Cayley, John. "Time Code Language: New Media Poetics and Programmed Signification." *Grammalepsy: Essays on Digital Language Art*. New York: Bloomsbury Academic, 2018. 95–114. *Bloomsbury Collections*. Web. 26 Apr. 2023. <<http://dx.doi.org/10.5040/9781501335792.ch-008>>.

Downloaded from Bloomsbury Collections, www.bloomsburycollections.com, 26 April 2023, 11:58 UTC.

Copyright © John Cayley 2018. Released under a CC BY-NC-ND licence (<https://creativecommons.org/licenses/by-nc-nd/3.0/>). You may share this work for non-commercial purposes only, provided you give attribution to the copyright holder and the publisher, and provide a link to the Creative Commons licence.

7

Time Code Language: New Media Poetics and Programmed Signification

One of the defining characteristics of poetic writing is its attention to the materiality of language, which has become an important critical concept in literary studies. We speak of “the materiality of text” or “the materiality of language” in general, as if this might be an abstract characteristic when, in fact, it is the critical marker of linguistic and literary embodiment, recognizable only in terms of that embodiment. As N. Katherine Hayles puts it, “*The materiality of an embodied text is the interaction of its physical characteristics with its signifying strategies.*”¹ The presence and operation of code is, in many, though not all, instances, a significant part of the complex physical makeup of electronic text and is often a *sine qua non* for the operation of its signifying strategies. In so far as we are interested in identifying and defining certain specific aspects of the materiality of language that are foregrounded by writing in networked and programmable media, we are called to pay close attention to the role of code and coding in this type of work. We must keep asking ourselves, what is code? What is the relationship of code and text in cultural objects that are classified as literary and that are explicitly programmed?

The context of this chapter is current and continuing discussion which addresses these questions. It refers implicitly and explicitly to other critical interventions that have begun to identify a genre of electronically mediated writing as “codework.” According to Rita Raley, “Broadly, codework makes exterior the interior workings of the computer.”² Code is indeed an archive of the symbolic inner workings of the computer. However, not only is it brought to the surface in the writing of new media, it may function to generate the language displayed on this surface, without itself appearing. In an earlier piece of mine, a prequel to this chapter, I argue that we must

be more articulate about the distinctions we make between code and text.³ These distinctions are creatively challenged by codework that brings “inner workings” to an “exterior,” especially when such work is manifested as a generative cross-infection of text and code-as-text, of language and code-as-language. In this earlier piece, I argued that “the code is not the text, unless it is the text.” Code that is not the text, code that remains unbroken and operative, may instantiate—as durational performance—the signifying strategies of a text. As such, it does not appear on the complex surface of the interface text as part of or as identical with it. There are, therefore, further distinctions within codework, between those works that bring the traces of an interior archive of code into the open, and those works that depend on the continuing operation of code, where the code, in fact, reconceals itself by generating a complex surface “over” itself. The present chapter addresses these distinctions and then takes on questions concerning the characteristics of a textuality whose very atoms of signification are programmed. What is textuality where it is composed from programmed signifiers? In particular, the temporal properties of such signifiers are highlighted, and the significance of this temporality is examined.

Literal performance literal process

Clearly, it is difficult to articulate and share a detailed, nuanced conception of what we do—how we perform and process—as we write and read and play with language. Out of our difficulties entire fields of critical thought emerge. I begin, for example, to use words to refer, provisionally, to phenomena, like words, which I assume have some kind of separate, atomic existence, however provisional or temporary. Word as word (re)presentation refers to word as thing (re)presentation. The implicit atomism—treating something as irreducible in order to try to assay its significance and affect—is always provisional, even where established by lexical authority, and is ever mobile. At one instant I refer to some word-sized atom of language, the next instant another, then, as suddenly, I recompile and shift “upward”—many levels in the hierarchies of code and language—and refer to the specific work or to “text” itself, which suddenly becomes not only a conceptual automaton in our minds but also an atom of linguistic matter in my discourse itself, even though my discourse is, as it were, contained within its significance.

Foregrounded in this way, the procedural, performative nature of the literal is demonstrable. Despite your understanding that, for example, these words are inscribed as writing—temporally stunned, deferred, and spatialized—you will sense words shifting their meanings as I write/speak and you read/hear. No matter how little attention you or I pay to what is going on as we process, it is easy to concede that, for example, the meanings of words like

“code” and “text” change during the shifting “now”—the distinct present moments as I write and you read—and may well change radically over the course of my intermittent writing/speaking and your intermittent reading/hearing. The generation of altered and new meaning is, after all, one of my explicit aims in addressing these terms.

