Routledge
Taylor & Francis Group

Digital Creativity

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ndcr20

Reading computer-generated texts: examining code as a reading strategy

Tuuli Hongisto

To cite this article: Tuuli Hongisto (27 Oct 2023): Reading computer-generated texts: examining code as a reading strategy, Digital Creativity, DOI: 10.1080/14626268.2023.2274451

To link to this article: https://doi.org/10.1080/14626268.2023.2274451

Published online: 27 Oct 2023.

DIGITAL CREATIVITY
https://doi.org/10.1080/14626268.2023.2274451

Routledge
Taylor & Francis Group

Reading computer-generated texts: examining code as a reading strategy

Tuuli Hongisto [a,b]

[a]Department of Philosophy, History and Art Studies, University of Helsinki, Helsinki, Finland; [b]School of Languages and Translation Studies, University of Turku, Turku, Finland

ABSTRACT

This article explores code as part of a reading strategy, focusing on a case study of works produced as part of a programming challenge entitled National Novel Generation Month (NaNoGenMo). For the challenge, participants needed to develop a programme that produces an output, most often a creative text, and make both the output and its source code available to readers. I approach the NaNoGenMo texts as works involving multiple audiences with different capabilities. I argue that examining the code alongside the produced text is a reading strategy that offers a different reading experience than that of reading the work without the code and that the source code can provide insights into the work's authorial intention. As computer-generated texts are becoming increasingly ubiquitous, discussion of their authorship is as relevant as ever. This article offers one perspective on this discussion and on the significance of authorial intention in computer-generated works.

ARTICLE HISTORY
Received 29 March 2023
Accepted 18 October 2023

KEYWORDS
Story generation; digital literature; electronic literature; reading strategies

Introduction

Digitalization has greatly broadened the scope of the study of literature. The development of new technology and platforms has given rise to new types of narratives as well as shaped existing genres, such as hypertexts. Entire fields of study, such as software studies and game studies, have emerged from the examination of these phenomena. Digital media has also complicated the communication process between the text and reader. The roles of sender and receiver are increasingly merged and mediated through digital platforms, as evidenced by collective modes of writing, such as writers crowdsourcing ideas or making poetry out of social media posts. This shift has also affected the concept of the 'reader' and theories on reading strategies. Reader (and player) theories have been both created and reformulated to fit digital contexts, as demonstrated by works on cybertext theory and non-linear texts (Eskelinen 2012; Pope 2020), games (Dubbelman 2013; Thoss, Ensslin, and Ciccoricco 2018), digital platforms (Roine 2019) and other digital technologies.

In this article, I examine reading strategies in the context of computer-generated texts. Digital technologies have broadened the possibilities for literary experimentation, and the resulting computer-generated works can prompt readers to try unconventional reading

CONTACT Tuuli Hongisto tuuli.hongisto@helsinki.fi

strategies. I explore this phenomenon through works produced in an online challenge, National Novel Generation Month (NaNoGenMo),[1] where participants develop programmes that produce creative texts. The participants develop a programme that generates a 50,000 word output and share both the code and the output online. In my study of NaNoGenMo texts, I focus on the following questions: How can examining the source code impact the reading of NaNoGenMo texts? What role does the perception of authorial intention play in their interpretation?

On reading strategies and National Novel Generation Month

All texts require knowledge and skills on the part of their readers to be understood: linguistic proficiency, a sense of historical context, recognition of cultural tropes, and so forth. In this article, I examine one skill that particularly affects the reading experience of my research material on computer-generated texts: the ability to understand code. Just as understanding intertextual references when reading a work like *Pale Fire* can help the reader to appreciate the work (Phelan and Rabinowitz 2012, 141), reading the texts produced as part of National Novel Generation Month requires specialized knowledge and skills.

I approach the different ways of reading NaNoGenMo works as reading strategies. The NaNoGenMo works have an exceptional quality: the process of their creation can be accessed through source code. I suggest that two main ways of reading NaNoGenMo texts can be differentiated based on whether the reading includes an examination of the source code and produced text. From now on, I refer to readers using these strategies as code-included readers (CI reader) and code-excluded readers (CE reader).[2] The article focuses on what reading the source code can add to the reading of NaNoGenMo texts with the help of the categories CI and CE reader.

In practice, the boundary between a CE and CI reader is not absolute. Understanding code evolves along a sliding scale, and the reader does not have to understand the code in its entirety for it to affect the reading experience. The work can also be appreciated without the code, just as any literary work can be appreciated without recognizing all of its references. With NaNoGenMo texts, one reader can also belong to both CE and CI audiences through first reading the work without the source code context and then re-reading it along with it. Nevertheless, there is a clear difference between reading code as part of a reading strategy or leaving it out. Making the distinction between the two readers demonstrates the effect that the code can have on the reading experience.

The works produced as part of the NaNoGenMo challenge are often unconventional and challenging to read. There are many ways to approach these kinds of texts that do not involve the source code. For example, experimental texts can be approached by abandoning a search for coherence as the primary goal of reading (Siltanen 2016, 9–11). These kinds of reading strategies can be used with NaNoGenMo texts regardless of whether the reader has access to the source code or not. As this article examines the effect that the source code can have on reading, the exploration of other kinds of strategies is for the most part beyond the scope of this article.

