16

FIRST-GENERATION POETRY GENERATORS

Establishing Foundations in Form

Christopher Funkhouser

CONTEXT, SIGNIFICANCE, ARTISTIC PARALLELS

The earliest works of digital poetry strictly involved coding, as there were no other possibilities. Poets initially used computer programs to unite a database and a series of instructions that established a work's content and shape. The language generated by programmed poems, or computer poems, was literally assembled to the specification of the programmer; formal, precise programming commands were written to perform particular tasks. In *Computers and Creativity,* Carole McCauley effectively summarizes the three essential stages in the process of creating such works: "determining a frame (single words, lines, or stanzas with particular grammatical features); creating a dictionary of words for use in the frame; and finally, adding any extra qualities or instructions (the machine should choose only certain rhymes or words beginning with a particular letter or should print the results in certain patterns on the page)."[1] As in any type of poetry, the computer poem relies on the author-programmer's senses, thoughts (or inspirations), and structural inclination to form words, phrases, and poems. As always, the poet enacts language amid a range of possible treatments.

During the 1960s, authors programmed poems using coding that was previously designed for mathematical and scientific calculation, synthesizing—through new languages—a body of information and instructions via the computer. The cooption of computer languages and machinery used to manage information—which is, in essence, the work of the digital poet—did not involve PCs at all. Programs were, out of necessity, run on institutional or corporate mainframes without sophisticated graphical user interfaces (GUI) or input devices such as

the computer mouse. The earliest works were produced using program-controlled or mainframe computers with integrated circuits that were being developed in the 1930s, '40s, and '50s, such as the Zuse Z22 and the IBM 7070. Instruction code, essentially a series of notations, would be entered and transposed onto punch cards that would be fed into another part of the machine and "read" in order to run the program.[2] A number of program languages composed of alphanumeric data became, using a designation Ted Nelson applies in *Computer Lib,* the "contrived" (intricate, technical) method of providing instructions or commands to a computer. Programming code at this juncture, writes Nelson, enabled "loops, tests and branches, and communication with external devices."[3] Languages developed in the 1960s, such as BASIC, TRAC (Text Reckoning and Compiling), APL (A Programming Language), and FORTRAN, were written on terminals and run on compatible mainframe computers controlled by the government, corporations, or universities.

The early works of text generation performed some type of permutation in that they transformed or reordered one set of base texts or language (e.g., word lists, syllables, or preexisting texts) into another form. Databases contained words or phrases, sometimes organized according by the parts of speech they represented, and utilizing multiple databases within a single work was a possibility. Programming code included instructions for the production of the poem, which would include formation of nuance. As demonstrated by works introduced in this essay, permutation procedures of algorithmically generated poems either randomly recombined database vocabularies into new arrangements of words, or merged preset word lists in controlled or random combinations, or combined words with slotted syntactic templates (providing grammatical frames to create an image of "sense"). While the scale and complexity of database operations have increased and become refined during intervening years, the basic dynamics found in the works discussed below, all created by 1969, have not changed. The other major development in digital poetry in the mid-1960s, cultivation of graphical works, also advanced and become a mainstay of the genre. In the first decade of computer poetry, significant expressive foundations of the genre were established.

It is not difficult to build a context for computer poems using literary works and discourse from the modern era, although it is clear that digital poetry's stylistic foundation was first established by premodernist writers. French symbolist writing, particularly Stéphane Mallarmé's late-nineteenth-century poem "A Throw of the Dice Never Will Abolish Chance" (1897), was unquestionably an artistic antecedent that directly impressed upon the disruption of textual space and syntax found in computer poems. The variations in typography, incorporation of blank space, and liberal scattering of lines often found in digital poems can be discerned as having roots in Mallarmé's work. Mallarmé was but one premodernist precursor whose atypical form of poetic presentation influenced the mechanics of digital

poetry. As divulged and reconstructed in the body of work that appears on Florian Cramer's *per.m]utations* Web site,[4] the programmed permutation works that emerged near the outset of digital poetry have even earlier predecessors in combinatory works that date back as far as 330 A.D. Samples and reinventions of writings by Optatianus Porphyrius (*Carmen XXV,* fourth century A.D.), Julius Caesar Scaliger (*Poetices,* 1561), Georg Philipp Harsdörffer ("Fivefold Thought Ring of the German Language," seventeenth century), and other works illustrate how the mechanics of computer poems have roots in works produced centuries ago.

Dada poets (circa 1915–23) also strongly influenced computer poetry. Dada poetry, as stated in a subsection of Tristan Tzara's *Dada Manifesto on Feeble and Bitter Love,* "To Make a Dadaist Poem," instructs readers to cut up newspaper articles into individual words and make a poem by random selection and reorganization. Dadaists challenged convention with other methods, including collage, the invention of new words (neologism), typographical distortion and desecration, transcription (or use) of nonsemantic sounds, and collaboration. All of these elements are found in early digital poems (i.e., the language and other media that serve as the database of the generated digital poem are akin to the words clipped from the newspaper, etc.). Dadaism is unquestionably a historical model that can be used as a context for many of computerized works. Radical unconventionality, when it appears in computer poetry, had already been permitted in twentieth-century literature (not only at the Cabaret Voltaire, but in Futurist exhortations, surrealist cut-ups, and elsewhere). Alteration and permutation of words and short phrases to subvert meaning (in semantic revolt) are certainly manifestations of cybernetic Dadaism. Instead of scrambling and reforming a single set of words, digital poets use multiple input texts and programmatic techniques to process, reorganize, and re-present texts, as new procedures for manipulating bodies of language become available. While not all text-generating initiatives follow in the Dada lineage, automatically randomizing texts with computer programs is a logical next step in the Dada progression.

EARLY EXPERIMENTS: MATHEMATICS AND PERMUTATION

The pursuit of composing poetry by using computer operations originated in 1959 when Theo Lutz made "stochastic" (i.e., random variation) poems written on a program-controlled Zuse Z22 computer. At the time, he was a student of Max Bense, who suggested using a random number generator to accidentally determine texts.[5] Examples of this work, which applied tools of mathematics and calculation (i.e., logical structures) to process language, along with descriptions of its attributes, were published by Lutz in a 1959 article ("Stochastic Text") in Bense's journal *Augenblick.*[6]

Lutz made a database of sixteen subjects and sixteen titles from Franz Kafka's novel *The Castle*. Lutz's program randomly generated a sequence of numbers, pulled up each of the subjects and titles, and connected them using logical verbal constants (such as conjunction) in order to create plain syntax, as seen in this excerpt from "Stochastic Text":

> Not every look is near. No village is late.
> A Castle is free and every farmer is far.
> Every stranger is far. A day is late.
> Every house is dark. An eye is deep.
> Not every castle is old. Every day is old.
> Not every guest is angry. A church is narrow.
> No house is open and not every church is silent.
> Not every eye is angry. No look is new.[7]

