Always Follow the Instructions: Rules and Rule Following in Visual Art

Acknowledgements

I wish to thank my research supervisors, Neil Cummings, Dr Tom Corby and Professor Roger Wilson for their support and assistance in the preparation of this thesis.

Thanks also to the staff of Chelsea College of Art and Design's Research Office.

There were many anonymous programmers on obscure lists who helped me along knowingly, or not. I am grateful to them all.

My brief work with Simon at www.hitherto.net bore *Noumena*.

I don't think I would have made it without my partner Helen, even though she disowns all responsibility.

For the memory of GVMC

Contents

Chapter 2: Text Machine

Chapter 3: Instructions Rule

Chapter 4: Inscriptions

Chapter 5: Code

Chapter 6: A Typology of Text Machines

List of Appendices:

List of Illustrations

ABSTRACT

– THE UNIVERSITY OF THE ARTS

– ABSTRACT OF THESIS submitted by Wayne Clements

– FOR THE DEGREE OF DOCTOR OF PHILOSOPHY, FINE ART and

entitled

– Always Follow the Instructions: rules and rule following in visual art.

– MONTH AND YEAR OF SUBMISSION: OCTOBER 2005

The thesis examines the role of instructions in art by developing a theory of a text machine.

This machine is explored through a discussion of its rules and instructions and its codes and inscriptions.

The text machine is defined independently of particular instances of its making, of specific technologies, but for the practice part of this submission text machines are simulated by computer. This occasions a discussion of the impact of one machine (the computer) upon another machine, the text machine. This became my research question. This question is posed in this form:

“What is the impact of the computer on the text machine?”

A complex response to this question is developed by a discussion of rules and instructions, codes and inscriptions and their interrelationships. Larger questions are also raised, such as the use of text machines in day-to-day situations.

Methodology

1. Introduction: Key Terms

I use the term "***text machine***" in this thesis. What is this entity?

First of all, it is "***text machine***", not "writing machine", to distinguish it from gadgets with keyboards and so on, used to write, such as the typewriter. The use of the term "writing machine", in fact, goes all the way back to the typewriter's invention and first use. So Mark Twain (1906), an early typewriter owner, used the phrase to refer to his machine: *THE FIRST WRITING-MACHINES* (a memoir of his first "type-machine", as he also called it, of the year 1875).

I am interested in a machine at the same time as real but also more elusive. What I mean by ***text machine*** emerges from what follows. For the moment, the machine may be understood as a machine that…writes a text. This machine, it will turn out, is in essence the rules and instructions required to make a text.

These two terms, ***rules*** and ***instructions***, also require elucidation. No easy definition is possible, as emerges in the discussion that follows in this thesis. Several reasons for this difficulty are explored in Chapter 3. A working definition is: ***rule*** is the injunction, the something to be done; ***instruction*** is the 'how to do it'.

The terms ***computer*** and its ***code*** cannot pass without comment. My understanding of a computer, it will quickly become apparent, is not specific to a particular brand, or type of operating system. The discussion is pitched at a

higher degree of abstraction: the computer here is a model of a machine and is contrasted throughout with my text machine.

Code has several meanings and many connotations that go beyond a narrow definition of *computer* **code**. The connection between the latter and forms of social code is explored (particularly Chapter 5). The computer code I know is *Perl*. However, I do not engage in prolonged discussion of specific codes. I return to this issue in Chapter 1.3 (III).

What is a ***text*** is also not unproblematic, although it might seem almost self-evident. This thesis adopts a rather functional definition of a text as a "character string in the ASCII mode". This may seem quite indifferent to issues of meaning or interpretation, or the question of what is a text, its boundaries, such as is explored, for instance, by Jacque Derrida (1979) in *Living On: Border Lines.* A text is here, "a differential network, a fabric of traces referring endlessly to something other than itself, to other differential traces" (p. 84). My adoption of the far more functional definition of a character string, however, is derived primarily from the experience of programming text machines for the computer. However, such texts will prove no less hard to demarcate.

The nature of *this* text does not go unquestioned. The ***thesis*** raises issues about its own identity. (This is most explicit where the question of mechanical authorship is explored). The method I adopt is at points to *perform* this 'question of the text'. In Fine Art PhDs this sort of strategy is not unprecedented. Thus Joan Turner and Darryl Hocking (2004) can claim: "It may be that the performance of writing in the visual arts dissertation is one of the foremost contemporary examples of academic writing, where opaque or playful modes of writing are valued" (p. 157). This notion of play appears in the next section that discusses some ideas of *Gilles* ***Deleuze*** concerning historical periods.

My thesis intersects with Deleuze at several points. Deleuze, as a cursory reading will reveal, is interested in many subjects shared by this thesis: codes,

rules, machines, Markov processes. Shared interests, however, do not mean shared opinions, as we are no doubt all well aware. Rather than adopt Deleuze's periodisation uncritically, I use it now to dramatise my own recent development.

2. Preliminary

In his *Postscript On Control Societies* Deleuze (1995) proposes a periodisation of history into:

1. *Sovereign Societies*
2. *Disciplinary Societies*
3. *Control Societies*

These are roughly sequential. (I say roughly, because Deleuze is too subtle a commentator to lapse into a reprise of the crudities of Stalinist 'stages' theory. Deleuze fortifies his model against this charge by allowing for the possibility of the coexistence of different models of organisation, a subtlety sometimes foreign to his epigones. Nevertheless, we are, according to Deleuze, living in a *Control* Society).

Following Foucault, according to Deleuze, Control Societies have superseded (since around the Second World War) the Disciplinary Societies of the eighteenth and nineteenth centuries with their "sites of confinement". Power is no longer primarily articulated by physical incarceration but by the codes, the passwords, which determine access to knowledge.

These latter societies are, according to Deleuze, associated respectively with different forms of technology. *Sovereign Societies*: levers, pulleys, clocks. *Disciplinary*: thermodynamic (which I take to mean engines and motors: steam, internal combustion, electrical). *Control*: "information technology and computers" (p. 180).

Looking back, I might conceive of my research as a headlong recapitulation in miniature of external developments – one on fast-forward, with stages comically accelerated as in a silent movie chase. Thus, I began by making a text machine. At that time I did not even conceive it that way. It was literally clockwork. It had a wind-up motor borrowed from a child's toy. The clockwork device was soon replaced by an electric motor (see Plate 3). In turn these mechanically simple machines were supplanted by digital computers programmed to perform more complex processes.

My thesis overwhelmingly deals with the final phase. A bias reflected in my research question (above).

The possibility of alternative forms of text machine largely remains just that, a *possibility*, one that serves to make the point: a text machine may be made as many different machines.

My research might appear to relive other histories in its course. I began with programming relatively simple text processes on my desktop computer. One of the first of these was literally from the early days of computing, as it was a remake of a 1960's artwork. I went from this 'template' (fill the gaps) approach to more complex methods of text generation in a few short hops (see Chapter 6). I went from programming at the command line to programming a website in a similarly brief period.

Does ontogenesis recapitulate phylogenesis? Does my own development in computing (ontogenesis) repeat that of computing in general (phylogenesis)?

I think here I must refuse this, and any further possible, isomorphic figures whether borrowed from biology or elsewhere. My development was uneven and mixed (but not for the same reasons or in the same way that the world economy exhibits combined and mixed development in Trotskyist theory: I must refuse this final scenario along with the others). There *were* sudden developments followed by periods of apparent regression as some old

problem returned for consideration. I made progress in one area, whilst others lagged behind. I could benefit from the example of others' work; at other times I was working alone. As a result, my progress was uniquely my own.

3. Why?

I began with a wish to investigate some possible uses of instructions in art. My interest in the text machine grew from this. What began as an untried speculation – that it might be possible to program some of these machines for computer – grew into the project described in this thesis. In fact, it became the main discussion of my research, and the question my research seeks to answer:

"What is the impact of the computer on the text machine?"

This turned out to have a rather complex reply.

4. What?

What should a methodology do in the arts? What should *my* methodology achieve, what is it *for*?

The second question reasonably requires answer (the first exceeds my task); there are several possible replies. These are ranked in what I consider a descending order of difficulty:

1. My methodology should provide a procedure for mechanically producing the rules sets and instructions for new text machines.

2. My methodology should enable me to prove if there is a text machine that produced a particular piece of writing.
3. My methodology should tell us how to evaluate the worth of the writing and the machine that made it.
4. My methodology should provide a theory of what a text machine might be. (This in turn should allow me to make an answer to my research question).

The fourth is what my thesis seeks to achieve. The other three, in my opinion are, for differing reasons all, fundamentally, unanswerable but not equally interesting to discuss.

The first ("a procedure for mechanically producing the rules sets and instructions for new text machines") seems to be a holy grail of anyone interested in instructions in this area: a rule set that can produce new rule sets: a machine of machines. Presumably this machine might also be able to produce itself. It would itself be a machine, and if it can make machines it might make itself, or produce the rule for its own production. But this seems self-contradictory. How could it have produced itself? If it did not, then it will be incomplete as it did not write itself, and is not the machine that may produce all text machines.

A formula that would enable one to produce work endlessly is more than can reasonably be expected in the arts. Yet in a sense, this thesis can claim to provide a means to the production of *some* text machines, although not *all*. The machines in Chapter 6, for instance, were made in regard of this thesis and contribute to it: this thesis may in this qualified sense be thought of as a text machine.

The second ("to prove if there is a text machine that produced a particular piece of writing") is also insoluble perhaps. It is a question I engage with. However, I do not think it productive of a truly useful answer in most cases. A text machine *both writes and is written*. Authorship here is going to be complex. Furthermore, the evidence of whom or what writes is usually not

available, only the texts are available and we cannot work back to the hand (silicon chip?) of the author. I talk about this in Chapter 4.

The third ("how to evaluate the worth of the writing and the machine that made it") is a serious question for those interested in writing, machines, and how to evaluate them. I have not attempted to answer the conundrum seriously. The reasons for this are in the text: my response has been rather to evaluate the significance of the question. My opinion of this question is that it is, basically, a controversy about the value of the human versus the artificial. Couched in such terms, I do not think it answerable for those determined to advance one side against another. It is, in other words, an ethical and aesthetic matter and cannot itself be turned into a mechanically performable procedure.