It follows, even from this simple, on-the-fly phenomenology of language, that atoms or instances of language (of whatever extent), though we treat them as “things,” are, in fact, processes. If they are ever static or thing-like, they are more like the “states” of a system, provisionally recognized as identifiable, designated entities. In themselves they are, if anything, more similar to programmed, procedural loops of significance and affect, isolated for strategic or tactical reasons, be they rhetorical, aesthetic, social, or political. This characterization is good linguistics and good critical thought. However, usually our perception and appreciation of linguistic and critical process are more broadly focused, bracketing the micro-processes that generate and influence significance and affect in the “times” taken to move from statement to statement, let alone those which pass so fleetingly and function so invisibly in the move from letter to letter.

Moreover, as Hayles demonstrates in her recent critique of prevailing notions of textuality, an abstracted conception of both “the text” (a physical and literal manifestation of the ideal object of textual criticism, more or less identified with an author’s intended work) and “text” (as a general concept), is allied to the apparent stasis and persistence of print, and still dominates our understanding of textuality in literary criticism.⁴ By contrast, for Hayles all texts are embodied in specific media. In her view, electronic texts represent a mode of embodiment through which literary works are able to perform a realization of a latent materiality, and perhaps also the revelation of such texts’ present and future informatic post-humanity, where they “thrive on the entwining of physicality with informational structure.”⁵ Hayles sets out some of the elements of an electronic text and emphasizes the dynamism of their symbolic relationships:

There are data files, programs that call and process the files, hardware functionalities that interpret or compile the programs, and so on. It takes all of these together to produce the electronic text. Omit any one of them, and the text literally cannot be produced. For this reason, it would be more accurate to call an electronic text a *process* rather than an object.⁶

Such a text, unlike that which has print for its medium, has no materially accessible existence prior to its generation when displayed on the screen: “electronic textuality ... *cannot be separated from the delivery vehicles that produce it as a process with which the user can interact.*”⁷

For an object to be identified as a process, at the very least, there must be some way for its state to change over time, and perhaps also the possibility of

enumerating the temporal sequence of such states, or some way to describe a procedure or procedures that generate the states and changes of state in the object. In other words, there have to be programs to be followed, to be run. In Hayles's analysis, however, the programming seems to reside chiefly in the delivery media of electronic textuality—the “data files, programs that call and process the files, hardware functionalities, and so on”—rather than operating from within the text itself, the text of interpretation.⁸ In earlier essays she has described and characterized a “flickering signifier” in digital textuality, but this flickering of signification is a function of the same *peripheral* processing of text and its image—both screen image and underlying encoded representations. Where the flickering is indicative of depth—like ripples on the surface of a lake—this is a function of code in the sense of encoding.⁹ We imagine depths behind the screen, within the box, underneath the keyboard, because we know that the surface text is multiply encoded in order that it can be manipulated at the many and various levels of software and hardware. However, much of this underlying programmatological manipulation is typically treated as insignificant for the purposes of interpretation. I know that the screens of text that I read are being ceaselessly refreshed with, perhaps, some subliminal perceptual flickering of their signifiers, but I do not necessarily read this process as part of what is being signified to me. Unless foregrounded by an author for particular rhetorical effects, the programmatological dimensions of screen rasterization, for example, do not play a direct role in the generation of significance or affect.

This is by no means to say that flickering signification does not operate in a poetics of new media. I believe that this phenomenon is crucial to both the theory and practice of literal art in programmable media and is generally applicable to textuality, including that of traditional media such as print.¹⁰ My present purpose, however, is to try to address the role of procedures that do directly affect rhetoric and poetics, to identify the subjects and objects of programming within discussions of code and coding, in so far as they inflect our understanding of writing and the performance of writing.

Five ways to write “code”

I have already suggested one source of possible misdirection concerning the relationship of code and signification. The debate is set out under the rubric of “codework” without fully articulating the ambiguities in the use of the term “code” itself. Thus Hayles, for example, concentrates on the role of “code” as *encoding* in signification, with “code” as operative *programming* implicitly consigned to the hardware and software periphery. Raley's minimal characterization—“Broadly, codework makes exterior the interior workings

of the computer”—evokes both encoding and programming aspects of code(work) since they are typically interior in her sense.¹¹ Raley goes on to suggest further distinctions in codework as identified by a prominent codework practitioner, Alan Sondheim:

Works using the syntactical interplay of surface language; Works in which the submerged content has modified the surface language; Works in which the submerged content is emergent content.¹²

However, Sondheim's set of distinctions does not evoke code as programming *per se*, and it remains focused on a written surface, however complex. It refers to the inscribed surface and what emerges from code into and through it. In order to help clarify the various ways that “code” is used in discussions of codework, I offer five provisional categories:

1. Code as (a special type of) language (viewed and interpreted as such)
2. Code as infecting or modulating natural language (the language works, but the code is “broken”)
3. Code as text to be read as (if it were) natural language; code which is infected or modulated by natural language (the code works, but the language is “broken”)
4. Code as system of correspondences, as encoding
5. Code as programming, as a program or set of methods that runs (in time) and produces writing, or that is necessary for the production of writing.