In this example, we can see patterns and repetitions of words, along with discursive leaps and quirky, unusual semantic connections (e.g., "No village is late"). The words themselves are not complicated, but when they are automatically or randomly arranged into syntax via computer program, the transaction imposes a nonrational ordering of subjects and thoughts. The text—seen above in translation, a further complication—is readable but disjunctive. Readers must connect and interpret abstractions in the poem (not a new phenomenon in reading or writing poetry), and derive meaning from the verbal associations while reading the text in and against its context. As Friedrich Block and Rui Torres observe in their essay "Poetic Transformations in(to) the Digital," this work propels Bense's theories at the time, which involved "the turn from idealistic subjectivity to rationalism and objectivity of art" and "the turn from mystic creation to statistic innovation."[8] Lutz's choice to build the first computer poems based on Kafka's book is intriguing, and adds a layer of significance to the endeavor. It is possible that Lutz chose Kafka's incomplete novel as a foundation out of respect for poetry, as a way to question the communicative values of machine modulated verse. While the processes of generating or consuming the poetry do not particularly reflect or require the reader to embody the type of mysterious bureaucracy experienced by the protagonist of the novel, an alienated, barren tone pervades the output of the program.[9] The best examples of output are successful because the reader, via the poet's condensation and computer processing of the materials, can rediscover the essence of Kafka's story, or somehow experience new perspectives derived from the original text. Lutz's selection of words, combined with his programming method, enables a speculative, self-reflexive, unconventional style of expression; the programming method consists of about fifty commands and could theoretically generate over four million different sentences.

"Stochastic Text" is a combinatoric poem that uses sparse, preset word lists in controlled and random combinations. The language also contains permutation: the same few words are used over and over each time the program is run. It is not a permutation of the entirety of Kafka's text; it is a variable, fragmented permutation of the words Lutz chose from the story. Lutz was at the crest of a wave that viewed mathematics, science, and creativity as cooperative disciplines that could forge new interrelationships through computerized mechanisms. Until the GUI became a more common feature of personal computers, Lutz's approach was the form's status quo—synthesizing source files and written programs used to access and activate the source files.

Brion Gysin explored a different yet related idea in his first permutation poem, "I am that I am," which is a cyclical, randomized representation of the three words contained in that phrase. Gysin's interest in the form stems not from technological or mathematical interests but from visual ones. In Richard Kostelanetz's anthology *Text-Sound Texts*, Gysin writes, "The whole idea of the permutations came to me visually on seeing the so-called, Divine Tautology, in print. It looked wrong, to me, non symmetrical. The biggest word, That, belonged in the middle but all I had to do was switch the last two words and It asked a question: 'I Am That, Am I?' The rest followed."[10] This work imposes a preestablished pattern on the words in a phrase, so they appear in different orders until all possibilities have been exhausted. Thus, a poem made with a three-word phrase will be six lines long ($3 \times 2 \times 1$); a poem that begins with a five-line phrase, such as "I am that I am," will be 120 lines long ($5 \times 4 \times 3 \times 2 \times 1$). The availability of computer technology automated the process of randomizing these permutations. The critical anthology *Brion Gysin: Tuning in to the Multimedia Age* shows four examples of computer-generated permutation poems, programmed with a random sequence generator on a Honeywell computer (using punch cards) by Cambridge University mathematics student Ian Somerville in 1960. The output appears in block formation:

I AM THAT I AM
I THAT AM I AM
I AM I THAT AM
I I AM THAT AM
I THAT I AM AM
I I THAT AM AM
I AM THAT AM I
I THAT AM AM I
I AM AM THAT I
I AM AM THAT I
I THAT AM AM I
I AM THAT AM I

I AM I AM THAT
I I AM AM THAT.[11]

In a 1964 piece called "Cut-Ups Self-Explained," also included in *Brion Gysin*, Gysin declares, "The permuted poems set the words spinning off on their own; echoing out as the words of a potent phrase are permuted into an expanding ripple of meanings which they did not seem to be capable of when they were struck and then stuck into that phrase."[12] Gysin held iconoclastic—and what would now be considered postmodern—views regarding poetry and the freedom of language: "The poets are supposedly to liberate the words—not to chain them in phrases. Who told poets they were supposed to think? Poets are meant to sing and to make words sing. Poets have no words 'of their very own.' Writers don't own their words."[13]

Gysin's creations mirror the method of a "proteus" poem, and can also be seen as an adaptation or transformation of the traditional *renga* poem. (*Renga* is a style of Japanese linked poetry in which stanzas are semantically related and create a type of code; words or phrases are repeated in successive stanzas.) However, instead of keeping an entire line intact, the poet creates a poem from one line, in which the words are internally cycled in a random pattern. In Gysin's work, the process is repeated over and over until every word has appeared in every possible position in the line of the poem.

OPEN FORM POEMS

Shortly after experiments by Lutz and Gysin, variant independent approaches to automatically generating text emerged, as did public installations of the work that showed how different approaches to the pursuit existed in the earliest years. The first major exhibition to occur was *Cybernetic Serendipity: The Computer and the Arts*, curated by Jasia Reichardt in London in 1968 (in the introduction to the exhibition catalogue, she credits Max Bense for giving her the idea to organize the show). Among the musicians and graphic artists who had begun to use digital technology, a handful of artists whose primary mode of expression was verbal were also included in the event, showing that creative potential for the computer was perceived by artists working in all forms. Through works shown in the exhibition catalogue, we see how quickly more sophisticated computer poems came into being as programs written in BASIC, TRAC, APL, and FORTRAN were able to engineer flexible techniques and the modernist propensity to synthesize disparate voices and cultural details. At the same time, the objectivity apparent in Lutz's work also continues, and the influence of Bense's theory on artificial poetry during this historical moment becomes even more obvious. For example, the early *Tape Mark* poems by Nanni Balestrini (1961) recombine appropriated texts by Lao Tzu *(Tao te ching)*, Paul Goldwin *(The Mystery of the Elevator)*, and

Michihiko Hachiya *(Hiroshima Diary)*. Such reinscription is a trait commonly found in computer poetry, and in this work Balestrini's avant-garde and leftist political proclivities consequently played a significant role. His choices of input texts were not accidental, and the combinatoric program, which unites and constructs chains of words from three authors, critically and unavoidably portray a scenario of nuclear disaster as a result of the inclusion of Hachiya's text. At *Cybernetic Serendipity*, the program repurposed sections of these texts to generate six-line poems with four metrical units per line, which were later edited for grammar and punctuation.[14] Each of the *Tape Mark I* poems has a different syntactical structure, even though the same words are often present from poem to poem. A passage from the exhibition catalogue, translated by Edwin Morgan, highlights the poems' qualities of permutation:

> Hair between lips, they all return
> to their roots, in the blinding fireball
> I envision their return, until he moves his fingers
> slowly, and although things flourish
> takes on the well known mushroom shape endeavoring
> to grasp while the multitude of things comes into being.
>
> In the blinding fireball I envisage
> their return when it reaches the stratosphere while the multitude
> of things comes into being, head pressed
> on shoulder, thirty times brighter than the sun
> they all return to their roots, hair
> between lips takes on the well known mushroom shape.[15]

The use of set phrases in these poems ("hair between lips" or "well known mushroom shape") gives a *renga*-like quality when more than one is presented at a time. The positions of the poem's phrases and the meaning they produce change in each example. Though the shapes of each stanza are similar, Balestrini's programming method can generate a variety of poems (within finite parameters) from words composed for other purposes; the program, like Lutz's, devours multiple texts in order produce combinatoric permutation poems. The brief phrases in Balestrini's dictionary collect and intricately reconfigure excerpts from previously written texts to generate hybridized, contemplative, and haunting expressions.

