few basic errors in indentations and spacing, my program was functional (if not particularly elegant), allowing a reader to read through this interactive edition of the text, making choices about which path to follow. As far as the reading experience was concerned, however, the program was unsatisfying. My goal was to add the graphical representation and have it interact with the program somehow.

28 Cliff introduced me to graphviz, an open source *graph* (network) visualization project that is freely available for Python. Given my background in mathematics and my research on *graph theory*, I felt immediately at ease with the way this program operates. In *graph theory*, a *graph* is defined as a plot of nodes and edges. In graphviz as well, in order to make a *graph*, I had to define all requisite nodes and edges. The program then generates a visualization of the graph that is spatially efficient. As an exercise, I made a *graph* of the famous *graph theory* problem of the *Bridges of Königsberg*[14].

29 With this *graph* theoretical program in my Python arsenal, I was able to make my own *graph* of *Un conte à votre façon.* [15] Still not enough, I aimed to integrate the two programs, and Cliff and I decided to give my original program the structure of a *graph* so that it would be able to generate a graph with graphviz at the end. My program now has nodes and edges that correspond with Queneau’s prompts and choices. With this structure, I was able to write a program that enters the *graph* at the first node and, depending on the choice of the user, proceeds through the story, printing out the selected nodes and the corresponding edges. The program records the path the reader takes and at the end, prints out a *graph* with the reader’s path filled in green, to represent the little peas. While my little program does not take into account the full potential of the intersection of *graph theory* and literature as proposed by Queneau’s text, I am very pleased with how it functions. For instance, I can leave to the reader the mathematical exercise of finding the minimum number of read-throughs required to hit every node. While there is still more that can be done, the graph my program generates is itself informative — side by side with the text, the reader can learn more about the potential of this three little peas story.

30 Unlike Queneau's original graph, mine does not depend on my own aesthetic preferences or interpretation of the text. The vertical layout, determined by graphviz's spatial constraints, seems to have — without any knowledge of the content of the nodes — understood something fundamental about the structure of Queneau's tale. This representation of the graph clearly demonstrates that there is only one “true” story, with a few minor diversions that do not change the final outcome. The left-hand side of the graph (nodes 2, 3, and so on) demonstrates a refusal to read the primary tale.

31 This is an odd, almost disconcerting outcome of my project that seems to confirm many criticisms of digital humanities scholarship. If reading the text is unnecessary to its interpretation, then is this not an abject refusal of traditional humanities work? I would argue that this method does not necessarily supersede a traditional close reading, but rather provides a legible visualization of the potential of similar graph theoretical texts. Any choose-your-own-adventure story is composed of nodes and edges which can be programmed — as I have done — using graphviz. Indeed, my program can be adapted very easily to create an interactive edition and graph of any such story. I would encourage digital humanists who wish to try their hand at exploratory programming to apply this method to more complex works of which a traditional close reading might be obscured by the number of nodes and edges.

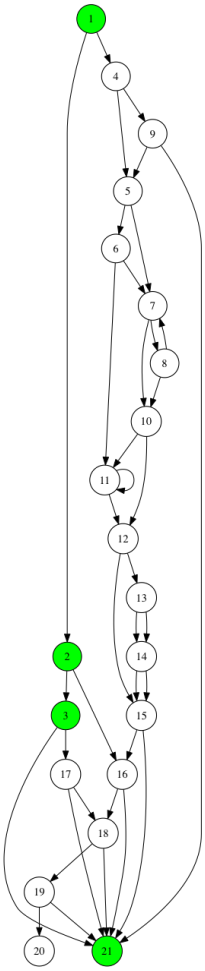

Figure 4.