The first three categories characterize texts according to properties of the constituent language. The texts are viewed as interface texts to be read in a fairly traditional manner. The language has been composed and laid out—in any number of complex contexts, including of course the online, shifting context of the web—and then it is read and interpreted. These categories cover the majority of literal art production which goes under this new rubric of “codework.”

Code as language in itself and in its own terms, category one, is something of a specialist study, and its full critical appreciation is as much the concern of computer scientists as literary critics. Nonetheless, writers such as Loss Pequeño Glazier seriously address code in its own terms as a potential poetry, not simply as linguistic fodder for the most common type of codework, the second category, where code infects or modulates natural language.¹³ This second type of code-infected writing, epitomized in many ways by the work of Mez, is widely practiced and represents not much more than the extension of the long-standing enrichment of natural language which occurs whenever history or sociology produces an encounter between linguistic cultures and

subcultures.¹⁴ My previous chapter on codework critiques this mode of writing, not least by comparing it to the encounters which occur between commensurate human languages. Codework comes off somewhat badly from the comparison, because code is a jargon or sublinguistic structure, not a full-blown, culture-supporting language. An encounter between, for example, English and Unix, is, in a sense, an encounter between a language and some smaller part of itself, rather than with an alien and commensurate linguistic entity.¹⁵ It is also the case with code-infected interface text that the code is, in the programmer's sense, "broken" after its incorporation into the text we read. It has lost its operative, performative "power" in the very instant that it is brought to the surface of interpretation.¹⁶ What of the code that remains hidden, which may well be operating as we read? This is the code that I want to read more critically.

Category three, codework that is manifest as written code, presented and intended to be read by non-specialist human readers—those who are not programmers and not, as it were, "manual" compilers or interpreters—is a special case and is less common. "Perl poetry" is a genre known mainly to programmers and hackers. In my earlier chapter, I described examples, citing work by Jodi and Cosic, as well as one of my own experiments. In this type of writing the code may be *functional* and unbroken (although not *functioning* as it is read). However, in most instances, natural language elements are introduced (in a way that allows the code to remain functional) or cultural framing is provided which renders the code readable—significant and affective—for humans. In a manner complementary to the conditions pertaining to code-infected language, the human cultural elements tend to be "broken" or at least heavily constrained in these forms.

Thus, the codework categorized according to my first three usages of "code" produces texts to be read, interface texts subject to interpretation by readers. The code is not running to generate the text; nor is it significantly present in the text in such a way that might alter or inflect the manner of reading. Code is not functioning to address writing as a formal procedure in these cases; it is not involved with the form and matter of the language used, although it is, clearly, making a contribution to its content. The language of code is visible on the surface of the interface text, but code has not necessarily been present at the scene of writing.

As we come to consider encoding—my fourth category—as an aspect of writing in programmable media, code does begin to emerge as integral to the material of the language used, necessary for its properties and methods, although, I argue, this aspect of code is still not fully indicative of its potential role in the active and continuing modulation of signification or as an engine for new literal and literary rhetoric in new media.¹⁷ As we have already seen, Hayles's flickering signifier acquires much of its conceptual power from the depths and layers of encoding it allows us to discover and recover in programmatological systems. It is clearly demonstrable that text,

stored and displayed in digital media, is multiply encoded, and awareness of this circumstance is certainly significant for our understanding of the materiality of language in new media. However, this is not an entirely new conception of textuality. The idea that the signifier is multilayered—with shifting and floating relationships of correspondence between the layers—is well known and widely accepted in criticism. Famously, Barthes (as Hayles acknowledges) brought our attention to the layered underlying semiotic codes prevalent even in *readerly* texts.¹⁸ He showed how elements of the interface text might instantiate and evoke many and various instances of the corresponding codes simultaneously. Moreover, and by contrast with the sense of code initiated by Barthes, the type of encoding highlighted by the flickering of Hayles' digital signifiers is, in one sense, largely sublinguistic, or on the outer margins of paratext. Although we can be made aware that the codes of digital media make words we read on screen flicker beneath it, we do not really care—for the purposes of interpretation—whether the text we read is encoded as extended ASCII or Unicode.

