May 2020

Computational Configurations: Behind the Bricolage of *The Truelist*

Emma Jayne Stanley
Clemson University, ejaynestanley@gmail.com

COMPUTATIONAL CONFIGURATIONS: BEHIND THE BRICOLAGE OF *THE TRUELIST*

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Arts
English

---

by
Emma Jayne Stanley
May 2020

ABSTRACT

Nick Montfort's book of computational poetry, *The Truelist,* allows the reader to understand the code written by Montfort to produce the book as both "bricoleur" and "bricolage." At the core of my examination, I will ask: how does computational poetry, specifically that of *The Truelist,* provide us with a new definition of what it means to embody a coded space, specifically a space that, according to Montfort, is constantly redefining itself for new readers? To focus specifically on examining the literariness of the text, I will primarily use Jacques Derrida's concept of the "bricolage" from "Structure, Sign, and Play in the Discourse of the Human Sciences" (specifically focusing on the output of the code) in an attempt to understand how Montfort's poetry-producing computations redefine poetry's embodiment of space and place, and how the code itself becomes a bricoleur, particularly with regard to associative imagery and how this computational "communication" forbids the reader to engage passively with the text and its potential variations.

DEDICATION

To my loving parents, Jayne and James Stanley. Thank you for always supporting me in my endeavors, and for constantly reminding me that my true intelligence comes from God. I love you.

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my thesis committee. Dr. Haynes, Dr. Hunter, and Dr. Hooley, I am thankful for each of you, and all of your help and encouragement.

Additionally, I would like to thank the professors from my undergraduate institution, Piedmont College, for their continued support throughout my graduate career at Clemson University. I wouldn't be the academic I am today without you all, so I am infinitely grateful for you—especially Dr. Stephen Whited, Dr. Hugh Davis, Reverend Tim Garvin-Leighton, and Dr. Timothy O'Keefe.

Also, I would like to thank my dear friends for their endless support. Matthew Stapleton, Kaitlyn Samons, Sarah Watkins, Natalie Freel, Young Joo Choi, Hee Joo Choi, and Vivianne Zsoldos: I love you all, and thank you all for believing in me when I couldn't believe in myself.

TABLE OF CONTENTS

I. Introduction

Computational poetry distinguishes itself as a form of literature but exists outside of what one considers "traditional" or "print" literature. There are forms of computational poetry one can interact with in an interactive fiction-like structure, but there are also poems produced solely by input/output from a code that then become fixed in one state once printed. In Nick Montfort's *The Truelist,* the reader receives (in a book format) the output of a code developed by Montfort, which has created a book-length poem for the reader to explore. According to Montfort, "the poem invites the reader to imagine moving through a strange landscape that seems to arise from the English language itself," and "the unusual compounds are open to being understood differently by each reader, given that person's culture and individual background." The poem's outreach to the reader's cognitive abilities and imagination creates a new experience for each reader of the text—especially since the text seems to create moments of nonlinearity within its constant juxtaposition of seemingly unlikely (yet mostly concrete) associative images.

At the core of my examination, I will ask: how does computational poetry, specifically that of *The Truelist,* provide us with a new definition of what it means to embody a coded space, specifically a space that, according to Montfort, is constantly redefining itself for new readers? To focus specifically on examining the literariness of the text, I will primarily use Jacques Derrida's concept of the "bricolage" from

"Structure, Sign, and Play in the Discourse of the Human Sciences" (specifically focusing on the output of the code) in an attempt to understand how Montfort's poetry-producing computations redefine poetry's embodiment of space and place, and how the code itself becomes a bricoleur, particularly with regard to associative imagery and how this computational "communication" forbids the reader to engage passively with the text and its potential variations. Montfort's coded space perfectly embodies Derrida's definition of bricolage and the bricoleur. Other Derridean concepts I will use include aspects from "Signature Event Context," including the "trace." Additionally, I will examine specific passages of Montfort's text, focusing on literary devices which prove helpful in my analysis/argument for a Derridean interpretation of how the poem redefines (or finds a new way to communicate the idea/discourse of) space and place where the code itself functions as a creator of bricolage.

Working largely with associative imagery, Montfort's code uses various concrete words to create new interactions with language. While associative imagery plays a large role in print poetry, especially with more abstract pieces of work (for example, modernist literature), the need for association in computational literature is even stronger. Associative imagery allows the reader to create new connotations or connections between the images in the work (along with one's associations with background knowledge known prior to one's engagement with the work), so Montfort's claim that one's engagement creates a new literary landscape through imagination perpetuates this notion within the seemingly random output of the code.

It might seem like a rather obvious assertion that one's background creates a new engagement with the text; however, this is a particularly interesting engagement with computational poetry. Authors of print literature, as well as computational literature, have focused their diction and syntax on creating particular images, but with computational literature, the poem is produced as the output of a code that has created these sequences. But this does not take away from the literary merit of the text—as a print author would choose a certain syntactic engagement with the image, Montfort has accomplished a similar task: creating a code with deterministic sequences that create a seemingly random structure, but uses specific words (chosen by him and input into the code) and sentence patterns (also written in the code) to produce the book-length poem as the output. This output, once printed in his book, gives us opportunities for association within this computational space—or, more specifically, the bricolage.