Although completely programmatic, the poetical collage techniques seen above are reminiscent of *The Cantos* and William Carlos Williams's *Paterson*, which juxtaposes poetry, the language of the people and natural world of his locale, and correspondence with other writers into a sequence of writing encompassed in the poem. Like Williams, Pound, and Eliot in a previous era, poets working with computers confronted social and artistic fragmentation in the world around them and—whether consciously or not—used technology to atomize and hybridize

texts that both subverted and reflected the complex of cultural information. Authors working on the page and screen in the postatomic era used fragmentation to legitimize discontinuity and challenge the stability of language as a point of meaning; this process of reassembling disparate pieces via technology was often used as a means to impart a sense of coherence.

A November 1962 issue of *Time* magazine brought one of the first examples of computerized combinatory poetry to a large audience. Its books section featured a brief notice titled "The Pocketa, Pocketa School," introducing "Auto-Beatnik" as a computer programmed to create poetry.[16] This unattributed exposé prints and informally discusses two examples of Auto-Beatnik poems, and offers a serious interpretation of one of them. While the tone of the article is curious, and it concludes by surmising, "There is a chance of a whole new school of poetry growing up," it essentially seems to make fun of the process, and takes it with only faux seriousness. The author of the piece suggests, "The machine needs help," but admits that "by drastically cutting down its choice of words—so that the incidence of a subject word reappearing is greatly increased—engineers can make the machine seem to keep to one topic." This point is significant, as it reveals that since the earliest observations on the subject, the creation of the computer poem was seen as something that involves limitation in order to succeed (or, it could perhaps be said, even to exist). This idea is often overlooked in commentaries that celebrate the endless possibilities of electronic text.

The poems published in *Time* contain syntax and are thematic as a result of its narrowed vocabulary (3,500 words, 128 simple sentence patterns):

> Few fingers go like narrow laughs.
> An ear won't keep few fishes,
> Who is that rose in that blind house?
> And all slim, gracious blind planes are coming,
> They cry badly along a rose,
> To leap is stuffy, to crawl was tender.[17]

One can see unconventional connections and phrases, but none beyond the boundaries of poetic license. Action, description, question, projection, and judgment—all poetic traits—are present. The second excerpt is similar to the first only in that it uses the simile "like" in the first line and that the poem contains unusual inflections:

> All girls sob like slow snows.
> Near a conch, that girl won't weep.
> Stumble, moan, go, this girl might sail on the desk.
> This girl is dumb and soft.[18]

Most interestingly, the program can emulate free verse, and aesthetically resembles a strain of Beatnik poetry (though here I am not referring to sophisticated and utterly profound works such as Allen Ginsberg's *Howl*, etc.). The first line certainly echoes the style or tenor of Jack Kerouac's poetry, especially in recognizing the suffering of all organisms (Kerouac was known for his use of Buddhist themes). A girl close to nature "won't weep," but the one with the desk is "dumb and soft." The poem raises poetic questions: Is this a critique of culture? From one perspective, it reads as a masculinist put-down; from another, as subjective information from which, perhaps, something else may be deduced. In this Auto-Beatnik poem, the program does not reveal sensitivity but does reflect the stream-of-consciousness qualities of many Beat works.

In 1964, Jean Baudot began to experiment with automatic sentence generation, samples of which were published that year in a collection titled *La Machine à écrire*. An excerpt of Baudot's combinatoric work appears in the *Cybernetic Serendipity* catalogue, which includes this passage:

> The address and a radical promotion will unite the receptions before the occasion.
> The gourmand dog moves the shovels.
> As the blue furniture was moving, a car and the loud-mouthed (garrulous) adventure sometimes turned against their friend.
> The flakes and painful terrors will respect a verdure, but a savage girlfriend will sometimes cheat on the vigorous noise.
> The marvelous tree and an oven will contain less enemies, yet a journalist finds himself incessantly faced with a dreadful morsel.
> The monotonous cheek and the wild cat use the assistant since the useless cauldron leaves the agreed program.
> The peaceful course water the pain according to the religious candle, but a skilled director can't hold the air.
> The application will monitor the brave fear for the benevolent enemy.
> Dark fires were inviting the day, but an intelligent man attracted the loss with the gay order.
> The crowns and the rain look at each other in the owl.
> The young trees and the nervous lion will come closer.[19]

Baudot, in *La Machine à écrire*, describes his process as involving three steps, the first two of which are random. To begin, a sentence structure is generated that "acts as a mould which, while syntactically correct, will have to be filled with actual words; only word-categories are indicated." Secondly, words are selected from a dictionary and placed into the structure. After this, the program applies grammatical rules to "polish" the spelling. The program was capable of creating two hundred sentences per minute. While using this process, Baudot declares, "There is no way of predicting what kind of sentences will be generated, or what words will be associated."[20] The sentences in this example are peripatetic in that

they exhibit a lack of continuity from line to line (no nouns are repeated; no particular themes are present). The aphoristic sentences are clearly readable but are awkward, and are full of unexpected juxtapositions, which undoubtedly some readers would perceive as haphazard nonsense. The distortions in some lines, like "The crowns and the rain look at each other in the owl," though elegant and poetic, barely resist embodying a nonsensical status. Within some lines, however, marvelous phrases are presented, some of which—even many years later—remain rife with cultural commentary ("a journalist finds himself incessantly faced with a dreadful morsel"), and perhaps even comments on computer poetry itself. From a point of view that questions the computer's rise in stature, the phrase "The application will monitor the brave fear for the benevolent enemy" can be read as an omen for the form: the tools are used because they are available and seemingly powerful, despite hegemonic attributes. Content and results of the creative efforts need to be observed for problems and flaws as well as for their possibilities.

