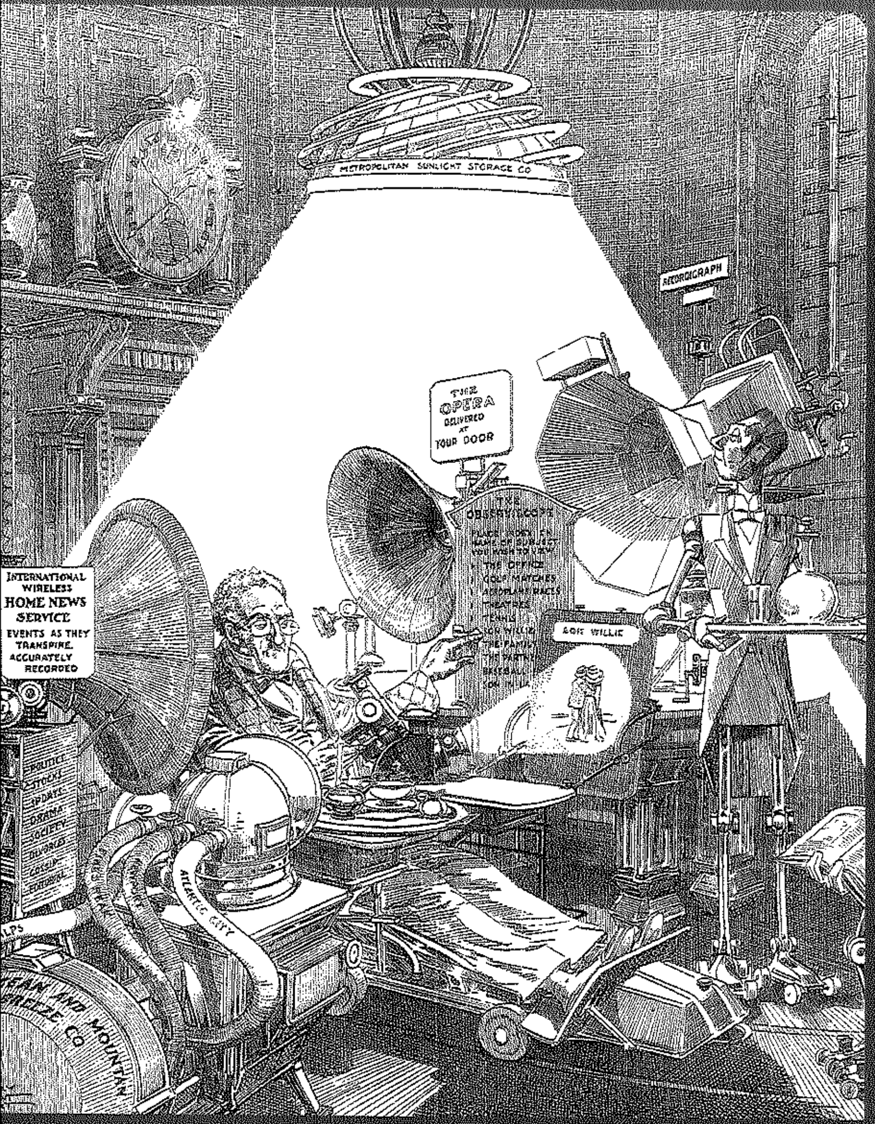
MEDIA ARCHAEOLOGY

Approaches, Applications, and Implications


METROPOLITAN SUNLIGHT STORAGE CO
RECORDIGRAPH
THE OPERA DELIVERED AT YOUR DOOR
INTERNATIONAL WIRELESS HOME NEWS SERVICE EVENTS AS THEY TRANSPIRE ACCURATELY RECORDED
SON WILLIE
ATLANTIC CITY
MOUNTAIN BREEZE CO

Edited by Erkki Huhtamo and Jussi Parikka

14

Digital Media Archaeology

Interpreting Computational Processes

Noah Wardrip-Fruin

The media archaeology approach has often unearthed forgotten moments from predigital media and, bringing them into the present media context, has both seen them anew and used them to illuminate the media culture of today. This chapter, instead, attempts a media archaeology of the more recent past—unearthing forgotten moments from the early history of digital media.

In particular, this chapter is a prototype of an archaeology of specific digital works and systems. Such an investigation cannot limit itself to published accounts of the system outputs, or even to stored interaction transcripts. It is important to understand how the digital artifact and the systems that supported it actually functioned—the operations, the processes.

The example discussed here is Christopher Strachey's 1952 love letter generator for the Manchester Mark I. This work, likely the first experiment with digital literature and digital art of any kind, is fully documented in notes and program listings found in Strachey's papers at the Oxford Bodleian Library. Fully engaging this work turns one not just to an explication of its operations but to their interpretation.

This, in turn, points toward a central issue both for the development of a digital media archaeology and for the future study of digital media generally: How do we engage a work's processes? Digital media are not simply representations but machines for generating representations. Like model solar systems (which might embody a Copernican or geocentric perspective while still placing the sun and planets in similar locations), the operational and ideological commitments of digital media works and platforms are visible more in the structures that determine their movements than in the tracing of any particular series of states

or outputs. As a step in such a direction, this chapter concludes by drawing on some of the analysis of Strachey's generator in considering a much more recent work: Marc Böhlen's Amy and Klara.

STRACHEY'S LOVE LETTER GENERATOR

People must have wondered if Christopher Strachey's father would amount to anything. Born to one of England's prominent families, Oliver Strachey was addicted to puzzles, expert at chess and bridge, a lover of crosswords, a trained pianist, and apparently ill suited to anything that mattered. He was not a good enough pianist to play professionally, he took an administrative job with the East India Railway Company and hated it, he was unhappily married and divorced. Then, at the outset of World War I, he took a post as a cryptographer with the War Office Code and Cypher School—and came into his own. His love of puzzles, and skill at them, made him a gifted codebreaker. He spent the rest of his career in cryptography and was honored as a Commander of the British Empire (CBE) in 1943.

In the mid-1940s there was reason to wonder if Christopher Strachey would ever share his father's experience of bringing the special shape of his mind to bear on a suitable task. He had been a bright child, playing imaginary three-dimensional tic-tac-toe on his mother's bed in the early morning and explaining mathematical concepts to his nurse at five. According to Strachey's biographer Martin Campbell-Kelly, his academic accomplishments were not a match for his intellect—though they did manage to get him into Cambridge University. At the university he continued to pay more attention to his social and intellectual life than to his academic performance. He graduated without much distinction in 1939, spent World War II working as a physicist, and in 1945 left that work to become a schoolmaster.

There was nothing to indicate that seven years later Christopher Strachey would find himself—through happenstance, through interests slightly askew from those around him—one of the people to do something "first." However, though it is not yet widely known, in the summer of 1952 he undertook the first known experiment in digital literature, and perhaps created the first digital art of any kind, when he completed his love letter generator for the Manchester Mark I computer.