Part III: Algorithmic Substitutions

Jean Lescure's S+7

The *S+7* method was invented by Jean Lescure and immediately gained popularity in the early Oulipo, most likely due to its precise definition, simplicity of execution, and quick and often hilarious results. Proposed on one of the first Oulipo meetings on 13 February 1961, it enjoys a privileged position as one of the group's first official constraints [Lescure 2014]. A simple algorithmic procedure, the *S+7* begins with a preexisting text, locates all the nouns (S stands for substantif, or noun) and replaces them with the noun that comes seven entries later in a dictionary of the author's choosing. Early Oulipian publications such as the first collected volume, *La Littérature Potentielle*, experimented with many different types of texts to get some sense of the results. 32

Lescure's article detailing the method claims that since the *S+7* operator is a purely mechanical function, the results also depend upon the chosen text and dictionary used [Oulipo La LiPo, 139]. While not immediately humorous on its own, the *S+7* owes its entire effect to the structure of the original text and the nature of the dictionary and it is therefore recognizable syntactic structures that become the most humorous when nouns have been swapped out with others that are alphabetically not too far away. For instance, here are some English-language *S+7* exercises produced by an online generator[16]: 33

1. In the bench Governor created the help and the economist. And the economist was without forum, and void; and day was upon the failure of the deep. And the Spring of Governor moved upon the failure of the weddings. And Governor said, Let there be link: and there was link. (*Genesis* from the Bible)

2. Lolita, light-year of my lifetime, firecracker of my longings. My single-decker, my south. Lo-left-winger-ta: the tirade of the toot taking a triumph of three stepparents dowse the pallet to tar, at three, on the teeth. Lo. Left-winger. Ta. (*Lolita*, Vladimir Nabokov)
3. It was the best of tinctures, it was the worst of tinctures, it was the agitator of witch, it was the agitator of foolishness, it was the equivalent of bellow, it was the equivalent of incredulity, it was the secondary of Lilac, it was the secondary of Darkness… (*A Tale of Two Cities*, Charles Dickens)

These are funny precisely because the original texts are so recognizable. The syntactic structure coupled with a few surprising substitutions in vocabulary derail the reading experience and help us understand that nothing is sacred — not even great literary classics or the Bible. 34

The Oulipo's further work on the *S+7* dealt more with variations on the genre. Mathematically speaking, Le Lionnais pointed out that the *S+7* is a more specific version of *M±n*, in which M represents any taggable part of speech and n represents any integer value [Oulipo La LiPo, 140]. Queneau's *La cimaise et la fraction* is an excellent example of an *M+7* in which adjectives, nouns, and verbs of La Fontaine's timeless fable *La cigale et la fourmi* are substituted accordingly [Oulipo La LiPo, 148]. *Homomorphisms* came next, fixing a specific facet of a preexisting structure and changing everything else. For instance, *homosyntaxismes* reproduce the syntactic structure of a work with a new vocabulary; *homovocalismes* reproduce the same vowels, but change all the consonants; similarly, *homoconsonantismes* freeze the consonants while changing the vowels [Oulipo Atlas, 159–161]. 35

Oulipo Computer Experiments

The Oulipo did experiment with some recognizable texts early on, once again in the context of their collaboration with Starynkevitch. In January 1963, Dmitri Starynkevitch was an invited guest at a meeting. Lescure insisted that he make a program for producing *S+7*'s on the CAB 500, and Starynkevitch spoke of the difficulties that arise when programming *S+7* given the lack of a dictionary. For a human author, applying the *S+7* method is tedious. For a computer of the 1960's, it was virtually impossible. Starynkevitch explains such difficulties: 36

> All the difficulty obviously comes from the amount of material you need — that is, to introduce into the machine. It is currently possible for us to work with a few sonnets of pages of text. (That is what we did for the *Cent mille milliards de poèmes*.) But if we need to perforate an entire dictionary (which is the case of the *S+7*), that is currently impossible. [Bens 2005, 188]

[17]

Eventually, they proposed to correct various issues by hand. By the following meeting, Starynkevitch had programmed the *S+7* method and sent Queneau examples of the program applied to the *Exercices de style*, *Genesis* (they found that God was particularly potential), Shakespeare, and Hugo. In addressing these questions, Starynkevitch had to do a great deal of work by hand. He first had to generate his own small dictionary of nouns (which he chose at random, according to his letter), separate it into masculine and feminine nouns, and then apply the procedure to both genders separately. With issues of plural, he had to make corrections by hand, as well as with exceptions to the grammar rules he had taught to the machine. 37