Story-generating algorithms, experimental literature and NaNoGenMo

NaNoGenMo texts are part of a broader context of programmes that produce stories, so-called story generator algorithms (Gervás 2012), which have been developed for decades as part of research on computational creativity. The development of story generating algorithms has involved efforts to formulate narrative features such as plot and character to a

computer readable form. For example, outlines of plot that gave an overall structure to the texts, called story grammars, have been developed (Kybartas and Bidarra 2017, 242). Many NaNoGenMo programmes use similar approaches, and participants also occasionally reference early story-generators such as Tale-Spin (Meehan 1976) and Minstrel (Turner 1994) as inspiration or resources for their work.[3]In addition to the study of computational creativity, computer-generated literature has its roots in the tradition of experimental literature. As Scott Rettberg (2018, 5) points out, computer-generated literature along with other forms of literature that examine the capabilities of computer technology are inherently experimental, since their writers engage with new narrative and poetic approaches that emerge from the computational context. The OuLiPo group, founded in France in the 1960s, is an example of experimental working methods that influenced later computer-generated literature. The group is essential in the field of constrained and procedural writing, where the writer systematically uses self-selected rules that go far beyond normal writing conventions (Baetens 2012). OuLiPo examines the interface between literature and mathematics: the constraint can be thought of as a mathematical theorem, and the texts produced based on it are different possible iterations of the constraint (Baetens 2012). The legacy of procedural literature shows in the NaNoGenMo challenge in that it emphasizes the process of producing a text, not just the end result. The challenge also constrains the writing process of the participants by requiring that the text should be computer-generated and 50,000 words in length. Many participants challenge themselves with more restrictions, and as a result, sub-challenges and related challenges to NaNoGenMo have emerged. For example, in Nano-NaNoGenMo, the source code for the produced text can be no longer than 256 characters,[4] and in NaNoLipo (directly inspired by OuLiPo), for each day of the challenge new restrictions are set for the output.[5]

The NaNoGenMo challenge provides ample material for researching the reading strategies employed with computer-generated texts. Although the challenge is relatively niche and not familiar to the general public, it has yielded a great deal of texts to explore. NaNoGenMo has been organized annually since 2013, and each year 20–100 people complete the challenge. The NaNoGenMo projects are diverse both in terms of their methods and output: the programmes vary in their complexity, and the texts are constructed for different purposes, such as parody and wordplay. Even though the name of the challenge suggests that the produced texts are novels, in practice the results are often something quite different from a novel, and thus, they require different approaches from their readers than do novels in general. The versatility of NaNoGenMo texts calls for a versatility of reading strategies.

Computer-generated literature and authorship

In the field of computer-generated literature, the question of who or what should be considered the author is always present. The dynamic of human and a machine is can show in different forms: the programmers and writers involved in the creation of the texts can for example strive for minimizing their own role, approach the use of a computer programme as a tool of expression, or play with the ambiguity of authorship of the work. For the reader, the authorship of computer-generated literature is also a point of interest, for it is what sets the computer-generated works apart from most other works of literature and arouses curiosity. This can affect the reading experience as well, as the reader can ponder which parts of a text are the result of human design and which are not.

No computer-generated text stands on its own without human influence. Therefore

computer-generated texts are best approached as collaborative efforts of human–computer interaction. In general, computer-generated texts are affected by the programmer, the programme and the input (the data the programme uses) (Henrickson 2018, 188–189; Lebrun 2017, 413). This is the case with NaNoGenMo texts as well: their authorship cannot be attributed solely to the programme or the programmer.

The amount of control the programmer has over the output can vary greatly depending on how the programme functions. The programmer can set restrictive rules for the programme to follow, for example regarding the available vocabulary or the narrative structure. Alternatively, the programmer can use methods that restrict the production of the output less, which makes it possible for the text to contain sections that the programmer could not have anticipated when coding the programme. For example, there is a tremendous difference between a neural-net-based programme that reads novels and tries to emulate them and a script that produces the word 'meow' 50,000 times.[6] In the former example, the programmer is not in control of the text on a sentence level, and thus, the role of the programme in the construction of the text is substantial, while in the case of the latter example, there is no possibility for generation of anything unpredictable or surprising.

In addition to the programmer's influence, programmes that process large quantities of data (such as novel corpora) and try to emulate it are greatly influenced by the data that is used. The use of various source texts, such as Wikipedia articles or novels from databases like Project Gutenberg, is typical of NaNoGenMo projects. This is one way in which the works utilize the methods of experimental literature: the combination of different texts has been typical, for example in works of so-called 'uncreative writing', where texts by other authors are combined and brought into new contexts (Goldsmith 2011, 1–4). With computational methods, copying, combining and otherwise utilizing textual material is particularly easy: the vocabulary and stylistic factors, such as the length of sentences, of the works in the corpora can be imitated. For example, a programme that is trained on a sci-fi corpus and a programme that is trained on the works of Shakespeare are going to produce vastly different results. In this way, the authors of the works in the corpora influence the produced work and are arguably participants in the authorial process. The questions of how authorship and authorial intention are defined need to be evaluated on a case-by-case basis when doing a close reading of NaNoGenMo texts and their source code.

For the reader, the amount of control the programmer has in relation to the produced text is closely related to perceived authorial intention. What is the concept or idea behind the work? To what extent is the programmer's intention behind the produced text, and to what extent is it not? Reader's view on these questions affects their reading.

