DHQ: Digital Humanities Quarterly
2023
Volume 17 Number 2

Poetry as Code as Interactive Fiction: Engaging Multiple Text-Based Literacies in *Scarlet Portrait Parlor*

Jason Boyd <jason_dot_boyd_at_torontomu_dot_ca>, Toronto Metropolitan University https://orcid.org/0000-0002-2752-6729

Abstract

In Prismatik’s *Scarlet Portrait Parlor* (2020) poetry and code uncannily appear one and the same. This results in a work that is both familiar and strange, and this, along with*Scarlet Portrait Parlor*’s brevity, simplicity of construction, and immediate recognizability as a work of literature (a sonnet) that is also executable source code producing a work of electronic literature, has the potential to intrigue students and textual scholars unfamiliar with and perhaps resistant to Critical Code Studies (CCS). A study of Prismatik’s work also has the potential to refine some simplistic judgements in CCS scholarship about the efficacy of code that emulates natural, human language. This case study aims to elaborate the value of *Scarlet Portrait Parlor* as a rich example of how poetry, programming, and interactive fiction can be intertwined if not blurred in a single text and to act as a catalyst for generative discussions about the overlapping and intertwining of natural languages, programming languages, creative writing, and coding.

One of the goals of Mark C. Marino’s *Critical Code Studies* is to persuade humanities scholars that the code that operates around us everywhere in our daily lives — mundane, but life-shaping code — is an interesting, important, and intellectually rewarding focus of inquiry in the humanities: as Marino observes, this class of computer code demands humanistic study because it constitutes “an ideology that is doubly hidden by our illiteracy and by the very screens on which its output delights and distracts” [Marino 2020, 40]. Given Marino’s concern that Critical Code Studies (CCS) should not be limited “to the study of code written as literature” [Marino 2020, 41], it is not surprising that *Critical Code Studies* deemphasizes so-called “natural language” programming languages and the making of “code that has aesthetic value and additional meaning” [Marino 2020, 41]. Chapter 7, while it does discuss code as poetry, deliberately chooses as its case study Nick Montfort’s poetry generator *Taroko Gorge* [Montfort 2009], the code of which (written in the programming language Python 2) is considerably removed from any resemblance to natural language texts. Marino argues that *Taroko Gorge* is “code that generates a work of electronic literature, or rather code that is a work of electronic literature” [Marino 2020, 204] — a claim meant to disrupt and challenge traditional understandings of what “literature” and “poetry” are. 1

To claim, as Marino does, that *Taroko Gorge* is a work of literature is provocative because its source code and traditional creative writing appear to be strikingly different if not inimical: that stark opposition is useful for those wishing to breach as well as those wishing to defend the wall that both sides see separating “code” and “literature” in popular and scholarly discourse. In this paper, however, I suggest that Prismatik’s *Scarlet Portrait Parlor* [Prismatik 2020][1] — a traditionally-structured poem which is simultaneously a work of functional code and, when executed, a work of interactive fiction — is arguably even more provocative for the opposite reason: poetry and code uncannily appear one and the same, and thus the presumedly obvious wall between them is not easily located for tearing down or reinforcing. This results in a work that is both familiar and strange, and this, along with *Scarlet Portrait Parlor*’s brevity, simplicity of construction, and immediate recognizability as a work of literature that is also executable source code producing a work of electronic literature, has the potential to intrigue students and textual scholars unfamiliar with and perhaps resistant to CCS. A study of Prismatik’s work also has the potential to refine some simplistic judgements in CCS scholarship about the efficacy and value of code that emulates natural, human language (in this case, the English language). 2

3 This case study of *Scarlet Portrait Parlor* aims to elaborate the value of the work as a rich example of how a poem, program, and interactive fiction (IF) can be intertwined if not blurred in a single text and show that the different reading competencies and literacies proper to each can enable a fuller appreciation of the work's operations and meanings. Marino asserts that “[c]ode should be read, and read with the kind of care and attention to detail that a programmer must use” [Marino 2020, 31]: *Scarlet Portrait Parlor* additionally must be read with the kind of care and attention that a scholar of poetry and an experienced player of interactive fiction must use. As an *amalgam* of poetry, code, and playable story, it is a useful example with which to interrogate some questionable assertions and arguments in *Critical Code Studies* about the relations between programming and human languages and, in particular, about so-called “natural language” programming languages and about Inform 7, an example of this type of programming language. I hope this case study will bring some added nuance to future discussions in CCS at the intersection of human and computer languages and coding and creative writing. Prismatik’s work is a compelling example of how natural language programming languages can be flexible enough to be used for writing that is both expressive in terms of human language and functional in terms of executable code and points to a possible future where the paths of writing and programming (as conventionally understood) are not necessarily divergent.

Scarlet Portrait Parlor: Overview

4 *Scarlet Portrait Parlor* (SPP) is short enough in length that it can be quoted here in full:

```
"Scarlet Portrait Parlor" by Prismatik             [A]

[A sonnet]                                         [B]

The Scarlet Portrait Parlor is a room              [1]

When play begins: say "Darkness falls again."      [2]

Inside the Portrait Parlor is a loom.              [3]
"A [loom] that weaves the inner thoughts of men."  [4]

The heavy guilt is carried by the player.          [5]

Instead of dropping guilt: say "You cannot!";      [6]

Below the Portrait Parlor is a Lair;               [7]

Inside the Lair are portraits left to rot;         [8]

[Now,] understand "your secrets" as the portrait;  [9]

[Now] understand "your actions" as the loom;       [10]

"Our mind and thoughts compose the awful fortress. [11]

To have our deeds exposed is mankind's doom."      [12]

Instead of putting [all your dark] guilt on:       [13]
        End the story finally saying "It's done."; [14]
```