Holistically, my paper will provide context for computational poetry (how it was first formed, and where it is currently) define the importance of examining computational poetry with the same merit as print literature (structurally and contextually), and specifically focus on Montfort's text for the formation of my argument that computational poetry is both bricoleur and bricolage simultaneously. I will explicate how and why Montfort's text is a significant piece of computational literature, particularly one that redefines the communication/engagement with major facets of literary devices and techniques within computational poetry that comprise the lived experience of the bricolage of coded space.

II. Framework: The Logic of the Bricoleur

A deep dive into Derrida's understanding and synthesis of "bricoleur" and "bricolage" requires backtracking into the infrastructure created by Claude Levi-Strauss, which is explicated in Levi-Strauss's "Science of the Concrete" in *The Savage Mind*. At the core of Levi-Strauss's argument, we are essentially posed with an overarching question: what is myth? This avenue of inquiry requires us to evaluate many of the aspects of how we understand the world around us, and the kind of language we use to discuss it. While "Science of the Concrete" poses many significant points, we should analyze the following statements in order to begin a clear articulation of the framework I will be using to understand the dynamic of Montfort's work. Levi-Strauss asserts:

> "Myths and rites are far from being, as had often been held, the product of man's 'myth-making faculty,' turning its back on reality. Their principle value is indeed to preserve until the present time which remains of methods of observation and reflection which were (and no doubt still are) precisely adapted to discoveries of a certain type: those which nature authorized from the starting point of a speculative organization and exploitation of the sensible world in sensible terms. The science of the concrete was necessarily restricted by its essence to results other than those destined to be achieved by the exact natural sciences but it was no less scientific

> and its results no less genuine. They were secured ten thousand years earlier and still remain at the basis of our own civilization” (Levi-Strauss 10-11).

Levi-Strauss’s concept of “myth” helps Derrida understand how to synthesize what exactly is true, or logical, in the world-- or perhaps, what isn’t, what doesn’t have to be, and why. Specifically, in “Structure, Sign, and Play,” Derrida asserts that bricolage is a “mythopoetic function” (“Structure, Sign, and Play” 287). Derrida comes to this conclusion by noting a stance espoused by Levi-Strauss:

> There is no unity or absolute source of the myth. The focus or the source of the myth are always shadows and virtualities, which are elusive, unactualizable, and nonexistent in the first place. Everything begins with structure, configuration, or relationship. The discourse on the acentric structure that myth itself is, cannot itself have an absolute subject or an absolute center. It must avoid the violence that consists in centering a language which describes an acentric structure if it is not to shortchange the form and movement of myth. (“Structure, Sign, and Play” 286).

To tie together these concepts, bricolage is a mythopoetic function, according to Derrida, because “mythological discourse… must itself be *mythomorphic*” (“Structure, Sign, and Play” 286). So, in summary: if code = bricoleur, then bricoleur creates bricolage. The bricolage, for the Montfort text, is the associative imagery, or more specifically, the “myth”-- the undefined product, the object produced by what is available and what is known, the random series of images. Therefore, the bricoleur and bricolage are both mythopoetic, and enact the very thing that they are.

III. Contextualizing Computations

Although my paper will be focusing on Nick Montfort's *The Truelist*, providing context for both the history and present functionality of computational literature is imperative in one's engagement with understanding the transformative elements of the genre. Computational poetry has many forms, and Alison Knowles and James Tenney's 1967 poem "A House of Dust" is representative of one of the earliest computational poems, and actually was the inspiration for the *Using Electricity* collection of computational poetry books, which includes *The Truelist*. Here is the code that produces the poem with important aspects bolded for explication:

```
material = ['SAND', 'DUST', 'LEAVES', 'PAPER', 'TIN', 'ROOTS', 'BRICK', 'STONE',
'DISCARDED CLOTHING', 'GLASS', 'STEEL', 'PLASTIC', 'MUD', 'BROKEN
DISHES', 'WOOD', 'STRAW', 'WEEDS']

location = ['IN A GREEN, MOSSY TERRAIN', 'IN AN OVERPOPULATED AREA',
'BY THE SEA', 'BY AN ABANDONED LAKE', 'IN A DESERTED FACTORY', 'IN
DENSE WOODS', 'IN JAPAN', 'AMONG SMALL HILLS', 'IN SOUTHERN FRANCE',
'AMONG HIGH MOUNTAINS', 'ON AN ISLAND', 'IN A COLD, WINDY CLIMATE',
'IN A PLACE WITH BOTH HEAVY RAIN AND BRIGHT SUN', 'IN A DESERTED
AIRPORT', 'IN A HOT CLIMATE', 'INSIDE A MOUNTAIN', 'ON THE SEA', 'IN
MICHIGAN', 'IN HEAVY JUNGLE UNDERGROWTH', 'BY A RIVER', 'AMONG
OTHER HOUSES', 'IN A DESERTED CHURCH', 'IN A METROPOLIS',
'UNDERWATER']

light_source = ['CANDLES', 'ALL AVAILABLE LIGHTING', 'ELECTRICITY',
'NATURAL LIGHT']

inhabitants = ['PEOPLE WHO SLEEP VERY LITTLE', 'VEGETARIANS', 'HORSES
AND BIRDS', 'PEOPLE SPEAKING MANY LANGUAGES WEARING LITTLE OR
NO CLOTHING', 'ALL RACES OF MEN REPRESENTED WEARING
PREDOMINANTLY RED CLOTHING', 'CHILDREN AND OLD PEOPLE',
'VARIOUS BIRDS AND FISH', 'LOVERS', 'PEOPLE WHO ENJOY EATING
TOGETHER', 'PEOPLE WHO EAT A GREAT DEAL', 'COLLECTORS OF ALL
TYPES', 'FRIENDS AND ENEMIES', 'PEOPLE WHO SLEEP ALMOST ALL THE
TIME', 'VERY TALL PEOPLE', 'AMERICAN INDIANS', 'LITTLE BOYS', 'PEOPLE
FROM MANY WALKS OF LIFE', 'NEGROS WEARING ALL COLORS', 'FRIENDS',
'FRENCH AND GERMAN SPEAKING PEOPLE', 'FISHERMEN AND FAMILIES',
'PEOPLE WHO LOVE TO READ']

print('')
print('A HOUSE OF ' + choice(material))
print('      ' + choice(location))
print('            USING ' + choice(light_source))
print('                  INHABITED BY ' + choice(inhabitants))
print('')
```