Finally, this type of relationship is simultaneous or synchronous. The flickering is a sign of a synchronic correspondence. The flickering may only be apparent in brief moments of time but, significantly, the relationships do not function temporally, nor are they modulated by time. This simultaneity of encoded correspondences is crucial, I believe, to the distinction between, on the one hand, code as encoding, and, on the other, code as the archive of functional programming. We have to distinguish between: (1) flickering as a function of the chained hierarchies of codes and language where the signifier flickers because it is reducible to something else which flickers; for example: the work, a text, is persistent on the screen but I know that it flickers because I know that the screen refreshes and because the keyboard is waiting for new input or because some paratextual procedure is being applied to the word image (e.g., changing its font and color); and (2) flickering because the signifier or chain of signifiers is produced by code and because the signifier may itself be programmed.

At this point, I am beginning to discuss code as operational programming in textuality, my fifth category. My aim is to distinguish the characteristics of textuality that incorporates (or is the subject of) code in this sense. This “strong sense” code is integral to all textuality, although it might be objected that this claim would be hard to substantiate before the historical advent of demonstrably programmable media. There have always been programs, I would answer, and these programs are a necessary aspect of the materiality of language—an ever-present aspect of mediation between a text's physical characteristics and its signifying strategies. The difference lies in where—literally, and also within cultural structures and hierarchies—these programs run, and it also depends on who writes and runs them. There is a continuity from what I will call “paratextual programming” and the kind of programming that is ever more familiar from the proliferation of programmable media.

Paratextual programming runs quasi-invisibly within traditional structures of writing, reading, and interpretation. The programmatological dimension of writing has always already been operative, and therefore, the traditional temporally stunned conception of textuality has always already been inadequate to literary and especially to poetic practice. However, the coding applied to textuality *in new media* allows us to perceive, if not the coding itself, then the unambiguous effects and consequences of that coding.

Punctuation colon programming

To demonstrate a continuity between paratextual programming and programming proper we can work with an example, a sentence within a paragraph at the beginning of this chapter. By paratextual programming I mean the (integral) aspects of inscription which frame or infect or undermine or position the text to be read, that is, the interface text. I use “paratext” in the sense of Genette, but I am highlighting its programmatological dimension.¹⁹ In contrast to “strong sense” coding or programming, paratextual programming becomes a perceptible part of the interface text—it appears on the same surface, often using the same symbol set (although as often employing the tropes and figures of non-linguistic media)—whereas coding per se remains invisible and inaccessible. In fact, “codework” in the sense of the instantiation of code-infected interface text (typically my category two) can be seen as paratextual programming using what is also occasionally referred to as postmodern punctuation. Raley calls this—approaching even closer to the textuality of programmable media—“punctuation particular to the apparatus” and cites one of the prime current practitioners of the art: “Talan Memmott calls this set of punctuations ‘technical ideogramatics.’”²⁰

Even writing punctuated in a manner that is “particular to the apparatus” can be quoted, unpacked, analyzed, and stunned to paraphrase, as writing in general or in traditionally recognized forms. As promised, I will demonstrate what I mean with a simple example, from a piece of language I have already used, “Word as word (re)presentation refers to word as thing (re)presentation.” Obviously, the visible marks of paratextual programming here are the parentheses. The primary specific intended effect of the parentheses is to provide a double reading, at once poststructural—through the evocation of the word “representation” and through the use of the parenthesis themselves putting presence/absence of a signifier into play, and also Freudian—through the implicit use of his phrases “word presentation” and “thing presentation.”²¹ What the punctuation does is set up a time-based revision of the atomic meanings of and within the sentence. I can, as I have done, recast these meanings and map them to a paraphrase based on the traces and marks in the interface text itself. This recasting is a process in itself, separate from the surface language of the interface text but archived

within it. Its implicit “code” evokes a widely used and well-understood rhetorical and interpretative “program,” the program of paraphrase. In this light, paraphrase can be seen as nothing other than the simplified (proper) naming of procedural loops within more complex language that we so name in order to be able to identify and atomize their procedures of meaning-generation for the purpose of re-articulation. Any text where codes and the codes of punctuation are integrated with the interface text, including much of the codework of Mez and of Talan Memmott, can be unpacked and analyzed in these terms, as inflected and driven by paratextual programming.