The concept of 'authorial intention' and its importance in interpretation has been debated for many decades. As outlined by Rabinowitz (2018, 88), and following the publication of W. K. Wimsatt, Jr. and Monroe Beardsley's *The Intentional Fallacy* (1946) and Barthes's influential *The Death of the Author* (1967/1977), understanding the intention of the author has widely been regarded as insignificant and pointless when interpreting texts. However, this view has been challenged, for example, in rhetorical narratology, as the author's intention—or at least the reader's impression of their intention—is brought into the analysis. As James Phelan (2017, 27) puts it, readers generally assume that texts reflect human agency and are an attempt to communicate. The reader therefore constructs some kind of understanding of the author and their intentions. The trust that the author of a text is trying to tell the reader something, which Leah Henrickson (2021, 36–37) refers to as

the *hermeneutic contract*, is complicated in computer-generated works, as the reader cannot be certain that the author of the text wants the reader to understand it (Henrickson 2021, 36–37). This is relevant for the reading experience of computer-generated texts such as the works of NaNoGenMo. When the reader is trying to figure out which parts of the text are dictated by the programmer and which parts are not, they are trying to figure out which elements in the text are a result of authorial intention. I refer to authorial intention in this sense: as the reader's effort to understand to what extent the text is the result of intentional communication.

When it comes to the two main reading strategies of NaNoGenMo texts—reading with and without the code—the process of constructing the authorial intention behind the work can be different. As with reading any text, the reader of NaNoGenMo texts is faced with fundamental questions about both the form and content of the text. Why has the text been constructed in such a way? What is it trying to achieve, and does it succeed? The source code can provide insights into these questions. It can give the reader a glimpse into the authorial intention behind the work—the parts for which the programmer is responsible. In a reading done without the code, these kinds of observations cannot be made as straightforwardly. The CE reader must rely solely on the text and more conventional ways of constructing authorial intention.

The contexts and paratexts of NaNoGenMo

Multiple layers of contextual information can guide the reading of NaNoGenMo texts, especially if the texts are read in their original surroundings. I approach these surrounding elements as *paratexts*. As defined by Gérard Genette (1997, 1–2), paratexts are texts that place the work in its context and sometimes provide commentary, such as titles, prefaces and illustrations.

The paratexts created as part of the projects consist of a variety of materials, such as an entry written by the programmer or the names and contents of the shared files. Some paratexts, such as the titles of the works, can often be found in the text files as well. I will mostly focus on the paratext of the source code. Depending on the project and the programming language used, it can be found in one or in multiple files. When I refer to the *text* in the context of NaNoGenMo projects, I mean specifically the text file shared by the programmer.[7] The text file can be quite simple, containing almost no formatting, such as line breaks, but many contributors choose to share the text in a more reader-friendly form, such as a pdf file with pagination.

Both the texts and paratexts of the various NaNoGenMo projects are accessible through GitHub, where the challenge is hosted. Reading and interpreting this material can involve navigating the platform and browsing and comparing different text files. The user of the platform can also engage in a dialogue with the participants by leaving comments. GitHub is a platform for programmers, and this is reflected in how the texts are presented to the reader. Typically, when one clicks the code repository of a NaNoGenMo project, such as the one depicted in Image 1, the output is just one of the files on display, so visually the code and output are presented as equals. Many participants also share the output directly on the GitHub interface, which means that the visual environment surrounding the text is a constant reminder of its origins. For example, when a text document is displayed on GitHub, line numbers are automatically shown—a feature that is typically associated with code files on the platform.

The participants in the challenge target their work mainly at people who have knowledge of programming. This aim is evident in how they address their readers when presenting their