The code is essentially functioning as a blueprint for the syntax and the variables of words able to be used within the code. The code knows that it's creating a house of material + location + light source + inhabitants. So, we could easily get a house of plastic in a green, mossy terrain, using candles, and inhabited by people who eat a great deal. There are certainly some interesting—and potentially problematic in a modern perspective—"inhabitant" options, especially with regard to referring to indigenous people as "American Indians."

Here are some examples that were produced by the code:

```
A HOUSE OF STRAW

          UNDERWATER

                    USING CANDLES

                              INHABITED BY PEOPLE SPEAKING MANY

                                 LANGUAGES WEARING LITTLE OR NO CLOTHING

A HOUSE OF PAPER

          IN AN OVERPOPULATED AREA

                    USING ALL AVAILABLE LIGHTING
```

INHABITED BY ALL RACES OF MEN REPRESENTED

WEARING PREDOMINANTLY RED CLOTHING

A HOUSE OF STONE

IN A HEAVY JUNGLE UNDERGROWTH

USING ELECTRICITY

INHABITED BY VEGETARIANS

(Knowles and Tenney)

An even earlier example, potentially even the first, is the 1952 computational poem by Christopher Strachy, titled “Love Letters,” produced on the Ferranti Mark 1. Nick Montfort has reproduced the computational poem on his website, providing readers with access to the code, while noting that he found the original source for the code from photographs of Christopher Strachy’s notes (“Nickm.com”). Here is a segment of the code for analysis, with the important categories bolded:

```
first = ['DARLING', 'DEAR', 'HONEY', 'JEWEL']
second = ['DUCK', 'LOVE', 'MOPPET', 'SWEETHEART']
adjectives = ['ADORABLE', 'AFFECTIONATE', 'AMOROUS', 'ANXIOUS',
'ARDENT', 'AVID', 'BREATHLESS', 'BURNING', 'COVETOUS', 'CRAVING',
'CURIOUS', 'DARLING', 'DEAR', 'DEVOTED', 'EAGER', 'EROTIC', 'FERVENT',
'FOND', 'IMPATIENT', 'KEEN',]
nouns = ['ADORATION', 'AFFECTION', 'AMBITION', 'APPETITE', 'ARDOUR',
'CHARM', 'DESIRE', 'DEVOTION', 'EAGERNESS', 'ENCHANTMENT',
'ENTHUSIASM', 'FANCY', 'FELLOW FEELING', 'FERVOUR', 'FONDNESS',
'HEART', 'HUNGER', 'INFATUATION', 'LIKING', 'LONGING', 'LOVE', 'LUST',
'PASSION', 'RAPTURE', 'SYMPATHY']
adverbs = ['AFFECTIONATELY', 'ANXIOUSLY', 'ARDENTLY', 'AVIDLY',
'BEAUTIFULLY', 'BREATHLESSLY', 'BURNINGLY', 'COVETOUSLY',
'CURIOUSLY', 'DEVOTEDLY', 'EAGERLY', 'FERVENTLY', 'FONDLY',
```

'IMPATIENTLY', 'KEENLY', 'LOVINGLY', 'PASSIONATELY', 'SEDUCTIVELY', 'TENDERLY', 'WINNINGLY', 'WISTFULLY']
verbs = ['ADORES', 'ATTRACTS', 'CARES FOR', 'CHERISHES', 'CLINGS TO', 'DESIRES','HOLDS DEAR', 'HOPES FOR', 'HUNGERS FOR', 'IS WEDDED TO', 'LIKES', 'LONGS FOR', 'LOVES', 'LUSTS AFTER', 'PANTS FOR', 'PINES FOR', 'PRIZES', 'SIGHS FOR', 'TEMPTS', 'THIRSTS FOR', 'TREASURES', 'WANTS', 'WISHES', 'WOOS', 'YEARNS FOR']

As one can see, the code is quite literally producing a love letter. It begins with a word from the first line, titled "first =", followed by an adjective from the list, then a noun, an adverb, and a verb, resolving itself into a love letter. Here's an example of a rather strange output the code produced… one of it's so-called "Love Letters" out of its endless stream of letters:

> "JEWEL DUCK,
>
> YOU ARE MY AFFECTIONATE FERVOUR. MY FONDNESS WINNINGLY THIRSTS FOR YOUR DARLING RAPTURE. YOU ARE MY AVID LUST: MY AMOROUS ENCHANTMENT. MY DEVOTED RAPTURE PRIZES YOUR CURIOUS TENDERNESS.
>
> YOURS EAGERLY,
>
> M.U.C" (Strachy)

Not limited to poetry, computational literature has also explored prose works, which range from codes that produce both a prosework output, but also interactive fiction games. Starting in 1977, programs like Tale-Spin continued to pave the way for the computational intricacies of Monfort's text. James Meehan, in his attempt to create a program that "writes stories by using knowledge about problem solving, physical space, interpersonal relationships, character traits, bodily needs, story structure, and English" that "simulates rational behavior" (Meehan 91). However, much unlike computational poetry, Meehan actually coded this to be an unfixed world where the code itself could make decisions about the plot regardless of the variables, in what was essentially a completely randomized, non-deterministic state. Meehan noted that sometimes the code needed to learn more about certain subjects and their parts of speech within the variables, as this was once an output:

"Henry Ant was thirsty. He walked over to the river bank where his good friend Bill Bird was sitting. Henry slipped and fell in the river. Gravity [having neither legs, wings, nor friends] drowned." (Meehan 92)

Meehan then asserted that "[i]t didn't know enough about gravity, obviously" (Meehan 92). Although the code used to make Meehan's poem, which was:

```
X tries to move Y to Z
Preconditions:
X is self-moveable
If X is different from Y,
Then DPROX (X, X, Y)
And DO-GRASP (X, Y)
DKNOW (X, where is Z?)
DKNOW (X, where is X?)
DLINK (X, loc (Z))
Act: DO-PTRANS (X, Y, loc (Z))
Postcondition: Is Y really at Z?
Postact: If X is different from Y,
Then DO-NEG-GRASP (X, Y) (Meehan 93)
```

was structurally sound, the code did often produce an illogical statement unless it had the correct amount of knowledge, which is unlike computational poetry where the goal is to produce a seemingly illogical, associative statement-- at least for Montfort's text.

Obviously there are changes in imagery when one encounters the non-linearity of a computational poem, especially when the text is able to be edited. But is this change true for the semantics of the line? Scott Rettberg speaks to this concept in *Electronic Literature* with regard to a text generator called *Travesty* developed in 1984 by Hugh Kenner and Joseph O'Rourke. In *Travesty*, the interactor enters a "source text" into the program, and the program analyzes the semantics and "linguistic probability" of the text using n-grams, and uses this input to create an output of text that follows similar semantic and linguistic patterns as the inputted text (Rettberg, "Combinatory Poetics" 38). Some interactive fictions require that the user inputs a certain phrase by typing—for example, "go left" to get the player character to move to the left down a new pathway, which opens up a new set of pathways. Other interactive fiction games offer two or more clickable options, where the interactor does not have to input text.

Although a text generator functions in a different way (e.g., the interactor inputs language into the program and an output is created based upon the syntactic structure) than most interactive fictions, the concept is still the same as computational poetry. There is still a sensibility of having non-linearity, and an upholding of the structure of the original text. *Travesty* doesn't create a "travesty" of language if it upholds the structure—perhaps the same could be said about clickable interactive fiction if it upholds the structure of the original text, or edited forms of a computational poem. Although the imagery and point of entry is altered in a digital platform, the text is still accessible—but in a way that requires an engagement from the individual who codes the computational poem in order to imagine new realities (by potentially changing the variables in the code).

Through asking these questions of computational poetry, we are exposed to new potential truths, and new potential logics (that we perhaps learned in hindsight from a certain series of associations with images in a poem that's predicated on associative imagery). Through our engagement with the text and our engagement with new forms of logic (or the "trace"), communication is defined and redefined for us as we continue as viewers of the computational output. The acknowledgment of the discovery of new truths is integral in one's engagement with associative imagery within the output of the code. Without acknowledging the impact of one's time spent working through the series of images, the impact of computational poetry is not fully realized. To glean the knowledge that computational poetry gives us, one must look inside of the text, experience it first-hand, experience the trace, and understand the parameters of communication. Through understanding and engaging with these aspects in computational poetry, we then understand the functionality of the poem through an avenue of inquiry that leads us down new pathways of truth within the logic puzzle.

IV. Code as Bricoleur: Analyzing *The Truelist*

Understanding the functionality of electronic literature, particularly computational poetry, is predicated on our synthesis of logic. Whether the genesis of our logical deduction derives from a sense of emotional subjectivity, or solely a set of objective facts (although it is often difficult to distinguish between pure logic and emotional influences), we complete various processes on a regular basis without being cognizant of our actions. For instance, when one is first learning to drive a car, one learns how to solve the proper "puzzle" for the objective of learning how to start the car, shift gears, and drive safely. This becomes a natural process for us at some point—one that we do not have to think about each time we do it. However, our emotions have the capacity to impact the logic puzzle, subverting the qualities of the objective that once seemed so logical, so easy to us.