Hypertextual dissolutions

Spatially organized, navigable texts can often be understood in the same way, where precisely the spatial organization and navigation is to be read as paraphrase, gloss, elaboration, annotation, and so on, all coded into operations that produce a successively revealed interface text. Making reference to spatially organized, navigable textuality immediately evokes hypertext. Indeed, hypertext does, for me, occupy a transitional or intermediate position between the textuality of what I have called paratextual programming exemplified in a postmodern punctuation of print text and a textuality that is generated by programs or that is itself programmed. For Philippe Bootz hypertext is simply the application of an operator to a literary dataspace.²² In Bootz’s theory, “the Procedural Model,” the application of a hypertext operator or class of operations to a proto-“hypertext” is what generates nodes and links while, at the same time, coding those methods and commands that enable what we call navigation into the hypertextual structure. For Bootz, it is important to see that the hypertextual operator is simply one of a virtually infinite number of such operators that might be applied to the literary dataspace, the proto-hypertext that would in fact become something quite other than hypertext if different operators were applied. It is also noteworthy that the procedures and programming of hypertext are relatively simple—the response to a set of documentary problems rather than to poetic or, indeed, narrative ones.²³ As famously discussed on the relevant internet lists in the late 1990s, there seems to be little content “inside” the links of traditional hypertext.²⁴ Hypertext took the spatialization of text beyond print media and brought the trope of navigation to prominence, but the composed language of its constituent nodes or *lexia* retained the print-like quality of having been impressed on a surface—discoverable, visitable, but with little programmatological “depth.” The classic hypertextual link does little other than provide the instantaneous replacement of one composed fragment of integral text by another. At times, this process is not appreciable, even metaphorically, as a spatial displacement. How is the replacement of text on the surface of a

unitary screen more of a spatial displacement than, for example, turning to a place—figuratively, literally, and physically—“further on” or “deeper into” or “at the back of” a book? The programming involved in hypertext seems relatively shallow and more closely allied with paratext and textual framing than with the potentialities I have it in mind to address.

Overriding the “read” method: Rosenberg’s programmed signifiers

Discussion of hypertext leads us to the work of Jim Rosenberg, which provides a further transitional demonstration of code operative in and through language, a crucial and interesting point of intersection of paratextual and strong-sense, fifth-category codework. Rosenberg explicitly contextualizes his practice within and against the traditional study and theory of hypertext, and yet his work is difficult to reconcile with classic link-node models of hypertext. On the one hand the actual coding in his work is, arguably, simpler than the implementation of a hypertext operator. In pieces such as Rosenberg’s *Intergrams*, *Diffractions Through*, and *The Barrier Frames*, his actual coding produces little more than the substitution of successive screen images showing texts, syntactic diagrams (in most cases), and textual frames in response to the position of a mouse or other pointing device.²⁵ On the other hand, Rosenberg has built elaborate, articulated relationships into the language and linguistic structures of the texts which are handled by his actual code, such that the positioning of the pointer—part of the work of the reader—becomes a device that reveals the programmatological dimension of his work.

It seems to me to be crucial to Rosenberg’s work that often when the mouse or pointing device is not in contact with an area containing or enclosing text, the visual field of the work is unreadable, or, more precisely, its constitutive texts are unreadable. In these states of the work (one might call them “rest” states), the reader is initially presented with “zoomed out” diagrams outlining large-scale syntactic relationship between areas of text (which are shown as graphic “representations of writing” rather than writing per se, Figure 7.1). If the reader “zooms in” on one of these areas, such that it fills the visual field and constitutes a new phase of the interface text, the words displayed on the screen (in the “rest state” of a zoomed-in assemblage) are still unreadable but for different reasons (Figure 7.2). Rosenberg typically composes his texts in overlaid clusters that together are dense enough to make reading the constituent layers impossible. It is precisely the movement of the mouse that brings one or another layer to the reading surface where it then becomes readable (while covering the other layers) (Figure 7.3). Move the mouse away and the work returns to an unreadable rest state.

FIGURE 7.1 Screenshot from The Barrier Frames. Courtesy of Jim Rosenberg.

FIGURE 7.2 Screenshot from Intergrams. Courtesy of Jim Rosenberg.

Rosenberg's work resists these reductions in the most obvious and effective way. When his work is space, it is not readable, and it has no emergent, repeatable linearity. Only within restructured time can it be read. Moreover, even less than in the case of hypertext can it be reduced to linearity. Without being indeterminate (Rosenberg's texts are not generated by quasi-random processes), these texts are nonetheless constructed in a manner that makes it next to impossible for writer or reader to anticipate or control the mouse or pointer's positions when addressing the work in a way that would allow, for example, the repeated performance of particular sequences of textual revelations. It would be impossible, that is, without learning to manipulate one of his works like a musical instrument, gaining the necessary control and skill to know which "notes" to strike and when. The point is: the reading, or recital, of one of Rosenberg's texts obliges its readers to address the inherent restructuring of time, specifically the time of reading. Rosenberg's coding of programmable media for literal art *guarantees* this specific aspect of his text's materiality and also, perhaps even more importantly, gives both writer and reader access to the manipulation of this dimension of literal textual matter.²⁸