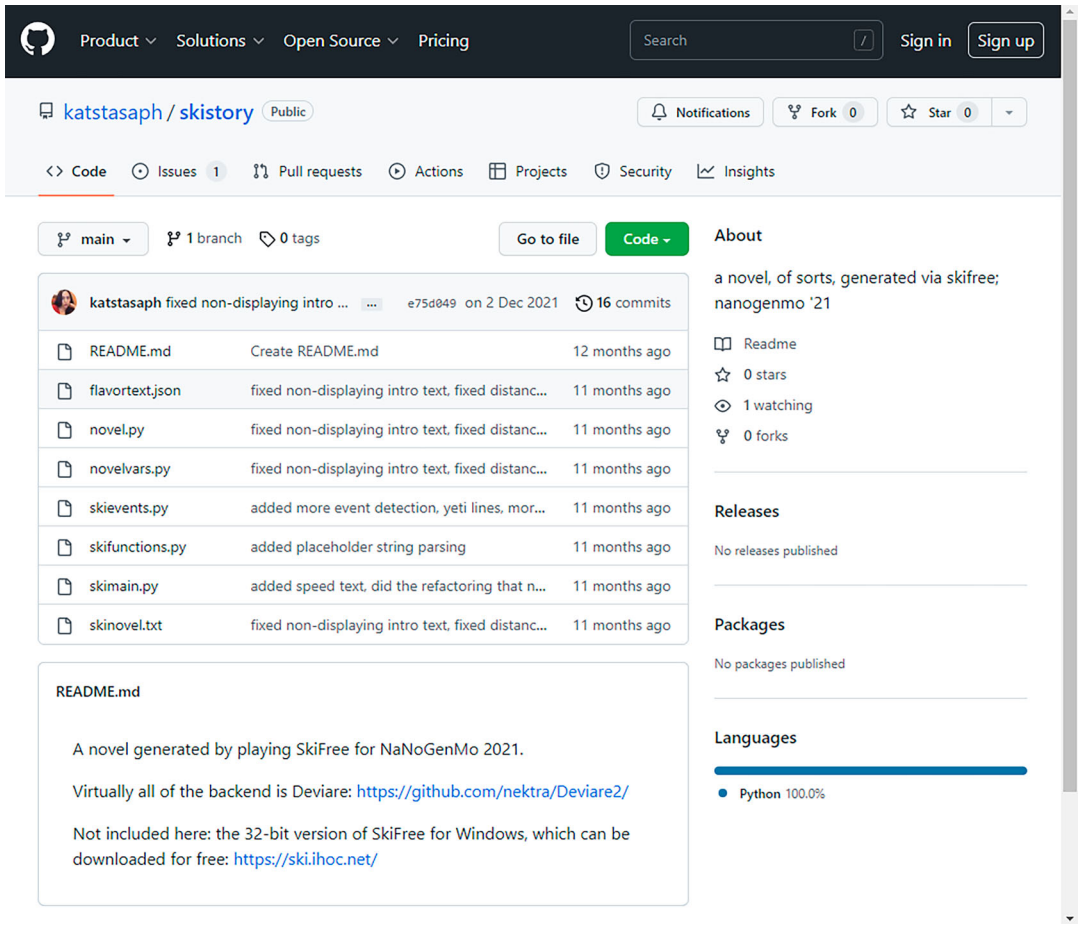

Image 1. A screenshot from a typical NaNoGenMo GitHub repository.

work. One can participate in the challenge by posting an entry on the challenge's webpage, on the platform. The entry must contain links to the source code and the output, which in most cases can also be found in a GitHub repository. For the most part, these introductory comments by the participants outline how the programme functions and some go over the code in detail. This draws the reader's attention to the source code and shows that practising and showing others one's programming skills is part of the point of participation.

The NaNoGenMo community has a culture of peer learning and sharing. Coding is creative work, and the same operating principle can be implemented in many different ways. The participants are interested in how others have solved problems related to the challenge. They also often comment on each other's work and suggest ways to modify the programme's operation. Many participants are also inspired by each other's work and, thanks to open source code, can use parts of each other's work to create new projects. Rettberg (2018, 49) compares this kind of sharing culture within electronic literature to OuLiPo's writing games, where writing is seen as 'a form of productive play'.

When taking all these various issues into account, it seems clear that when it comes to the actual readers, the NaNoGenMo texts are primarily produced by programmers for

programmers. However, it is possible to read NaNoGenMo texts outside of this context. The works are usually stable and can be read linearly, and the reader cannot influence the progression of the work in any direct way. In this sense, the works differ from many other works of electronic literature, such as generative works, which are texts that have no stable output and where the reader can choose to see different versions of the text (see e.g. Ackermans 2017), or hypertexts and games, which allow the reader to affect the content or order of the work (see e. g. Eskelinen 2012; Pope 2020). It is relatively easy to separate the output from the GitHub platform, for example by copying the text to a file. Some NaNoGenMo projects have even been published in printed form (see, e.g. Agafonova, Tikhonov, and Yamshchikov 2020; Montfort 2014). The fact that the texts are read both with and without knowledge of the source code is reflected in my categorization of CI and CE readers. It also means that it is possible, albeit rare, to encounter NaNoGenMo texts without being aware of their creation process. Thus, there can be two kinds of CE readers: those who are aware that the text they are reading is computer generated, and those who are or are unaware of it.

The code files for the NaNoGenMo projects resemble other paratexts, such as forewords, in the sense that they contextualize the work for readers. They have a similar function as that of footnotes in editions of novels aimed at students, which provide the readers with context that helps them to 'join the authorial audience' (Phelan and Rabinowitz 2012, 140). However, while contextualization of the work is a key aspect of the role of the source code in the reading experience, there is much more to the process. The code provides a view onto how the text has been constructed and its underlying aims in a way that is incomparable to that of footnotes, prefaces or other paratexts traditionally found in printed works. As the examples in the next section show, seeing the 'anatomy' of the text through code can enhance the reading.

Reading with the source code

The significance of reading the code is manifested in the content of the NaNoGenMo texts in multiple ways. This is especially clear with highly conceptual projects, where knowledge of the source material and the creation process of the text are needed to appreciate the work. The integral role that the source code can play in the reading is illustrated by the following example, an excerpt from a poem from one of the NaNoGenMo projects:

> Sin
> A wing, a color
> Otherwise me
> Course lover
> Act brain

At first glance, there does not seem to be much coherence to the poem. The reading experience is different, though, for a CI reader who sees how the programme functions. The poem is produced by a programme that writes an output where each word has a number of characters corresponding to a digit of pi.[8] The programme can take any text as its source code. In the final work, the programmer used *Anne of Green Gables* to generate the poem. The programme allocates the words of the text it has received to specific lists based on their length and then, following the digits of pi, selects words from a corresponding wordlist. A CI reader can track this process by reading the code. For example, as can be seen from the code excerpt shown in Image 2, each word from a given input text (fwords) is allocated to wordlists based on their length.

For a CI reader who knows what to look for, the reading is altered after seeing the code:

> Sin **(3)**
> A **(1)** wing, **(4)** a **(1)** color **(5)**
> Otherwise **(9)** me **(2)**
> Course **(6)** lover **(5):**

```
for line in fwords:
     for words in line.split():
         if words.isalpha():
            if len(words)==1:
             oneletter.append(words)
             onewordCount +=1
            if len(words)==2:
             twoletters.append(words)
             twowordCount +=1
            if len(words)==3:
             threeletters.append(words)
             threewordCount +=1
            if len(words)==4:
             fourletters.append(words)
             fourwordCount +=1
            if len(words)==5:
             fiveletters.append(words)
             fivewordCount +=1
            if len(words)==6:
             sixletters.append(words)
             sixwordCount +=1
            if len(words)==7:
             sevenletters.append(words)
             sevenwordCount +=1
            if len(words)==8:
             eightletters.append(words)
             eightwordCount +=1
            if len(words)==9:
             nineletters.append(words)
             ninewordCount +=1
fwords.close()

thefirstword=(threeletters[(random.randint(1,threewordCount))-1])

thesecondword=(oneletter[(random.randint(1,onewordCount))-1])
```

Image 2. An excerpt from the source code of a poem generator.