The fourth ("a theory of what a text machine might be") is developed over the course of my thesis. It is, in my view, at least interestingly answerable. We do not have anything like a developed theory of a text machine. We have (it will be seen) the use of a phrase, or several related phrases. If my thesis succeeds in being this theory, it is a step forward. A thesis that provides a theory in a previously untheorised area can make a claim to have made a contribution to theory. What I have done is to develop some of the concepts and issues that are relevant to a study: these are the text machine, its rules, instructions, codes, and inscriptions; these are explored in some detail, getting approximately a chapter each.

This is *what* I aimed to accomplish. *How* did I hope to do it?

5. How?

The way I decided to explore the text machine was to make some machines. I did not, as I explain in the Introduction to this thesis, choose to make machines with motors and gears and sprockets. Instead, I learnt computer

programming, at least enough to program my machine. This approach began as a method of testing my claims: if I could program a text process, it was a text machine. Whatever the difficulties and doubts that might attach to this method, the method itself became a subject of investigation and interest; it became the basis of my research question, “what is the impact of the computer on the text machine?” This question came to involve a discussion of many things, such as instructions and program, text and code.

Essentially, what I did was to make work that engaged with the issues I encountered in my research and then reflect upon and evaluate the results. I placed the more successful results on my website (www.in-vacua.com). I made work and documented it. I presented work in public (at CHArt 2004 – see **Appendix 1**). I exposed it to peer review (by successfully submitting work to online software art groups, *Rhizome.org* and *runme.org*). I wrote and submitted to journals articles concerning issues related to my work (see Appendices).

These processes and events predictably gave rise to new questions and this in turn prompted new work. This was a relatively unstable process, as work changed ideas and ideas work.

However, this instability in a research process is not unprecedented and conforms to one (there are three: the “positioning of a practice”, the “theorising of a practice” and the “revealing of a practice”) in Katy Macleod’s (2004) list of types of practice-based research in art and design. Of the revealing of a practice she writes:

“Thus, the written text was instrumental to the conception of the art projects but the art projects themselves exacted a radical rethinking of what had been constructed in written form because the process of realising or making artwork altered what had been defined in written form.” (No page numbering).

This “seesaw” process is one familiar to me, as my increasing involvement in computers led me into investigations I had not dreamt of not long before. The consequences of this process appear in the body of the thesis.

The process of finding out what others had done in programming text machines, programming my own, and trying to understand what the machines were, lead me to write the typology in Chapter 6. These concepts were produced by my investigation: they are abstractions – and have a generality because of that. But they are also “real abstractions” in Peter Osborne’s (2004) phrase: abstractions of reality derived from an investigation of realities. These in turn are productive of knowledge as they enable me to make sense of the confusing epiphenomenona that is machine text. Furthermore, the work I now make and will make in the future will be made in the light of this advance in my understanding of my subject. In other words, how I work is informed by the development in my theoretical perspective that was itself the product of my making work and thinking about it.

Carole Gray and Julian Malins (1993) in their, *Research Procedures/Methodology for Artists & Designers* suggest that methodology is necessary if meaningful research is to be carried out, and that this methodology be *rigorous*, *accessible*, *transparent* and *transferable*. At the same time they note there “are no well-defined strategies on which researchers can draw” in the arts (p. 2). Things *may* have improved a little in the years since this paper was written, but I was unable to draw on an already developed, clearly defined system of inquiry in my research. I had to try to develop my own.

The approach of exploring a problem, familiarising myself with it, producing my own work and deriving a comprehension of the subject is one that recurred throughout my practice. This thesis is the accumulation of such insights. Its reflections upon the outcomes of research activity meet the criteria Gray and Malins demand.

My way of working usually meant identifying a technique for making a text, then undertaking background work on this approach. I then would find computer programs, if available, that made these sorts of texts. Then I would try to program my work. For instance, with the Markov chain algorithm, there was no off the peg solution to my problem, in the shape of a complete

program I could use. So I had to write the program myself using an algorithm I had found. Once I understood how it worked, I could make variants of my machine and go on from this base to write about the subject[1].

From piecemeal making I have moved on to the development of a system of methods, of principles that, for Gray and Malins, constitute a methodology.

Possibly I was slightly assisted because my project veers somewhat towards science subjects. (This is reflected in the fairly large number of scientists in my bibliographical references). Certainly, I was able to set myself a task – and carry it out in most cases. I do not suggest, however, that the results in this thesis can meet the criteria of provability and repeatability that may be associated with the best scientific findings. Nevertheless, the research has value.

[1] See Appendix 4.

Indeed, a writing machine can be a tool…but it does not have to be

Martin Heidegger

Chapter 1: Introduction

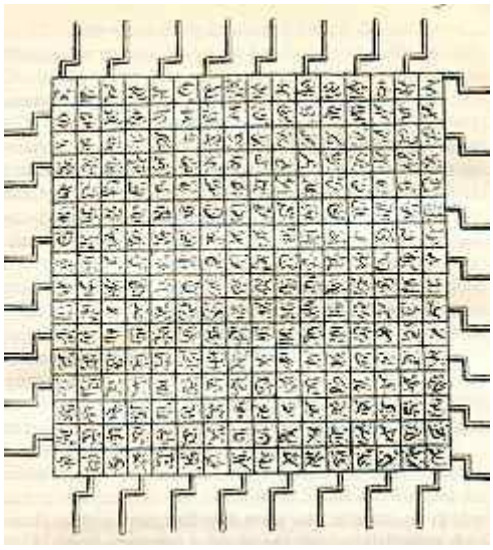
1. Text Machine: a working definition

As Richard Bailey (1974) acknowledged, quite a long time ago, it is established that nearly everyone who writes about machines that write starts with Jonathan Swift's (satirical) 1726 description of the Grand Academy of Lagado. Inserting this thesis into that tradition, I do the same, but with the purpose of giving an idea of what I mean by 'text machine'. The Lagadonian's machine was "mechanical not electronic" (Bailey op. cit. p. 283). Essentially it consisted of fixed syntactic structures and a randomising process for the contents, some idea of which may be gained from the illustration that is Plate 1. This is Swift's description:

"It was Twenty Foot square, placed in the Middle of the Room. The Superficies was composed of several Bits of Wood, about the Bigness of a Dye, but some larger than others. They were all linked together by slender Wires. These Bits of Wood were covered on every Square with Paper pasted on them, and on these Papers were written all the Words of their Language, in their several Moods, Tenses, and Declensions, but without any Order. The Professor then desired me to observe, for he was going to set his Engine at work. The Pupils at his Command took each of them hold of an Iron Handle, whereof there were Forty fixed round the Edges of the Frame, and giving them a sudden Turn, the whole disposition of the Words was entirely changed. He then commanded Six and Thirty of the Lads to read the several Lines softly as they appeared upon the Frame; and where they found Three or Four Words together that might make Part of a Sentence, they dictated to the

Four remaining Boys who were Scribes. This Work was repeated Three or Four Times, and at every Turn the Engine was so contrived that the Words shifted into new Places, as the square Bits of Wood moved upside down." [2]

Plate 1



The handles were turned and new sentences churned out and dutifully noted down. (Although a satire on what Swift describes as, "improving speculative Knowledge by practical and mechanical Operations", ibid. p.175, its idea of a basic, randomising method has proved quite durable. However, it is not the only option by any means: there will be *many different machines*). Whether electronic or mechanical, what I mean by text machine is: a machine that in its functioning writes a text. It is for this reason, I am not, as I have stated, referring to machines that may be used to write, such as typewriters, but do not themselves write. A text machine may be operated by a person, switched on, switched off, fed materials, but a text machine *writes texts*.

The text machine here is a contingent definition that I progressively destabilise during the course of my thesis. The concepts I rely upon to give

[2] Swift (1963) pp. 175-176.

the machine its description are its rules and instructions, the codes or languages it may use, and the inscriptions it may produce. None of these terms are unproblematic and they enter into complex, constellated relations as I develop my theme.

Almost too late for this thesis, I read Florian Cramer's (2005) *WORDS MADE FLESH*. His pamphlet discusses many things that appear in this thesis (and some that do not, as Cramer ventures into Lull (Ramon Lull, the medieval Catalan monk and Cabbalist) and Lullism and other occultisms I do not visit (see particularly pages 36-41 of his text). Nevertheless, although I have admiration for Cramer's writings, whilst reading his text I became aware not merely of the similarities in our interests, codes, poetry, Claude Shannon, the Oulipo, and many more, but also the substantial difference between the two texts, his and mine.

Codes, texts, rules and instructions interact both in my investigation and in Cramer's paper. What distinguishes our two texts is my bringing together of several strands under the theory of a 'text machine'. To be sure, he makes a couple of references to 'writing machines', but, as with others who use this or similar phrases, there is no developed understanding of what such a machine might involve.

Why propose such a machine – and what is it any way? To understand this it is possible to consider alternatives I might have chosen. Why, particularly, not say "system", as in "writing system"? Would not this heading be able to subsume within it all the various text machines of which I speak? The answer is, no. The reason is the same reason that I cannot substitute the word "machine" into the phrase "road system", and so make "road machine" (which is instead a common term for a car). Something else is required. Similarly, with roulette systems, such as the martingale[3] system, the system may be clearly described. But to set it in motion one needs a person, a gambler – or an automated game player, a gambling machine.

[3] I do not recommend it. Essentially it is, if you lose, double your bet. If you win, cash up and leave.

“Machine” is being used in my thesis in a way similar to that Deleuze uses it in his discussion of machines: the formation of materials – or individuals – into collectives. (Mumford, 1967, expresses a similar idea. The social in Mumford is already a machine, a mega-machine, and makes possible within it machines both technical and social). The machine here is something that does something: it is something that takes an input and transforms it to an output, or will if provided with the necessary circumstances.

The text machine(s) take their place among other machines, organising both their materials and their human users. But how do they achieve this? They do so by the deployment, significantly, of their rules and instructions, codes and texts – and us, their human ‘users’.