Technological experiments in combinatoric poetry continued throughout the 1960s. Jackson Mac Low, already a prominent poet based in New York City, created his first computer poems while he was a resident at the Los Angeles County Museum in the summer of 1969 using a PFR-3 programmable film reader that was designed for graphics applications, connected to a DEC PDP-9 computer.[21] The program Mac Low worked with, he explains in *Representative Works: 1938–1985*, selected and permuted words from a list of short messages he had composed, or randomly ordered lines of messages. Further permutations also occurred in printing the output text, as only every tenth line was printed. The program's database (the "message lists") and processes allowed Mac Low to create "an indeterminate poem, of which each run of the printout is one of an indeterminable number of possible realizations."[22] "Printout from 'The'" used a message list containing "about 50 messages"; the following excerpt is indicative of the poem's style:

THE WIND BLOWS.
THE RAIN FALLS.
THE SNOW FALLS.
THE STREAMS FLOW.
THE RIVERS FLOW.
THE OCEANS RISE.
THE OCEANS FALL.

THE BUSHES GROW.
THE MOSSES GROW.
THE FERNS GROW.
THE LICHEN GROWS.

THE TREES SWAY IN THE WIND.
THE FLOWERS SWAY IN THE WIND

THE INSECTS ARE HATCHED.
THE REPTILES ARE HATCHED.
THE MAMMALS ARE BORN.
THE BIRDS ARE HATCHED.
THE FISHES ARE HATCHED.

THE PEOPLE SAIL ON RAFTS.

. . . . . .

THE INSECTS GATHER FOOD.
THE BIRDS GATHER FOOD.

THE PLANETS SHINE.
THE MOON SHINES.
THE SUN SHINES.

THE TREES DRINK. THE FUNGUSES DRINK.

THE MOSSES TURN TOWARD THE LIGHT.
THE FLOWERS TURN TOWARD THE LIGHT.
THE TREES TURN TOWARD THE LIGHT.[23]

Clearly, the programming leads to extensive repetition. Mac Low's mediation of the materials, however, contains enough compelling variation (by virtue of the way its message lists, or output, subtly permute and combine the poet's lists). Although it may not be cubist per se, the program, in effect, functions in the manner of a modernist writer such as Gertrude Stein, featuring poetic distinction and familiar belletristic traits. Repeated themes and phrases are organized so the segments of language work together. Since the elements in the database are coherent, referencing nature and its processes, some semblance of sense and meaning—as well as an epiphanic push at the end (the fortunate turning toward the light)—substantiates this effort as a literary work.[24]

VISUAL STRUCTURES

During the mid-1960s, Emmett Williams—known for many achievements, including his participation in the Fluxus movement—produced two computerized permutation texts that further exemplify how diversified the form had become, even in its earliest years. In the most accessible work from this era, "IBM" (1966), Williams programmed a poem that he had originally formulated in 1956. The process, as described in *A Valentine for Noël*, involved randomly choosing twenty-six words from a dictionary (or by chance operation) and then associating each of them with a letter of the alphabet to create "an alphabet of words."[25] A three-letter title was chosen, and the first line of the poem was determined by substituting words for letters in the title. Letters of words in one line were then used to make subsequent lines.

The vocabulary in Williams's original version of the poem produced (beginning with "IBM" as input) "red up going" as the title, followed by the lines "perilous like sex / yes hotdogs / evil jesus red black evil."[26] These ten words and forty-six letters led to a ten-line, forty-six-word poem; the volume of the poem expanded exponentially. Williams, however, was not interested in this possibility beyond processing the letters of each of the words in the second set of lines one more time—probably because configuration of the lines became quickly redundant as a result of the small vocabulary. He writes of the computer as "the muse's assistant," and says that the poem as such amounts to an "eternal project," but considers it "no great accomplishment."[27] In the rendition of "IBM" shown in *A Valentine for Noël,* Williams enhanced the process by imparting a "cylindrical" effect by shifting the vocabulary attached to the letter after the third process of substitution was made (i.e., at that juncture, the initial "a" word became the "b" word, "b" became "c," and so on) to create new poems. In this version of "IBM," each letter is matched with each word in the vocabulary one time; thus twenty-six distinct pairs of poems appear, each stemming from "IBM" as input. A visual aspect was added when Williams used a Diatype machine to increase the size of a word each time it was repeated, as seen in this 1973 example of a second-generation poem originating from the phrase "fear hotdogs money" (figure 16.1).

Repetition of specific words leads to repetition of lines, which might not be as interesting were it not for the visual component that directly represents the informational temperature of the poem.[28] In contrast to Gysin's permutation output (which results from a processing of itself), Williams's combinatoric texts feature self-contained generative dimensions. Words are presented not to convey syntactically correct meaning, but are the result of the combination of letters found in a preceding iteration of the contrived structure, and are reused as many times as the letter initially appears. As they are repeated and increase in size, a verbal and visual amplification is presented. The word thus contains more weight and becomes, perhaps, thematic (as do "fear" and "idiots" in figure 16.1). This process increases the variables, and using a limited database (twenty-six listed words in "IBM"), a vast number of different poems can be made from a small amount of input text. Williams crafted the poem by formulizing the work and adding a visual framework, making it something other than an oversized permutation fraught with repetition. He used a specific, premeditated equation(s), as did artists and writers involved with Oulipo, which added qualities to the work that would not be present otherwise.[29]

Williams's other work, "Music" (1965), employed an IBM computer to identify the 101 most common words in Dante's *Divine Comedy,* and used those words to create a series of computer poems. Though the work is out of circulation, an intricate description is found in Jacques Donguy's essay "Poésie et ordinateur." Williams's poem recognized and especially emphasized the *Divine Comedy*'s

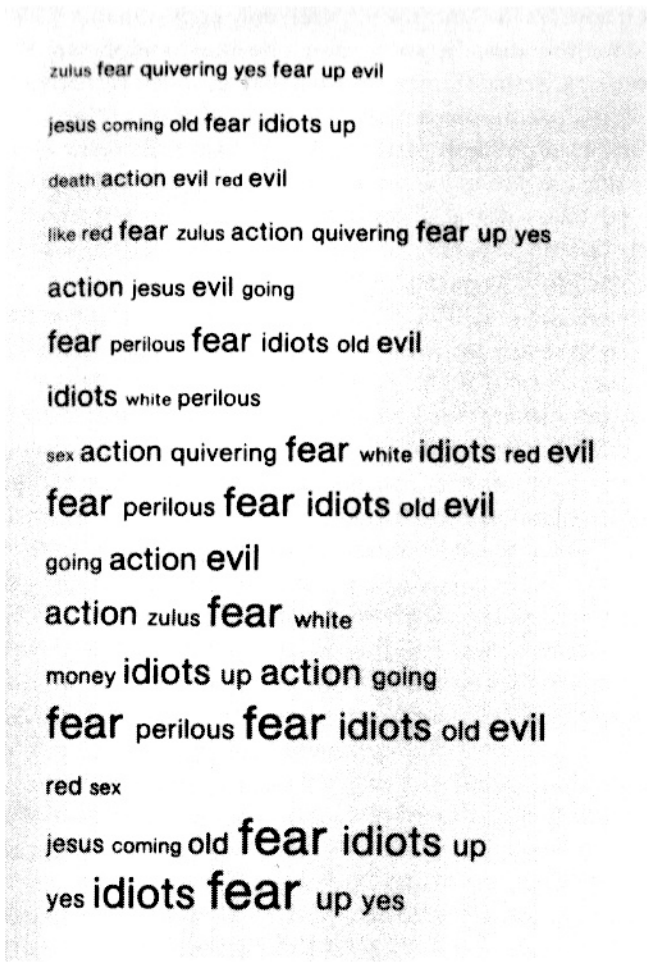


FIGURE 16.1. Emmett Williams, "IBM," in *A Valentine for Noël: Four Variations on a Theme* (Barton, VT: Something Else Press, 1973), 5–7. Print. Reproduced with permission by Something Else Press, New York, NY, © Ann Noëll Williams.