[Prismatik 2020][2]

Scarlet Portrait Parlor is authored in Inform 7, often described as a natural language programming language because, as the text of *SPP* demonstrates, it closely resembles English syntax and sentence construction. As a submission to a competition the objective of which was to create a work in Inform 7 “with beautiful source code text” [Veeder 2020a], the composition of the program was at least as important if not more important than its executable (playable) functionality: the source code in and of itself, that is, was to be read and judged on its aesthetic merits beyond its functionality as a program. This “extra-functional significance” [Marino 2020, 34] is a key interest of CCS, which aims to study “how code serves as a communication medium including and beyond the realms of practical application and specific machines” [Marino 2020, 8]. 5

In *Critical Code Studies*, Marino recounts the reaction of programmers when he tells them he wants to interpret their code: “they suspect I want to read their code as an English major would read…a sonnet by Shakespeare. Will I treat their methods as stanzas? Their routines as rhymes about roses?” [Marino 2020, 29]. This amusing misunderstanding arises due to the programmers’ assumption that literary scholars only approach texts as “literature” (even when they are not “literature”). However, in the case of *Scarlet Portrait Parlor*, such an otherwise laughable approach to code as poetry is not only appropriate but required. The SPP source code is in the form of an English sonnet, and it has been intentionally composed to communicate a coherent meaning or message as a sonnet. As such, it requires knowledge of poetic form and skill in interpreting verse in order to be effectively evaluated. However, to fully understand the constraints faced by the poet/programmer of this sonnet/program requires more than just a facility with parsing verse: it requires a knowledge of the syntax and structures of Inform 7 in order to understand how the programming language is present in and has shaped the sonnet and its meaning. 6

But *Scarlet Portrait Parlor* is more than a literary work written under the double constraint of the English sonnet form and valid Inform 7 syntax: as a computer program, it is also a work that can be run and played using an interpreter. In this state, the reader must have some knowledge of the navigational conventions of parser based IF as well as how Inform 7 and the interpreter respond to reader input. Having read the sonnet/program and played its IF execution, and, with this dual knowledge, going back and forth between these states for further revelation, the intriguing challenge for the reader is to develop an argument about how these three states of *Scarlet Portrait Parlor* together contribute to the work’s meaning. It is this need to oscillate between SPP’s states that makes the work valuable for thinking about how literary form, code structures, and parser-based texts can interdependently shape and shift how a text’s meaning is constructed, as well as how it is read and interpreted. 7

Scarlet Portrait Parlor as a Sonnet

As the subtitle of the work explicitly (but perhaps unnecessarily) indicates (line B), SPP is a sonnet, “one of the oldest, strictest, and most enduring poetic forms” [Richardson]. More precisely, it is an English sonnet: it consists of fourteen lines of iambic pentameter, that is, five feet — a foot being comprised of two syllables — with the stress or emphasis on the second syllable of each foot (known as an iamb) rhyming *ababcdcdefefgg*. Another way to describe the structure of the English sonnet is to say that it consists of three quatrains (four-line stanzas) of alternating rhyme with a closing rhyming couplet (two lines). Thus, in terms of compositional form or structure, Prismatik sets himself a considerable *poetic* challenge, as a sonnet is difficult to compose both in terms of its formal requirements and its relative brevity. How successful is *Scarlet Portrait Parlor* in rising to this challenge? It is fourteen lines, it does largely follow the English sonnet rhyme scheme, and it is mostly written in iambic pentameter. In addition to the “forced” or “eye” rhyme of “on” and “done” in the couplet, one might look askance at the rhyming of “portrait” and “fortress” in lines 9 and 11 — another forced end rhyme, although the first (and stressed) syllables of the words (port-, fort-) do rhyme. These same lines are the only two in the sonnet that have so-called “feminine” endings (an extra unstressed syllable). A final noticeable departure from strict English sonnet form is the final line, which consists of four trochees (stressed-unstressed feet, the reverse of the iamb) and a final iamb (or perhaps spondee, if it is felt that the two final syllables should both be stressed equally). These departures from strict English sonnet form should not be presumed to indicate a poetic failing or incompetence on Prismatik’s part, as deliberate and careful modifications to the form are often the hallmark of great sonnets and sonneteers. That lines 9 and 11 are a rhyming pair in the same quatrain and that they are in the last quatrain before the concluding couplet, and that the switch from iambic is confined to the final line, are potentially signs of the *volta* or “turn” that is expected in the traditional sonnet (an issue explored later in this paper). 8

Scarlet Portrait Parlor as Code

Parsing SPP as source code requires a shift to a perspective where the versifier has become a coder and where the speaker of the text is addressing not a human but a computer that is tasked with executing the compiled source code. A knowledge of the Inform 7 programming language enables one to discern the extent to which the sonnet's language and format, and thus message, is shaped by and constrained by code. One aspect of the SPP text is difficult to incorporate into any reading of it as poetry: the punctuation. This includes the use of commas, semicolons, and periods, line spacing and indentation, as well as quotation marks and square brackets. As the use of much of the punctuation appears to be a response to the text-as-sonnet as it overlaps the text-as-code, it offers a useful segue way into a consideration of *Scarlet Portrait Parlor* as Inform 7 code. 9