Consider a moment in your life when you acted on emotional impulse. We often alter our responses to a logical premise when we encounter problematic situations. Or, perhaps, not simply a problematic situation, but a situation in which the logical outcome is shattered. However, during our emotional responses, we glean knowledge—perhaps not in the moment, but in hindsight. We learn some of what is "logical" from our engagement with the "illogical," or "*non-linearity*," which I define as: *the place of subversion from the initial logical deduction; the place that exists outside of a defined context or lens through which we are viewing*. In other words, we learn what is logical from the absence of logic—the absent-present—when what is *absent* is *objectivity* and what is *present* is the *realization* of the lack of objectivity.

Within this liminality of logic existing within the absent-present, we find what Jacques Derrida refers to as the "trace." In her translation of Derrida's *Of Grammatology*, Gayatri Chakravorty Spivak asserts that "Derrida suggests that what opens the possibility of thought is not merely the question of being, but also the never-annulled difference from 'the completely other.'" Spivak continues, noting that "[t]he structure of the sign is determined by the trace or track of that other which is forever absent" and "[t]his other is of course never to be found in its full being" as the "trace is the mark of the absence of a presence, an always already absent present, of the lack of the origin that is the condition of thought and experience" (Spivak xxxv-xxxvi).

When one considers this theoretical application to electronic literature, specifically computational poetry, the application helps us understand and deconstruct the logic puzzle. Throughout the many forms of computational media, computational poetry is the epitome of where the engagement with what I defined as "*non-linearity*" takes place. Since a lack of linearity exists, especially when reading Nick Montfort's *The Truelist*, the associative imagery begins immediately upon the beginning of the first section of the book-length poem. Montfort's code has become the bricoleur, providing us with a bricolage of words:

Now they saw the foothills,

And the airking,

The earthworm,

The sliphound exceeding the king,

The heartwoman,

The shiphound,

The hardpath river leading the ship,

The traplight welcoming the work,... (*The Truelist* 1).

I will further explicate the coding techniques later in my analysis, but it is important to note here that we are seeing a deterministic sequence. This means that the code, given the parameter and variables, created a specific outcome that would happen each time the code runs unless someone were to change the variables given to the code. While this might seem problematic in the sense of the code being a bricoleur creating bricolage, I argue that Monfort could have fed the bricoleur any series of words for the bricolage, but the code, using what it had, created a poem full of associative imagery. This will be explicated further in my analysis of the second section of the poem.

Given that Montfort has done a significant amount of research not only in computational poetry, but also interactive fiction, it is important to consider that there are significant portions of interactive fiction that might fall under the same category, especially since the coding infrastructure is quite similar, except that the interactive fiction coding allows for direct engagement from the reader by inputting a word, or clicking an option for a pathway. Given that interactive fiction has many options for pathways, the reader/spectator, or what Nick Montfort defines in “Fretting the Player Character” as “the interactor” who engages with the “player character,” has the ability to engage with a storyline within the non-linearity (Montfort, “Fretting the Player Character” 139). However, Montfort argues that the “interactor does not really play the player character, however generic or specific the character is” since interactive fiction lies more in the realm of “putting on the character as a pair of eyes" rather than engaging in a dramatic display such as one does in a game such as *Dungeons & Dragons* (*D&D*). (Monfort, “Fretting the Player Character” 139). Although *Dungeons & Dragons* is primarily played in person (meaning that the “interactor” would be engaging with another human rather than solely an interactive digital platform), the argument that it requires a more “dramatic” taking on of a persona is understandable due to the “multi-party role-playing contexts”—but, Montfort offers an additional argument, claiming that even “multi-user online environments” require a higher level of engagement with the interactor/player character dynamic than is available in interactive fiction (139). In the case of interactive fiction, and perhaps also *Dungeons & Dragons*, the “bricoleur” seems to most apply to the human, rather than the code, as the interactor is required to take the smallest pieces of information and build upon them to create something. For interactive fiction, perhaps that’s a direction for the player character to go toward; for *Dungeons &*

Dragons, perhaps it's a description of a half-elf gunslinger that one builds upon given the most basic character information.

Although the dynamic between the gamer and the game, or the "interactor" and "player character," as Montfort refers to them, is more intense in *D&D*-like roleplaying games (RPGs), there is something to be said about one's engagement with computational poetry, logic, and truth(s) when interacting-- or simply viewing-- a coded poem. If we continue with the Derridean logic of the absent-present, we might use his assertion in "Linguistics and Grammatology" as a continued framework:

> That trail [of the absent-present] must leave a wake in the text. Without that wake, abandoned to the simple content of its conclusions, the ultra-transcendental text will look suspiciously like the precritical text… [w]hat we here call the erasure of concepts should mark the places of that meditation to come. For example, the value of the transcendental arche must make its necessity felt before letting itself be erased. The concept of the arche-trace must accede to both that necessity and that erasure. It is in fact contradictory and unacceptable within the logic of identity. The trace is not only the disappearance of origin, it would say here—in the discourse that we use and according to the trail that we follow—that the origin did not even disappear, that it was never constituted except as a back-formation by a nonorigin, the trace, which thus becomes the origin of origin. (Derrida, "Linguistics and Grammatology" 66)