seven most-used nouns, its most-used adjective *(dolce),* and the word *amor* (which appears the same number of times as *dolce*). The most-used word in the *Divine Comedy, occhi* (eyes), appears 213 times, which establishes the number of lines in Williams's poem. In the poem, words are placed into nine columns, in alphabetical order. As the poem progresses, the lesser-used words disappear; the lines become increasingly shorter as each key word is used the same number of

times that it appears in Dante's work, until only *occhi* remains.[30] Unfortunately, due to its obscure status, the poem cannot be found and shown. Williams borrows a condensed verbal framework from Dante, which is mechanically represented into lines that diminish in relation to the number of times they appear in *Divine Comedy,* until a single word remains. The ideogrammatic appearance that would take shape as a result of this design scheme (a downward-pointing wedge) portrays more than a degree of concretist aesthetics, as represented in the shaping of works in anthologies of concrete poetry, such as Gerhard Rühm's "wand" and Williams's "do you remember?"[31]

As hardware and graphical programs were developed in the '60s, poets increasingly began to use digital tools to create visual poems. Examples of graphical digital poems began to emerge in the late 1960s. Marc Adrian, an Austrian sculptor and filmmaker who also made visual poems, created "Computer Texts" (figure 16.2), which were featured in the *Cybernetic Serendipity* exhibition. Adrian was one of several filmmakers who pursued the use of computer language and/or natural language and other mechanical fabrications that incorporated arbitrary functions and permutation algorithms, along with methods for the breakdown and sequential composition of images and text. In the example of "Computer Texts" documented in the exhibition catalogue, the computer randomly assembles poems by using a database of 1,100 alphabetic symbols to place 20 words at a time.[32] Reviewing the output shown in the exhibition catalogue, it appears that Adrian organized the interface using a grid system. The symbols retrieved from the database—which were letters or groupings of words—appear, at times, in layers upon each other, in rows and columns on the screen.

Adrian partially disguises the grid element by varying the size of the font and not using every line or block. He adds a fluid aesthetic quality to the poem by diminishing rigid shaping via this technique. The rounded sans-serif font also helps mask the x–y coordinates responsible for the symbols' arrangement, accentuating its visual properties. What looks like a combinatory poem is actually a graphical permutation. This example shows the piece to be verbally controlled, as only a single vowel, "o," is used; this restriction does not impede the poem, but rather imparts a design that reflects a particular technique and emphasis on both the language's appearance and its sound. Further, while some of the words (e.g., *cool, loco, old, do,* etc.) are known, other combinations reveal the experimental essence of the poem (e.g., *colpo* and the overlapped words). Neologism and graphical elements (overlapping words and smooth, scattered lettering) were not new to poetry; futurist, constructivist, Dada, and concrete poets had already implemented such textual conditions without the benefit of computers. Yet Adrian's piece is important for several reasons. These "Computer Texts" are among the first examples of work presented with unconventional "syntax," permutation, and

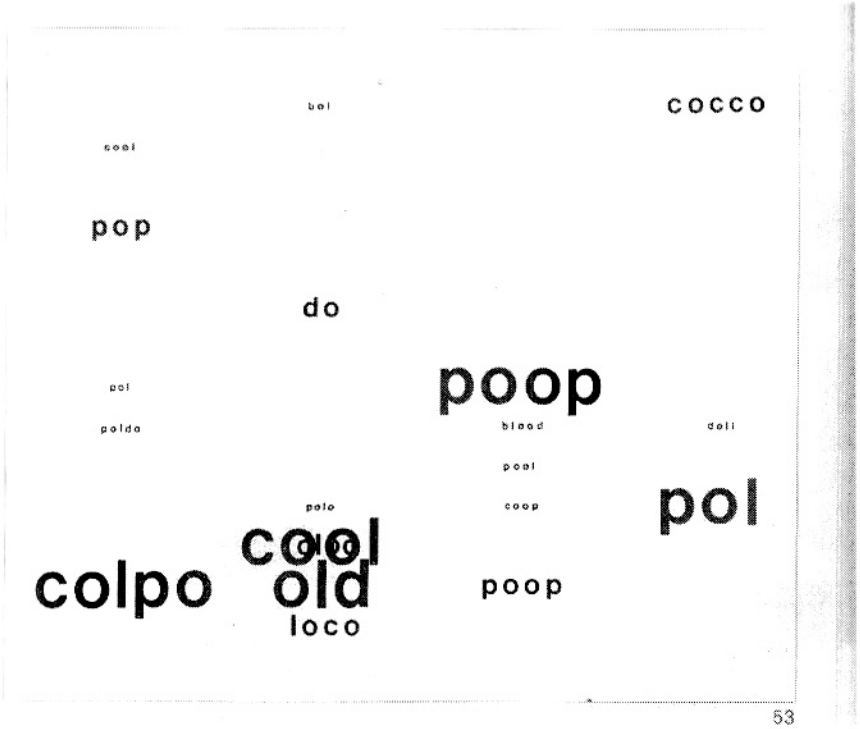


FIGURE 16.2. Marc Adrian, illustration for "Computer Texts," in *Cybernetic Serendipity: The Computer and the Arts,* ed. Jasia Reichardt (London: Studio International, 1968), 53. Courtesy The Studio Trust, www.studiointernational.com.

aleatoric reordering of fragments of language by a computer, a technique profoundly exploited by Hugh Kenner and Joseph O'Rourke, in their TRAVESTY program; John Cage, in his *I–VI;* Jackson Mac Low, in his book *42 Merzgedichte in Memoriam Kurt Schwitters;* and by others in later years. Pieces of words are combined by programming and hardware to present abstract artistic communication. Adrian used the machinery to place and displace language and meaning; readers are thereby challenged to build an understanding of a text they have played a role in activating. To see one example of the poem, as above, cannot fully illustrate the overall effect of the program. As the computer can perpetually reassemble the symbols, readers would normally see at least several screens; the poem would be accumulating content along the way. The reader has the prerogative to walk away from the work at any time, and would presumably do so when he or she is finished "reading" and thus co-composing.