...

The development of my ideas involved passing through a number of areas, some of which are formally only distantly related. To give some sense of how I arrived at the formulation of a machine, as the organising theory for my discussion of instructions and art, I will now go through some of the contextual background to this thesis.

2. Contexts

Below I go through a series of relevant contexts. There is, however, a *problem* of context. My thesis is not really about computers, although it discusses them. It is not exactly about electronic literature, although it discusses this too. Nor is it about images, although it is a thesis in fine art. It relates to all these, but is not precisely about any one of them.

It is a thesis about a text machine; but there is a lack of research in my area. I may put the problem like this: there is a considerable amount of writing about the *writing*, but not much about the *machine* that writes. Nevertheless, I have

chosen to discuss this machine. To do this I have had to try to draw quite widely from sources that are not often encountered together.

Here I wish to discuss how electronic literature, computers, and visual art might relate to one another and how from the interstices between them the shape of a text machine might begin to emerge.

(I) Visual Art: Instructions and Text

My thesis relates to visual art through the questions first of instructions and secondly of text: the text machine I theorise is fundamentally the application of rules to text. ***Instructions and rules are what I am investigating; the field in which I am operating is that of visual art***.

Text is, since the 1960s, a well-established material of the visual artist, particularly the conceptually oriented practitioner. For instance, we have Art and Language's 1969 'Introduction' to their *Journal of Conceptual Art*. This saw the authors raise the possibility that their editorial itself might count as a conceptual art work (p. 99).

There have been many since (and not a few before) who have made text-based artworks, for instance Bruce Nauman, Jenny Holzer, or Hamish Fulton, and some of the recent work of Tracy Emin for example.

Instructions also are an established visual artists' practice. These too are usually in textual form. Consider (again) the history of conceptualism; Yoko Ono had transformed art to text by the time of her 1962 show in Japan.

"…in 1962, I did an exhibition of instructions as paintings at Sogetsu Art Center in Tokyo. I did a show of instruction paintings at AG Gallery in New York, but that was exhibiting canvases with instructions attached to them. Displaying just the instructions as paintings was going one step further,

pushing visual art to its optimum conceptualism (my italics); it would open up a whole new horizon for the visual arts" (Ono, 1995, p.5).

Whether or not one shares the belief, common for the time, in the antithesis of language ("the instructions") and art objects (paintings) to achieve a dematerialisation ("optimum conceptualism"), art as text had arrived.

However, I did not choose to leave matters there, as they were in the 1960s when according to Bruce Altshuler (2003), "the international art world was exploding with art-by-instruction" (p. 3). What I did essentially was to apply instructions to the production of texts. This application I characterise as a text machine.

It is this, my interest in the application of instructions to text that marks a point of departure from what might have become an archaeological interest in a variant of conceptual art. However, an interest in instructions also distinguishes my interest from one in electronic literature per se, as shall be seen below.

Where an interest in instructions, as a particular facet of Conceptualism, *is* still keen is in computer-based media. This is in part, I have concluded, because many of these artists are in tune with the critical or counter-cultural aspects of high Conceptualism. Also, their need to clearly formulate their work so that a computer may execute it draws them to areas of Conceptualism that are instruction oriented. However, the connection between art and computing was made early, in Jack Burnham's 1970 *Software* show that joined conceptual art and computing under the single term (see also Burnham, 1968).

A difficulty may occur where there is a misunderstanding of what Conceptualism is, if it be conflated with merely following clearly stated procedures alone; thus, a purported association with Conceptualism can flatter any sort of programmed art. Lev Manovich demolishes many of the

hopes of computer artists who wish to associate themselves with conceptual art in his essay *Don't call it art*[4]. I will not repeat his arguments here.

One event that turned me towards programming and away from the continuing tradition of performance art and conceptual art was my encounter with Hans Ulrich Obrist's *Do It* website[5]. This is a compendium of artist's instructions, interviews and texts. However, the website does nothing to reflect on its own instructional foundations: these foundations are bequeathed to it by virtue of its computerisation. The website is a repository of scripts. Yet their performance occurs offline. I was uncomfortable walking above an abyss, as I saw it, once I noticed it was there. I had similar misgivings with a work by Keith Tyson (*Replicator* (http://adaweb.walkerart.org/influx/tyson/) where the web was called upon to convey instructions, but performance and documentation had to be conducted off line. I concluded that this gap contributed to the work's lack of usage ("there were 2 Replicators and there won't be anymore" the web page opines). This prompted me in part to make user involvement with my work achievable from the keyboard/screen.

That the possible convergence of computers, rules and conceptualism is 'in the air' might be suggested by the inclusion of RSG's (Radical Software Group) doctored game art in a recent show: *Logical Conclusions: 40 Years of Rule-Based Art*[6]. In other words, I believe that my research in this area is – at the very least – timely.

[4] Manovich's reply to Ars Electronica 2003's *Art and Code.*

[5] It one of several on the Internet: see for example, 'The Institute of Infinitely Small Things', *100 (11) Instruction Works*: http://www.ikatun.com/100-11/

[6] At Pace Wildenstein, New York City, 2005: http://www.pacewildenstein.com/Exhibitions/ViewExhibition.aspx?guid=a73e5d4d-f2d6-4cc2-9030-6e3b784c4ebd

(II) Electronic Literatures

My interest in accounting for the text *machine* I hope may explain what might seem a cavalier attitude to the considerable amount of work that has been done on hypertext fiction and digital poetics and other electronic text. (It is not hostility: it is a different emphasis). These latter have attracted a considerable amount of interest, including academic work.

I often have had to remind myself, I was researching the text machine, not a particular computer program, software package, genre of electronic writing or the Internet itself. Much of the web is in fact textual, at least on two levels. On one level, programs are written, and on another level, interaction is often by typing. I have focused on producing a theory of a text machine not on theorizing the Internet itself.

Some aspects – the Internet's poetics, its narrative genres – are dealt with by theoreticians of these genres (see Los Pequeño Glazier, 2002, for the former, Michael Joyce, 1995, the latter). In so far as I do consider these issues that are relevant to my issues but not my issues, I do so from a peculiar (a double headed word) perspective.

There are other forms of more specifically Internet literature of this type I do not deal with to any great extent, for instance, Muds (Multi User Dungeon) and Moos (Multi User Dungeon Object Oriented), AI (Artificial Intelligence) programs and chatterbots (programs that simulate conversation). These are all a form of writing – a program – and they are *written to* and they often write back. I direct the interested reader to literature on the subject, particularly Janet Murray (1997) and Espen Aarseth (1997). An interesting combination of the Oulipian and the Moo is to be found in Katherine Parrish (2005).

Whilst I consider the programming of text at length, and I employ a computer and I now work on the Internet, I am at the same time, when I write about these things, writing about something else also: this is the text machine. It is not, finally, completely identifiable with a particular form of technology outside

of itself, such as a computer, and still less with a particular body of writings. But my work and my theory have become more committed to working with computers and the Internet specifically over the course of time.

Let me insist: *I begin with instructions.*

It is worth restating my argument here, because although I will elaborate little now, I believe it will make much of what is to come a little clearer: I was particularly interested in the application of instructions to text. This (instructions for manipulating or generating, at any rate, inscribing a text) I characterised as a text machine. This requires a minimum description as "something that does something to text". My preference for machines that write (they also, in some senses, read) is in part because I am interested in text and in part because as Cramer (2001a) says:

> "In digital systems, literature is a privileged symbolic form for this very reason. We may automatically search a collection of text files for all occurrences of the word 'bird', but doing the same in a collection of image files or bird songs in a collection of audio files is incomparably tricky and error-prone, depending on either artificial intelligence algorithms or manual indexing…" (p. 1).

That is to say, I found the theorisation and construction of my machine to be the more viable because it dealt with text. (The machines I discuss largely fall into the categories I use in Chapter 6).

Over the years, there have been many valiant attempts to program literature. A good deal of this has now migrated to the Internet where a number of the programs are available (an interesting link page for some of these programs is at http://www.evolutionzone.com/kulturezone/c-g.writing/index_body.html. There is another at http://www.eskimo.com/~rstarr/poormfa/poemtool.html). Charles O. Hartman (1996) sketches some, but not all, of this tradition in *Virtual Muse.* (Much of this history is still to be written).

Another pre-Web initiative began in France in 1981 (see Harry Mathews and Alastair Brotchie's 1998 *Oulipo Compendium*) when the Oulipo (in English, 'workshop of potential literature') inaugurated their computer research group,

the ALAMO, followed by a group in the USA, and one in Italy, the TEANO. Paolo Ferrara (2003) provides descriptions of a number of the Italians' anagram, sonnet writing and other programs. The Oulipo requires a thesis to itself. Their more purely textual and algorithmic orientation sets my own investigation at some distance from theirs.

My research is not primarily historical (although I have researched this history and even contributed to it – see my *Computer Poetry's Neglected Debut* in the appendices): it was both theoretical and practice based

It must be acknowledged that the history of programmed text is to a large degree one of disappointments. I come back to this later in the thesis. But I have been aware throughout of how the computer has seldom written anything of what has been agreed to be of lasting merit, and the reasons for this are complex. What I wish to say now is, it is bad enough reinventing the wheel, but one should really not reinvent a square wheel – unless of course it is for the purposes of research.

(III) Computers (Software)[7]

This interest in developing theory motivated my use of the computer as a universal Turing machine. This, as I explain in Chapter 2, is a machine that simulates other machines. The computer would simulate my abstractly specified machine and this might tell me something about what otherwise might seem a rather ethereal entity.

This use of the computer, according to Lisa Jevbratt (2001b), is a rather antiquated one:

"Because of the traditions in which computer languages were developed, they are commonly thought of as symbolic logical abstractions of thoughts and

[7] There is a considerable literature about computer programming as an 'art'. This idea is often associated with the work of Donald Knuth (1975). I am not writing a thesis about programming. Elegance and economy of code and algorithm is not my subject.

natural languages, and computers as the universal machines manipulating these symbols. The praise for these special machines stems from their ability to simulate any other medium. However the scene has changed dramatically since the first code breaking machines and other early versions of computers. Every computer now exists in relation to a network, whether it is connected or not. Every software is potentially a networked software.“ (No page numbering).