These early examples were never published, and the *S+7* computer experiments were never mentioned in any publications. My suspicion is that they were displeased with the *S+7* computer program for two main reasons, despite the fact that the procedure itself is ostensibly purely algorithmic and easily applicable to literal computers. First, while early computing was able to produce *S+7*'s, so much was done by hand that the Oulipo very quickly abandoned the idea of using computers to do *S+7*'s. What is the point of automating a procedure if such an automation requires the creation and perforation of a noun dictionary that has no other purpose? Additionally, correcting gender discrepancies that the computer could not understand was likely just as time-consuming as doing the entire thing by hand in the first place. 38

While the A.R.T.A. project focused more on strictly combinatorial productions (where there were a fixed number of 39

elements that could be recombined as the user wished, as in the case of Marcel Bénabou's *aphorisms* [Oulipo Atlas, 311], there would eventually be an amusing computer program produced that functioned along the same lines as the *S+7*. In July 1981, a conversation between Paul Braffort and Jacques Roubaud (a mathematician and poet, another member of the second generation of Oulipians) saw the birth of the *ALAMO* (*Atelier de Littérature Assistée par la Mathématique et les Ordinateurs*) [ALAMO expérience]. One of the group's most successful programs consists of "emptying" Rimbaud's *Le dormeur du val* of its meaningful words (nouns, adjectives, verbs), and replacing them with common words from Baudelaire's lexical vocabulary [ALAMO website]. As *Le dormeur du val* is one of the most famous French-language poems and has a particularly recognizable syntactic structure, even ordinary people who have been through the French school system would find a hybrid poem humorous.

My program

My last digital annex is much more modest in scope, consisting only of an *S+7* generator. Since this is such a canonical Oulipian technique that is often mentioned but very rarely analyzed, my hope was to provide the reader of my chapter 2 with a program to generate his/her own *S+7*'s. In this way, when I claim that well-known texts produce more comedic effects, the reader can generate an *S+7* of a well-known text and a lesser-known one in order to enrich my analysis with more examples. I had also hoped that such a program might be able to provide multiple dictionaries so that the reader can experiment with the effect of different types of dictionaries on the procedure and resulting texts. A final outcome was supposed to be the ability to subtract 7 from previously written *S+7*'s, allowing the reader to confirm whether or not an Oulipian had faithfully produced an *S+7* without cheating. While my chapter 2 includes a diverse and varied selection of texts and procedures, only some of which are substitution-based such as the *S+7*, my annex would aim to provide one small, interactive example of the type of potential literature the early Oulipo was examining. 40

Since my other annexes dealt exclusively with original texts written in French (Queneau's *Cent mille milliards de poèmes* and *Un conte à votre façon*), I programmed them using the original French texts, though I could have easily provided a translated version. Given that the *S+7* is a procedure, however, and that my dissertation committee consists entirely of native English speakers, I allowed myself to circumvent the difficulty of gendered nouns by designing my program in English rather than in French. My next preliminary choice dealt with the texts I wanted to provide to the reader. I chose a set of very canonical English writings: *The Declaration of Independence*, *Genesis*, *The Raven*, *A Tale of Two Cities*, and *Moby Dick*. My final step was to decide which excerpts of each text to include. I decided on the most recognizable parts (generally the incipits) given what I explained above about the *S+7* method. 41

This annex provided an excellent excuse to acquaint myself with *Natural Language Processing* using the NLTK's textbook and programs (https://enaming.com/exclusive/domain/nltk.com/). While I could easily tag the nouns in my chosen texts manually, such laborious work would inevitably produce the same dissatisfaction as early Oulipian computer experiments with Starynkevitch. Nevertheless, my first step had to be to create dictionaries of nouns using nltk. Theoretically, what Cliff and I have envisioned allows the reader to produce dictionaries with specific vocabularies which makes for more interesting S+7 variants. The first noun dictionary I produced came from an online edition of Edgar Allan Poe's complete works. An *S+7* of the *Declaration of Independence* that uses a dictionary of Poe nouns to produce the substitutions will not only be amusing in terms of diction, but will also serve an analytic purpose. The *S+7* can be best understood as a commentary on an original text rather than an original piece of constrained writing. While applying this method to a pre-existing text does indeed produce a new document immediately, the larger implications of the method imply that any text is the *S+7* of another. In other words, all literature is potential. 42