THE GENERATOR'S GENESIS

Strachey came to this almost out of nowhere. In 1950, just two years earlier, he had no official connection to research communities of any sort—not mathematics, not engineering, and certainly not computation. He had developed an interest in modern computers through scattered articles, books, and radio addresses on the topic. He had never seen one.[1] And he had not shown a particularly keen interest in creating literature or art. But two things in his background can be seen as setting the stage.

First, certain circumstances of his upbringing may have made it more likely that a playful, creative experiment would occur to him as a possible use for a computer. Strachey was born in 1916, five years after his father married Ray Costelloe—an active suffragist and a trained mathematician and electrical engineer who came from an American Quaker background. In 1919 the family moved to Gordon Square, where Christopher's grandparents also lived. Gordon Square was then the center of the Bloomsbury group of artists and intellectuals, and Christopher's uncle Giles Lytton Strachey—a prominent member of the group—had just shocked the country with the 1918 publication of Eminent Victorians (a skewering of Cardinal Manning, Florence Nightingale, Thomas Arnold, and General Gordon). The family's neighbors included Virginia and Leonard Woolf, Clive and Vanessa Bell, and John Maynard Keynes.

Second, there was the happenstance of his choice of university. Strachey attended King's College, Cambridge, which was then quite small (about two hundred undergraduates). While there he met a junior research fellow named Alan Turing, who was, at just that time, undertaking perhaps the most fundamental work ever performed for the discipline of computer science (a discipline still some years from being founded). According to Campbell-Kelly, it is unlikely that Strachey spoke with Turing about computing at King's, but he did get to know him.[2] And as the Manchester Mark I computer was built, it was Turing who wrote the programming manual. Though Strachey was officially only a teacher at Harrow School, his personal connection with Turing was enough to allow him, in 1951, to ask for and receive a copy of the manual. And it was this that enabled Strachey's sudden appearance in the world of computing.

Strachey had first seen a modern computer earlier that year. He had been introduced to Mike Woodger of the National Physical Laboratory (NPL) by a mutual friend and had spent a full January day learning about the Pilot ACE computer then under construction at NPL (based on a design of Turing's). When he returned to school after winter break he began working on a program to make the ACE play checkers. Then he learned of the Mark I that had just been installed at Manchester, which Woodger informed him had a significantly larger "store" than the ACE—making it better suited to Strachey's programming interests. After receiving a copy of the programming manual from Turing, Strachey visited for the first time in July and discussed his ideas for a checkers-playing program with Turing. These ideas impressed Turing, and he suggested that the problem of making the machine simulate itself using interpretive trace routines would also

be interesting.[3] Strachey, taken with this suggestion, went away and wrote such a program. As Campbell-Kelly writes:

> The final trace program was some 1000 instructions long—by far the longest program that had yet been written for the machine, although Strachey was unaware of this. Some weeks later he visited Manchester for a second time to try out the program. He arrived in the evening, and after a "typical high-speed high-pitched" introduction from Turing, he was left to it. By the morning, the program was mostly working, and it finished with a characteristic flourish by playing the national anthem on the "hooter." This was a considerable tour-de-force: an unknown amateur, he had got the longest program yet written for the machine working in a single session; his reputation was established overnight.[4]

The attempts to recruit Strachey began immediately, and by November Lord Halsbury of the National Research and Development Corporation (NRDC) had convinced him to take a position as a technical officer. Strachey, of course, was still teaching at Harrow School—but in 1951 and 1952 he spent long sessions during his school breaks at Manchester, working on his checkers program and two assignments already given him by the NRDC. He also attended computing colloquia at Cambridge University and even gave a two-part BBC radio address about computers that spring. In his second BBC talk he described the multimodal interaction (image on a cathode ray tube, text on teleprinter) and unusual proto-personality of his checkers program:[5]

> In addition to showing a picture of the board with the men on it on a cathode ray tube, and to printing out the moves on a teleprinter, the machine makes a sort of running commentary on the game. For instance it starts by printing "Shall we toss for the first move? Will you spin a coin." It then calls, in a random manner, and asks "Have I won?" There's no cheating in this, at any rate as far as the machine is concerned. The player has then to feed his moves into the machine according to certain rules. If he makes a mistake the machine will point it out and ask him to repeat the move. If he makes too many mistakes of this kind, the remarks printed by the machine will get increasingly uncomplimentary, and finally it will refuse to waste any more time with him.[6]

By June 1952 Strachey had wound up his responsibilities as a schoolmaster and officially began full-time computing work as an employee of the NRDC. That summer he developed—probably with some input from others—a Mark I program that created combinatory love letters.[7]

It is unlikely that Strachey had digital art, of the sort we create today, in mind.[8] For one thing, there would have been little thought of an audience. As with his checkers-playing program, the love letter generator could be reported to a wider public but experienced directly only by a small audience of his fellow computing researchers. At the same time, it certainly was not an official assignment from the NRDC; rather, like many creative computing projects, it was undertaken for enjoyment and to see what could be learned.

Not everyone in Strachey's small audience enjoyed it equally. Turing's biographer Andrew Hodges reports that "those doing real men's jobs on the computer, concerned with optics or aerodynamics, thought this silly, but . . . it greatly amused Alan and Christopher."[9] Looking at the program's output today, we can understand why Turing and Strachey's colleagues thought the project silly. Here are two examples that Strachey published in an article in the art journal *Encounter* (immediately following texts by William Faulkner and P. G. Wodehouse):

Darling Sweetheart

You are my avid fellow feeling. My affection curiously clings to your passionate wish. My liking yearns for your heart. You are my wistful sympathy: my tender liking.

Yours beautifully
M. U. C.

and

Honey Dear

My sympathetic affection beautifully attracts your affectionate enthusiasm. You are my loving adoration: my breathless adoration. My fellow feeling breathlessly hopes for your dear eagerness. My lovesick adoration cherishes your avid ardour.

Yours wistfully
M. U. C.[10]

There could be a variety of reasons why, reading these, we might not share Turing and Strachey's great amusement. Perhaps we are further removed from a certain type of purple prose, or from that early computing culture focused on "real men's jobs."[11] But another reason seems more likely—that it is not simply the output that amuses; that the resulting letters are not really the interesting part of the project.