Act **(3)** brain **(5)**

Knowledge of the operating principle of the programme transforms the reading experience. Still, knowing how the programme operates leaves a lot of room for interpretation—for example, why was *Ann of Green Gables* chosen as the source text, or what is the meaning of arranging the work according to digits of pi. However, being familiar with this organizing principle in the first place is what makes asking these questions regarding the interpretation possible, marking a significant difference between the CE and CI reader.

In some NaNoGenMo texts, code and text are intertwined in a way that highlights themes of the work. This can be seen, for example, in utilizing structures typical of coding in the text. It is challenging to produce long coherent texts with computational methods (van Stegeren and Theune 2019, 65), so NaNoGenMo participants have had to use their creativity to produce such texts. One way to achieve this is to utilize one of the cornerstones of programming: the loop structure. With a loop, one can for example produce short texts that are iterations of the same segment of code, and tie them together into a longer text. Some participants have taken advantage of the inherent repetition of the loop structure by making it the text's central stylistic device. This is the case with a text named *Any day in 2020*[9] that is a diary full of log entries that all contain descriptions of the same activities (such as listening to a podcast, watching videos, playing games) for the whole 50,000 words (see the excerpt below). The text is produced by rearranging and repeating a group of sentences written by the programmer.

> **Wednesday the 20th of May 2020**
>
> Did some writing on the new book. Made some moves in a game of nomic. Got a new board game in the post. I hope this US election won't end badly. Read an interesting article. Watched some short videos.
>
> ---
>
> **Thursday the 21st of May 2020**
>
> Made some moves in a game of nomic. Watched some short videos. Did some background reading for the new book. Listened to a podcast. Thought how I should walk more, I used to walk all the time. I hope this US election won't end badly.
>
> ---
>
> **Friday the 22nd of May 2020**
>
> Played a board game. Watched some short videos. Listened to a podcast. Did a cryptic crossword. Read an interesting article. Played some videogames. Did some background reading for the new book.

The loop structure in the code creates entries that resemble each other, which in turn creates the eerie impression of the protagonist living essentially the same day over and over again. With repetition, comes the association of mechanicalness: in the text, the life during the pandemic is made to seem like it could have been computer-generated. In *Any day in 2020*, the code and the text form counterparts to each other: the loop structure is used to create a mechanical impression, so that the connections between a human and a machine are present not only in the production of the text but also in its thematics. In this way, the reader's attention is drawn not only to the end result, but also to the process of producing the text. Reading the work in its original context with the source code amplifies this effect.

Reading the code can also be a part of the work's appeal in and of itself. When examining the code, a CI reader can see how the text is made—the text's 'anatomy'—which can be a point of interest. Especially in the case of rule-based programmes, where the programmer has a great influence on the output, reading through the code reveals how the text is constructed. Programmes that produce text types with strictly defined features; poems with a particular rhyme structure illustrate this point quite well. For example, in a NaNoGenMo project that produces a collection of limericks, the variation of lines can be examined in the code. Below is an example of a limerick produced by the programme:[10]

> There once was a woman meditating porthole
> She rewired but couldn't flagpole
> With nannies febrile
> She emerged awhile
>
> Her work as a bartender was sinkhole

As can be seen from the example, the poem follows the rhyme structure *aabba* characteristic

of limericks. The poems are constructed line by line: the rhyme structure must be considered, as the following excerpt shows:

```
WRITE line1 allwords 1
WRITE line2 allwords 1
WRITE line34 allwords 2
WRITE line5 allwords 1
```
[11]

In this source code excerpt, lines three and four use a different wordlist (allwords 2) than the rest of the poem. Lines 1, 2 and 5 share the same wordlist of rhyming words (e.g. porthole, flagpole and sinkhole), while lines 3 and 4 use a different wordlist and thus share a different rhyme (e. g. febrile and awhile). This is how the use of different rhymes is ensured for these lines of the poem. The lines are constructed by selecting a sentence structure and assigning values from appropriate wordlists to complete the line. The excerpt below shows the selection of possible first lines for the limericks:

```
There once was a _PERSON.MANWOMAN_
_VERBING_ _RHYMEA_
There was a _ADJ_ _PERSON.MAN-
WOMAN_ _PREP_ _RHYMEA_
_PERSON.NAME_ was a _PERSON.MAN-
WOMAN_ who _VERBED_ _RHYMEA_
a _PERSON.MANWOMAN_ called _PER-
SON.NAME_ who _RHYMEA_
There was a _PERSON.MANWOMAN_
_PREP_ _PLNOUN_ _RHYMEA_
```
[12]

As the excerpts indicate, there are five possible sentence structures for the first line. Words written in uppercase letters represent different word lists, such as names (PERSON.MANWOMAN), different verb inflexions (VERBING; VERBED), prepositions (PREP) and so forth. The word lists can be examined in the GitHub repository as well, making the process of constructing the limericks even more transparent. Jumping between different files to form an understanding of how the text is constructed allows the CI reader to appreciate the features that have been taken into account but also to take a critical stance on the text if something is found to be lacking, such as a clear, concise point in the case of limericks. This process can provide a CI reader with more insights than the mere output– for example, by making genre features very visible in a way that might be illuminating for a reader who is unfamiliar with them.