Devising this program has forced me to consider the *S+7* from a critical perspective, rather than a purely inventorial one. While I had initially included the *S+7* as a basic example of an arithmetical Oulipian technique, I now see it as a first attempt in a long line of Oulipian research on productive substitutions. These valuable insights, while not necessarily unique to digital humanities work, were facilitated and brought to light as a direct result of this project. The collaborative nature of the project is one reason: discussions with Cliff Wulfman who is not only a literary scholar, but also a computer programmer, enabled both of us to understand this procedure simultaneously as inspired by early computers and algorithmic methods. 43

Studying and implementing natural language processing with nltk allowed me to better understand the Oulipian mindset that spawned these early procedures, elucidating the historical context of why the group ultimately abandoned these efforts. The early Oulipo, inspired by the potential of computer programming, demonstrated an algorithmic approach to literature in the early constraints the members produced. My programming experiences helped me to understand the essential difference between these types of algorithmic procedures and the work that the group eventually settled on following the members' dissatisfied responses to the computer experiments of the 1960s and 1970s, a clear preference for abstract mathematical thought — patterns and structure — rather than the procedural tendencies of applied mathematics. 44

Finally, the creativity required to carry out such a project brings a new freshness to my research and analysis that I hope will enable it to say something truly new and interesting about what appears to be a trivial, superficial method meant to produce one-off silly results. While in this case, my code is likely not adaptable for other digital humanities projects, I do believe that future scholars can benefit from a greater understanding of nltk in order to learn more about other algorithmic production that coincided with the development of computational linguistics and early humanities computing in the 1960s and 1970s. An understanding of nltk has the additional benefit of promoting a certain kind of Oulipian creativity in literary analysis, allowing a digital humanities scholar to invent a code that can analyze texts, completing this fascinating loop of research and invention. 45

Conclusions

While the official goal of this project had to be to produce a product, the *real* product has been the important insights I have gained from the experience. In a sense, the product was a pretext in order to take the idea of the Oulipo and in a quasi-Oulipian fashion, think about formalizing it on computers. From learning to program in Python (even though Python is not the language early Oulipians used in their computer endeavors), I have gained a greater understanding of early Oulipian techniques and why the Oulipo abandoned computer programming relatively quickly and relegated this sort of activity to the *ALAMO*. I also have a greater appreciation for how exactly to use digital humanities tools to explore new avenues in these texts that were created with algorithmic procedures in mind. 46

That said, as with Oulipian computer-texts that lost their interest when programmed onto a computer, digital tools are best when properly understood and critically implemented, at least with regards to Oulipian work. Creating a computer program of Queneau's *Un conte à votre façon* alone is not necessarily a critical act, as it tends to obscure the potential of the constraint from the user of the program. However, using a computer to find more substantial intersections between the graphical representation of *Un conte à votre façon* and the text itself is an Oulipian endeavor. Being an Oulipian reader means playing a game set up by an Oulipian author, and they're a clever bunch. If you don't want to be trapped forever in the labyrinth they created, then you need to think like they do. For computer texts, that means learning to think like a computer, and understanding wherein lies the potential. 47

Beyond the narrow use of digital humanities methods in Oulipo studies, this experience has convinced me that the broad nature of the digital humanities and how the field defines itself is productive. Indeed, the nebulous nature of definitions of the digital humanities allows for a great variety in approaches and can foster creativity. Especially for literary studies, such creative approaches are perhaps most appropriate. 48

Notes

[1] I am greatly indebted to Cliff Wulfman, who is no longer at the Center for Digital Humanities but who was a staff member during my project. His advice was invaluable, his recommendations of beginner's Python courses were always useful, his technical expertise was unmatched. But most importantly, I have gained immeasurable insights from our conversations surrounding the creation of these annexes, which I will discuss in the remainder of this paper.