When we read an example of output from the love letter generator, we are seeing the surface manifestation of two other elements that remain hidden from us: the generator's data and processes. These two elements were what Strachey worked on, and any one of the vast number of possible output texts is only an unpredictable manifestation of them. It is likely that this unpredictability is part of what amused Strachey and Turing, but we will only partially understand it, or any other aspect of the system, if output texts are all we consider. Yet we are unfortunately likely to do so. In fact, most reports of the generator (including those in the excellent texts of Campbell-Kelly and Hodges) provide only sample outputs. The two exceptions to this that I have found are David Link's "There Must Be an Angel" and Strachey's own article in *Encounter*, both of which detail

the entire set of processes (at a relatively high level of abstraction) and a portion of the data.[12]

The views of the generator that include its data and processes, as well as its output, are views that consider the work as a system. This chapter takes such a view as its starting point for interpretation—finding a richness in the work unavailable to interpretations that focus only on the surface. This sort of interpretation is nearly demanded by early digital artworks, for which there was little or no consideration of an audience. But, as this chapter will argue, from a media-archaeological perspective it is an approach we can fruitfully bring into contact with digital art today.

UNDERSTANDING THE GENERATOR

If we are to view the generator as a system, we must consider its surface output, the data it employs, and the processes it executes.

We can begin with the surface. The generator's outputs have been used in discussions of queer identity, but the generator has rarely been considered carefully as a literary project. Certainly there are reasons for this—Turing and Strachey were both gay, and at least Turing openly so, at a time when homosexuality was illegal in England. It might also seem from widely reproduced outputs of the generator (e.g., that found in Hodges) that it was a love letter generator that "could not speak its name"—the word *love* being conspicuously absent.

But this does not explain the almost complete lack (with the exception of Jeremy Douglass's contribution) of attempts to read the generator's output in literary terms—to give it close consideration.[13] Surely our existing tools for literary work are sufficient to perform a reading of surface text. No, a lack of means for approaching the generator's output does not seem the likely cause of this silence. Rather, it seems more likely that scholars have not approached the generator's output from a literary perspective because it simply does not feel human. The letters preserved by Strachey are not texts that anyone would write—yet, unlike certain modernist and postmodern texts, they do not achieve a paradoxical interest value based on the knowledge that they were written by a person. These inhuman texts were, in fact, produced by a machinic process.

In part the inhuman feeling comes from the mismatched awkwardness of declarations such as "You are my avid fellow feeling" in the first letter reproduced above. But it may be even more dramatic in the patterns of permutational word repetition in the second example. In the first sentence we have "sympathetic affection" followed by "affectionate enthusiasm." In the next sentence, "loving adoration" followed by "breathless adoration." The following two sentences echo previous phrasing with "breathlessly hopes" and "lovesick adoration"—after which the "letter" is abruptly over.

The generator's texts do not seem like a fumbling attempt to express love, but like something other than expression, something that is not working toward the pleasures traditionally found in reading. How, then, can we read the love letter generator? If we are going to find more that interests us, in this project that "greatly amused" Strachey and Turing, we're going to have to look beneath the surface.

An entry can be found in the name of the hopeless romantic of the above letters. "M. U. C." is the Manchester University Computer, or Mark I. M. U. C. played the part of a love letter author by carrying out the following process, as outlined in the same article in the *Encounter:*

> Apart from the beginning and the ending of the letters, there are only two basic types of sentence. The first is "My—(adj.)—(noun)—(adv.)—(verb) your—(adj.)—(noun)." There are lists of appropriate adjectives, nouns, adverbs, and verbs from which the blanks are filled in at random. There is also a further random choice as to whether or not the adjectives and adverb are included at all. The second type is simply "You are my—(adj.)—(noun)," and in this case the adjective is always present. There is a random choice of which type of sentence is to be used, but if there are two consecutive sentences of the second type, the first ends with a colon (unfortunately the teleprinter of the computer had no comma) and the initial "You are" of the second is omitted. The letter starts with two words chosen from the special lists; there are then five sentences of one of the two basic types, and the letter ends "Yours—(adv.) M. U. C."[14]

With this entry, we will now examine the generator's data and processes.

The Generator's Data

What can be read in the generator's data—its sentence templates and, especially, the "lists of appropriate adjectives, nouns, adverbs, and verbs" available to the sentence-assembling process?[15] These are not a traditional text; rather, they represent the spectrum of possibilities for each open slot in the generator's structure, for each run of the program.

One thing we can do with the data is compare it against the ideas we have developed while reading examples of surface text. By looking at the complete list of available words (table 14.1), we can see, for example, that the absence of the word *love* from certain printed examples of the generator's output was simply an accident of randomness rather than a deliberate, telling lack built into the system. The generator's complete vocabulary contains *love, loves, loving, lovingly, lovesick,* and *lovable.* But, given the nature of randomness, we might not have realized this—even after reading many examples of the generator's surface output—without examining the system's data.

Another thing we can do with data is look for patterns in it. Data may be carefully authored or selected to work with processes in particular ways. It may have telltale absences or conspicuous repetitions. In this case, however, what seems

TABLE 14.1 The Love Letter Generator's Data

Category	Words
Adjectives	anxious, wistful, curious, craving, covetous, avid, unsatisfied, eager, keen, burning, fervent, ardent, breathless, impatient, loving, lovesick, affectionate, tender, sweet, sympathetic, fond, amorous, erotic, passionate, devoted, dear, precious, darling, little, lovable, adorable
Nouns	desire, wish, fancy, liking, love, fondness, longing, yearning, ambition, eagerness, ardour, appetite, hunger, thirst, lust, passion, affection, sympathy, fellow feeling, tenderness, heart, devotion, fervour, enthusiasm, rapture, enchantment, infatuation, adoration, charm
Adverbs	anxiously, wistfully, curiously, covetously, eagerly, avidly, keenly, burningly, fervently, ardently, breathlessly, impatiently, lovingly, affectionately, tenderly, fondly, passionately, devotedly, seductively, winningly, beautifully
Verbs	desires, wishes, longs for, hopes for, likes, clings to, wants, hungers for, thirsts for, yearns for, lusts after, sighs for, pines for, pants for, woos, attracts, tempts, loves, cares for, is wedded to, holds dear, prizes, treasures, cherishes, adores
Letter Start	dear, darling, honey, jewel, love, duck, moppet, sweetheart

apparent is a lack of careful shaping. Strachey wrote, in his *Encounter* article, that "the vocabulary is largely based on *Roget's Thesaurus.*" Here we can see that the data looks like a verbatim transcription from that source. From this we can begin to ask ourselves questions, but only preliminary ones. For what sort of processes would one choose to copy the data from a thesaurus, rather than carefully select each element? Is this data a determining factor for the work? What would happen if it was replaced by thesaurus entries associated with different interpersonal relationships, or with an entirely different topic?

As these preliminary questions reveal, most of what it might be interesting to interpret about this data can be considered only in the context of the generator's processes. In general, in process-intensive work, data is interesting primarily when considered for how it will be employed in processes. And so it is to this challenge, of interpreting the generator's processes, that we must turn.