I spend some considerable effort distinguishing my text machine from machines it might be mistaken for, not least the digital computer. In so doing I have given some space to portraying what I think a computer is. This material dominates Chapter 2 and I will not repeat it here.

But, as Jevbratt says, the scene has changed.

It is true, my work has changed also. Much of it now uses materials only really available on the Internet and uses these materials in ways that are only possible with a computer program. The work is accessible by a web connection by users, and databased by online organisations.

It is useful to remind myself I used the computer initially to investigate the rule-based constitution of textual procedures. If I could program them, so the argument went, then it was an instruction artwork. This, in essence, is a non-computer question, in so far as it is not specific to particular codes, operating systems or hardware. It is, if you will, an abstract machine simulated by a universal machine. It is the transposition of one typification of a machine to another.

This is where what may be seem an indifference to code lies – and a beneficent indifference to all that is constructed from code. I may never be a software artist, if by this code is to be my medium. Code is not a medium. Really, rules and instructions are a method and code is the intermediate language through which it may pass. Therefore, many of my interests are not code-specific. They should not be, in a thesis that is interested in procedures that can be written in numerous codes – or not in code at all.

I chose *Perl*, it is true, because of its adaptability as a text manipulation language. *Perl* is what is called a “high-level scripting language”. What this

means in practice is that instead of having to write out many lines of code, *Perl* may require just a line: it is very compressed. Writing in another language is a different experience. But I am not writing a thesis about programming; therefore, I do not take this discussion forward.

And there is reason, in any case: I could not say, with Alex Galloway (in an interview with Jevbratt, 2005a) that “my medium is Perl”, although I now write some *Perl*. This is because for me a particular code, the particular program it writes, is at the same time incidental to the machine. The fact that codes are interchangeable in most of what I have done underlines this. The indifference is not mine; the indifference is, I claim, the indifferent interchangeability of code. In other words there is a difference between the day-to-day practical matter of programming and the theoretical issues arising from it.

Here I depart from a narrow confinement to a techno-definitional approach to medium. I am not tied to a primarily physical definition of medium: neither the matter nor the morphological regularities of a system. Thus I am not primarily interested in mapping or configuring code structures as the “soil” of the Internet as is Jevbratt (soil, her word), and others: land or landscape artists wandering the web with sketchpads – or driving a software Smithson bulldozer.

Code for me, in its inter-changeability, has something contingent and non-essential about it as well as being practical and utilitarian in its individual codes and their uses. This puts all attempts to ‘make code a medium’ (matter) or a subject (topography) in doubt. It is neither modelling clay nor soil. (So when I program, I am aware that a task might be imagined without this or that computer; that, if a computer is used, it can be done in one of many languages and that even the algorithm I may use is replaceable by others). There is nothing specific to all this unless it is a lack of specifics.

Nor, therefore can I follow a discernible trend in some software theory and try to repeat the return of a self-reflexivity associated with modernism (see Chapter 5). Software theory will never convincingly replicate a passage in that history and expect to achieve a definition of its own *specificity* and *purity* (see,

Thierry de Duve, 1999, particularly his Chapter 3). This is assuming that such a repetition, should it be possible, is at all desirable, something I here openly repudiate.

However, in my practice and my theory I have been happy to come to an accommodation, in fact a fascination, with software, primarily because I have become involved with writing it. This engagement is reflected in the body of this thesis.

(IV) Research

Research (in terms of completed PhD theses) available to me is rather scant and not a little patchy. That it is scattered across several disciplines is not very surprising perhaps. Relevant areas include visual art, electronic literature, and computing – particularly where computing touches upon literature and art.

There is Maria Mencia's (2004) thesis about multimedia poetry, which investigates the transposition of visual poetry to computer. Scott Rettberg's (2002) thesis on hypertext fiction includes an interesting discussion that goes beyond his main subject. It also incorporates into the thesis some fiction text. I have followed this lead by incorporating generated text in this thesis.

Hisar Maruli Manurung's (2003) thesis concerns an ambitious attempt at programming a Natural Language Generation poetry generator: MCGONIGLE. This is a science not a literature thesis. Another PhD thesis that straddles disciplines with a strong science orientation is Paul Margerison's (1994) on algorithmic computer art.

Some other research has been published in book form, most notably Aarseth's (1997) text on cybertext from which I have benefited greatly. Also,

Alexander Galloway's *Protocol* (2004) is based on his PhD research on the connections between computing, codes and society.

In visual art there are several areas that seem comparatively well covered: the Internet as a site for art, for instance, through Josephine Berry's (2001) thesis, or Beryl Graham's (1997) thesis, and more recently Sarah Cook's (2004) thesis on the curation of new media art. Nick Lambert (2003) has written a D.Phil thesis on the history of computer art before the Internet that covers some of the issues in this thesis. But there is, despite continued fashionability, comparatively little real research at PhD level on new media art (something noted by Cook, 2004, for instance: "In the literature, it is repeatedly noted that there is a paucity of scholarship on the aesthetics of new media art", p.34).

I have searched, largely in vain, for research on the text machine, or some variant term. In its interests Steve Hodges's (2004) M. Sc. thesis came closest to my own concerns as it discusses the relationship of code, language and writing. That this thesis is an information science thesis is some indication of the distance I have traveled in my research[8]. Nandy Millan's (2001) thesis is another science thesis (computer science, in this case) that has some shared interests with my own. To explore connections between the computer and art, Millan includes a discussion of A.D.A.M., a poetry-generating applet written in *Java*. This use of a text machine – using writing to investigate issues in visual art – is certainly reminiscent of my own (even though I cannot share Millan's confidence that the "program illustrates the use of the computer as an originator of art, since the only role of the human artist in this particular case is the one of writing the computer program").[9]

[8] However, I should say here I feel the thesis suffers from a lack of clear distinction between its concepts, for instance between program code and algorithm, which as I explain are quite separate matters.

[9] Primarily because: "It consists of a number of text files containing a limited corpus of words which have previously been ordered and classified according to different syntactic categories: adverbs, prepositions, nouns, pronouns, adjectives, verbs, and so on. All these files are kept in a separate directory that is called once the applet is initialised." Its work falls into Bailey's (1974) category of "computer-assisted" poetry, where much has been prepared for the computer. As Bailey says, such works "reflect what its creator thinks a poem should look like" (p. 286). Here we touch upon a much wider debate about digital poetics.

There are a few other theses that are well worth reading. One is Bill Seaman's (1999) thesis about his multi-media work. Another, and not least of these, is a thesis by one of my research supervisors, Dr Tom Corby (2001). His work and example gave me the confidence to press on into an area that hitherto I knew little about and had the less expertise.

3. Conclusion

My research took me into areas I had not explored before. I began with an ambition to investigate rules and instructions. Early on I decided to confine myself to rules and instructions applied to text. A speculation about using a computer to explore this question turned into my full-scale research project as I learnt to write my own programs. The relationship of these two sorts of writing became one of the major themes of the thesis as I tried to answer what became the major question of my research.

…So the assistant
Points to the old cogwheels, the old handles
Set in machines…

Thom Gunn

Chapter 2: Text Machine

1. Introduction

The text machine has existed in a few instances as a sort of contrivance: as a device, something actually made. Still not very often, it has existed on paper alone (as for example in Swift's fictional machine, or Franz Kafka's famous execution-and-writing machine from *In the Penal Colony*). But the machine, as written specification, has been more frequently, on closer scrutiny, a writing *system*, as differentiated at the start of this thesis from a writing or text *machine*. Discussing two such 17th German occult systems, Cramer (2005) effectively differentiates between machine and system:

"Kuhlmann thought, just as Harsdörffer, of human language as something inherently computable. [It] therefore suffices as a potentiality and thought experiment on language and writing, and needs *[n]either an actual machine*, *nor its output* to make its point" (p. 62, my italics).[10]

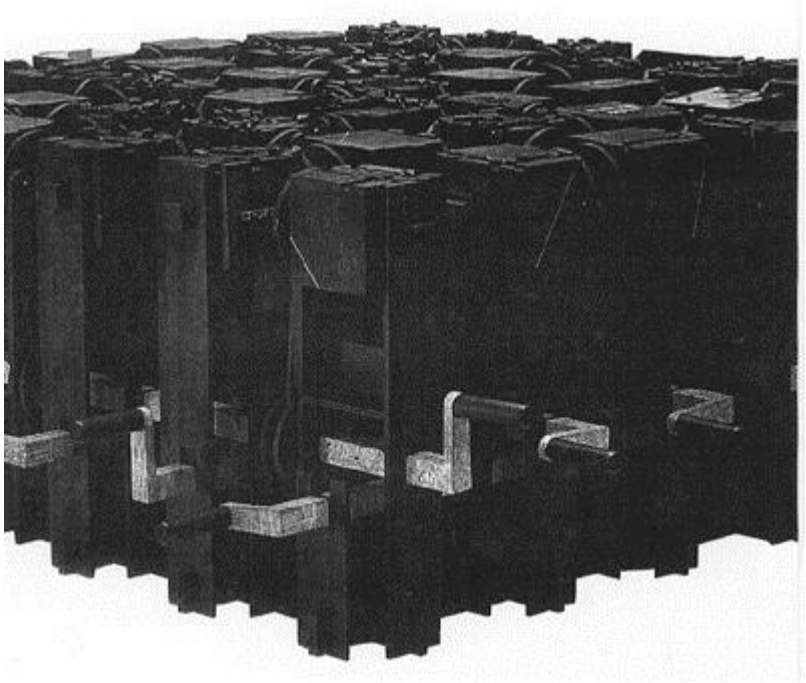
Only occasionally have these machines been built as a contraption, gadget, mechanical contrivance. One example is the concrete poet and Benedictine monk Dom Sylvester Houedard's poetry machine, a kind of coin machine comprising spinning barrels with words instead of icons. Houedard's

[10] The corrections in square brackets are corrections checked with the author.

machines were exhibited at the V&A Gallery in 1972: “I reached the ‘coin machines’ where poems could be constructed at the spin of a cylinder, at the push of a lever, at the bending of an arm…”[11]

There have been a few others; Daniel Libeskind’s (Plate 2) *Writing*, *Reading*, and *Memory Machines*[12]. These are described in his (1991) essay, *Three Lessons in Architecture*.