The punctuation is at times puzzling in terms of the meaning of the text as a sonnet: for example, why not use a comma or colon instead of a period in line 3? Why not use a semicolon instead of a period in line 5? Why include a semicolon at all in line 6? Why not use a comma instead of a period in line 11? (The semicolon that oddly concludes SPP is possibly a mistake). Also puzzling is that most of the text's lines (except for lines 3-4 and 13-14) are separated by blank lines, which is not a poetic convention. In Inform 7, however, it *is* a convention to separate discrete code statements or blocks with blank lines for ease of reading and debugging (see the *Inform Recipe Book* for examples [Nelson a]). The end of a code block can be indicated without punctuation (as we see in line 1 of SPP) or with a semicolon, but only if it is followed by a blank line, which Inform 7 understands as designating the ending of the previous code block. Therefore, if one wanted to eliminate the blank lines in *Scarlet Portrait Parlor*'s source code and still have an executable Inform 7 program, one would have to end each line (except line 13) with a period (the exclamation mark within the quotation marks in line 6 is considered equivalent to a period, as would be a question mark). The decision to end lines 6 to 10 with semicolons, as well as the decision to not put a blank line between lines 3 and 4, are stylistic choices informed by the text as a sonnet: in an attempt to reinforce SSP's status as a sonnet, Prismatik has opted to use semicolons to convey a sense of flow/connection between lines (although this sense ends up being more visual than semantic) and has consequently been obligated to use unpoetic blank lines. Although the indentation of the final line could be regarded as a form of typographical emphasis, conventionally both lines of the couplet would be indented in laying out a sonnet typographically. While in this instance it is unnecessary, the indentation of the final line is due to the convention (and often a requirement) in Inform 7 that conditional statements indent those portions that follow an initial line that ends with a colon. In this instance, the couplet constitutes an if/then statement. 10

In addition to the uses of punctuation explained above, quotation marks, square brackets, and the remainder of the text not enclosed in this punctuation indicate the differentiation of the three basic types of content in Inform 7 source code. Content in quotation marks designates the creative writing originated by the writer/coder and is outputted during play in response to a player's commands for the player to read. Quotation marks, then, from a source code perspective, are not used as they are in conventional texts (for example, to designate dialogue or quotations). Content in square brackets (never used in traditional poetry) are comments (i.e., annotations of the source code, a standard feature in programming languages) meant solely for the coder to read; they are ignored when an Inform 7 program is compiled and run by an interpreter. One exception to this is the use of square brackets *within* quotation marks (as seen in line 4), which are used for modifying the display of content within the quotation marks when played as an IF. Content neither in quotation marks nor square brackets are instructions for the interpreter to execute, which IF readers do not see, at least in the form in which they appear in the source code. Overall, most of the punctuation in SPP is indicative of the demands of Inform 7 rather than the meaning-making needs of the sonnet. 11

Turning from punctuation to word and phrase choices, many of the words in SPP have an extra level of significance in Inform 7 that goes beyond their dictionary meanings. Inform 7 was designed to create parser-based interactive fiction (IF) and creating an IF typically consists of building a virtual environment (spaces, objects, living creatures) that "the player" (the IF reader) moves in and interacts with, usually by playing some kind of entity who can interact with the environment. This explains the first line, which seems unnecessary to the sonnet reader but is necessary for an executable Inform 7 program: a "room" is one of the fundamental object "kinds" in Inform 7 (the source code "needs at least one name of a location where the drama can unfold. For reasons of tradition, such locations are normally called 12 ‘rooms,’ though people have used them to represent anything from grassy fields to states of mind and other metaphorical places” [Nelson b]. Inform works usually consist of a collection of rooms linked by directions (another object kind). The virtual space of SPP is quite modest: a Parlor, below which is a Lair.

Part of Inform’s flexibility in terms of program composition is that it can make inferences about the status of named objects in the program: for example, it is not necessary to include a phrase stating (as was done with the Parlor) that the Lair “is a room” because Inform 7 assumes that an object that is connected to an existing “room” by a direction (in this case, “Below”) is also a room (line 7). These two rooms each have a single object (a *thing* in terms of Informs 7’s object kinds) in them: a “loom” (line 3) and “portraits” (line 8): Inform 7 automatically infers objects that are stated to be “in” or “inside” objects that it knows to be rooms are *things*. It is this kind of built-in understanding, along with the flexibility in which commands can be phrased, that enables Prismatik to write a highly constrained form of poetry like the sonnet in Inform 7, whereas a programming language with more rigid syntax would make this unachievable. 13

Lines 2 and 14 use two very common phrases for the start and end of an Inform 7 work: “When play begins” and “End the story finally”. The first phrase represents a rule that causes something to happen when an IF is newly run: in this case an epigraph or preamble to the IF is outputted (using “say”). The second phrase represents a rule that ends the story with a message when a certain condition is true, in this case, when the player tries to put the “guilt” on any thing in the environment (as no particular “thing” is specified after the “on” [line 13]). The “say/saying” rule in Inform 7 (seen in lines 2, 6, and 14) is equivalent to the “print [on screen]” command in other programming languages. The “understand” rule (lines 9-10) is commonly used to anticipate possible variations of how a player might refer to objects (e.g., Understand “phone” as the telephone) or how they might describe actions they wish to take (e.g., Understand the commands “dial” or “phone” or “telephone” as “call”). 14

Prismatik uses one feature of Inform 7 — the built-in properties of objects — very effectively with the “guilt” of the text. SPP states that this guilt is “carried by the player” (line 5, a standard Inform 7 operation), and, consequently, the “guilt” forms part of the player’s *inventory*. While colloquially and figuratively we talk about degrees of guilt in terms of a physical quality like weight (“heavy”) and as something that one “carries” around with them (as a psychological “burden”), in terms of the SPP text as source code, guilt is a generic and material *thing* object (therefore, no different in kind than the loom or the portraits), and as such, it has the default property *portable* (it can be taken and dropped). Prismatik references the *portable* property in the rules about how guilt can be manipulated in lines 6 and 13. These use the *instead* rule for what is essentially an if/then conditional: if the player tries to drop the guilt they are carrying, or tries to put it on anything in either of the rooms, they will be prevented from doing so, and a message reacting to the attempt will be outputted (“You cannot!” [line 6]). 15