The Generator's Processes

My approach—of interpreting systems—now comes down to a crucial question: How can we begin to read processes? That is to say, how can we begin to interpret what a work does, what it can do, instead of only what it says?

First we need to identify some features of the work's processes from which to begin our interpretation. One approach to this is comparison—considering two or more processes together, and seeing which shared and differing features emerge.

This is the approach taken here. I begin by comparing Strachey's love letter generator with two other works in which processes play a significant role: one is an influential literary work, and the other is the contemporaneously developed version of Strachey's checkers-playing program.[16] The features that emerge through this comparison, when considered in context, will form the starting point for interpretation.

One Hundred Thousand Billion Poems The Oulipo (Ouvroir de Littérature Potentielle, or Workshop for Potential Literature) was founded in 1960 by Raymond Queneau and François Le Lionnais. It was founded after Queneau ran into Le Lionnais, a friend of his, while at work on a difficult and unusual project that he did not feel he had the strength to continue.[17] Queneau reports, "He suggested that we start a sort of research group in experimental literature. That encouraged me to continue working."[18] The project was Queneau's *Cent mille milliards de poèmes*, or *One Hundred Thousand Billion Poems* (1961).

This work consists of ten sonnets, each having fourteen lines. While one might expect, then, that this work would be more suitably titled Ten Poems, something in the construction of each poem causes the number of potential poems to be much larger than ten. To wit: a reader can construct alternate poems by reading the first line of any of the original sonnets, followed by the second line of any sonnet, followed by the third line of any sonnet—and find that the whole work is artfully constructed so that any reading of this sort produces a sonnet that functions syntactically, metrically, and in its rhyme scheme. This is made easier by the way the poem is printed, with each poem on a page cut into fourteen strips that can be turned individually. Each line of poetry rests on an individual strip of paper, so that new poems can be composed by turning strips to reveal lines originally used for one sonnet or another.

This process, carried out by the reader, creates a dizzying number of possibilities. When one chooses which of the first lines to read, there are ten possibilities. Next, having read one of the ten first lines, one can choose any of the ten second lines—meaning that there are one hundred (10 × 10) possibilities for reading the first two lines. After reading a second line, one can choose any of the ten third lines—meaning that there are a thousand (100 × 10) possibilities for reading the first three lines, and so on. This type of work is called "combinatorial literature," and Oulipo member Harry Mathews, while incorporating a quotation from earlier writing by a fellow Oulipian, Claude Berge, writes of combinatorics that

> [its object is] the domain of configurations, a configuration being the preset arrangement of a finite number of objects, whether it concerns "finite geometries," the placement of packages of various sizes in a drawer of limited space, or the order of predetermined words or sentences."
>
> Arrangement, placement, order: because these are the materials of Oulipian combinatorial research, what generally results can be called rearrangement, replacement, reordering, subsumed by the generic term permutation.[19]

While combinatorial literature is concerned with the arrangement of fixed elements, it is important to note that not all the elements have to be employed in each result—not all the packages have to fit in the drawer. Certainly a major feature of Queneau's *One Hundred Thousand Billion Poems* is that only 14 of its 140 lines are used in the production of any of its potential sonnets. And from this we can see that Strachey's love letter generator is a work of combinatorial literature—one of those that preceded the first work of the Oulipo, a historical circumstance the Oulipo came to call "anticipatory plagiary."

What can we say about the love letter generator's processes, in comparison with those of *One Hundred Thousand Billion Poems?* To begin, we can observe that its processes are random and carried out by a computer, whereas Queneau's Poems are always the product of reader selection. The generator's processes are also quite a bit more combinatorial than those of the Poems. The generator carries out a combinatory process in the selection of nearly every word when creating sentences that follow the pattern of "My—(adj.)—(noun)—(adv.)—(verb) your—(adj.)—(noun)." In each of these word selections, the number of potential choices is also not small. For example, there are 31 possible adjectives that could occupy the open space after the sentence's initial "My" and 29 possible nouns for the slot following that, creating 899 possibilities for just the first three words of each sentence of this form (424,305,525 for a complete sentence).[20] *One Hundred Thousand Billion Poems*, on the other hand, is combinatory only on a line-by-line basis, and there are only ten options for each line.

But at least as important as the degree of combinatorial operation in these works is the nature of what is being permuted—the data that is being arranged by these processes. In Queneau's piece, the chunks of data are quite large: full lines of the sonnet. Because the chunks of data are large it is possible, though it requires a high-wire act of writing, for Queneau to enforce structure through the artful construction of each line. *One Hundred Thousand Billion Poems* not only maintains scansion and rhyme in all its permutations (which simply requires constructing only ten sonnets with identical schemes in these dimensions) but also syntactic sense, which requires artful parallel constructions across the sonnets. Further, the different original sonnets have different topics but evocative, potentially related imagery that enriches the possible reader experiences. As Stephen Ramsay characterizes reading One Hundred Thousand Billion Poems: "Though one might create a poem using random selections, there is nothing inherently aleatory about the process.... Rather, one consciously and deliberately looks for interesting combinations of lines and poetic effects. In building my sonnet, I found myself unable to resist the urge to make the wild horses of the Elgin marbles seize 'the thumb- and finger-prints of Al Capone.' . . . One has the clear sense of having discovered something with these combinations—of having liberated certain energies in the work—while at the same time having beat Queneau at his own game."[21]

Once again, we see the potential power of data-intensive approaches. In contrast, the love letter generator achieves greater combinatorial possibility by working with smaller units of data. It carries out more operations on smaller units—it is, in Chris Crawford's terminology, "process intensive."[22] Because the data units are small (individual words), and because the selection included in the work is not carefully shaped in any obvious way (e.g., only rhyming nouns), the love letter generator does not seem to achieve shape through data the way that *One Hundred Thousand Billion Poems* does.

It might be possible for the generator to, instead, achieve shape through process. For example, the processes could be elaborated to avoid particularly poor results (e.g., excessive repetition) or to enforce particularly pleasing patterns of some sort (e.g., linked imagery between sentences, alliteration, or even rhyme). Though some of these might require slightly more elaborated data, this does not seem the most important facet of the fact that more complex processes were not used to give more structure to the generator's output texts, to make them better love letters. So let us remember this fact and return to it after comparing the generator with another example process.

Strachey's Checkers-Playing Program Strachey completed the first version of his checkers-playing program for the Mark I before he began work on the love letter generator. His original design for the program focused on an approach from game theory that is now relatively well known as a "game tree search" or "minimax" algorithm. Strachey describes it as follows in his *Encounter* article:

> In the scheme actually used the machine "looks ahead" for a few moves on each side. That is to say that it selects one of its own possible moves, discovers all the legal replies for its opponent, and tries them out one by one. For each combination of its own move and its opponent's reply, it then finds all its own possible second moves and so on. As there are, on an average, about ten legal moves at each stage of the game, the number of moves it has to consider gets very large indeed if it is to look ahead more than a very few steps. After a certain number of these stages (the number being determined by the time taken) the machine evaluates the resulting position using a very simple scheme. It notes the value of this position and ultimately chooses the move which leads to the best possible position, always assuming that its opponent makes the best of the possible moves open to him.[23]

With this much description we can begin to draw some basic distinctions between the love letter generator and the checkers-playing program. Some distinctions,

such as that one set of processes selects words while the other selects game moves, are so straightforward that they will be passed over. More important is a comparison, for example, of how these selections are made.