[2] I ultimately abandoned the idea of making an annex for my first chapter on Set Theory, as none of the set theoretic constraints discussed in my chapter would have been particularly engaging on a computer. Furthermore, the main purpose of my first chapter was a historical analysis of the intellectual inheritance of the mathematical field of set theory and not close readings of Oulipian constraints. Additionally, in this paper I will not discuss my fifth annex, which took the form of a Tinderbox hypertext, based on the geometric representation of the table of contents of Italo

Calvino's *Le città invisibili*. While productive in its own way, this annex is fundamentally different from the other three, given its creation using outside software and its purpose as a research and note-taking tool, rather than as an interactive, digital edition of the text itself.

[3] Original French: "Seule une machine peut lire un sonnet écrit par une autre machine."

[4] The 1999 computing initiative was less an experiment in exploratory programming than it was an editorial and commercial enterprise, resulting in the sale of a CD-ROM. While the nature of the programmed texts in the CD-ROM share certain commonalities with the previous experiments, I will not include this in my analysis.

[5] Original French: "On souhaita que M. Starynkevitch nous précise la méthode utilisée ; on espéra que le choix des vers ne fut pas laissé au hasard."

[6] Original French: "un possible accord entre l'informatique et la création littéraire."

[7] Original French: "Le recueil imprimé est très joliment conçu mais la manipulation des languettes sur lesquelles chaque vers est imprimé est parfois délicate."

[8] Including (but not limited to): Bev Rowe's digital edition including original translations with excellent footnotes (http://www.bevrowe.info/Queneau/QueneauHome_v2.html); and Gordon Dow's less impressive version (http://www.growndodo.com/wordplay/oulipo/10%5E14sonnets.html), but with the additional feature of displaying the number of sonnets generated at this site (currently far below the total 10^{14}).

[9] https://github.com/Princeton-CDH/digital-oulipo

[10] Original French: "Ce texte…s'inspire de la présentation des instructions destinées aux ordinateurs ou bien encore de l'enseignement programmé. C'est une structure analogue à la littérature « en arbre » proposée par F. Le Lionnais à la 79e réunion."

[11] Original French: "si non, passez également à 15, car vous ne verrez rien."

[12] Original French: "L'ordinateur, dans un premier temps, 'dialogue' avec le lecteur en lui proposant les divers choix, puis dans un second temps, édite 'au propre' et sans les questions, le texte choisi. Le plaisir de jouer et le plaisir de lire se trouvent donc combinés."

[13] Zed Shaw, *Learn Python the Hard Way*. (http://learnpythonthehardway.org)

[14] "The seven bridges of Königsberg" is a famous historical problem in mathematics and simple introduction to the basics of graph theory. Let us say you are a tourist in the Prussian city of Königsberg (now Kaliningrad, Russia) and want to see the whole city, which is intersected by the Pregel River, containing two large islands. Seven bridges connect these various positions: two leading from the left bank to the first island, two more leading from the first island to the right bank, one between the two islands, and one each between the second island and each of the banks. Is it possible to devise a walk through the city that crosses each bridge once and only once? Leonhard Euler solved the problem in 1736, laying the foundations of graph theory in the process.

[15] See next image. Note that in my graph, a path has been highlighted in green. This is one possible path through the story, highlighted on the graph by the program at the close. In fact, this path is the "lazy" one, in which the reader refuses to read the story of the three little peas, the tall skinny bean poles, and the average mediocre bushes.

[16] Source: http://www.spoonbill.org/n+7/. These examples exhibit simple flaws caused by imperfect parts of speech tagging.

[17] Original French: "Toute la difficulté vient, évidemment, de la quantité de matériau dont vous avez besoin — donc : à introduire en machine. Il nous est possible de travailler sur quelques sonnets ou quelques pages de texte. (C'est ce que nous avons fait pour les *Cent mille milliards de poèmes*.) Mais s'il faut perforer tout un dictionnaire (c'est le cas des S + 7), cela nous est actuellement impossible."

Works Cited

ALAMO expérience Braffort, Paul. 2007. *ALAMO, Une expérience de douze ans.* December 9. Accessed June 21, 2016. http://www.paulbraffort.net/litterature/alamo/alamo_experience.html.

ALAMO website ALAMO. *Historique.*Accessed June 21, 2016. http://www.alamo.free.fr/pmwiki.php?n=Alamo.Historique.