This was not the only example of a filmic digital work structured with language made by Adrian in the 1960s. As he describes in a 1970 interview, he gathered four different films that assimilate digital technology and language under the general title *FILMBLOCK* (1962–64):

> The films TEXT I and TEXT II are a mere permutation; TEXT I results from a memory program of a computer. The words were chosen by the challenge that they can be read in English and German alike with no change of meaning. GO is another theme of permutation. It shows clearly how meaning in the consciousness of the spectator generates "by itself," based on a pure formalistic device. ORANGE, determined by a random generated scheme of free visual and verbal associations, is a montage that circulates round the idea and the picture of an orange.[33]

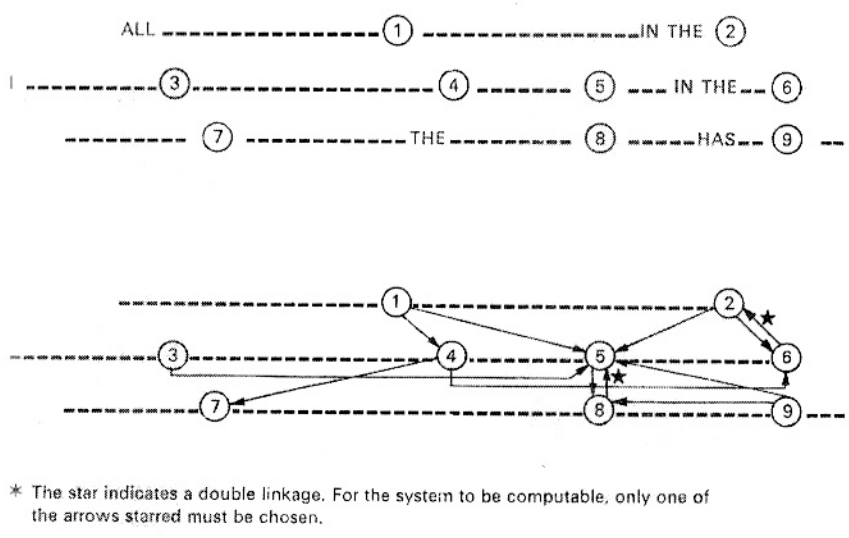
According to an announcement for a 2003 film festival in Germany posted on nettime.org, Adrian used a computer originally developed to record heart rates, which was connected to a cathode-ray terminal, to "write" directly onto the unexposed film material.[34] However, his practice was extended beyond a single idea or title. As Gerhard Ruhm writes in "The Phenomenon of the 'Wiener Group' in the Vienna of the Fifties and Sixties," Adrian was instrumental in fostering a "methodical inventionism" that was developing at this time in Austria, because "he referred to the usefulness of the Fibonacci series for the permutative processing of the accidentally, intuitively or schematically created stock of words," an idea that everyone else involved with the group began to explore.[35]

CONVENTIONAL STRUCTURES

From the very early stages of computer-generated poetry, programmers used classical forms as models in their experimentation. Though not as common as efforts to create open verse, several programs were written to codify the mechanics of established poetic forms. In order to effectively impose structure, which sometimes includes specific metrical or syllabic parameters, some programmers cultivated syntactical templates to help organize content and "slot" words into rigid patterns. However, just as poets such as Ted Berrigan loosely interpret the definition of the sonnet, few computer poems of this sort rigidly adhere to the tenets of classical structures (even if they do so nominally). The process of automating classical forms in itself would not be extremely difficult, but to write a program that shows versatility in output (i.e., one that does not essentially write the same type of poem over and over, as do the slotted works) requires flexible interpretation of form.

Of all verse forms that poets have attempted to program, haiku has dominated, probably due to its formulaic limitations. Charles O. Hartman connects the popularity of haiku generators to the motivations of Imagist poetry. In *Virtual Muse,* he writes, "As both poets and programmers have realized, for different reasons, the reader's mind works most actively on sparse materials."[36] Haiku's restricted sensibility of the formulaic obviously appealed to many authors, and its brevity made the unwieldy task of programming unique poems more manageable. The form's units of line (usually three) and metrical patterning (five syllables in the first line, seven in the second, and five again in the third) are literally and conceptually inscribed by various programs.

Margaret Masterman and Robin McKinnon Wood developed a "slot" structure to generate orderly haiku at *Cybernetic Serendipity.*[37] Their "Computerized Japanese Haiku" were written in TRAC and feature nine slots that are filled with words from nine different databases. The slots enable grammar to be preprogrammed and, by setting up a thematic association between one another, establish a poem's semantic center (slot 5, figure 16.3). Slot 1 relates to slots 4 and 5, slot 2 relates to slots 5 and 6, and so on (see arrows on the lower part of figure 16.3).



* The star indicates a double linkage. For the system to be computable, only one of the arrows starred must be chosen.

FIGURE 16.3. Margaret Masterman and Robin McKinnon Wood, illustration for "Computerized Japanese Haiku," in *Cybernetic Serendipity,* 54. Courtesy The Studio Trust, www.studiointernational.com.

Cybernetic Serendipity featured several poems created by this program, from which these examples are selected:

1 Poem
eons deep in the ice
I paint all time in a whorl
bang the sludge has cracked
. . .
3 Poem
all green in the leaves
I smell dark pools in the trees
crash the moon has fled.[38]

The poems reveal how generated works can be monochromatic in structure when syntax is unvarying and predetermined. The program produces syntactically and mathematically correct poems that effectively follow the haiku format, but repeatedly creates the same type of short poem. Words selected by the database are the only variable in the formula—in *Virtual Muse,* Hartman describes this approach to composition of computer poems as similar to the principle "used by Steve Allen in the old 'Mad Libs' game."[39] On the other hand, the computer program clearly and capably creates haiku (although without the nuance one finds in work by masters such as Basho, Issa, or Santoka); while there is some variation in syllabic content, each example approximates haiku (if not completely reflecting its classical attributes, as "3 Poem" does).

In the article "How to Write Poems with a Computer," published in the *Michigan Quarterly Review* in 1967, John Morris provides an accounting of his experience as an author of computer-generated haiku. Since "you could think of a poem," writes Morris, "as a kind of list, made up of sub-lists for its stanzas," he pursued his idea to test out "a new language" that was "designed to deal with lists."[40] Both the form and the vocabulary of the dictionary in Morris's work were borrowed from a collection of Japanese haiku. His program was able to generate two randomized haiku per second, and he made thousands of them, though the essay presents the only two generated works that the author valued, including this example:

Distance, I listen:
Far weird savage frozen spring,
Old song, echo still.[41]