Let's start at the beginning. When the checkers-playing program begins the process of selecting a move, it starts by looking at the current state of the board and then projects forward. This means that the program must constantly keep track of what is often called "state information"—it must "maintain state" over the course of the game—in order to know where to begin. And in selecting each move it projects forward many possible states, with the choice based on the best possible outcome of a series of moves.

By contrast, what kind of state does the love letter generator maintain? It must know what stage in the generation process is taking place—beginning, ending, or main body. It must also know when two sentences of the form "You are my—(adj.)—(noun)" appear consecutively in the main body, so that it can follow the rule requiring that "the first ends with a colon . . . and the initial 'You are' of the second is omitted." But the determination of which sentence types will be used is random, and so is the selection of the word that will fill each open slot for an adjective, noun, adverb, or verb. None of what has already been decided plays into the current decision, and certainly no forward projection of future possibilities is carried out.

Why is this? Certainly it is not because the computer was incapable of it, or Strachey was incapable of it—the checkers-playing program, after all, was written before the love letter generator. In part it may have been that the mathematical operations of playing a zero-sum game were more amenable to an approach that made complicated decisions based on state information. But more important than speculation, for our purposes, is the simple fact that a state-free design was chosen for the generator's processes.

Before discussing this issue further, however, let's look at another facet of the checkers-playing program. While the game tree search algorithm it used was not unknown at the time of Strachey's work, in the context of real-world computer checkers (in which speed issues required a limited number of projected future moves) it produced an unexpected behavior. Strachey reported this unexpected result to computer scientists at the Association for Computing Machinery's national meeting in 1952 and then put it in layman's terms for his Encounter article:

> There is, however, one feature of the machine's game which was quite unexpected and rather interesting. The way in which the machine values the positions it reaches when looking ahead is extremely crude. It counts each man as one point and each king (being obviously worth more than an ordinary man) as three points; the value of any position is the difference between its own points and its opponent's points. A large number of the positions it examines will, of course, have the same value, and it chooses between these at random.

> Suppose now its opponent has a man in the seventh rank so that he is about to make a king in his next move, and the machine is unable to stop him. The machine will effectively lose two points at its opponent's next move, and a human being would realise that this was inevitable and accept this fact. The machine, however, will notice that if it can sacrifice a single piece its opponent must take this at once. This leads to an immediate loss of only one point and, as it is not looking far enough ahead, the machine cannot see that it has not prevented its opponent from kinging but only postponed the evil day. At its next move it will be faced with the same difficulty, which it will try to solve in the same way, so that it will make every possible sacrifice of a single man before it accepts as inevitable the creation of an opponent's king.[24]

This type of behavior, in which there are complex (and likely some unexpected) results from the interactions of simple rules, is often in the digital arts called "emergent behavior." In this case, the behavior that emerges is not desirable (it leads to bad checkers playing) but it is notable for being both a completely logical outcome of the design of the system and an outcome that even the system's author did not foresee. Part of what sparks interest in process-intensive digital art is the possibility it seems to hold out for more positive forms of emergence—which will be able to surprise not only the system authors but also the audience.

Encounter readers were not given an explanation of how Strachey sought to address this problematic result, but he did give more information to the audience at the ACM meeting:

> In order to avoid this difficulty, the second strategy was devised. In this the machine continues to investigate the moves ahead until it has found two consecutive moves without captures. This means that it will be able to recognise the futility of its sacrifice to prevent Kinging. It is still necessary to impose an over-riding limit on the number of stages it can consider, and once more, considerations of time limit this. However, as no more *[sic]* is continued for more than two stages unless it leads to a capture, it is possible to allow the machine to consider up to four stages ahead without it becoming intolerably slow. This would mean that it would consider the sacrifice of two men to be of equal value to the creation of an opponent's King, and as there is a random choice between moves of equal value, it might still make this useless sacrifice. This has been prevented by reducing the value of a King from 3 to 2⅞.[25]

What Is the Generator's Game?

The above gives some indication of the level of complexity that Strachey's curiosity-driven (rather than NRDC-assigned) Mark I programs were able to achieve. Given this and our previous discussion, a series of observations present themselves for our interpretation. It is a potentially puzzling combination of facts. How should we consider the love letter generator's deliberate simplicity, its statelessness and randomness, and the fact that its vocabulary is a transcription from a thesaurus? This may not seem a puzzling set of facts on their own, but it seems more puzzling once we are reminded of the fact that this was not a project tossed off and then forgotten. In addition to Strachey and Turing's amusement at the time, Strachey also wrote of the love letter generator for Encounter two years later, and the project made enough of an impression that it has appeared in many accounts of his work, Turing's work, and early work with the Mark I.

David Link argues that the love letter generator is based on a "reductionist position vis à vis love and its expression. Like the draughts game that Strachey had attempted to implement the previous year, love is regarded as a recombinatory procedure with recurring elements."[26] Given the discussion above, I believe we should go further. The love letter generator is not just any recombinatory procedure with recurring elements but specifically a process designed to fail. Just as, when Polonius enters the stage, the audience waits for the next spectacularly vapid truism to escape his lips, I picture Strachey and Turing watching the teleprinter, knowing the processes that were going on within the Mark I, and waiting for the next formulaic jumble of those words most socially marked as sincere in mainstream English society. To put it another way, the love letter generator—in the way it operated—was a blunt parody of normative expressions of desire. It played the role of the lover as an inept spouter of barely meaningful, barely literate sentences, composed with repetitive randomness while one finger still rested in the thesaurus. Further, the examples chosen for preservation by Strachey appear to be those with particularly strong surface markers of mindless permutational patterns.

As a linguistic process designed to fail spectacularly and humorously, through randomness, the love letter generator is certainly not alone. Perhaps the best-known examples are the *Mad Libs* books, first published later the same decade (1958) by Roger Price and Leonard Stern.[27] Like the love letter generator, *Mad Libs* are defined by a process that fills in adjectives, adverbs, nouns, and verbs within given sentence structures. But *Mad Libs* can also request exclamations, geographical locations, numbers, colors, parts of the body, and so on. Further, most of the words in any given *Mad Libs* text are not blank but instead form a skeleton of a traditional text on a particular subject, with only strategic words left open. In addition, *Mad Libs* are not combinatorial. Rather than making all the possible words of each open sort part of the work, *Mad Libs* fills in their blanks by drawing on the suggestions of players who do not know the subject matter of the text for which they're providing material. For example, rather than choosing among verbs on a provided list, players of *Mad Libs* are free to use any verb they can recall.