Examining the text of SPP from the dual perspectives of an English sonneteer and an Inform 7 programmer is revealing in several respects. It becomes clear that the poetic and programming constraints of each shaped the particularities of the text. For example, line 5 would have been valid in Inform 7 if it had been written as “The player carries the heavy guilt,” but this would have made the line one syllable short and not preserved the iambic pentameter. Therefore, the line had to end with “player.” In terms of word choice, like the choice of “loom,” which was dictated by the need for a rhyme for Inform 7’s object kind “room,” “Lair” was dictated by the programming rather than poetic conceit (the need to rhyme with “player”). The order of the lines/code blocks is primarily dictated by the rhyme scheme rather than a logical programming structure: the lines could be placed in a number of different arrangements (e.g., once editing the lines so that they end with periods, 1, 3, 4, 7, 8, 11, 12, 5, 6, 2, 13, 14) and still be executable and function as an Inform 7 program. Of course, it then would no longer be a sonnet. 16

Reading *Scarlet Portrait Parlor* as a Sonnet

Now that we have looked at how Inform 7’s syntax constrains the punctuation and wording/phrasing of the sonnet, we can ask: does this interfere with SPP’s ability to convey a meaning or message as a sonnet? The first quatrain (lines 1-4) describes a “scarlet portrait parlor” containing a loom “that weaves the inner thoughts of men” (presumably into portraits). While the first line is unnecessary in terms of the meaning of sonnet, as it hardly needs to be pointed out that a parlor “is a room”, as we have seen, it is demanded by the Inform 7 programming. The second line may appear 17 unconnected from the other lines in the quatrain — what does “When play begins: say ‘Darkness falls again.’” signify? What is the “play”? Is it weaving on the loom? And who is telling whom to “say”? Is it the speaker telling themselves or the speaker telling an addressee (as sonnets often have a speaker addressing a person or an idea)? The first line of the second quatrain (line 5) introduces “the player,” who could be viewed as the sonnet’s addressee and who is earlier told (by the speaker) to “say” when “play begins.” The next line tells the player that “You cannot!” drop the heavy guilt that they are carrying (lines 5-6): is this an imperious command or a despairing realization? The sonnet goes on to describe a “Lair” below the parlor (line 7) where there are “portraits left to rot” (line 8) and assists our interpretative work by telling us directly (lines 9-10) that the loom symbolizes “your actions” and the portrait[s][3] “‘your secrets’” (earlier referred to — assuming these portraits are what are woven on the loom — as “the inner thoughts of men” [line 4]). If the player’s actions (weaving) risk the exposure (through portraiture) of the player’s secrets or inner thoughts, then this is perhaps why the portraits end up rotting in the Lair (a type of place associated with dangerous beasts and criminals). It is “Mankind's doom” to have those rotting portraits revealed when others breach “the awful fortress” (alias the “Lair”) that encloses and protects our “mind and thoughts.” Replacing the “your” of lines 9, 10, and 13 with “our” in line 11 turns this statement into a kind of universal adage. As the closing couplet (lines 13-14) reminds us, the player/weaver carries guilt (caused by their inner thoughts/secrets being revealed by their actions?) and suggests that instead of “putting [all your dark] guilt on” (on a portrait?) that they stop (“End the story finally”) and state “‘It’s done.’” — the “it” possibly being a portrait being composed or an action that has already been taken. Is the “turn” then in the couplet, by telling the addressee/the player that instead of burdening themselves with guilt over their actions exposing their secrets, they find closure by saying “what’s done is done”? In this interpretation, if it is “mankind’s doom” (i.e., inevitable fate) to have the motivations (“mind and thoughts”) behind its deeds exposed, then why feel guilty about this exposure, especially once the deeds have been done?

This parsing and interpretation of *Scarlet Portrait Parlor* suggests that it is a sonnet about how we (presuming the use of “men/mankind” is an outmoded way of referring to humanity) burden ourselves with self-representations that are informed by what our actions might reveal about our inner selves, our character, but that we should accept that actions *will* reveal character and stop feeling guilt or shame about this revelation, especially for actions that are already done. While this idea (actions as an index of moral character/psychological motivations) is not an original one, it is one worthy of expression and consideration. In terms of the artistry with which this idea is expressed, the contrasting symbols of parlor (a space of respectable sociality) and lair (a space of wildness, secrecy, and danger) and of the loom and weaving (a familiar trope for life and fate, from the Greek *Moirai* to Tennyson’s “Lady of Shallot”) are very striking, as is the imagery of the “portraits left to rot” [Prismatik 2020, line 8] (reminiscent of Oscar Wilde’s *The Picture of Dorian Gray*), the mind as a fortress, and psychological guilt as a physical burden. The modifications to the standard English sonnet scheme (discussed above) appear to be skillfully done to highlight the “turn” where, after the depiction of the parlor/lair in the first two quatrains (the proposition), the third quatrain (with the distinguishing feminine rhyme) leads to the resolution of the couplet (with the trochaic emphasis in the last line). One might criticize the lack of enjambment [Poetry Foundation] [4] and the inconsistency of the use of imagery relating to the arts, flitting from fine arts (portraiture) to artisanal crafts (weaving) to storytelling (line 13) and to theatre (assuming the Shakespearean premise that “all the world’s a stage” and interpreting “the player” as an actor), as well as fault the poem for explicitly decoding the symbolism of the portrait and loom in lines 9-10. However, no matter the specific interpretation, this analysis demonstrates that *Scarlet Portrait Parlor* can legitimately and profitably be considered and studied *as a sonnet*. 18