In the same realm, Derrida also upholds the idea that one cannot effectively communicate unless communication is defined within a particular context. Specifically, Derrida asserts that communication is subjective; however, when we are attempting to articulate and communicate an abstract concept, we must define specific parameters for the communication to exist—a “determined content, an identifiable meaning, a describable value” (Derrida, “Signature Event Context” 309). Without having ties to some sort of concrete objectivity, communication can be misconstrued. Within computational poetry, the code is the parameter—specifically, the pathways are parameters that create a context for communicating a narrative within the potential coded variable plotlines. Although there are multiple options for words that one could have put into a computational poem, the input is ultimately defined within the parameter of the code. The variables (words, in this case) are given a set of options that they cannot exist outside of. The means of communication are defined for the viewer—however, they can be redefined if different words are input into the code. Regardless, the context exists within the holistic perspective of the basic construct of the code, but upholds the idea of defining and redefining a message so that the means of communication unveil new truths, or new ways in which we understand logic in the absence of having one singular objective answer, given the intense amount of associative imagery in computational poetry.

New forms of engagement, new truths about reality, and new ways to consider a text utilize the concept of the absent-present (analyzing the associative images based upon our personal life experiences). In relation, according to Scott Rettberg, interactive fiction takes on a modernist sensibility—specifically, Rettberg asserts that "[p]articular concerns of literary modernism included a heightened interest in the materiality of language, an increased use of referentiality and intertextuality, and ways of using language to represent interiority and the flow of human consciousness" (Rettberg, "Hypertext Fiction" 56). Virginia Woolf's *To the Lighthouse* is a prime example of these features, and Rettberg agrees, declaring that what we see in Woolf's work is a "turn toward representing events not primarily through their manifestation in the exterior world, but as an interior process that can itself be represented not only through shifts of narrative voice and point of view, but also through the material form of the text itself—its pacing, its use of grammar, and its patterns of association" (Rettberg, "Hypertext Fiction" 56). The biggest distinction here with relation to interactive fiction is likely the notable shifts of narrative voice, point of view, and associative patterns. Pacing and grammar play into each of these as well—a context is provided, pathways are formed, the revealing of information works at different paces based upon one's choices in the narrative, and the articulation (grammar) within the text echoes the sensibility of the work, which paces the interactor in many "time signatures," as James Wood might say, in *How Fiction Works* with regard to narrative structure (Wood 43). Is this criteria not also applicable to computational poetry? The only difference in one's engagement with a computational poem versus an interactive fiction game would be one's ability to choose pathways; in computational poetry, the pathways are chosen for you by the code, which produces an output for the reader. Computational poetry, specifically Montfort's, upholds the associative qualities; Woolf's text does each of these things as well, and Rettberg provides an example of the moment where "Woolf describes words as making a 'pattern on the floor of the child's mind'" (Rettberg, "Hypertext Fiction" 56). Later on in Woolf's novel, Woolf (and her narrative style) unveils and structurally echoes a breathtaking articulation of how one stumbles upon the "truth" in life. Woolf asserts:

> … the old question which traversed the sky of the soul perpetually, the vast, the general question which was apt to particularise itself at such moments as these, when she released faculties that had been on the strain, stood over her, paused over her, darkened over her. What is the meaning of life? That was all-- a simple question; one that tended to close in on one with years. The great revelation had never come. The great revelation perhaps never did come. Instead there were little daily miracles, illuminations, matches struck unexpectedly in the dark… (Woolf 161).

Throughout Woolf's text, we continue to see her narrative work in a stream-of-consciousness style, which is similar to that of computational poetry. Within computational poetry, the reader experiences the pathways of consciousness as they flow from the code into the output of the coded space. Are these pathways the "matches struck unexpectedly in the dark?" Small moments of truth? Moments that make us reconsider all of our initial thoughts, forcing us to expand our consciousness and consider new outcomes? Let's look at the second section of Montfort's *The Truelist* to further explore this idea.

The second section of *The Truelist* begins similarly to the first section, due to the coding variables. Montfort's code creates the following lines within the first two stanzas of section two of the code's output:

> Now they saw the overcloth,
>
> The playbound past,
>
> The cloudpath horn opposing the horse,
>
> And the hardeye,
>
> The waterfield supporting the head,
>
> The moonhills delighting,
>
> The ringsack supporting the ship,
>
> And the windman, … (*The Truelist* 8)

Some noteworthy aspects of the poem's structure that require explication include: the syntax of the first line in the first stanza, the structure of each stanza and position of the text, the associative imagery created within the diction, and the endless series of commas, which lead us all the way to the end of the second section, where, as we will soon note, the punctuation changes to a semicolon and finally a period.