Plate 2



I made several (Plate 3) around the year 2000. They were machines that projected rotating text that could be read either upside down or back to front.

[11] Ana Hatherley (1972) *the art of letting things happen, a letter to sylvester houedard*’ (p. 41). Sadly, these machines may no longer exist. I have seen part of one in the home of the poet Bob Cobbing (one of Houedard’s publishers). It is a wrought iron frame, about 18 inches high.

[12] All three were destroyed in a fire: see Mathews and Brotchie, (1998) p. 177.

Plate 3

These join the few other machines artists have made occasionally. (Many of these are not machines that *write* at all but some other sort of machine: Marcel Duchamp's *Rotoscopes* or Brion Gysin's *Dreamachine*).

A number of writing procedures or systems have been invented. Some of these have been the work of poets and writers and some have been by scientists interested in generating texts. They are by no means all the same either in their details or in their context or use. I make reference to several of these text-making strategies in this thesis. Whether it is the Oulipo or Claude Shannon[13], however, these systems become automated and functional with computerisation. This observation echoes one by David Harel (1988) about algorithms and computing. Algorithms predate the computer by a good thousand years. However, computers, Harel notes, give a huge impetus to their use and creation. This is also so with text machines. The opportunity to program them means a growth in their development. This is something I

[13] See Chapter 5.

return to in Chapter 4. In the rest of this Chapter I shall try to define the text machine and to differentiate it from other machines, including the computer.

The subject of my research is rules and instructions in visual art. The method of investigation I have adopted is to theorise and construct an artwork I call a text machine. Text machines have rules and instructions. They are not solely rules and instructions, but the importance of one to the other will be shown.

It will become clear, not all texts imply the presence of a text machine. And certainly not all artworks are machines, text or otherwise. I am describing a particular sort of artwork and an unusual sort of machine.

I describe what a text machine is – and what it is not. I distinguish between three manifestations of the machine: these may be called the abstract, “paper and pencil”, machine; the “limited function machine”; and “the simulated”, imitated by another, machine.

These three are to be understood as the moments of any single text machine. A text machine may be defined abstractly; may be built as a limited function machine; or may be imitated by another machine, that is to say, a computer. Later, I will describe my own efforts to simulate text machines on computer.

A text machine comprises: its Rules and Instructions, its Codes, and its Inscriptions (and the “text-matter” from which these Inscriptions are formed). However, a text machine must be understood as a combination: the ensemble of its aspects. It is not reducible to one of these elements alone.

I now go on to describe a theory of the text machine.

2. Text Machine (Real Machine)

I am not the first to use the phrase ‘text – or writing – machine’. These terms, or similar ones, turn up quite often. For instance, Rettberg (2002): “Works of electronic literature should be understood as text machines functioning in network environments” (p. 5) or as the title of N. Katherine Hayles’s (2002) *Writing Machines*, or Italo Calvino’s (1997) *Literary Machines*. But a phrase is not a theory, and I have tried to work through the implications of what a text machine might be.

The conception of the machine as a procedure, in some sense, was there from early on. Thus, in describing his cut-up technique in *Minutes to Go* (in, Beiles et al, 1960 p. 5), Gysin wrote:

> the writing machine is for everybody
> do it yourself until the machine comes
> here is the system according to us

How such machines, not only Gysin’s, might exist I will now go on to discuss. For it to be a text machine, in the sense I mean it, I propose it must be possible for the machine to exist in any of three states: abstract, limited function, simulated. It is not required for the machine to be in all of these states simultaneously. In fact, if it can be described in its abstract state, then it is possible for it to exist in the other two. For this reason, the abstract text machine has a kind of priority over the physical and the simulated.

For the practice part of my submission, I have focussed on the construction of simulated machines on computer. These simulations are, in part, a way to test my ideas, although the difficulties of verification, or indeed, what verification in this context may mean, have also not escaped me.

I might be reminded a text machine is not the only machine in the history of art it is possible to identify or imagine. This, as noted above, is quite true. For now, my claim is that a text machine be taken as a model of any comparable machine.

Belinda Barnet (2004) develops a theory of the evolutionary patterns of "technical machines". Trying to accommodate both an understanding of that machine's duration over time, and recognition of the changing temporal qualities of the machine, she concludes the machine "can be identified by a group of procedures or processes that remain stable throughout the evolutional lineage" (Barnet, no page numbering). Therefore for her, as I have tried to suggest above in the case of the text machine: "The technical object is not concrete"; or perhaps we should say, its concretion is incidental to its definition and temporal persistence. To take one example she uses (the computer), the modern digital machine and the 1930's to early 1940's analogue machine "seem completely unrelated". But they are (by Alan Turing's, see below, definition) the same machine.

Such observations may seem to leave me vulnerable to an accusation of functionalism, such as Hayles (2001) levels at 'cybertext' theory: "Like all functionalist theories, cybertext theory elides materiality in order to create a template based on function, generally casting a blind eye to how these functions are instantiated in particular media (no page numbering)." My desire is to redress, if possible, the balance in the analysis of each part of my compound noun: both the text (which has had the greater attention) *and* the machine. But a consideration of function is unavoidable if the machine is to be considered in any depth. It is difficult to conceive of a machine of any sort with no consideration of its function. Hayles, herself, does not present a developed theory of what such a machine might be in her (2002) *Writing Machines*. But this is my ambition: a developed theory of the text machine in this thesis. To contemplate function does not make one a *functionalist.* That is a particular (ideological) orientation to function. Nor does it mean that we necessarily ignore the functioning machine's instantiation in particular media, as will be seen below.

3. Machines, Discrete and Universal

The idea that a machine can be imitated by another machine appears in Turing's papers detailing the formalised working of a computer. A Turing Machine is equal to an abstract description of its functioning. It is essentially a table of rules such as could be carried out by a physical machine, or also perhaps a human, described as a sort of ideal clerk supplied with paper and pencil. [14]

(There are many accounts of Turing's and Alonzo Church's modelling of computer function. See W. Daniel Hillis's account in *The Pattern on the Stone* for a largely non-technical introduction).

There is in Turing a distinction between "discrete-state machines" that perform a single function and "Universal Machines" that can perform the functions of any discrete machine. A Universal Turing Machine is a model of how a digital computer works in abstract form (without it being a technical specification of the actual hardware and software). A 'discrete machine', however, only performs limited tasks.

I will not repeat Turing's detailed description of these machines that involve a supply of paper, a system of notation and the ability to write a symbol, or not, or to erase it. Turing describes how a discrete machine might operate. He also shows how a Universal Machine, provided with the table of rules of a discrete machine, might perform the functions of that machine.

It may be apparent that my demand that a text machine may be described abstractly, made as a limited functioning machine, or simulated (by a computer), makes it seemingly equivalent to a (discrete) Turing machine. So where, if anywhere, does it differ? To answer this question I must go into some detail about what constitutes a Turing Machine as regards other seemingly similar machines. I will do this by way of a discussion of two

[14] This why Kripke (1982) says, "'Machine' often seems to mean a program" (p. 33) (although it is not necessarily specifically a *computer* program he cautions).

typologies of the machine as they appear in writings by Kenneth L. Ketner (1988) and by Nick Montfort (2004). If we are to say why a text machine is not a Turing machine, we need to know what a Turing machine is.

4. Peirce's Theorematic Reasoner and Chomsky's Finite Automaton

Ketner contrasts Turing machines with C. S. Peirce's interest in logical machines. On Ketner's account (see also Peirce 1991), Peirce was little interested in machines that repeated a predefined process, although he was not against their use. However, for him, these machines represented a relatively uninteresting reduplication of reasoning processes already established.

According to Ketner, Peirce's interest was in "theorematic machines" (Ketner suggests the term *Peirce machines*) rather than those that follow "deterministic algorithms" (p. 50). This is part of a larger distinction between deterministic and theorematic reasoning (Peirce's phrase, ibid. p. 49) in mathematical method in which Peirce stressed the "hypothetical, experimental, observational, and creative" (ibid. p. 50).

Clearly, I am not concerned here with mathematical matters. However, Ketner notes the difference between the wider category of "numerous instances of nondeterministic (sic) machines", and a lesser category, of which would include: "a nondeterministic machine that could accomplish the theorematic method" (pp. 50-51). If the text machine is not a Peirce/theorematic machine, could it be, nevertheless, one of the group of non-deterministic machines Ketner mentions? A non-deterministic machine, for Ketner, might be a "device, or recipe, that emulates a roll of a dice using at one stage of its operation some nondeterministic element, perhaps a random number generator or a cosmic ray detector" (p. 51).

Before I say if this is so, there is a difficulty with Ketner in that he conflates the Turing machine with a deterministic machine as such. He writes, “a Turing machine, then, is a definition of such a deterministic method” (p. 55). But in fact, Turing (1950) airs the idea of a “digital computer with a random element” in *Computing Machinery and Intelligence* (p. 5). Turing proposes “instructions involving the throwing of a die or some equivalent electronic process” (ibid.). Turing notes that we cannot tell by observing if the machine has a random element. Therefore, a non-deterministic machine is imaginable (or what passes a sort of Turing ‘non-determinism test’). The conclusion must be Turing machines are not deterministic by definition.

Having returned from this excursion, it is possible to ask if a text machine is deterministic or not. The ‘Kozlowski machine’ (*Noumena* at http://www.in-vacua.com/noumena.html, a program that processes web pages) is deterministic in the sense it does not use a random element, as Ketner and Turing describe. I have made machines that do, but this is not among them. In the case of *Noumena* for instance, it might be argued that we do not know what a user may input in the form of a web address to process. But by the same argument we do not know what might be passed to a (deterministic – *some* are) Turing machine to calculate. The conclusion must be, a text machine may be either a non-deterministic machine or deterministic in the sense I have used. I will argue below that, whilst there are resemblances between a text machine and a discrete-state Turing Machine, the two are not identical, but this is not because of the issue of determinism.