Reading *Scarlet Portrait Parlor* as “Beautiful” Code

Scarlet Portrait Parlor was an entry in *Event One of the Second Quadrennial Ryan Veeder Exposition for Good Interactive Fiction* (and won third place). Event One’s challenge was “to create a game in Inform 7 with beautiful source code text” [Veeder 2020a]. Veeder notes that Inform 7 was in part chosen because “Inform 7 code has a unique potential to be beautiful by way of resembling beautiful English sentences (as well as many other vectors of beauty)…” [Veeder 2020a]. As Veeder admits, what makes “beautiful code” is debatable, and he proposes two possibilities: cleverness and elegance [Veeder 2020a]. But his earlier comment about choosing Inform 7 suggests that style and expression (in the context of composition in the English language) is also a factor. While cleverness and elegance are important aspects of style in writing in terms of judging its beauty (wittiness, originality, rhetorical impact, use of 19 figurative language) they mean differently in terms of coding, where cleverness and elegance usually refers to brevity, clarity, simplicity, correctness or validity, reusability, and modifiability. In his very brief judgement of SPP, Veeder commends its compactness and correctness, and notes that he deducted points for a few “inelegancies” (for a score of 8/10) [Veeder 2020b]. He does not state what these inelegancies are, but, given that compactness and correctness are qualities more valued in code than verse (although one might argue they are as if differently valued when it comes to a form like the sonnet), these are probably features like the use of comments in lines 9, 10 and 13, which could be criticized as a misuse of the commenting function: using comments to “pad” lines to meet the metric requirements of the sonnet instead of finding an executable coding solution. Another possible inelegancy Veeder might have considered when scoring SPP is line 9, which is non-functional, since “portrait” has not been created as a *thing*, given that it has not been placed in a room or on the player as have the loom (line 3), guilt (line 5), and the portraits (line 8). Inform 7 understands single and multiple things as different objects, even if they have the same name (“portrait” ≠ “portraits”) (see the “Scarlet Portrait Parlor as Interactive Fiction” section for a further discussion of this line).

20 Some questions Veeder does not address, but which would be pertinent and interesting to ask in the case of a CCS approach to SPP is: Can code be considered beautiful because it takes the form of a poem? Or does a carefully crafted sonnet that is also an executable program only lead to inelegant code? Or does such code inelegance become beautiful because it is structured poetically while still remaining executable? While it is commonplace to think of poetry as open and malleable and coding as limited and rigid, in this case it is the poetry that its limited and rigid (the English sonnet form) and the coding that is malleable enough to accommodate itself (while obeying its own constraints) within the sonnet’s constraints. Does SPP demonstrate that Inform 7 code is beautiful because it shows that Inform 7 programming can also be used for creative writing for the production of traditional and electronic literature?

21 These questions could also have been asked in *Critical Code Studies*, in particular in the discussion of Inform 7 in Chapter 5, which examines the FLOW-MATIC programming language in order to mount an argument reinforcing the distinctions between natural language programming languages, the human languages they try to emulate, and “good” programming. FLOW-MATIC was designed to use English words and phrases in order to make it more accessible to non-programmers. Marino notes that FLOW-MATIC programs are difficult to read today because they lack now-common programming structures and argues that this “illustrates the ways legibility is not dependant on similarity to natural language but overall clarity of expression” [Marino 2020, 144]. As an example of FLOW-MATIC’s lack of clarity of expression, Marino provides the example of the operations REWIND and ZZZZZZZZZZZZ, which, he argues, “demonstrates the contradictions of this English-like programming language because the program essentially uses both ZZZZZZZZZZZZ and REWIND to indicate that instruction [the rewind instruction]” [Marino 2020, 146].[5] When considering Inform 7, Marino’s judgement that having different words or methods for a single operation are “contradictions” is a mistaken one. Richness or diversity of expression in a programming language is only indicative of the lessening of a “dedication to legibility” [Marino 2020, 146] if one starts from the dubious premise that programming languages, including natural language programming languages, should strive for (or can only function using) a one-to-one equivalence between named operations and methods and what those operations and methods do. Inform 7 shows that need not be the case and that is a premise that CCS should interrogate, not codify: as with natural languages and legibility, there is no essential problem with programming languages using synonyms for the same operation (“carrying”, “holding”) or variant syntax for instructions (“The player carries the heavy guilt”, “A heavy guilt is carried by the player”). In these cases, it is the very similarity to natural language (English) that makes it easy to discern that these different words and phrases are doing the same work. Perhaps the problem with FLOW-MATIC was not that it used English words and phrases, but that it did not use them more extensively, more “naturally,” in the way Inform 7 does.

22 In his efforts to demonstrate that “the distance between the appearance of English and the processing of natural language offers a valuable lesson in the dangerous temptation to read code by merely imposing reading strategies from other semiotic systems” [Marino 2020, 135], Marino ends up creating a problematic example in Inform 7 to illustrate that “natural-seeming programming languages can lull a newbie into thinking the language can understand a statement rather than process it” [Marino 2020, 150]. First, the *Writing with Inform* manual provides a direct and unambiguous response to the question, “Does Inform really understand English?”:

> No. No computer does…[Inform] is a practical tool for a particular purpose, and it deals only with

> certain forms of sentence useful to that purpose. Inform source text may look like “natural language,” the language we find natural among ourselves, but in the end it is a computer programming language. Many things which seem reasonable to the human reader are not understood by Inform [Nelson b, 2.16].

Note that here Nelson is saying that Inform does not understand English in all its complexities and vagaries; he is not saying that Inform does not understand what might be called “Inform 7 English”: Inform *needs* to understand this English in order to process it.