In Rosenberg's work the coding is in the system, but it is also within, and a part of, the writing because of the way the text must be read, because of the simple fact that the only way to read is by working with the text, manipulating it with a programmaton's pointing device. Rosenberg has recast reading and has changed the properties and methods of the signifier. He instantiates a signifier that has radically different properties to that of print culture. One way of figuring this difference is to extend an analogy with object-oriented programming and say that Rosenberg has extended the class "Text" and overridden its "read" and "write" methods. In Rosenberg, writing is (among other things) a method of layering, overlaying, and compositing texts, and reading is (among other things) a tentative work of revealing the clustered layers in order to pass the literal data they contain on to the "read" method of an underlying or parallel Text object of the "parent" class, the Text object of print culture.²⁹

While we want to emphasize the fact that the signifier is a temporal, durational object, we also have to consider that literal and literary time is itself restructured by textuality. Textuality is temporal and as such restructures the culture of human time. That textuality was always temporal is clear. We are familiar with the textual generation of linear and narrative time. We are familiar with writing as deferral, especially as a function of its spatiality, its translation of time into space. We are comfortable with the figures and tricks of narrative reordering—flashback and the like—although chiefly in the frameworks of historical time and narrative drive.³⁰ However, textuality as instantiated in programmable media realizes the potential for a more radical restructuring of the culture of human time, and Rosenberg's literal art provides an instance of how this happens through

the absolute necessity to work with it, in time, in order to read.³¹ It is, in this sense, a (if not “the”) type of “ergodic literature,” where non-trivial effort is *necessary* for reading.³² However, as Espen Aarseth shows with his provisional “textonomy,” time enters into the art of letters and is restructured through many other rhetorical methods and procedures, not only through ergodic manipulation, but also—giving a far from exhaustive list—through animation, text generation, quasi- and pseudo-random modulation, and various combinations of all of these, not to mention the kind of live textual collaboration that *networked* programmable media allow.

Text in the docuverse

Rosenberg’s marked and continuing investment in hypertext per se invites us to re-examine the claim—much touted in the hypertextual “golden age”—that textuality gives way to hypertextuality in new media.³³ Rosenberg sees his work in terms of hypertext and is an active participant in the research community associated with both the technical and theoretical development of hypertext. When viewed from the perspective of computer science (or computer science in the service of the humanities), as a system implemented in software, “hypertext” has both a more precise meaning and also a range of ever-evolving meanings closely dependent on the changing capabilities of actually existing systems. Thus, the web is a variety of hypertext—providing nodes, links, and navigation—but the basic capabilities of HTML in the standard server-browser implementation are severely limited when compared with more developed hypertext systems or speculative structures.³⁴ Many hypertext theorists and researchers—including and perhaps especially Ted Nelson—would say that the web falls short of even the fundamental requirements for a properly hypertextual system. I am not so much interested in explaining or elaborating these technical distinctions. Still, I want to consider the implications of the proposition that Rosenberg sees his work, his literary objects, as reducible to hypertext, and I want to do this in relation to a theory of hypertext that is particularly “totalizing”—Nelson’s vision of the docuverse.

This chapter provides context for a re-examination of Nelson’s vision in terms of my arguments concerning the poetics and the temporal materiality of textual art in new media. More specifically, I am discussing programmed signification—strategies of signification in Hayles’ terms—in which codes and coding operate to generate or modulate texts, substantively. The attempt to reconcile such strategies with Nelsonian hypertext yields, I believe, crucial perspectives on both hypertext and the materiality of textual art. Nelson is a visionary theorist particularly sensitive to text as “evolving, Protean structure”³⁵ and yet, paradoxically, his docuverse—along with the properties

and methods of its Xanadu system—is not only “the original (perhaps the ultimate) HYPERTEXT SYSTEM”³⁶ but also the final instantiation of the textual materiality of authorized editions, of the ideal, abstracted, persistent, authorized text that is currently the dominant object of attention in both literary and academic discourse.

For Nelson “a document is really (Figure 7.4 appears here in his text) an evolving ONGOING BRAID.”³⁷ This definition accords perfectly with a materiality of text for which structured durations of time are necessary to its strategies of signification. Nelson’s system also specifies and provides a way to view text in various successive states that arise during the spiraling, branching process of composition, “instantaneous slices” captured from the evolving braid as “versions” of the text or some part of the text. For Nelson, a text very much has a history as well as a synchronous existence, and his rendition of hypertext aims to represent this chronological dimension and to do so well. However, his nodes are *time-stamped* not *time-based*, as in the phrase “time-based art.” The docuverse captures states of the looping and spiraling braids of textuality, but not the looping and spiraling itself. In later versions of the docuverse these nodes are conceived of as the “spans” of a “permascroll.”³⁸ The totalizing and ultimate instantiation of a Nelsonian docuverse is a representation of *the* permascroll.