To explain, one should note that similarly to the first section of the book-length poem, this second section begins with the words “Now they saw the…”. This time, instead of seeing “foothills,” as we did in the first section of the poem, we see “overcloth.” This is due to the random-like, but deterministic, structure of the code. By examining the code, we can break down various sections of it to understand how it produces the output, and the kind of “bricolage” is being created by the series of words the code has been given to work with alongside commands. Through an intricate system, Montfort codes the beginning of the poem using the Python language:

```
VERB_I = 'alight danc delight dream hurtl listen ponder mourn sleep'.split()
VERB_T = 'welcom exceed fac follow lead oppos reflect shadow support'.split()
PRE_A = 'back down flat fore free hard high long over soft true'.split()
PRE_N = """air bear bird blood blue blur bone book cloud corn cross dead
  door ear earth fire fish fly foot hand head heart home horn horse house land
  life light lock love man moon night past play plum port post rest ring river
  root salt sand ship sick side sky slip snow star stone tail trap turn void
  water whip wind woman wood work""".split()
POST_A = 'away back bound less like path side'.split()
POST_N = """bed bird board boat book cloth eye field fire fish hand head hills
  horse town house king wing light line list lock man ring room sack ship stone
  storm tail hound water way weed land woman wood word work worm yard""".split()
POST_S, T, ALL, OUT, X = 'hood ness'.split(), '    ', [], '', 3223
```

This section provides the code (or, as I would argue, the “bricoleur”) with the necessary words and information it needs to create the poem, making it result in a form of “bricolage,” as the code has used the available means to create a poem. To focus on the line of code that created the “overcloth” phrase at the beginning of the first stanza in section two, we should look at:

```
        PRE_A = 'back down flat fore free hard high long over soft true'.split()
```

and

```
        POST_N = """bed bird board boat book cloth eye field fire fish hand head hills
          horse town house king wing light line list lock man ring room sack ship stone
          storm tail hound water way weed land woman wood word work worm
yard""".split()
```

For the purpose of my analysis, I bolded ‘over’ and ‘cloth’ as used in the code above. This line of code, when resolving itself, essentially links to other sections of the code as it resolves which tells the code abut its functionality. In particular, we know that the code understands that there are variables in the word choice. Later on in the code, we begin to see how the input understands the order in which to create the outputted syntax. Let’s examine the next section of the code, where I will bold the important sections of the code:

```
def shift(limit):

   'Xorshift to create a random-like but deterministic sequence.'
   global X
   X ^= (X << 19) & (2 ** 64 - 1) # ANDing by (2...) keeps X a 64-bit value.
   X ^= (X >> 35) & (2 ** 64 - 1) # Bit shifts >> and << along with OR give
   X ^= (X << 4) & (2 ** 64 - 1) # a random-like sequence.
   return X % limit # MOD gives a difficult-to-predict number in [0, limit-1].

for com in PRE_N + PRE_A: # Generate all PRE-POST combos except tautonyms.
   for pound in POST_N + POST_A + POST_S:
      mid, before, after = '', '', ''
      mid = ['', ' and '][com[-3:] == 'man' and pound[-3:] == 'man']
```

```
if pound in POST_S: # Substantive compounds have a possessive before.
   before = POST_N[shift(len(POST_N))]
   before = before + ['’s ', '’  '][before[-1] == 's']
elif pound in POST_A: # Adjective compounds have a noun after.
   after = ' ' + PRE_N[shift(len(PRE_N))]
mid = [mid, '-'][com[-1] == pound[0] and not pound == 'ness']
if mid == ' and ' or not (com == pound):
   ALL.append('the ' + before + com + mid + pound + after)
```

When observing the sections of the code that I have bolded, we begin to understand how the code resolves itself. It is creating a “random-like but deterministic sequence,” following all of the rules of the values and variables within the code. The code then knows to generate a word from a specific series of words, and where to place it in a sentence, due to the usage of “pre” and “post” with regard to adjectives, nouns, and verb compounds. The usage of “pre” and “post” tell the code whether or not a certain word goes in a specific spot. This is indicated by “POST_A” and “POST_N,” as well as “POST_S,” respectively, and allows the bricoleur (the code) to create the bricolage construct from the series of words and rules. Other words and rules available to perform the bricolage includes the usage of the suffix “-ness”-- mid = [mid, '-'][com[-1] == pound[0] **and not pound == 'ness'**]-- and other words/parts of words that are able to be appended to the phrases-- ALL.append(**'the ' + before + com + mid + pound + after**).

However, despite the variables in the code, the fact that the code is deterministic provides us with the understanding that the code, in the state that Montfort has written it, will always produce the same book-length poem. Essentially, Montfort produced a bricoleur-- the code-- and a series of variables for it to use random-like structures for the creation of bricolage. To that end, though, one might wonder: if the code produces the same poem every time, how is it bricolage? How is it not simply a given output, following the same series of resolutions?

I believe that Montfort actually did take into consideration the ability for an art-like structure of words to be created, whether as produced by his original series of words, or by pleading to the reader to use the code and edit it however they please. Montfort provides the full code in the back of the book, at the very end of the poem. He prefaces it by asserting that "[t]his one-page (80 column x 66 line) program generates a book-length poem," and declares that "[c]opying and distribution of this file, with or without modification, are permitted in any medium without royalty provided the copyright notice and this notice are preserved" (*The Truelist* 141). This further supports the "code as bricoleur" concept, as Montfort is allowing anyone who pleases to provide the bricoleur (the code) with new variables, phrases, and parameters, which would produce a slightly different outcome depending on the exact structure of the newly-edited code. So, for example, you could edit the line of code titled "POST_N," where N = noun, and add new words, creating a line of code such as: **POST_N= ''' phone book home town country hills sand sky woman'''''** and so on, which would then use those words within the bricolage, which continues the random associative imagery one receives when engaging with the book, as the nouns are not specific, and could apply to any phone, any book, any home, for example.