Much of this essay is spent conceiving of a series of programmatic steps that would enable better results. Morris separates the internal dynamics of computer poetry into two distinct areas: the algorithmic (step-by-step procedure or instruction) and the random. The algorithm has three purposes: choosing, testing, and correctly ordering the words. Establishing the sequence of haiku, because of the loose and abrupt syntax embodied in the form, is a particular challenge because both algorithm and "anti-algorithm" are needed in the program, the latter feature essentially being a bypass mechanism that allows elements such as surprise and discordant ideas into the work. The semantic aspect of the algorithm, writes Morris, will be "about the size of an encyclopedia" and must be able to give "texture" to the poem.[42] Fundamentally, Morris expects of the machine what one would expect of a gifted poet, writing: "The computer must pay attention to rhythm and sound, and must somehow link texture with semantics to make each one complement the other—all without becoming obnoxiously evident in its task. It must grow banal when speaking of banalities, cool or crisp for the displeased mistress, hot and languid for a summer shower. At times it must play with the sheer sounds of words. (Whitman's 'Weapons shapely, naked, wan.')"[43]

He suggests that randomness is the counterbalance to the algorithmic, but also that computerized randomness is not connected to a work in the same manner as a poet's own internal, intentional efforts of randomness. In the end, Morris essentially argues that computers and programming are incapable of capturing the nuances of poetry; in many ways the essay is a defense of a highly idealized form of written poetry, which he defines finally as "a communication from a particular human being."[44] Morris's outlook on poetry is conservative—he expects a machine, or a human-machine collaboration, to do the same thing that a human does, which places rigid expectations on the work. Formally scrutinizing computer poems is valuable, as is holding them to standards. The context for computer poems includes traditional poetry, which in Morris's comparison triumphs; it is, however, also clearly wider in scope and something that resists archetypal structure. Digital creative writing has retained its integrity in open, variable forms, and new standards for judging their success are equally valid. In the considerate but conventional viewpoint expressed by Morris in this essay, technological shifts in the work (such as use of graphical tools) and trends in experimental poetry are absent. Morris speaks very generally about his work as an indicator of the possibility for digital authorship, and often sees his own work as an end, rather than a beginning. Clearly, designing and programming computerized poetry are a formidable task. Morris, in effect, positions producing quality results as an impossible task. Fortunately, other authors did not share his pessimistic outlook on the endeavor, as works created in this discipline continued to evolve during the following four decades.

OBSERVATIONS

An impressive range of styles of expression is found in the output of computer poems created in the first decade of their invention. The creative spirit and impetus of the era, to combine randomness with order through intricate, technical

art, altered the human relationship with language. Cyborgian poetry, works co-created by humans and digital machinery, emerged from these experiments. Digital authors proved that language could be digitally processed into shapes or sequences to create partially synthetic poetry, largely depersonalized and inclined toward abstraction. Some of these endeavors were rooted in Bense's theories, but computer poetry was a largely disconnected movement, with few central figures or theories.

Writing a computer program that will generate captivating text involves multiple imaginative steps. Carole McCauley quotes an unidentified essay by Margaret Masterman that addresses the intentions of this type of poetry: "The ultimate creative act for the computer poet lies in writing the thesaurus and in filling in the semantic directives. Thus the human creative process is pushed one stage further back; and the poet composes a poetic system, which can produce for him any number of poems formed from a given frame, among which he then chooses, rather than himself straightforwardly writing one poem, and then altering it."[45] In Masterman's valid assessment, computer poems rely extensively on derivation and programmatic divination. Derivative approaches have always been used to make poetry with text generators. Early methods involved using algorithmic coding to manipulate previously existing texts to re-present or rearrange the words (e.g., Lutz's "Stochastic Text," Balestrini's *Tape Mark* works), or creating programs that selected words from a database and combined them into lines or sentences. Randomization, repetition of words, discursive leaps, and quirky, unusual semantic connections are the predominant characteristics of these poems. Permutation and patterning are the traits that most directly connect this work to the history of poetry. Japanese *tankas* (circa 800 A.D.), medieval triolets, sonnets, and so many other forms of poetry consist of highly patterned language. As we see above, poet-programmers are inclined to forge many textual arrangements and develop fresh approaches to composition, without sharing common central concerns or aesthetic uniformity.

Early computer poems often show great effort (in terms of preparing code and selection of database material) to give digital poems a sense of cohesion. Despite the random effects imposed on the poems by complex programming, one can find an intentional plotting of associated fragments of language and thought, similar to those found in modernist poetry. Another style emulates the Dadaist practice of reordering the words of one text in order to make a new text, a practice called "matrix" poetry by later practitioners. This approach invites and permits poets to use previously composed texts within new, perhaps seemingly unrelated contexts, as Marcel Duchamp did (using other premises) in his "readymade" artworks. The principles involved with the poesis of such works—especially in works that re-generate themselves—move away from creating singular artifacts. Such models of expression are, as Peter Bürger has remarked, "not works of art but manifestations."[46] Later practitioners and theorists of computer poetry, such as Jean-Pierre Balpe, view and celebrate the nature of such work as infinite; programs can endlessly create unique transformative poems. The moment at which the poem is generated is a fusion that occurs between the software/algorithm and the interface; at one moment the materials are one thing (a set of words in a database), and at another they are something else (words shaped into a poem). The production of serial texts in this manner, mutations and manipulations of the language of a database, opens the possibility of a continuous perpetuation of language and ideas. This work emerged during a period when poets, critics, and others were first exploring the relation of language to the world, paying particular attention to language as a system with variable properties. When we encounter the various forms of digital poetry, we see a representation of our highly technological world; within the myriad types of expression, the artist often seeks to expose, and sometimes subvert, the various binary oppositions that support our dominant ways of thinking about literature (and, perhaps, about communication in general).

NOTES

1. Carole Spearin McCauley, *Computers and Creativity* (New York: Praeger Publishers, 1974), 113.

2. The *Cybernetic Serendipity* catalogue (*Cybernetic Serendipity: The Computer and the Arts*, ed. Jasia Reichardt [London: Studio International (special issue), 1968]) reports that the operations involved with the successful production of Nanni Balestrini's *Tape Mark* poems required the author to create 322 punch cards and input 1,200 instructions into the computer (55).

3. Ted Nelson, *Computer Lib / Dream Machines: New Freedom through Computer Screens—A Minority Report* (Chicago: Hugo's Book Service, 1974), 15.

4. http://permutations.pleintekst.nl.

5. Three years later, Bense published one of the first essays about composing "artificial" poems, "Über natürliche und künstliche Poesie" (On natural and artificial poetry), in *Theorie der Texte (Text theory)* (Cologne: Kiepenheuer & Witsch, 1962).

6. "Stochastic Text," *Augenblick* 4 (1959); republished at www.reinhard-doehl.de/poetscorner/lutz1.htm (accessed July 7, 2003); see also www.stuttgarter-schule.de/lutz_schule_en.htm (accessed July 9, 2005).

7. Trans. Helen MacCormack, www.stuttgarter-schule.de/lutz_schule_en.htm (accessed July 9, 2005).

8. Friedrich Block and Rui Torres, "Poetic Transformations in(to) the Digital," www.netzliteratur.net/block/poetic_transformations.html (accessed February 14, 2012).