The permascroll is another important point of view from which to examine the Nelsonian docuverse. It is the linear and literal representation of *every textual event*. It is all writing, everything written, everything inscribed as language, as and when it was so inscribed. Hypertext can then be generated from the permascroll through operations that display linked windows onto spans of the scroll. Textual history and textual criticism can be recast as a vast but particular and privileged set of pointers to those spans on the permascroll representing various textual events that are—culturally, institutionally—significant for the archive and interpretation of a writing

FIGURE 7.4 Illustration from Literary Machines 93.1, p. 2/14. Courtesy of Ted Nelson.

tradition. A book, for example, may be viewed as one (complex) window on the scroll, its coordinates and output parameters determined by a particular culture's definition of "book" as textual framing device containing the end product of many processes (e.g., editing) that examine and select from prior collections of spans delineated on the permascroll and representing the chronological development of the text.

The textual event

To my mind, Nelson's vision is truly that: visionary, magnificent. It *is* an ultimate system, the epitome of a textual universe composed of *editions*, composed, that is, from minimal, transcribable textual events. And yet it begs a question that throws our underlying concerns into high relief: "What is a textual event?" Nelson's concerns—as opposed to those of literary artist-practitioners—were first and foremost documentary, as Philippe Bootz has noted.³⁹ Nelson proposed a reconfiguration of the documentary universe that is more than equal to the task of handling all the nuances of textuality and its criticism as currently instituted and implemented in traditional literary media.⁴⁰ The textual event is defined culturally, by cultural institutions and by media technologies. In our own context, the institutions that dominate literature and language art are editorial bodies—universities, publishers, the world of letters—and for these authorities the textual event is still ultimately determined by a simple test: "Can it be printed?" In recent years, this formulation may have been slightly modified (by the web in particular) to: "Can it be printed *out*?" Nelson challenges and reconfigures the forms of display and the engines generating our textual points of view but he does not fundamentally challenge the notion of textual event.⁴¹ Specifically, he does not address the necessity, I propose, to allow the textual signifier to include—as inherent constituents of its materiality—temporality and programmability.

Consider how Rosenberg's diagram or intergram poems might be transcribed on the Nelsonian permascroll. As it happens, all the textual elements of a Rosenberg piece are determinate: they may be conceived as authorially composed and transcribable editions. As such all the elements of a diagram poem or intergram and all of its states could be rendered by the permascroll and its engines, except that, crucially, there is no obvious—or *institutionally recognized*—way to represent (1) overlay and simultaneity or (2) the dissolution and resolution of these textual properties in necessarily temporal and ergodic processes.⁴² Both of these features are intrinsic to the aesthetic, and to the significance and affect of Rosenberg's work, to its meanings. Appended to the scroll and its engines, one could imagine the record of code—or of abstracted representations of algorithms—that

would allow all of the features of a poem to be rendered and reproduced, but these would not be part of the scroll or its native systems; they would not be instituted as a recognized part of the docuverse, even the Nelsonian docuverse, let alone the traditional world of letters.

In a sense, I am revisiting old arguments concerning the specific technical capabilities of our emergent media of inscription. Circa 1994 hypertext arrives in the world of letters, and certain practitioners and theorists complain of its inadequacies, its inability to implement a wide range of the intrinsic potential for an extended literacy in networked and programmable media.⁴³ However, besides pin-pointing “what gets left out” of textuality by both print and hypertextual culture (our focus is the temporality and programmability of the signifier, the textual event), the Nelsonian example highlights the complicity of institutions with the implicit cultural resistance to certain forms of practice. The resistance is not just a function of technology. Nelson is radical in demanding a revolution that extends to literary institutions. He imagines a total migration of literary content to new media, and goes so far as to propose a total reconfiguration of critical apparatus and intertextuality, of tools for quotation and reference.⁴⁴ He also provides, to my mind, a workable mechanism that upholds basic principles and moral rights established by copyright, while shifting their control and management away from the actual-existing copyright hegemonies that threaten to dominate and constrain cultural production in new media.⁴⁵ That is radical.

Temporal and literal institutions

The institutions that are not challenged by the Nelsonian paradigm are those cultural institutions that authorize and maintain a definition of the fundamental atom of inscription and its relationship to a particular, privileged type of temporality. The minimal unit of text—of the symbolic, of language-in-Western-culture—is the letter, an abstraction we conceive as timeless. Strings of letters that are structured into words, sentences, paragraphs, chapters, books, and so on, we also think of as having a temporality that is deferred. We say that writing renders time as space, while also, of course, always allowing its power to represent (arbitrarily complex) temporal structures *in content*.⁴⁶ For any particular text, we accept any of its recorded histories or chronologies that can be expressed in terms of these atoms of inscription. This acceptance allows and accounts for the complexities of textual criticism, for the relatively sophisticated notion of “text” that these practices require—for text as a history of *editions*, however provisional and reworkable. Hypertext provides a navigable visualization of the relationships between such fundamental units of inscription, while Nelson’s particular