There is a recent attempt by Montfort (2004) to contrast two machines he calls “cybertext” and “hypertext”. Montfort uses the ‘Chomsky Hierarchy’ to draw a distinction between these machines. Montfort uses only two of the Chomsky Hierarchy’s four types. One is what Chomsky calls “finite automata” and the other is the Turing machine. “The paradigm of the hypertext is the least powerful computational machine, the finite automaton. The prototypical cybertext is of the fourth and most powerful computational class, a Turing machine” (no page numbering), according to Montfort.

We need not go into the Chomsky Hierarchy here. Nor do we need to worry about Montfort's low opinion of hypertext (based as it is on that literature's lack of computational strength). The important question for my argument is, can Montfort's "cybertext" be a Turing Machine? If so, Montfort will have moved our ideas on in this area. Montfort says a Turing Machine "can run Quake III, display GRAMMATRON, or beat Garry Kasparov in chess". However, the two examples he gives of cybertext, *Eliza* (see Weizenbaum 1978, for Eliza's simulation of a non-directive therapist) and *Racter* (to which the authorship of the fiction work *The Policeman's Beard is Half Constructed*, is attributed)[15], obviously cannot do any of this. They are both computer programs and they produce texts. They run on a Turing Machine (that is, a computer), they are not themselves Turing machines. They might be examples of *discrete-state machines,* with the qualifications I make immediately below, and they seem to qualify as text machines. But this is not something Montfort says, and he makes no distinction at all between universal and discrete-state Turing Machines. Montfort's categorisation, therefore, is not a viable tool for understanding "cybertexts" or, indeed, machines of any sort.

5. Text machine – Turing Machine?

If a computer may model a text machine, is it not a simplifying matter to say that a text machine *is* a computer? This way we have the advantage (applying Occam's razor) of throwing away a complicating part of the explanation. A Turing machine is the theoretical model of any computer. If so, is not the Turing machine the theoretical model of the text machine as a sort of 'text computer' also? Libeskind (1991) makes just this adventure when he says of his writing machine of cogs and pulleys, "it's a little computer I built" (p. 45). However, I hesitate to follow him.

[15] Racter is a computer program. *The Policeman's Beard is Half Constructed* (1984) is a collection of poetry and prose attributed to Racter and sometimes credited as the computer's first book.

A Turing machine must have several things a text machine does not require: a Turing machine has a simple and unambiguous notation. It also possesses clear rules and instructions to follow. A text machine does not necessarily have these. Text machines may exist without conversion to code and program. What happens when the computer simulates them is that one must decide on a definite, or several definite, interpretations of the machine. For instance, if we wish, although it is not really a *text* machine, to *program* Young's *Composition 1960 #10, to Bob Morris*[16]*,* we have to decide what the instruction *is*, decide the meaning of its terms. I return to this in the next chapter where it is proposed this is by no means straightforward. For the moment, I wish to emphasise the difference between an instruction in human language for a human to interpret and carry out and one, for computer, where the possibilities are more literally spelt out. In short, a text machine may allow more ambiguity than may a Turing machine.

6. Between a Turing and an Abstract Machine?

A text machine cannot be confused with Deleuze and Guattari's (2003) "Abstract Machine". I will state the difference before I go on to note any similarities: *a Deleuze and Guattari machine cannot be made*.

By this I mean that it is not possible to make a 'Bach machine' or a 'Beethoven machine' (two of several machines mentioned in *A Thousand Plateaus*) either as a physical machine or as a simulated machine. Of course, the scores to their music may still be played, but theirs are 'machines' that have ceased to function: there is no new forming of unformed matters, in Deleuze and Guattari's terms; there is only faux Bach, ersatz Beethoven.

A Deleuze and Guattari Abstract Machine cannot be built principally because its rules are mentioned but not specified. One cannot really imagine

[16] "Draw a straight line and follow it". In Sohm, H. (1970). No page numbering.

replicating a Beethoven or a Bach machine with paper and pencil and a table of prescribed actions. This is because a Bach or a Beethoven never was a machine in the sense I have described above: their 'machine' is less reducible to a simplified procedure than is a text machine. This is because a text machine may be thought of as constantly tending toward the finitude of the algorithm without being constituted as such.

A Deleuze and Guattari Abstract Machine is said to have "rules" (p. 70) and "is not random" (p. 71) – but the rules are not stated. Their machines form unformed matters, and my coinage of text materials owes something to their usage. However, their Abstract Machines extend well beyond my own area of investigation, to "overcode" language, the body, the earth and more.

The text machine, if it can be made, must also have the possibility that it may be written. In effect, to be in to be in an abstract form, it *must* be written; if it is to be simulated it must be written so a machine can understand it, as code. The text machine requires a degree of specificity not provided by Deleuze and Guattari, but not so much as a Turing machine, as I have outlined. It may be permissible to situate most text machines, if only figuratively, somewhere between a Turing and an Abstract Machine.

A text machine does not only write, it is also written, or it allows of the possibility it may be written. This one fact marks the difference between my understanding of the text machine and the Abstract Machine of Deleuze and Guattari. It is also the source of an ambiguity at the base of the machine's being: is the machine art, or does it make art, or is it both? Can this distinction be fixed anyway?

(There are many different machines-of-the-text, if I may be allowed this construction. The *Oulipo Compendium,* Mathews and Brotchie, 1998, under the entry on "*Machines for writing*", quite properly remarks that all Oulipo strategies are in a sense writing machines. However, there is an ambiguity as to whether the techniques that are devised are as intriguing as the writings they produce – or more so? But it is too simple to say that a text machine is

the artwork rather than its writings. The distinction is not clear. I do not say that “the difference between machine and output is not clear” – this would be to contradict all of the foregoing – rather the confusion is how to value each. Text machines write. As I have said, they also are written, or more correctly, *may be written*. How we evaluate this status of writing *and* being written, and the relative merits and interest of the two, I shall return to in the fourth Chapter).

7. Several Machines of Conceptualism

Sol LeWitt’s well known formulation: “The idea becomes a machine that makes the art” seems to situate the machine as both antecedent and other to the art. This is a division, a precession, Alexander Alberro observes in LeWitt. Contrasting him with Lawrence Weiner (who *did not*), Alberro (1999) writes, LeWitt, “maintained the work should still take on a physical form” (p. xxiii).[17]

Alberro contrasts LeWitt’s with Weiner’s well-known position of leaving the decision of whether to give the work a physical form up to the “receiver”. Alberro’s reading of Weiner appears to characterise Weiner’s instructions as something that might – or might not – make art, but are not themselves art; they are connected with the artwork but are distinct from it. However, Weiner’s insistence that, “The piece need not be built”, I believe, allows us to interpret the instruction, in its (abstract) statement of the “piece", as “The Work”, as

[17] In Keith Tyson’s work too, I note, a similar distinction persists and is part of Tyson’s continued debt to Conceptualism. Speaking of his “Art Machine” he says, “It's like a Sol LeWitt mechanism. But it isn't just intellectual. I have it all written down on paper. It's a proper flow chart” (interview, Dave Beech, 2002. No page numbering). (Elsewhere – Saul Albert 2002, for instance – it is suggested it is a computer program, in *Prolog; Prolog* is a logic manipulation language). Whatever the Art Machine is, or is not, it is inaccessible to us: we know it principally by its products (and Tyson’s contradictory remarks). Crucially, the two remain different *as of kind*. The Art Machine is not itself present as work. *Nor can it be inferred*.

much as any physical fabrication. Alberro's (Ibid., p. xxiii) claim that in Weiner, it is "the eclipse of the authorial figure of the artist" that is achieved, is a misreading on this account, or at least is a partial reading; it is the eclipse of visual art as *object-only* that results. The work is both object *and/or* the abstract statement of its conditions. One further step is possible to imagine: the question of the work's simulation. I return to this in a moment.

An example: Weiner's instruction: "*One Hole in the Ground Approximately 1' x 1' x 1'. One Gallon Water Based White Paint Poured into this Hole*". This work can be instantiated physically (or not, according to the Weiner credo): there is the written instruction, and there is the physical instance. And there is nothing to say we might not, if we had not known it first, have worked back from the instance to the instruction: started from the instance and produced from it its text (more of this in the next Chapter).

But could we work from the instruction of this work by Weiner to its simulation? I am referring to the same sort of simulations as those I speak of above, where the machine and its activity are simulated by computer.

There is obviously a difference here: Weiner's instruction and the material it addresses constitute two *different* media; they exist in two different realms: text, a symbolic medium, and paint, the ground, physical substances.

However, computer instructions (a program), and the text-materials (data) of a simulated text machine, *can be held in the same medium*. As Cramer (2003, p. 101) notes, the previously assumed "clear cut-division, a material difference between the tool and the writing, the processor and the processed, no longer exists in software since computers adopted the Von Neumann architecture of storing instructions in the same symbolic realm".

This "Von Neumann architecture" constituted a revolution in computer theory:

"Every tradition of common sense and clear thinking would tend to suggest that 'numbers' were entirely different from 'instructions'. The obvious thing was to keep them apart: the data in one place, and the stock of instructions to

operate *on* the data in another place. It was obvious – but wrong." (Hodges, A., 1983, p. 302).

(John Von Neumann's paper containing these ideas is dated 1945). So it is for text as it is for numbers. The data and the instructions can be kept in the same 'place'.

However, there is still as yet no way to convert, to take another Weiner example, plaster and lathing to binary code. But the unification of instruction and material permits *full* simulation, instruction and materials, in the case of the simulated text machine, but not for the Weiner. So long as instructions, which can be written as code, are to be executed on other symbolic matter (such as text, but not of course, exclusively text), the machine thus constituted may be fully simulated with a computer.

Therefore, simulation of the Weiner is not possible in the sense that I have developed it. A computer animation of the work would not simulate the 'Weiner Machine': it would merely represent it. Here lies one reason for selecting symbolic media, text, to work with.