23 Second, the example that Marino provides of a sentence Inform does not understand (while we do) is: “‘The door in the room needs to be fixed.’” [Marino 2020, 150]. This puzzling example fails to pay attention to context. Like writing a sonnet, like writing a program, writing an IF is a very specific writing situation with very specific goals, with a “particular purpose,” as Nelson puts it. It is difficult to imagine a situation where one would actually want to write such an instruction in Inform 7 as Marino provides. As doors in Inform 7 do not independently fall into disrepair or get broken without the author’s authorization, if the IF author did not want a door that needed to be “fixed” in their IF, they simply would not create such an object, and therefore would never need write Marino’s example instruction (and what exactly does Marino expect the interpreter would do with such a statement if it understood it?). If such a sentence *was* included in an Inform 7 work, it would likely be as an instruction not to the computer but to the human player, to prompt them that they need to do something to fix the door, e.g.,

24 Instead of going through the door: say, “The door in the room needs to be fixed.”

25 The above example, and the Inform 7 language in general, complicates Marino’s claim that there is a significant “difference between understanding and parsing and processing” [Marino 2020, 150]. Rather, what we see happening simultaneously are two levels of understanding: the author/reader understands the instruction as an English phrase (however curious in construction) and, *at the same time,* as a piece of code that, once it is processed (“understood”) by the interpreter, will produce a specific output in response to a specific player input. With Inform 7, understanding its code is the same as understanding its English. The English (and the code, for that matter) is understood and understandable in the specific writing situation of creating an Inform 7 program: this makes Marino’s instruction “‘The door in the room needs to be fixed.’” of little sense and his hapless “newbie” make-believe.

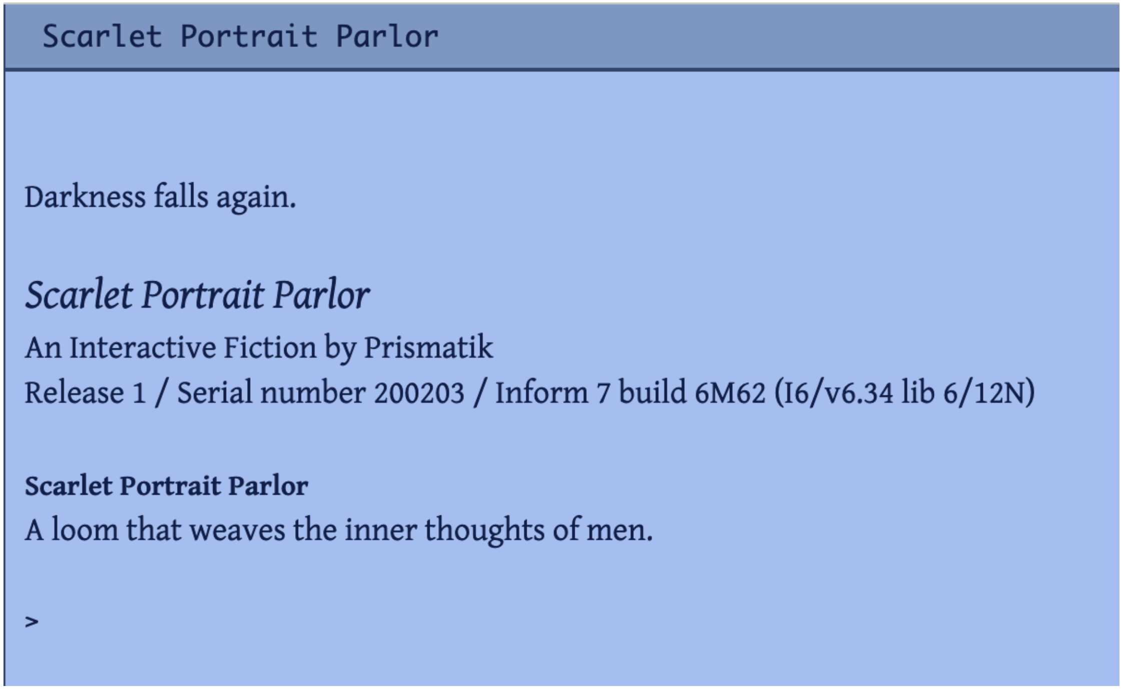
26 One of Marino’s aims in chapter 5 of *Critical Code Studies* is to show that “natural language” programming languages are, counter-intuitively, less legible, less clear in expression than more “code-like” languages. As shown above, there is a dubious premise in his argument on this front in relation to FLOW-MATIC, and his use of Inform 7 to bolster that claim instead weakens this argument further. Part of the problem with Marino’s use of Inform is that he fails to remember that it is a tool to elaborate virtual worlds, that is, as a tool not for facilitating business processes, but for creative writing. Clarity of expression as it relates to a business program is not the same in relation to a program that is a form of creative writing. *Critical Code Studies* is concerned with promoting code literacy and legibility: the ability to understand the meanings and effects of code. Currently, much code is only legible to programmers. One CCS response to this lack of accessibility is to advocate for everyone to become programmers or at least to develop a functional code literacy that would enable everyone to read the programming languages now in widespread use. But another and complementary response to the demand for a general code literacy would be for CCS to study, explicate the potential of, and advocate for programming languages that strive to emulate “natural languages” so that code literacy does not require a highly specialized literacy far removed from the common literacy that most people possess.

Scarlet Portrait Parlor as Interactive Fiction

27 Inform 7 programs are doubly a form of creative writing because, when executed as a compiled story file, they become Interactive Fictions, a form of potential literature that is realized when a reader/player transverses them by typing commands into an IF interpreter [The Interactive Fiction Wiki]. While it is an expectation that players of IF will not see the source code and do not need to as part of the IF experience, in the case of SPP, the source code is meant to be read by a human for a meaning other than its status as an IF program. The interpreter does not understand this other meaning, nor does it need to: the interpreter does not know it is a sonnet (or what a sonnet is), nor does it know or

understand the message of the source code as sonnet: to the interpreter, SPP is a series of instructions creating two *rooms*, three *things* in those rooms, and a set of instructions about how the player can interact with them, as well as instructions about what to do at the start when the program is run, and under what conditions the program can end.