9. Lutz also published examples of stochastic text, and similar works, in a book he cowrote with Rolf Lohberg titled *Cybernetic Computer and Electronic Brain* (New York: Bantam, 1986). Examples of his "Autopoems" are also found in Abraham Moles's *Art et ordinateur (Art and computer)* (Paris: Casterman, 1971) and in Pedro Barbosa's *A Ciberliteratura: criação literaria e computador* (Lisbon: Ed. Cosmos, 1996). The German artist Johannes Auer has created two websites that emulate the program, http://auer.netzliteratur.net/o_lutz/lutz_original.html and http://copernicus.netzliteratur.net/index1.html.

10. Brion Gysin, "I am that I am," in Richard Kostelanetz, *Text-Sound Texts* (New York: William Morrow, 1980), 373.

11. From *Brion Gysin: Tuning in to the Multimedia Age*, ed. José Férez Kuri (London: Thames & Hudson, 2003), 93. The programming details are not available; alternate versions of the poem, in which the words appear with a different sort of arrangement, are included in Emmett Williams, ed., *An Anthology of Concrete Poetry* (New York: Something Else Press, 1967), and in Kostelanetz, *Text-Sound Texts*.

12. Brion Gysin, "Cut-Ups Self-Explained" (1964), in *Brion Gysin*, 154.

13. Ibid.

14. No specific information on which program was used is available; it may have been Autocoder, which was the program used most commonly on the IBM 7070, or FORTRAN or RPG (Report Program Generator), which also ran on that machine.

15. Nanni Balestrini, "Tape Mark," trans. Edwin Morgan, *Cybernetic Serendipity*, 55. There are significant discrepancies between Morgan's shapely translations of "Tape Mark" and examples of output produced by the Syntext version of the program (1994). Syntext produces four blocks of Italian text each time the program is activated. There are no punctuation marks other than a final period, and words that happen to fall on the right-hand margin are split into two lines. As blocks of text, the poem requires the viewer to formulate interpretation, or separation of text into distinct units; its output on the screen lacks formatting and is blurred rather than sculpted, with poetic nuance. While the language is familiar, the lack of punctuation and line breaks emits a strikingly different poetic arrangement.

16. "The Pocketa, Pocketa School," *Time*, May 25, 1962. The article notes the Librascope Division of General Precision Inc. in Glendale, California, as the site of the computer. Charles O. Hartman, in *Virtual Muse: Experiments in Computer Poetry* (Hanover, NH: University Presses of New England, 1996), 2, lists R.M. Worthy as author of the program and reports that examples of Auto-Beatnik poems were published in a magazine called *Horizon* (1962): 2. Only one Auto-Beatnik poem can be found on the Web at present, "Poem No. 41: Insects," http://hem.fyristorg.com/stettin/hemsida/poem.html (accessed August 8, 2004).

17. "The Pocketa, Pocketa School," *Time*, 99.

18. Ibid.

19. Jean Baudot, *La Machine à écrire (A writing machine)* (Montreal: Éditions du Jour, 1964), 16, quoted in *Cybernetic Serendipity* and trans. for this chapter by Stephan Smith.

20. Jean Baudot in *Cybernetic Serendipity*, 58.

21. Mac Low also used the PFR-3 to compose poetic "stories" together titled "South," which are referred to and shown in McCauley's *Computers and Creativity*. The distinction of these works as "stories" is casual, referring to their narrative value; it is not a meaningful formal distinction.

22. Jackson Mac Low, *Representative Works: 1938–1985* (New York: Roof Books, 1986), 209.

23. "Printout from 'The,'" in ibid., 214–15.

24. Mac Low conducted numerous computer-aided compositions during this period, in which turns of phrase and sometimes the raising of questions—which have the effect of presenting a sudden thought—alter the narrative and transform the reader's perspective. One such poem, "South" (see note 20), is grouped with the PFR-3 poems in Mac Low's *Representative Works*.

25. Emmett Williams, *A Valentine for Noël: Four Variations on a Theme* (Barton, VT: Something Else Press, 1973), 3.

26. Williams, "red up going," in ibid., 5.

27. Williams, *A Valentine for Noël*, 4.

28. To obtain the "informational temperature of the aesthetic text" was an objective proposed by Haroldo de Campos as one of the tasks of concretism in 1960.

29. The Oulipo group, founded in France in 1960, advanced various forms of noncomputerized procedural poetry and writing that employed arithmetic and other programmatic constraints.

30. The words that compose the poem, besides *occhi, dolce* (sweet), and *amor* (love), are, according to Donguy, *mondo* (world), *terra* (earth), *maestro* (teacher), *ciel* (sky), and *mente* (mind). Jacques Donguy, "Poésie et ordinateur," in *Littérature et informatique: la littérature générée par ordinateur*, eds. Alain Vuillemin and Michel Lenoble (Arras: Artois Presses Université, 1995), 212–32, quotation in section 4.

31. Gerhard Rühm, "wand" (n.d.), in *Anthology of Concretism*, ed. Eugene Wildman (Chicago: Wild P, 1970), 102; Emmett Williams, "do you remember?" (1966), in *An Anthology of Concrete Poetry*, ed. Williams, 322.

32. Marc Adrian, "Computer Texts," in *Cybernetic Serendipity*, 53.

33. Adrian quoted in Peter Weibel, *Beyond Art: A Third Culture: A Comparative Study in Cultures, Art, and Science in 20th-Century Austria and Hungary* (Vienna: Springer Wien, 2005), 148.

34. See www.nettime.org/pipermail/nettime-ann/2003-November/000291.html (accessed July 22, 2004).

35. Gerhard Ruhm, "The Phenomenon of the 'Wiener Group' in the Vienna of the Fifties and Sixties," www.ubu.com/papers/ruhm_vienna.html (accessed February 2012). Years later, Adrian produced a book, *die maschinentexte* (Sydney and Granz: Gangan Books, 1996), in which some of his experiments in this area are documented.

36. Hartman, *Virtual Muse*, 31.

37. Masterman was a member of the Cambridge Language Research Unit. She was not a poet, but rather a scholar who wrote profoundly on the growth of scientific knowledge (including a widely cited essay, "The Nature of a Paradigm," *Criticism and the Growth of Knowledge*, eds. Imre Lakatos and Alan Musgrave [Cambridge: Cambridge University Press, 1970], 59–89) and who became extremely interested in machine translation.

38. *Cybernetic Serendipity*, 54.

39. Hartman, *Virtual Muse*, 31.

40. John Morris, "How to Write Poems with a Computer," *Michigan Quarterly Review* 6, no. 1 (Winter 1967): 17.

41. Ibid.

42. Ibid., 19.

43. Ibid.

44. Ibid., 20.

45. Masterman quoted in McCauley, *Computers and Creativity*, 115.

46. Bürger quoted in Marjorie Perloff, *Radical Artifice: Writing Poetry in the Age of Media* (Chicago: University of Chicago Press), 5.