Due to the disparity between the knowledge level of CI readers and CE readers, the NaNoGenMo texts are examples of a singular text having two kinds of audiences, one of which is more informed than the other. This kind of double audience can be recognized in works that are aimed at readers with different education or knowledge levels. Phelan and Rabinowitz (2012, 141) present *Huckleberry Finn* as one such example, and Brian Richardson (2007, 259) points out a similar phenomenon in children's literature, where different messages are communicated to children and adult audiences. In some cases, a more knowledgeable critical reader might even take pleasure in feeling superior to a naive reader of the text (Eco 1979, 9–10). This kind of setup can be inferred from many NaNoGenMo projects as well. The reader with the possibility and competence to read the source code can feel like they understand the work in a way that would be impossible for someone coming from outside the NaNoGenMo community. Despite this, CE readers can find meaning from the texts using different reading strategies that do not involve reading the source code. For example, in the case of the pi poem, a CE reader may appreciate how the rearranging of words from *Anne of Green Gables* presents Lucy Maud Montgomery's vocabulary in a new light or find interesting or amusing word associations.

Different readings of NaNoGenMo texts—critical perspective, authorial intention and the Eliza effect

As the examples show, examining the source code of NaNoGenMo texts can affect their

reading. The visual environment that the work can be found in (most often a GitHub repository) encourages the reader to take a look at the source code as well as the produced output. Nevertheless, the text can be read independently of the context. How do these two readings—reading the work with and without the source code—differ?

For one, for a CI reader the construction of authorial intention plays a greater role. The source code can indicate to what extent there is authorial intention in a text. For example, the names of the functions, wordlists and variables of the code can communicate the principles and goals for the produced text. For a CI reader, it can also be easier to grasp the 'point' of the work. This is especially true with projects where the process of developing the programme is more important for the programmer than the output. For instance, in the case of the aforementioned limerick programme, the programmer cites their desire to try out a certain tool for text generation, which is explicitly prioritized over producing as good quality output as possible.[13] Without having access to these kinds of insights provided by the source code, a CE reader can miss out on contextual information that would help manage expectations for the text.

Access to source code also makes it possible to evaluate the process of producing the text. Letting the reader see the code reduces the hierarchy between the author(s) of the programme and the reader and allows the reader to take a critical stance from a technical standpoint: what could have been done better in terms of how the programme functions? The context of the challenge as a chance for the participants to exhibit their programming skills even encourages this kind of reading of the texts. A similar type of evaluative stance is much harder to take without having access to the paratexts and the competence to assess them.

In the case of computer-generated texts, such as the NaNoGenMo works, people's tendency to ascribe more intelligence and complexity to computer systems than they actually possess can also affect the reading. This phenomenon is called *the Eliza effect*, named after the famous interactive computer system *Eliza,* created in the 1960s by Joseph Weizenbaum (Wardrip-Fruin 2012, 24–25; Weizenbaum 1966). It was possible to have a conversation with *Eliza* just by typing any set of words with a typewriter connected to a computer, after which *Eliza* would respond by typing an answer back (Weizenbaum 1966, 36). The answers were crafted based on a currently active script, the most famous of which was a script that mimicked a psychotherapist (Wardrip-Fruin 2012, 27; Weizenbaum 1966, 42). The script behind *Eliza* is quite simple: it is based on picking out keywords and phrases from the input given by the conversation partner and then inserting them into prewritten sentence structures (Weizenbaum 1966, 36–39). Despite its simplicity, *Eliza* was assumed by many of its users to be complex because interacting with it could resemble a coherent dialogue (Weizenbaum 1966, 36–39). As illustrated by the Eliza effect, knowledge that the text was being computer-generated might inspire readers to assign more meaning to the text than they otherwise might. The computational origins of the text make it exceptional, and thus more interesting.

The Eliza effect can impact the reading of NaNoGenMo texts as well. In addition to CI readers, readers who read the text without the code can be aware that the text is computer-generated. For example, even when reading published versions of NaNoGenMo texts, the paratexts such as forewords, can disclose the creation process of the work. Sometimes the computational origins of a work can also be inferred from the style of the text. As Wardrip-Fruin (2012, 36–38) and Weizenbaum (1966, 36) note, the Eliza effect often fades—or does not come into effect in the first place—when users of a programme can understand how it works. In this sense, CE reading could, depending on the work, bring more sense of

wonder to the reading experience than CI reading.

Finding meaning in some NaNoGenMo texts can be challenging for the reader. For example, in the case of the poems based on the digits of pi, there are no mechanisms for ensuring meaningful links between consecutive words. The words are selected using a function that picks the words at random, with the only requirement being the appropriate length of the word: this leads to a lack of semantic cohesion in the text. With these kinds of texts, both the CE and CI reader can of course apply unconventional reading strategies to interpret the text. For example, as put by Richardson (2012, 155), the reader can put aside the normal rules of reading and instead look for occasional 'islands of meaning'. Similarly, a reader of experimental poetry might deploy various reading strategies, not all of which aim at searching coherence (Siltanen 2016, 9–11). In the case of randomly generated NaNoGenMo texts, even without the aim of semantic cohesion, the texts can offer meaningful word pairings by accident. The sheer length of a minimum text of 50,000 words guarantees this result. Another matter is whether finding these 'islands of meaning' in a randomly generated text would remain an interesting task for the entire duration of the text.