The result is a process that everyone knows will fail. One player, who has seen the terms needed for the *Mad Libs* text, asks the others for the necessary types of words. The players then joyfully call out the most potentially inappropriate suggestions they can imagine. There is some humor in this, but much more in the anticipation—in waiting to see how this gathering of data will, when combined with the given data through the *Mad Libs* process, result in a ridiculous text. The anticipation is released as the final text is read, often to great laughter. But no one keeps the resulting texts around afterward. They are funny only as the anticipated, yet unpredictable, result of the *Mad Libs* process. And certainly, in this way, the love letter generator's products are more like *Mad Libs* than like the carefully crafted linguistic foolishness of characters like Polonius. However, it is important to note that Polonius is meant to represent a certain type of fool, while *Mad Libs* are not meant to represent any type of previously recognizable processes. *Mad Libs* are a humorous process, but not a representation—while I interpret the love letter generator as a representation.

Here, I believe, we come to a point at which we can understand the love letter generator, the first experiment in digital literature. It is a process designed to fail that employs a thesaurus-based set of word data and that can result in particularly inhuman surface texts (as seen in those selected for preservation by Strachey). We understand this combination in context—or, perhaps it is better to say two contexts: the technical context of the early stored-program computer on which Strachey worked as well as the social context of 1950s computing culture and the increasingly homophobic larger English society. Taking all this together, we can see the generator as a parody, through its operations, of one of the activities seen as most sincere by the mainstream culture: the declaration of love through words.

That is, I see the love letter generator, not as a process for producing parodies, but as itself a parody of a process. The letters themselves are not parodies of human-authored letters; rather, the letter production process is a parodic representation of a human letter-writing process. It is not a subtle parody, driven by a complex structures that circuitously but inevitably lead, for example, to the same small set of vapid sentiments stored as data. Rather, it is a brutally simple process, representing the authoring of traditional society's love letters as requiring no memory, driven by utterly simple sentence structures, and filled out from a thesaurus. The love letter generator, in other words, was as complex as it needed to be in order to act out a broad parody.

PLANS FOR ANOTHER GENERATOR

Coming to an interpretation of the love letter generator, however, does not bring us to the end of its story. Strachey's papers in the Oxford Bodleian Library also reveal his plans for a second version of the love letter generator. Here the parody would have been somewhat less broad. Also, while the sheer number of possible

$$\textit{There is no} \begin{pmatrix}\textit{girl}\\ \textit{woman}\\ \textit{one}\end{pmatrix} \textit{in} \begin{pmatrix}\textit{the universe}\\ \textit{the world}\\ \textit{existence}\\ \textit{England}\end{pmatrix}$$

$$\textit{who is nearly as} \begin{pmatrix}\textit{beautiful}\\ \textit{lovely}\\ \textit{delicious}\\ \textit{exquisite}\\ \textit{wonderful}\\ \textit{exciting}\\ \textit{attractive}\\ \textit{charming}\\ \textit{lovable}\\ \textit{nice}\\ \textit{inspiring}\end{pmatrix} \textit{as you.}$$

FIGURE 14.1. An example of a simple sentence grammar from Strachey's plans for a second love letter generator.

letters might not have been greater, the feeling of diversity would have been increased. Rather than each letter being an expression of inarticulate desire, the second version of the generator would have operated according to themes such as "Write to me," "Answer my letter," "Marry me," "Stop seeing that man," and "Tell your mother." Further, each output would have had a general style, such as "reproachful," "yearning," "impatient," "grateful," or "reminiscent." Strachey's notes contain many sample sentences for different permutations, such as:

I can't imagine why you are always seeing that man
How can you be so cruel as not to stop seeing that man?
Do I dare to ask you to stop seeing that man?
Don't go on seeing that man or I shall never speak to you again

His notes also provide many grammars for producing sentences along these lines. Figure 14.1 shows a simple one, and Figure 14.2 the ending of a more complicated sentence structure.

Strachey's plans are a pleasure to read through—with clever turns of phrase, amusingly structured processes and grammars, and occasional surprising nods (like the "among my wires" in figure 14.2) to the conceit that it is M. U. C. who declares love in these letters. But the project was abandoned, and Strachey's papers give no indication of the cause. Personally, my thoughts turn to the context of Strachey's work. I cannot help but wonder if it all began to seem less funny in the face of what had happened to the man with whom Strachey had stood and laughed while the first generator went about its foolishness.

When Strachey and Turing stood together in the summer of 1952, waiting for M. U. C.'s next pronouncement of love, Turing was on a forced program of hor-

$$\textit{...is}\begin{pmatrix}\textit{beastly}\\ \textit{wretched}\\ \textit{miserable}\end{pmatrix}\textit{without}\left\{\begin{array}{l}\textit{you}\quad\begin{pmatrix}\textit{in my arms}\\ \textit{sitting beside me}\\ \textit{among my wires}\end{pmatrix}\\ \textit{your}\begin{pmatrix}\textit{beautiful}\\ \textit{lovely}\\ \textit{delicious}\\ \textit{exquisite}\\ \textit{wonderful}\\ \textit{exciting}\\ \textit{attractive}\\ \textit{charming}\\ \textit{lovable}\end{pmatrix}\begin{pmatrix}\textit{tip tilted nose}\\ \textit{curly hair}\\ \textit{slender waist}\\ \textit{tapering fingers}\\ \textit{dimpled chin}\\ \textit{sparkling eyes}\end{pmatrix}\end{array}\right.$$

FIGURE 14.2. The conclusion of a more complex sentence grammar from Strachey's notes.

mone injections. Intended to render him impotent, they were also causing him to grow breasts and perhaps impairing his thinking. The injections were part of Turing's sentence—following his conviction, on March 31 of that year, for "gross indecency." Turing and Strachey may have laughed at the half-witted pronouncers of heterosexual love so roughly parodied by the love letter generator, but that meant laughing at a society all too willing to reject, incarcerate, and hormonally alter them for the simple fact of their homosexuality.

Two years after that summer, on the evening of June 7, 1954, Alan Turing took his own life.

LEARNING FROM THE GENERATOR

We might say that Lytton Strachey's *Eminent Victorians* and Christopher Strachey's love letter generator are both humorous critiques of conservative elements in English culture and those who held them in overly high esteem—one work operating through what it says, the other through what it does. The mode of the love letter generator, of expression through operation, is far more uncommon. But is it unique? Now, more than half a century later, we might turn to the question of whether what we have learned through an examination of Strachey's generator could help illuminate the contemporary landscape of digital media.