The development of a theory of a simulated machine was given impetus by my interest in a work by the Polish artist Jarowslaw Kozlowski, *Reality*[18]. *Reality* is a 1972 bookwork. It comprises a section of Kant's *Critique of Pure Reason* with the text removed, leaving the punctuation. The effect of this is to draw attention to the sentence structure over the sense of the text. My wish was to construe from Kozlowski's bookwork its instructions and set them in action, my preferred method being to make a computer simulation. This, in effect, was to construct a simulated 'Kozlowski Machine' (http://www.in-vacua.com/noumena.html): a machine that deleted text and kept the punctuation of web pages. The method I adopted to do this might be conceived of as a form of "reverse-engineering". This might seem a strange application of the term, although it is not so uncommon where computer

[18] There is further analysis of this artwork in Chapter 6.

scientists[19] become involved in working out how a text might have been generated[20]. (I have placed in the Appendix to this Chapter my reasoning concerning *Noumena's* reverse-engineering of *Reality*).

If it is accepted that we are talking about the same changing machine over time, it may still be objected that artworks are more than an abstracted procedure, that when simulated, the Kozlowski loses specificity. The choice of texts treated *is* important to the interpretation of the original work. With the Kozlowski, it is a section of Immanuel Kant's *Critique of Pure Reason* that is deleted (there is a case for arguing that the section is the one where Kant discusses the noumenon, the reality beneath phenomena, but I have not been able to confirm this[21]).

With the simulation of the text machines certain specifics *are* lost – but others are gained; new materials are treated. Text machines pull text-materials into them and form them anew. The machine itself may be described, but so may its inputs and its outputs; the machine requires only something to work on for its functioning and in so doing it subjects new textual resources to its process.

[19] Here is a programmer talking about how to reverse-engineer a text. Schwartz (1999): "I typed random sentence into www.google.com looking for some grammars. The most interesting hit I got led me to the Dilbert Mission Generator, located at http://www.dilbert.com/comics/dilbert/career/bin/ms2.cgi. I spent about an hour hitting reload repeatedly to reverse-engineer the output... I've cleaned up some of the choices, and fixed a few misspellings, so this grammar isn't quite what you see there." (The web address is wrong. Try http://www.dilbert.com/comics/dilbert/games/career/bin/ms_noun.cgi instead).

[20] More recently I found this passage about Quirinus Kuhlmann in Cramer (2005, p47): "Through this intertextuality, the poem renders itself a Solomonic machine. It is a computational reverse engineering of Solomon's wisdom, considering the proverbs as they are written in the Bible the fragmentary output of an occult machine."

[21] Book II, Chapter III: "THE GROUND OF THE DISTINCTION OF ALL OBJECTS INTO PHENOMENA AND NOUMENA". The noumenon is Kant's inaccessible "thing in itself". The phenomenon is the object of experience. Kozlowski, I believe chose this passage with care to draw attention to the syntactic structure of language as marked out by the punctuation.

8. Loosely Related

I should at this point make it clear that I am not using terms such as “abstract machine”, or another (that I have not used), “virtual machine” in the several senses that computer scientists use them. But the distinction may appear subtle, and although I do not wish to become involved in these more technical discussions, I must touch upon the matter now. It may seem that my comments on the text machine more or less parallel discourses in computer science. I mention this possibility now so as to ensure there is awareness of the issue, but also because comparisons may prove productive to my own theme.

I have indicated some of the differences between my use of the term *text machine* and other machines as they appear in computing science and the arts. “Abstract machine” in computer science in its “generic meaning is a behavioral model of a computer”[22] is, as I have tried to establish in the preceding chapter, not identical with my term.

A “virtual machine” in the sense of the “creation of a number of different identical execution environments on a single computer”[23] is still more a technical specification and further from my area of interest. However, “virtual” and “abstract” are sometimes used interchangably by computer scientists to refer to higher level programming structures that are effected ultimately at the (lower) level of the physical states of the computer. It is this issue I am interested in here.

This approach to the abstract (or virtual computer) has relevance to my discussion. High-level structures (such as lists, arrays and other programming constructs) are abstractions seen as having a low-level machine implementation. For Aaron Sloman (2002) these are virtual processes or mechanisms that “really exist” and have “causal consequences” (p.188). He

[22] http://en.wikipedia.org/wiki/Abstract_machine

[23] http://en.wikipedia.org/wiki/Virtual_machine

resists, therefore, a reductionism that might grant existence to mechanical features alone, such as voltage, wiring and the rest. Why indeed stop there and not recognise only atomic and sub atomic levels? In the end this may become, if it is not already, as Sloman suggests, a metaphysical question, one I, like he, will not pursue.

High-level abstractions may have implementation in various ways. So, for instance, Chris Fields (2002) differentiates between three levels in ascending order: (i) processes of the system's hardware that are the implementation of algorithms and data structures (ii) the algorithms and data structures themselves (iii) the computations realised by the algorithms executed (p.166).

These levels are, according to Fields (who in turn is following Marr), "loosely related". He writes:

"This argument is based on the observation that a given computation may be realized by many different algorithms, that a given algorithm may be implemented by many different physical processes, that input-output experiments cannot distinguish between different algorithms or implementations…" (op. cit.).

These arguments are applied to the composition and functioning of the computer alone in Fields's work. They do not extend further, to embrace physical processes and "input-output" that does not necessarily involve a computer. But there is no reason why we might not make such an extension to the text machine. We could substitute, for instance, the "writings of the text machine" for "computations"; "instructions" for "algorithms"; and "pencil and paper machine" or "nuts and bolts machine" for the computer's hardware.

For the text machine in *each* of its instances, at higher levels there are instructions and a language in which they may be expressed, and below that a physical process. They produce texts of a given sort or sorts (replacing "computations"). But the machine is not identical with the language of expression or the physical process of its instantiation. We may move across processes and languages, transposing languages as we go.

It is important to recall that a computer may be either a discrete-state or a universal machine. A discrete state machine (computer) performs distinct actions according to its rules and instructions. But discrete-state machines may not be computers at all (Turing[24] gives the example of a lighting system). It is also true that a computer need not be electrical (Turing's example is of Charles Babbage's Analytical Engine[25]). I do not use "discrete-state machine", preferring the less historically weighted "limited function machine".

The distinction, therefore, is not between computers and the rest, nor between electrical machines and the rest, but between machines that perform one or several functions and a machine that can "mimic", to use Turing's word, all the others: it is the "special property of digital computers, that they can mimic any discrete-state machine" (ibid. p. 7). It is because of this property that many discrete-state machines are not constructed at all. There is no need to make a 'nuts and bolts' machine like Babbage's, nor is there a need to fabricate a discrete-state electrical machine (although for archaeological reasons sometimes such machines have been constructed). Why do this when universal machines are commonplace?

These distinctions become important when we discuss text machines. The construction of a machine consisting of wheels and gears is certainly possible. It is possible to identify a machine, an electrical digital machine, which only performs various text operations and is thus not a conventional desktop computer (in other words, a computer that could not be reprogrammed: a kind of pocket text calculator[26]). The existence of a universal machine (the desktop computer) limits their use and availability. The construction of a machine of levers and connectors is more likely to be for historical or perhaps purely

[24] Op. cit. p. 6.

[25] "Since Babbage's machine was not electrical, and since all digital computers are in a sense equivalent, we see that this use of electricity cannot be of theoretical importance", ibid. pp. 5-6.

[26] An instance might be the 'Pocket Crossword Solver' made by the Lexibook Company. The one I own is from 2002. It performs several functions, including an anagram search.

theoretical reasons. Hillis's 'Tinker Toy Computer', a 'tic-tac-toe' player made from a children's construction set is one such (see, Hillis op. cit. pp. 16-18).

9. Conclusion

In this chapter I have described a text machine artwork. I have proposed that the machine can be described in the abstract, made physically, or be simulated. I have distinguished my concept of the machine from some other relevant ideas. I explained how simulation of the machine could be achieved because of qualities inherent in the computer as a Universal Machine. This favours, but not exclusively, the use of text, a material that may be stored and processed within the architecture of the computer as presently constituted.

We have seen the importance of rules and instructions. In the next chapter I explore this question in more depth. I also return to the problems of deriving an instruction and following an instruction, and the peculiar status of rules in art.

This moment seems to constitute a central point in my thesis so far and one that much of the preceding was tending to all along. The foregoing statements, I wish to suggest, represent an advance in a theory of a text machine, one that has relevance also to the theorisation of other "writing" and "art" machines that may nearly or exactly coincide with it.

In the next Chapter I shall test this contention by posing what I think are some substantial objections to rules and instructions and their use.

Appendix to Chapter 2

In this Appendix I develop a discussion of *how* it might be possible to reverse-engineer an artwork and some of the problems this poses. The ideas presented here will be taken up at points later in the thesis, not least in Chapter 3 (the difficulties of following a rule), Chapter 4 (particularly section 2, "*Reverse-Engineering a Text* ") and Chapter 5 (concerning Finnemann's recent consideration of flexibility in rule generation).

I begin with three propositions:

1. **Instructions are *producible*.**

2. Instructions are *expandable*.

3. Instruction-art has *structural identity*.

I will now go on to explain what I mean by these remarks. Will do so by reference to the relation of *Noumena* (at http://www.in-vacua.com/noumena.html) to *Reality* by Kozlowski.

1.*'Instructions are producible'*. I do not need a written instruction from Kozlowski's hand (to my knowledge there is none, nor have I seen one) to turn it into software. An instruction can be written after the fact. If it can be done, then that work was instruction-art. An instruction is construable, in a similar way that a grammatical rule is construable, from practice: a speaker does not necessarily need to know the rule to follow it (though they may). If it is there, however, it may be abstracted and consciously adopted.

2. '*Instructions are expandable'*. The instruction abstracted from the Kozlowski is restricted in scope to one text: "delete all the text from a passage from Kant's *Critique* leaving the punctuation". It is expanded in *Noumena* to "remove the characters from any text that may be displayed on computer, leaving punctuation". That instructions are expandable is important to my argument and wide ranging in effect. I return to it immediately below.