For an IF player who is not familiar with the source code, SPP as an IF would likely be baffling. After the epigraph ("Darkness falls again." (line 2)) and the banner text displaying the title and author (line A), "Scarlet Portrait Parlor" displays in bold (line 1), an IF convention for indicating the room the player starts in. After this comes the line "A loom that weaves the inner thoughts of men." (line 4), followed by the prompt awaiting player input (see Figure 1). That Prismatik gave some thought to how the SPP text would present via an interpreter (in other works, as an IF) is indicated in the source code by the enclosing of "loom" in line 4 in square brackets: this prevents the interpreter from displaying an automated description ("You can see a loom here.") which would have been generated from line 3, and which would be repetitive coming after and in addition to "A loom that weaves the inner thoughts of men." Normally after a room heading one would expect a description of the room that would evoke its atmosphere and general appearance as well as offer information about items in the room and paths to other spaces that would help the player determine what commands might be viable to type. The description that does appear (of the loom) would be something a reader would more typically expect *after* examining the loom, and if one does "x [examine] loom," one gets a built-in default message for items that have no author provided description: "You see nothing special about the loom." — an odd message, given that the reader has been informed the loom "weaves the inner thoughts of men." 28



```
Scarlet Portrait Parlor

Darkness falls again.

Scarlet Portrait Parlor
An Interactive Fiction by Prismatik
Release 1 / Serial number 200203 / Inform 7 build 6M62 (I6/v6.34 lib 6/12N)

Scarlet Portrait Parlor
A loom that weaves the inner thoughts of men.

>
```

Figure 1. *Scarlet Portrait Parlor*'s initial display when run using an interpreter [Prismatik 2020]

Seasoned IF players would likely at the start of play type two common commands hoping to obtain more information: "x [examine] me" (which for SPP returns the default generated message: "As good-looking as ever." — a phrase which has an ironic appropriateness for a sonnet about the unflattering revelations of self-portraiture) and "i [inventory]", which informs the player: "You are carrying: / a heavy guilt." (line 5). Although in other contexts this would be read figuratively only, as an IF, this would also be understood by the player literally, with "guilt" understood as a tangible item. A seasoned IF player will know that usually a few commands can always be tried with carried items (e.g., wearing, eating, smelling, tasting, touching, listening to, etc.), none of which generate anything beyond negatives (e.g., "You can't wear that!") except for dropping: if the player tries to "drop [or throw] guilt", they get the message: "You cannot!" (line 6), a response which, in terms of IFs, is somewhat unexpected, as usually most carriable things are (by default) droppable. 29

At this point the player, lacking guidance about how to proceed, would have to experiment. They might try the standard 30

IF directions (the eight cardinal and ordinal compass directions, plus up, down, and inside), one of which (“d” [down]) will take them, as the bold heading will indicate, to the “Lair” (line 7), followed by lines 11-12 and “You can see portraits left to rot here.” (generated automatically from line 8). They might try “taking” either the loom or the portraits and putting them in opposite rooms or together in the same room (which can be done) — but this does not have any particular effect. Prismatik almost certainly did not intend these actions to be relevant for SPP as an IF; however, because of the sonnet constraint, he was unable to specify that the loom and portraits were “fixed in place” or “supporters ”or “scenery”, which would have prevented such actions. An enterprising player might try putting the portraits on the loom or vice versa, but as these two things are not implemented as “supporters,” this would only generate a default message: “Putting things on the [loom/portraits left to rot] would achieve nothing.” If the player tries to put guilt (line 13) on either of these things, or “on me”, they would end the game with the message, in bold italic: “***It’s done.***” (line 14).

31 It would be hard if not impossible for the player to discern from such a playthrough that the source code was a sonnet, even though the playthrough suggested above more or less follows the line order (2, 1, [3], 4, 5, 6, 7, 11, 12, 8, 13, 14). Two lines, however, provide evidence that a knowledge of the source code is expected on the part of the IF player, unless this is intended as a particularly unfair puzzle: lines 9 and 10. It seems unlikely that a player would be able to figure out without knowing the source code that they could also refer to the portrait as “your secrets” and the loom as “your actions.” Knowing this enables the player to type the command “put guilt on your actions,” which ends the story in the same way as in the previous paragraph. The argument might be made here that “portrait” in line 9 is intentional (that is, not a mistake for “portraits”) in that it prevents the command “put guilt on your secrets” from succeeding: since a thing called “portrait” has not been created in the source code, such a command returns the default generated message “You can’t see any such thing.” Was this done because secrets are things that are hidden, making the default generated message particularly apt? If the “portrait” is one that is in process of being created on the loom in the Parlor, then its secrets are only in the process of being exposed, and this suggests that, when one has woven a portrait and revealed one’s “inner thoughts,” only then does the portrait join the others rotting in the Lair, on which the player can put the heavy guilt resulting from secrets and “deeds exposed.”

32 The activity involving guilt is the primary interactivity of SPP and putting “[all your dark] guilt on” the loom, the portraits, your actions, or the player all generate the same ending (“It’s done.”). Are these all meant to be “correct” solutions or endings, or is there one “correct” solution or perhaps a “best” solution? In terms of the source code, the player is prevented from putting guilt on *anything*, given the use of the “Instead of” rule rather than the “After” rule (e.g., “After putting guilt on…”) as well as because the loom, portraits and player are not designated as *supporters* on which things can be put. Therefore, the “It’s done” ending, which an IF player might legitimately interpret as meaning “It’s accomplished,” might not mean that when one knows the source code. There, what the “it” of “It’s done” in the concluding couplet/if-then statement in lines 13-14 refers to is ambiguous: does it means you cannot (or perhaps should not) put your guilt on your actions because “the story” is already done, over, in the past, irrevocable?