Consider Marc Böhlen's Amy and Klara.[28] At first it seems like a work far removed from Strachey's generator. Instead of reaching the audience as plain text on a teleprinter, Amy and Klara takes the form of two pink boxes from which large robotic eye-like speakers emerge. Then synthesized speech begins, with one robot commenting on an article from Salon.com. This quickly devolves into an uninteresting fight. For example:

Leave me alone.
What is wrong with you?
Leave me alone, please.
Weirdo.
Aha.
You are such a dork.[29]

Like the love letter generator's letters, the fights of Amy and Klara are not as well written as those produced by an average human writer. Further, like the teletype printing of Strachey's generator, the text-to-speech technologies that produce Amy and Klara's voices are flat, mechanical, and without nuance. If the goal were simply to create a compelling audience experience, it would be more effective to have the robots simply play a prerecorded fight between human voice actors.

But Amy and Klara uses text to speech for the same reason that the love letter generator uses a teletype rather than human handwriting. Each performance of Amy and Klara is an unpredictable expression of a much more complex underlying system, and the output must be able to vary widely for that expression to be possible.

When we begin to look at Amy and Klara as a system, we notice two things that may not be noticed by audience members. First, through slots in Amy and Klara's boxes, two cameras look at each other. Second, each robot also houses noise-reducing microphones. In other words, the robots of Amy and Klara not only "speak"—they also "see" and "listen."

In addition, the robots of Amy and Klara also "read." Each performs a statistical evaluation of the contents of Salon.com. This is the starting point for their dialogues, as the Amy robot chooses a topic identified by her reading of Salon.com on which to offer a comment. A text-to-speech system turns Amy's comment (assembled by an agent architecture in part based on AIML) into sounds sent through her speaker. Because the robots do not share data, the Klara robot only "hears" Amy's comment through her microphone—and must use automatic speech recognition technology to turn it into text. Given the limitations of software systems for text-to-speech conversion and automatic speech recognition, misunderstandings begin almost immediately. This is compounded by the fact that Böhlen has designed the Klara robot with a simulated thick German accent. As the robots find that they disagree, or that they believe they disagree, the exchange becomes unfriendly and, in time, simply an exchange of unpleasantries. Like the German accent, nasty exchanges are something for which text-to-speech systems (and agent architectures) are not generally designed. This required significant custom system work on Böhlen's part.

Here we can see that, just as Strachey's goal was not to reproduce the most effective human love letter, Böhlen's goal is not to reproduce the most engaging human fight. Rather, Böhlen assembles a system that, through its performance, expresses something about recognition and misrecognition, communication and miscommunication, mechanism and emotion. To interpret the work further as a system would require a careful examination of its surface, data, and processes. For our purposes it is sufficient to note that this system-oriented method of interpretation, strongly suggested by Strachey's early work in the digital arts, can offer an important perspective on digital works much closer to the present. This has implications both for future work in digital media archaeology and for the digital media field broadly.

Because processes are so central to digital media, an archaeology of digital media must move beyond what is done in most of the field's existing historical discussions. We can applaud the fact that historical work in digital media is already accustomed to studying technological systems and proposals other than those that achieve dominance—as media-archaeological approaches suggest. For example, digital media histories generally engage the ideas about hypertext in the writings of early pioneers such as Theodor Holm Nelson, Andries van Dam, and Douglas C. Engelbart, rather than only those of World Wide Web developer Tim Berners-Lee. On the other hand, the ideas expressed by the specific designs of the processes in nondominant systems are very rarely investigated. For example, I am aware of no critical work that investigates the processes in the "Green" and "Gold" Xanadu source code released from Nelson's project or the version of van Dam's still-functioning FRESS project demonstrated by David Durand and Steven J. DeRose.[30]

More generally, the digital media field must begin to grapple with the ideas embedded in its systems. Those working in the digital arts are often working in terms of processes (with inspirations ranging from John Cage to contemporary computer science) in ways that are invisible on the surface of their projects. Similarly, those working in commercial areas of digital media, such as computer games, construct systems that operationalize ideas of narrative structure, character behavior, linguistic interaction, and so on. Each of these is something that, in other domains, we are accustomed to scrutinizing closely, often seeking to understanding something of their underlying logic. But in the area of software, in which the underlying logic exists in an explicit encoding that can be examined, this takes place very rarely. As work continues in areas such as media archaeology and software studies, I hope we will develop a set of approaches and body of examples that will render this long-running lack a historical curiosity.

NOTES

1. Though he had used a differential analyzer, a kind of analogue computing machine, when working with differential equations during World War II.

2. Most of this account of Strachey's life and family is adapted from Martin Campbell-Kelly's

"Christopher Strachey, 1916–1975: A Biographical Note," *Annals of the History of Computing* 7, no. 1 (1985): 19–42, while the following material on Strachey, Turing, and the love letter generator also draws on Andrew Hodges's biography of Turing, *Alan Turing: The Enigma* (New York: Walker, 2000); Christopher Strachey's article "The 'Thinking' Machine," *Encounter* 3, no. 4 (1954): 25–31; Strachey's papers in the Bodleian Library (University of Oxford); and material from the British Broadcasting Corporation (BBC) archives. I am indebted to Oliver House and David Durand for archival work with Strachey's papers, and for the transcript of Strachey's second BBC address I am indebted to Allan Jones.

3. As David Durand points out (pers. comm.), having a machine simulate itself, as in the problem that Turing suggested to Strachey for his first Mark I program, is also the basic outline of Turing's demonstration of the halting problem—a lynchpin of Turing's argument in his essay providing the foundation for modern computation. See Alan M. Turing, "On Computable Numbers with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society* 2, no. 42 (1936): 230–65.

4. Campbell-Kelly, "Christopher Strachey," 24–25.

5. More significant than its questionable status as the first computer personality, Strachey's checkers program troubles the claim of A. S. (Sandy) Douglas's OXO to the title of "first graphical computer game." Douglas's program, which showed a game of tic-tac-toe on a CRT, was developed in 1952 for the University of Cambridge EDSAC.

6. Christopher Strachey, "Science Survey," transcript of radio address, BBC Home Service, 1952, sent to me in 2005 by Allan Jones, who has published on early BBC broadcasts on computing.

7. Campbell-Kelly, in "Christopher Strachey," notes some aesthetic advice from Strachey's sister Barbara, while Hodges, in *Alan Turing*, mentions collaboration with Turing, but neither source confirms the other's account on these points. In Strachey's writings he often fails to even credit himself (preferring to say that there is such a program and leaving aside who created it).