Although abandoning the search for coherence might work better for randomly generated texts than conventional reading strategies, the reading of them clearly differs from that of seemingly similar texts written by humans. Here the hermeneutic contract comes into play. Even seemingly unintelligible texts are typically seen as being the result of some kind of communicative intention that the reader is left to decode from the text (Henrickson 2021, 36–37). In the case of computer-generated texts, however, the reader cannot be certain of this—at least not without the context of source code. A randomly generated text is an extreme example of this lack of communicative purpose, at least when it comes to conveying semantic content.

For a CI reader, understanding the aims of the programme and the context in which it is produced can reduce the feelings of discomfort and frustration that texts with no cohesion can cause. For a CI reader, knowledge of the source code is a constant reminder that while NaNoGenMo projects can be approached as literary works, their point can be just as much about trying new things in programming as about the readability of the output.

Conclusion

The NaNoGenMo texts can be read either with or without their source code, and the question of which approach is more fruitful is not clear cut. What is clear is that there is a disparity in the knowledge level between these two approaches. The source code can give the CI reader information that helps them manage expectations and interpret the work. It also provides the reader with an opportunity to approach the text as a kind of map that can be used to find out how the work is constructed, which in and of itself can be interesting. In this sense, the reading experience can be considered lacking without the context of the source code. A CE reader, depending on their interpretation and preconceived notions on computer-generation could for example dismiss the work entirely because it is not up to par with human-written texts—which might not be the aim of the work. On the other hand, a CE reader can also be affected by the Eliza effect, allowing them to see something more meaningful and deep in the text than the operation of the programme would warrant—this is an experience that is inaccessible once the reader has read the source code.

As natural language generation algorithms are becoming increasingly more ubiquitous, the assumption that texts derive from human experience, with the purpose being that of communication, needs to be reconsidered. Compared to natural language generation programmes that are completely opaque to their

users and raise uncertainty on whether or not the produced text has any communicative intent, the NaNoGenMo challenge provides a sharp contrast: it celebrates open-source code that allows readers to appreciate—or criticize—the creation processes of the works.

Notes

1. Homepage of the challenge: https://nanogenmo.github.io/.
2. Many reader models distinguish between the actual, empirical readers and the ideal recipient of the text, who forms an appropriate interpretation of the work. This idea has been expressed in different forms, such as Umberto Eco's (1979) *model reader,* Wolf Schmid's (2010) *abstract reader* and Wayne C. Booth's (1961/1983) *implied reader*. In the same vein, I refer to a 'reader' as an abstraction that can be inferred from the text rather than actual, empirical readers.
3. See for example threads where participants share resources for the projects: https://github.com/dariusk/NaNoGenMo-2015/issues/1, https://github.com/dariusk/NaNoGenMo/issues/11
4. The challenge was launched in 2019: https://nickm.com/post/2019/11/nano-nanogenmo-or-nnngm/
5. The challenge was launched in 2018: https://github.com/eseyffarth/NaNoLiPo2018
6. These are actual examples from NaNoGenMo projects. The first one is a neural network-based program trained on Phillip K. Dick novels (https://github.com/NaNoGenMo/2021/issues/38), while the latter is a tongue-in-cheek submission by one of the admins of the challenge (https://github.com/NaNoGenMo/2019/issues/105).
7. With the NaNoGenMo challenge, the source code and output are separate and it is the output that receives the label of being the produced 'novel'. Thus, the aim is not to make the code into something literary, which is a genre of its own (see, e.g. Marino 2020, 203–204).
8. Link to the project: https://github.com/NaNoGenMo/2021/issues/91.
9. Link to the project: https://github.com/NaNoGenMo/2020/issues/20
10. Link to the project: https://github.com/NaNoGenMo/2021/issues/13.
11. Source: https://github.com/verachell/Limericks-NaNoGenMo-2021/blob/main/limericks.txt.
12. Source: https://github.com/verachell/Limericks-NaNoGenMo-2021/blob/main/sentences/Line1.txt.
13. See the creator's submission at: https://github.com/NaNoGenMo/2021/issues/13.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Turku University Foundation under Grant 080971 and Emil Aaltonen Foundation.

Notes on contributor

Tuuli Hongisto is a PhD student majoring in comparative literature at the University of Helsinki. She graduated from the University of Helsinki in 2020 with comparative literature as her major (thesis topic: 'Essential narrative features in story generating algorithms'). Her PhD project focuses on the topic of reader- and authorship of computer-generated literary texts. Her research interests include electronic literature, computational creativity, and narratology.

ORCID

Tuuli Hongisto http://orcid.org/0000-0003-0228-7255

References

Ackermans, H. 2017. "Source Code: Linguistic, Literary, and Cultural Meaning-Making in Generative Literature." *Thinking Through the Digital*. http://conference.reprecdigit.se/#/contributions/hannah-ackermans.

Agafonova, Y., A. Tikhonov, and I. P. Yamshchikov. 2020. "Paranoid Transformer: Reading Narrative of Madness as Computational Approach to Creativity." *Future Internet* 12 (11): 1–12. https://doi.org/10.3390/fi12110182.

Baetens, J. 2012. "OuLiPo and Proceduralism." In *The Routledge Companion to Experimental Literature*, edited by J. Bray, B. McHale, and A. Gibbons, 131–143. London: Routledge.