33 The final two lines of SPP encapsulate how an understanding of the work is deepened and complicated by reading them simultaneously as the conclusion of a sonnet, program, and an IF. In terms of a sonnet, the text seems to call for a relinquishing of, or a refusal to be burdened with, a continual narrative of guilt or shame over what our actions might suggest about our “secret” inner life. However, as Inform 7 source code, guilt cannot be dropped, nor can it be put on anything (and there is nothing to put guilt on in any case in the source code), and this suggests a much more pessimistic, even fatalistic, response to the couplet of the sonnet, where the compiled program will abruptly terminate itself in response to *any* attempt by the player to get rid of guilt. As an IF, the message depends on what the player decides to put guilt on: the loom that weaves the inner thoughts of men, the portraits left to rot, or themselves. Putting the guilt on themselves or on the loom (“your actions”) would in this case be an enactment (if “It’s done” is read as “It’s accomplished”) of the very thing the sonnet and the Inform 7 program counsel against.

Conclusion

34 As I hope this explication of *Scarlet Portrait Parlor* conveys, what makes this work so challenging and so interesting is that it requires the reader to draw upon and engage multiple modalities and literacies to construct a reading. Ultimately, the value of this work is not necessarily that it provides some unitary message across its multiple modalities, but that it

provides a powerful and instructive example of creative procedural authorship that is working within multiple constraints (the English sonnet form, Inform 7 programming structures, and parser-based interactive fiction). As such, it is a work that is immensely valuable in revealing how literature, programming, and interactive digital work can intersect, and thus shows the value and range of Critical Code Studies. While Marino's intent in *Critical Code Studies* is not to delegitimate so-called "natural language" programming languages and "code written as literature" [Marino 2020, 41] as a focus of study in CCS, his study perhaps does not do justice to the potential this area has to turn what is seen as the wall separating "code" and "literature" into a bridge bringing together code and literature into a continuum that can lead to generative discussions about the overlapping and intertwining of programming languages, natural languages, creative writing, and coding. One important point of future discussion relates to whether natural language programming languages might be a means of synthesizing programming and other forms of human writing and of achieving a more widespread code literacy and code creativity than is the case given predominating programming paradigms and syntaxes.

Notes

[1] "Prismatik" is a pen name of "MathBrush" (a pen name of Brian Rushton). On the *Interactive Fiction Database* (IFDB) page for *Scarlet Portrait Parlor* (https://ifdb.org/viewgame?id=t15dwwysqvz2ptug) the work is listed as "by MathBrush (as Prismatik)."

[2] The letters and numbers in square brackets have been added and will be used in this paper to reference lines.

[3] The "portrait" in line 9 might appear to be a mistake for "portraits," if one assumes that this "portrait" is referring to the "portraits left to rot" of line 8, One could also assume that this "portrait" may refer to one currently being woven on the loom. For a further discussion of the portraits/portrait, see the "Scarlet Portrait Parlor as Interactive Fiction" section.

[4] In terms of meaning, lines 1 and 2 do not seem to be indicative of enjambment (the "running-over of a sentence or phrase from one poetic line to the next, without terminal punctuation" [Poetry Foundation]) — that is, we are not to read "The Scarlet Portrait Parlor is a room when play begins," which implies that the parlor will become something other than a room at a later time, which it does not. Although the first line is not end-stopped, in terms of the meaning of the text as poetry, it seems to demand some end punctuation.

[5] While it can be argued that, unlike REWIND, a series of 12 Zs is not a generally transparent command, it does have a certain logic (perhaps the logic that suggested its original development) given that, if such a word existed, it would occur at the very end in an alphabetized categorization, with the only option being to go back or 'rewind' to an earlier section. One could also speculate that a series of 12 Zs would be very visually distinctive when parsing a FLOW-MATIC program, which perhaps made it more useful in some contexts than REWIND. As I go on to argue, a consideration of the writing context is essential when evaluating text, including code.

Works Cited

Marino 2020 Marino, M. (2020). *Critical Code Studies*. The MIT Press.

Montfort 2009 Montfort, N. (2009). *Taroko Gorge.* Nickm.com. Accessed on 6 July, 2023 at https://nickm.com/taroko_gorge/.

Nelson a Nelson, G. *The Inform Recipe Book.* http://inform7.com/book/RB_1_1.html.

Nelson b Nelson, G. *Writing with Inform.* http://inform7.com/book/WI_1_1.html.

Poetry Foundation Poetry Foundation. "Enjambment." *Glossary of Poetic Terms.* https://www.poetryfoundation.org/learn/glossary-terms.

Prismatik 2020 Prismatik. (2020). *Scarlet Portrait Parlor*. https://rcveeder.net/expo/event1/prismatik1/.

Richardson Richardson, R. "Learning the Sonnet." https://www.poetryfoundation.org/articles/70051/learning-the-sonnet.

The Interactive Fiction Wiki The Interactive Fiction Wiki. "Interpreter." *The Interactive Fiction Wiki.* http://ifwiki.org/.

Veeder 2020a Veeder, R. (2020). *Event One of The Second Quadrennial Ryan Veeder Exposition for Good Interactive Fiction*. https://rcveeder.net/expo/event1.

Veeder 2020b Veeder, R. (2020). "Judgment of *Scarlet Portrait Parlor*. " https://rcveeder.net/expo/event1/prismatik1/score.html.