genius was to provide a generalization of these complex requirements and a potential reconfiguration of their underlying structures, one that was radically, institutionally implicated. However, because of the properties of their shared fundamental atom of inscription—particularly its deferred temporality—all of these forms of textuality may, if necessary (typically this will be for institutional reasons or to allow the text's accessibility to familiar, traditional, interiorized reading practices), be rendered as print, without affecting—institutionally—the interpretation and appreciation of the text's aesthetic, its strategies of signification, its generation of meanings, its significance and affect.⁴⁷ Furthermore, this printing out implicitly privileges a particular form of temporality: an (arbitrary) sequencing of elements of inscription that is ineluctable during any particular experience (or “printing out”) of a text's instantiation but which, in institutional terms, appears to be inevitable and necessary, to the extent that we may prefer one sequence of elements and come to designate this sequence as *the* text, as its standard, canonical edition.

This type of canonization will not work for a Rosenberg intergram. Neither will it work for a wide-ranging and growing corpus of work that is textual and also, to give only the most obvious examples of textual properties that are not, as it were, “(perma)scrollable”: animated, generated, indeterminate, the product or instantiation of real-time collaboration. To take one example—textual animation—we can see that in cinematic film titling, in advertising using time-based delivery media, and finally in the poetics of networked and programmable media, textual animation has a history and highly developed, if inadequately articulated, rhetoric specific to its textual materiality. The atoms of this textual matter cannot be simply recast as arbitrary sequences of letters, not without bracketing, masking or ignoring vital aspects of these texts' signifying strategies—specifically, for example, a whole range of transition effects from text to text. This means, unambiguously, that criticism must address the cultivation and articulation of temporality in this work as well as, if not also by way of, an analysis of the code that guarantees and drives literal temporality.

Code generates literal time

Code as programming has other contributions to make to the emergent tropes and figures of a rhetoric extended so as to articulate the signifying strategies of writing in networked and programmable media, reflecting its materiality and media specificities.⁴⁸ However, in so far as code generates the temporalities of writing in programmable media, it highlights what I believe is currently the most important thread in a program that criticism and theory must follow in order to accept these temporalities as integral

and inalienable properties of all atoms of signification in literal, indeed, literary art. Without its code—even when rendered as elaborate “scrollable” hypertext—Rosenberg’s work sacrifices vital aspects of its aesthetics, its strategies of signification, its power to generate significance, affect, meaning.⁴⁹ The effect of an intergram arises largely, as we have said, from simultaneities and from temporal and ergodic processes that dissolve and resolve these simultaneities. The cultivation and articulation of real, material time is built into the text through coding.

Much play, in the recent criticism of new media poetics, has been made of the *visibility* of a literary work’s engagement with the material specificities of its media, what might be called its material self-reflexivity. In *Writing Machines*, Katherine Hayles demonstrates brilliantly how Mark Danielewski makes us see and feel and hear the (empty/nothing/void in his *House of Leaves*).⁵⁰ However, we still—in this and other print culture examples—see and feel and hear the “leaves” using technologies of inscription that are profoundly familiar within the culture and institutions of literature and pedagogy in general: parallelism of textual streams (text and footnotes); commentaries, and commentaries on commentaries; multiple perspectives; typographic novelties; not least, temporal complexities represented as content while formally virtualized and *deferred* by writing. We must also however acknowledge and distinguish texts in media that are composed with code and that allow authors and readers to program aspects of temporality as integral parts of the text, as constitutive of its very materiality, and we must recognize a productive, critical opposition between writing as deferral/spatialization—of content including representations of time however complex—and writing as program and performance—in and of time. We need to elaborate this distinction for many reasons:

The real temporal dimension of the *materiality* of text has been underplayed and overwhelm by the stasis and persistence of authorized editions.

A materiality of text that embraces temporality offers a more general theory of textuality, backwardly applicable to work in durable media.

A significant body of work now exists that is made from programmed signifiers and can be displayed using time-based media, and this body of work remains literally unreadable and largely resistant to methods of interpretation that cannot cope with temporality in a sophisticated manner.

Work of this type includes performative pieces that are made of language and are expressed as literary art but which cannot be addressed using the existing tools of literary criticism.

Finally, much of this work is explicitly generated by and made, at least in part, from code, coding that has an unambiguous relationship with the programmatological engines of new media, the tools we now habitually use to write.

Code is presented to us as a special type of linguistic archive. It leaves traces on the surface of literary culture that cannot be denied or ignored, even in works that do not make art with these traces. Strangely, the code is hidden as it runs—driving the temporal atoms of literal signification, restructuring the culture of human time. The code of programmatology embodies a literal interior now calling to us for articulation and poiesis.⁵¹