V. Conclusion: An Ever-Present Past

At the end of *The Truelist*, we finally receive some variation in the syntax. Montfort has coded in such a way that the code has finished its job as being a bricoleur, asserting to the audience several stark lines of poetry, that end with a bold statement, requiring the reader to meditate on the final line: …

And the corneye,

    The overfish,

        The snowring dreaming,

            The deadlist,

And the tailboat welcoming the ring,

    The postcloth,

        The bloodfire;

            Now they saw the truelist. (*The Truelist* 140).

In the final stanza of the poem, we are met with a new punctuation mark, as well as a different ending than the previous ending stanzas included in the earlier sections of the text. On a structural level, one notices the bigger pause that the semicolon causes when one reads the text. This pause is leading us up to the final phrase: "Now they saw the truelist." This leaves the reader potentially wondering: How am I supposed to interpret this? Do we read this as "true" + "list," placing everything we've just read within a sensibility of objectivity? In addition, the phrase is in a strange present, but also past-tense, as the line uses the word "now," but then the past-tense "saw," which, interestingly enough, is the same syntactic phrase used in the beginning of each section. Why is it important that we need to feel both past and present at the same time? In a sense, the deterministic sequencing of the code (or bricoleur, as we've established) has provided a parameter of objectivity for the reader-- the bricolage is the "truelist," (or "true" + "list") as the list of images we read throughout *The Truelist* are objectively true given the parameters of the coded space and input. This is quite different from the ending of all of the previous sections, which ended in "And then the [noun]," giving the reader the sensibility that there was more to see within the truelist. The truelist itself is the code, which is also the bricoleur, with the variables being the bricolage, in the ever-present past, as the code can be run infinitely, but produce the same outcome (if, of course, Montfort's original diction is used within the code).

On a final note, let us look back at a statement from Jacques Derrida:

> History has always been conceived as the movement of a resumption of history, as a detour between two presences. But if it is legitimate to suspect this concept of history, there is a risk, if it is reduced without an explicit statement of the problem I am indicating here, of falling back into an ahistoricism of a classical type, that is to say, into a determined moment of the history of metaphysics. Such is the algebraic formality of the problem as I see it. More concretely, in the work of Levi-Strauss it must be recognized that the respect for structurality, for the

internal originality of the structure, compels a neutralization of time and history. For example, the appearance of a new structure, of an original system, always comes about--and this is the very condition of its structural specificity-by a rupture with its past, its origin, and its cause. Therefore one can describe what is peculiar to the structural organization only by not taking into account, in the very moment of this description, its past conditions: by omitting to posit the problem of the transition from one structure to another, by putting history between brackets. ("Structure, Sign, and Play" 291).

To conclude, Derrida is synthesizing a "neutralization of time and history" (291). Although there always seem to be new objective structures, these are merely "past conditions," as everything that is in the present is almost immediately in the past—similarly to Montfort's interesting ever-present past notion in the verb tenses of his text. This is a prime example of creating a comprehensive cosmology of computational poetry and turning a space into a coded "place" when the code is given a defined context in which it can become a bricoleur, creating artistic bricolage-- or in this case, creating the bricolage that upholds the associations within the truelist. As the reader creates their own associations, the code's ability as bricoleur to create the bricolage provides an impactful look into how we view the transition from space to place within a coded list of possibilities, allowing us to imagine this ever-present past in which Montfort's text exists.

Works Cited

Derrida, Jacques. “Linguistics and Grammatology.” *Of Grammatology*, translated by Gayatri Chakravorty Spivak, Johns Hopkins University Press, 2016. 29-79.

Derrida, Jacques. “Signature Event Context.” *Margins of Philosophy*, translated by Alan Bass, University of Chicago Press, 1982. 307-330.

Derrida, Jacques. “Structure, Sign, and Play in the Discourse of the Human Sciences.” *Writing and Difference*, translated by Alan Bass, University of Chicago Press, 1978. 278-293.

Knowles, Alison and James Tenney. “A House of Dust.” *https://nickm.com/memslam/a_house_of_dust.html*, accessed 7 October 2019.

Levi-Strauss, Claude. “The Science of the Concrete.” *The Savage Mind*. The University of Chicago Press, 1962.

Meehan, James. “Tale-Spin, an Interactive Program that Writes Stories.” *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977.

Montfort, Nick. “Fretting the Player Character.” *Second Person*. MIT Press, 2007. 139-146.

Montfort, Nick. *Nickm.com*, 2020.

Montfort, Nick. *The Truelist*. Counterpath, 2018.

Rettberg, Scott. “Combinatory Poetics.” *Electronic Literature*. Polity Press, 2019. 20-53.

Rettberg, Scott. “Hypertext Fiction.” *Electronic Literature*. Polity Press, 2019. 54-86.

Spivak, Gayatri Chakravorty. Translator's Preface. *Of Grammatology*. By Jacques Derrida. Trans. by Chakravorty. Johns Hopkins University Press, 2016. Xxvii-cxi.

Strachy, Christopher. "Love Letters." 1952. Reproduced by Nick Montfort, 2014. *https://nickm.com/memslam/love_letters.py*, accessed 2 January 2020.

Wood, James. "Flaubert and Modern Narrative." *How Fiction Works*. Picador, 2008. 39-46.

Woolf, Virginia. *To the Lighthouse*, Houghton Mifflin Harcourt, 1955.