Bens 2005 Bens, Jacques. 2005. *Genèse de l'Oulipo 1960-1963.* Paris: Le Castor Astral.

Bourbaki 1939 Bourbaki, Nicolas. 1939. *Éléments de mathématique.* Vol. 1. Paris: Éditions Hermann.

CAB 500 picture Nantan, G. 2013. *Calculatrice électronique CAB 500.* March 30. Accessed June 21, 2016. http://www.histoireinform.com/Histoire/+infos2/chr4infg.htm.

CAVF Queneau, Raymond. 1973. "Un conte à votre façon." In *La littérature potentielle*, by Oulipo, 277-280. Paris: Gallimard.

CMMP Queneau, Raymond. 1961. *Cent mille milliards de poèmes.* Paris: Gallimard.

Dow 2009 Dow, Gordon. 2009. *100,000,000,000,000 Sonnets.* May 6. Accessed June 21, 2016. http://www.growndodo.com/wordplay/oulipo/10%5E14sonnets.html.

Gauthier 2009 Gauthier, Joëlle. 2009. *Machines à écrire.* November 11. Accessed June 21, 2016. http://nt2.uqam.ca/en/repertoire/machines-ecrire-0.

Hodges 2012 Hodges, Andrew. 2012. *Alan Turing: The Enigma.* Princeton: Princeton University Press.

Lescure 2014 Lescure, Jean. 2014. *S+7.* Accessed June 21, 2016. http://oulipo.net/fr/contraintes/s7.

Molinet 1999 Molinet, Jean. “Sept rondeaux sur un rondeau.” *Anthologie de la poésie française du XVIe siécle*. Rookwood Press, 1999. 16. Print.

Montfort 2015 Montfort, Nick. 2016. *Exploratory Programming in Digital Humanities Pedagogy and Research* in *A New Companion to Digital Humanities*West Sussex: John Wiley and Sons.

Montfort 2016 Montfort, Nick. 2016. *Exploratory Programming for the Arts and Humanities* Cambridge: The MIT Press.

Online CAVF Queneau, Raymond. 2000. *Un conte à votre façon.* November. Accessed January 16, 2013. http://www.infolipo.org/ambroise/cours/immediat/images/queneau_cavf.pdf.

Online CMMP Queneau, Raymond. 2012. *Raymond Queneau — Cent mille milliards de poèmes, 1961.* August 17. Accessed June 21, 2016. http://openset.nl/blog/?p=192.

Oulipo Atlas Oulipo. 1981. *Atlas de littérature potentielle.* Paris: Gallimard.

Oulipo La LiPo Oulipo. 1973. *La littérature potentielle.* Paris: Gallimard.

Propp Morphologie Propp, Vladimir. 1965; 1970. *Morphologie du conte.* Paris: Éditions du Seuil.

Ramsay 2011 Ramsay, Stephen. 2011. *Reading Machines: Toward an Algorithmic Criticism.* Urbana: University of Illinois Press.

Rimbaudelaires ALAMO. *Rimbaudelaires.* Accessed June 21, 2016. http://www.alamo.free.fr/pmwiki.php?n=Programmes.Rimbaudelaires.

Rowe Online CMMP Rowe, Beverley Charles. 2012. *Queneau.* November 26. Accessed June 21, 2016. http://www.growndodo.com/wordplay/oulipo/10%5E14sonnets.html.

S+7 Christian, Peter. *The N+7 Machine.* Accessed June 21, 2016. http://www.spoonbill.org/n+7/.

Starynkevitch article Starynkevitch, Dmitri. 1990. "The SEA CAB 500 Computer." *Annals of the History of Computing* (American Federation of Information Processing Societies) 12 (1): 23-29.

Têtes folles picture Trier, Walter. 2015. *Têtes folles.* October 8. Accessed June 21, 2016. http://www.legrenierdepascal.com/2015/10/8192-tetes-folles.html.

Wolff 2007 Wolff, Mark. 2007. *Reading Potential: The Oulipo and the Meaning of Algorithms*Digital Humanities Quarterly 1.1

lpthw Shaw, Zed. 2015. *Learn Python the Hard Way.* July 19. Accessed June 21, 2016. http://learnpythonthehardway.org.