Barthes, R. (1967) 1977. "The Death of the Author." In *Image, Music, Text*, translated by S. Health, 142–148. London: Fontana Press.

Booth, W. (1961) 1983. *The Rhetoric of Fiction*. Chicago: University of Chicago Press.

Dubbelman, T. 2013. "Narratives of Being There: Computer Games, Presence and Fictional Worlds." PhD diss., Utrecht University.

Eco, Umberto. 1979. *The Role of the Reader: Explorations in the Semiotics of Texts*. Bloomington: Indiana University Press.

Eskelinen, M. 2012. *Cybertext Poetics. The Critical Landscape of New Media Literary Theory*. London & New York: Continuum.

Genette, G. 1997. *Paratexts: Thresholds of Interpretation*. Cambridge: Cambridge University Press.

Gervás, Pablo. 2012. "Story Generator Algorithms." In *The Living Handbook of Narratology*, edited by P. Hühn, et al. Hamburg: Hamburg University. http://www.lhn.uni-hamburg.de/article/story-generator-algorithms.

Goldsmith, Kenneth. 2011. *Uncreative Writing: Managing Language in the Digital Age*. New York: Columbia University Press.

Henrickson, L. 2018. "Tool vs. Agent: Attributing Agency to Natural Language Generation Systems." *Digital Creativity* (Exeter) 29 (2–3): 182–190. https://doi.org/10.1080/14626268.2018.1482924.

Henrickson, L. 2021. *Reading Computer-Generated Texts*. Cambridge: Cambridge University Press.

Kybartas, B., and R. Bidarra. 2017. "A Survey on Story Generation Techniques for Authoring Computational Narratives." *IEEE Transactions on Computational Intelligence and AI in Games* 9 (3): 239–253. https://doi.org/10.1109/TCIAIG.2016.2546063.

Lebrun, T. 2017. "Who is the Artificial Author?" In *Advances in Artificial Intelligence*, edited by Malek Mouhoub, and Philippe Langlais, 411–415. Vol. 10233. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-57351-9_47.

Marino, M. 2020. *Critical Code Studies*. Cambridge, MA: The MIT Press.

Meehan, J. R. 1976. "The Metanovel: Writing Stories by Computer." PhD diss., Yale University, Computer Science.

Montfort, N. 2014. "New Novel Machines: Nanowatt and World Clock." Trope Tank Technical Report TROPE-13–03, July 2014.

Phelan, J. 2017. *Somebody Telling Somebody Else: A Rhetorical Poetics of Narrative*. Chicago: The Ohio State University Press.

Phelan, J., and P. J. Rabinowitz. 2012. "Reception and the Reader." In *Narrative Theory: Core Concepts and Critical Debates*, edited by D. Herman, J. Phelan, P. J. Rabinowitz, B. Richardson, and R. Warhol, 139–159. Columbus: Ohio State University Press.

Pope, J. 2020. "Further on Down the Digital Road: Narrative Design and Reading Pleasure in Five New Media Writing Prize Narratives." *Convergence (London, England)* 26 (1): 35–54. https://doi.org/10.1177/1354856517726603.

Rabinowitz, P. J. 2018. "The Intention Debates." In *A Companion to Literary Theory*, edited by D. H. Richter, 87–99. Newark: John Wiley.

Rettberg, S. 2018. *Electronic Literature*. Cambridge: Polity.

Richardson, B. 2007. "Singular Text, Multiple Implied Readers." *Style (University Park, PA)* 41 (3): 259–274.

Richardson, B. 2012. "Reception and the Reader." In *Narrative Theory: Core Concepts and Critical Debates*, edited by D. Herman, J. Phelan, P. J. Rabinowitz, B. Richardson, and R. Warhol, 155–159. Columbus: Ohio State University Press.

Roine, H.-R. 2019. "Computational Media and the Core Concepts of Narrative Theory." *Narrative (Columbus, Ohio)* 27 (3): 313–331. https://doi.org/10.1353/nar.2019.0018.

Schmid, W. 2010. *Narratology. An Introduction*. Berlin: de Gruyter.

Siltanen, E. 2016. *Experimentalism as Reciprocal Communication in Contemporary American Poetry: John Ashbery, Lyn Hejinian, Ron Silliman*. Amsterdam: John Benjamins Publishing Company.

Thoss, J., A. Ensslin, and D. Ciccoricco. 2018. "Narrative Media: The Impossibilities of Digital Storytelling." *Poetics Today* 39 (3): 623–643. https://doi.org/10.1215/03335372-7032788.

Turner, S. R. 1994. *The Creative Process: A Computer Model of Storytelling and Creativity*. Hillsdale, NJ: Lawrence Erlbaum Associates.

van Stegeren, J., and M. Theune. 2019. "Narrative Generation in the Wild: Methods from NaNoGenMo." *Storytelling*, 65–74. https://doi.org/10.18653/v1/W19-3407.

Wardrip-Fruin, N. 2012. *Expressive Processing: Digital Fictions, Computer Games,*

and Software Studies. Cambridge, MA: MIT Press.

Weizenbaum, J. 1966. "ELIZA-a Computer Program for the Study of Natural Language Communication between Man and Machine." *Communications of the ACM* 9 (1): 36–45. https://doi.org/10.1145/365153.365168.

Wimsatt, W. K., and M. C. Beardsley. 1946. "The Intentional Fallacy." *The Sewanee Review* 54 (3): 468–488.