8. At the time of Strachey's projects, when the first stored program computers were just coming into existence, artistic applications of computers were essentially unheard of. According to Jasia Reichardt, the prominent curator of the 1968 computer art exhibition *Cybernetic Serendipity*, computer art's "first tentative steps date back to 1956." Jasia Reichardt, *The Computer in Art* (London: Studio Vista, 1971), 7. The earliest examples cited in current surveys of digital art, such as Christiane Paul's *Digital Art*, are from more than a decade after Strachey's generator. Christiane Paul, *Digital Art* (London: Thames and Hudson, 2003). It is, of course, quite possible that further research will reveal even earlier digital artworks than Strachey's generator. For example, C. T. Funkhouser has written of a 1959 digital poem created by Theo Lutz using one of Zuse's electronic digital computers—which may lead us to imagine that an earlier work of digital literature/art, using one of Zuse's earlier systems, might be uncovered through further research. But whatever happens, we do know that the field of digital literature has more than a half century of history, almost as long as that of the digital computer itself and perhaps the longest of any of the digital arts. See C. T. Funkhouser, *Prehistoric Digital Poetry: An Archaeology of Forms, 1959–1995* (Tuscaloosa: University of Alabama Press, 2007).

9. Hodges, *Alan Turing*, 478.

10. Strachey, "'Thinking' Machine."

11. Regarding the prose, Strachey, in his *Encounter* article, characterizes the generator's output as giving a "Victorian" impression (ibid., 26). But as George Landow (pers. comm.) points out, this seems the same view of Victorian culture found in Giles Lytton Strachey's *Eminent Victorians*, which is probably more amusing than accurate.

12. David Link, "There Must Be an Angel: On the Beginnings of the Arithmetics of Rays," in *Variantology 2: On Deep Time Relations of Arts, Sciences and Technologies*, ed. Siegfried Zielinski and David Link (Cologne: König, 2006), 15–42.

13. Jeremy Douglass, "Machine Writing and the Turing Test," presentation in Alan Liu's Hyperliterature seminar, University of California, Santa Barbara, 2000, www.english.ucsb.edu/grad/student-pages/jdouglass/coursework/hyperliterature/turing/.

14. Strachey, "'Thinking' Machine."

15. Of course, for many works of digital literature it is a challenge to get access to the data. While Strachey, like most computer scientists, published an account of his project's processes, it is rare to publish a project's complete data. In this case, the relevant papers of Strachey's at the Oxford Bodleian Library were consulted. These contain a complete program listing for the generator, from which its data were extracted (folders C 34 and C 35, box MS. Eng. misc. b. 259). Unfortunately, most early work in the digital arts was not so scrupulously preserved. The importance of preservation issues for digital literature, combined with some practical suggestions for current authors, is the subject of Nick Montfort and Noah Wardrip-Fruin's *Acid-Free Bits: Recommendations for Long-Lasting Electronic Literature,* 2004, Electronic Literature Organization, www.eliterature.org/pad/afb.html.

16. This chapter is not alone in making these comparisons: both Link and I made them in 2006. Noah Wardrip-Fruin, "Expressive Processing: On Process-Intensive Literature and Digital Media" (PhD diss., Brown University, 2006); Link, "There Must Be an Angel."

17. Raymond Queneau, *Cent mille milliards de poèmes* (Paris: Gallimard, 1961).

18. Quoted in Jean Lescure, "A Brief History of the Oulipo," in *Oulipo: A Primer of Potential Literature,* ed. Warren F. Motte (Lincoln: University of Nebraska Press, 1986), 32.

19. Harry Mathews and Alastair Brotchie, eds., *Oulipo Compendium* (London: Atlas Press, 1998), 129.

20. Given that each love letter contains five sentences, each of one of the two types, one can add together the number of possibilities for each of the two sentence types and then take the resulting number to the fifth power in order to determine the number of possibilities for the main body of the letter (leaving aside the letter's opening and closing words). I calculate this to be 753,018,753,081, 800,000,000,000,000,000,000,000,000—a number much greater than one hundred thousand billion.

21. Stephen J. Ramsay, "Algorithmic Criticism" (PhD diss., University of Virginia, 2003), 54.

22. Chris Crawford, "Process Intensity," *Journal of Computer Game Design* 1, no. 5 (1987), www.erasmatazz.com/page78/page31/page229/page241/ProcessIntensity.html.

23. Strachey, "'Thinking' Machine," 27.

24. Ibid., 28.

25. Christopher Strachey, "Logical or Non-mathematical Programmes," in *ACM '52: Proceedings of the 1952 ACM National Meeting* (Toronto) (New York: ACM Press, 1952), 49.

26. Link, "There Must Be an Angel," 25.

27. Leonard Stern, "A Brief History of Mad Libs," 2001, www.penguinputnam.com/static/packages/us/yreaders/madlibs/history.html.

28. Marc Böhlen, "Amy and Klara," in *Proceedings of ISEA 2006 Symposium / Zero One San Jose,* ed. Steve Deitz (2006), http://isea2006.sjsu.edu/content/view/261/49/.

29. Marc Böhlen, "Amy and Klara: Towards Machinic Male-dicta and Synthetic Hissy Fits," 2006, www.realtechsupport.org/new_works/male-dicta.html.

30. Udanax.com and Project Xanadu, "Xanadu Secrets Become Udanax Open-Source," 1999, www.udanax.com/; Steven J. DeRose, "Fress: The File Retrieval and Editing System," 2003, www.derose.net/steve/writings/whitepapers/fress.html.

15

Afterword

Media Archaeology and Re-presencing the Past

Vivian Sobchack

What may be called "presence" ("the unrepresented way the past is present in the present") is at least as important as "meaning."
—EELCO RUNIA, "PRESENCE"

Archaeologists should unite in a defense of things, a defense of those subaltern members of the collective that have been silenced and "othered" by . . . imperialist social and humanist discourse. . . . This story is not narrated . . . , but comes to us as silent, tangible, visible and brute material remains.
—BJØRNAR OLSEN, "MATERIAL CULTURE AFTER TEXT: RE-MEMBERING THINGS"

Both of these epigraphs, the first taken from a groundbreaking theoretical essay by a Dutch philosopher of history and the second from a "manifesto" by a Norwegian archaeologist, strike me as particularly relevant to the task of making sense of "media archaeology," however heterogeneous and literally unruly this undisciplined discipline might be. Much like the far-ranging essays in this volume, both epigraphs are dramatic articulations of a fairly recent, decidedly materialist, and generally antinarrative and antihermeneutic discourse focused on the conditions under which the absent past can be said to have "presence" in the present.[1] Thus this discourse is also concerned with the conditions for—and effects of—both "immediacy" and "mediation," even as it has not directly addressed the various entities and forms specifically designated as "media." As I will argue, this discourse of presence (a "presence in absence") and its particular concern with the past and the conditions under which it can be re-presenced (as well as historiographically communicated) are central to media archaeology. What, however, in the context of this discourse is meant by the term *presence?*