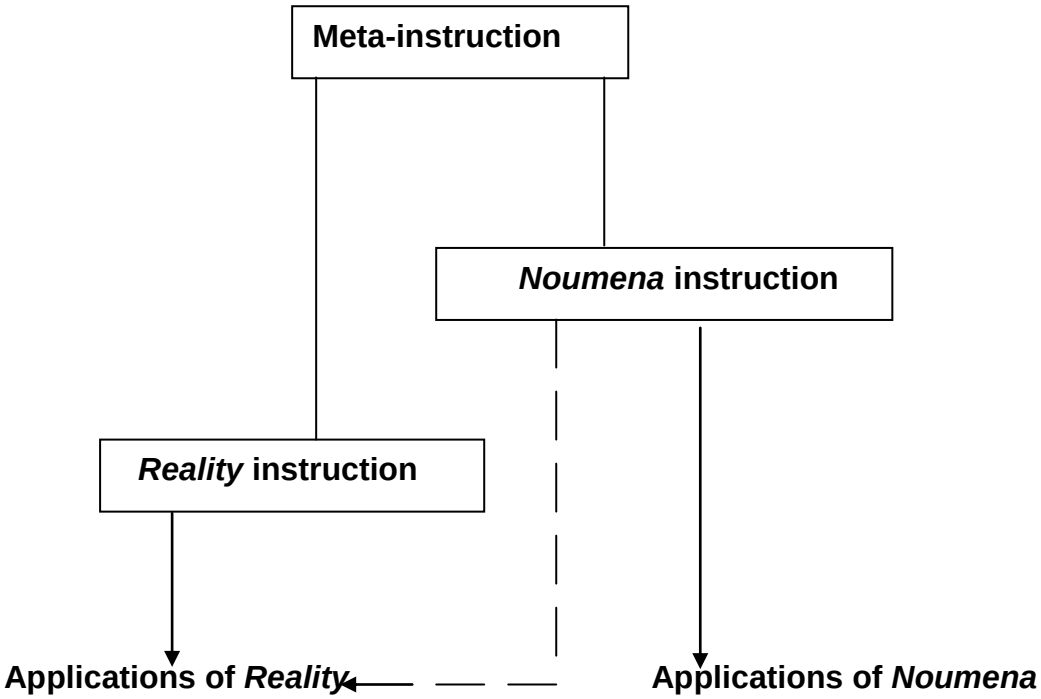
3. '*Instruction-art has structural identity'*. This has two aspects. One relates to the instruction, the other to its application.

Noumena does not in fact remove all text. Occasionally some is left (for example on the 'submit' buttons on a web page). This could be corrected, but it is not important because greater structural identity takes precedence over subordinate detail.

Secondly, that '*Instructions are expandable'* (as above) means that the instruction itself may undergo change, so long as this is not beyond recognition. In changing the instruction's scope I have necessarily altered some of its qualities. But not completely, it is the Kozlowski instruction transposed to different media. Instruction-art should thus be seen as a system of rule development, not of passive rule following. We are used to the idea of variability of the *performances* of this script or that score. The idea that the instruction itself may undergo dynamic development is less familiar.

Fig 1 is intended to represent these arguments schematically.

Fig 1



The meta-instruction ("remove the characters from **any** text, leaving only the punctuation") encompasses both subordinate instructions. *Noumena's* is not the meta-instruction because it is limited to texts that may be displayed on computer, and that is not all texts.

The instructions are placed in descending order. This represents their relative generality. In practice many gradations are possible, and thus many more instructions.

There is a broken line from *Noumena's* instruction to the applications of *Reality.* This is because it is possible to use its software to treat a section of Kant, so potentially at least, some applications of *Reality's* instruction might count as *Noumena's.*

9. Conclusion

It is my argument that the theory proposed above accounts for the particular case of '*Noumena-Reality'.* It establishes in what sense it is possible for one to be a software version of the other. However, the theory might, and I believe should, be applied to any similar relationship between works, and regardless of media. As such, I wish to make the claim that this theory effectively answers a central problem posed by my research. To summarise:

I have proposed that a rule may be derived; it does not need to be given.

I have also proposed that a rule may be increased in scope and transposed for use in different media, and I have indicated why and in what ways this is possible.

This suggests how an artwork might be *reverse-engineered*. I return to the concept of reverse-engineering. When I do, I will give the concept itself further explanation.

Inside the computers themselves everything becomes a number: quantity without image, sound, or voice. And once optical fiber (sic) networks turn formerly distinct data flows into a standardized series of digital numbers, any medium can be translated into any other. With numbers anything goes. Modulation, transformation, synchronization; delay, storage, transposition; scrambling, scanning, mapping – a total media link on a digital base will erase the very concept of medium. Instead of wiring people and technologies, absolute knowledge will run as an endless loop.

Kittler (1999)

Chapter 3: Instructions Rule

1. Introduction

In the last Chapter I began to establish what the text machine was. In this Chapter I continue to elaborate on this discussion.

I have said that the text machine has rules, codes, and inscriptions. In this Chapter I discuss the text machine's rules. I will talk about the problems that rules and rule following may create for my theory. I will then go on to show the kind of rules we might be thinking of in a discussion of a text machine.

But before this, in the initial sections of the Chapter, I intend to draw in a larger debate, as the effects of digitisation call into question the role of distinct media in cultural production. I attempt to extend the scope of this discussion to engage with an idea of what I call the post-mechanical.

2. “Post-Medium”

There is a debate played out about the status of media when they are converted to digital code. This discussion takes place in the writings of Rosalind Krauss (1999), Manovich (1999, 2001), Kittler (1999) and Mark Hansen (2004), amongst others. It is part of a wider debate about the consequences of digital conversion not only of text, sound, film and photography and other visual media, but also of human identity, as retold in Hayles’s (1999b) account of the posthuman, but associated with Moravec (see also Hayles 1999a) who controversially suggested downloading a human consciousness to disk. Yet, whilst what it is to be human and cultural forms are called into question, the machine itself customarily remains hidden, as it were (to adapt a figure from Marx), behind the backs of the cultural producers.

The theorists I have mentioned, in their different ways, give consideration to the effects of the digitisation of sound and visual media, such as photography and the cinema. However, these discussions focus primarily on the image, on the photograph and the film, not on the machine. In my writings I will not be dealing in any detail with these previously distinct media, or with their machines. Rather, I will consider only the text machine and how it too might be made into a signal, be transmitted and reconstructed. That this may be possible is due to what the machine is, its mode of existence. It will be seen that digitisation dramatises this issue, but for me, does not create its conditions.

With digitisation, at the core, there is a realisation that what were formerly different media, be they film or text, photograph or sound, are no longer distinct in their storage and transmission conditions: all, at bottom, are binary digits. But there are, if you will, *two* levels[27]. Hansen (2004) stresses the “human perceptual ratios” (p. 1), where data is experienced as differing phenomena, be it an image, or whatever. But there is also the circumstance,

[27] In fact this is a rather basic division. According to Richard Feynman, *Lectures on Computation*, there are thirteen levels to an operating system. See Matthew Fuller (2003) p. 21.

as Kittler (1999) puts it that: ”Inside the computers themselves everything becomes a number” (p.1) – although it is more correct to say “a value”[28]. Of course, let us be reminded, *we* are *not* inside the computer, or at least not yet. As Kittler realises, we do not experience an undifferentiated data stream and it is this that leads him to observe, despite their informational basis, “there are still media” (p. 2). That is to say, we continue to distinguish between media, in a way that the computer need not in its mechanistic indifference. However, the tension between our perceptual experience and the digital, undifferentiated ground of being of the media we experience persists.

I wish to go further, rather than turn, as does Hansen, I feel too soon,[29] to the phenomenological in pursuit of understanding new media. I wish to focus not merely on new, grounding continuities between formerly separate media, but also those between machines. Secondly, I want to assert that this is a continuity that is prior to digitisation.

3. Post-Mechanical

My wish is to extend the debate about media to the machine, a machine viewed as something that may be converted to signal and transmitted. The “abstract body” must be provided with an actual body, nuts and bolts, pencil and paper, or the hardware of a computer, if it is to function. However, for me the machine is not identical with any of its actual examples. As will be seen below, this is not a result of computer use, but in fact precedes a particular technology.

[28] This because 0s and 1s are a convention for representing what are in fact switches: a series of on and offs. They are not really numbers.

[29] I disagree with Hansen precisely in this, not that “[n]o matter how “black-boxed” an image technology…may seem, there will always have been embodied perception as/at its origin” (p. 9), but that there is no way to gain access to this origin, nor that if we could, would we be greatly advantaged in our understanding by it. Rather than to turn away, to an originary myth, I will try to look harder at the box.

A text machine may be thought of as consisting, substantially but not entirely, of rules such as *may* be passed to a computer to execute as an instruction. A text machine, it will be remembered, does not have to be converted to computer and program, but it should in principal be possible.

It will be noted, I have just introduced a distinction between “rule” and “instruction”. Implicit in this is a distinction between a rule that specifies what *shall be done,* and instruction, something that *can be followed*: the instruction gives detail to the rule[30]. This will become important as I continue, when it will be seen that the same rule requires appropriately differing instructions if it is to be executed, for instance, by a human rather than a computer. But it may be useful to think of an example used in Chapter 2, where a random number function or some other computational device simulated random processes, such as the throwing of dice. These events have the same rule (“random occurrence goes here”) but are different instructions (“throw dice here”, rather than, “random function occurs *here* in computer script”).

What I am suggesting is that the machine, its functions, what it does, can be encoded and passed in appropriate form, to a computer to execute. The rules and instructions are (see previous Chapter) a machine in the abstract such as a computer can enact. This is a conception of a post-mechanical machine: a text machine *requires a medium, but is not medium dependent*. Furthermore, I must add immediately and because of the preceding remark, *the texts the machine produces, similarly, require a medium but are not medium dependent*.

This latter proposal, relating to the text, is controversial on its own (and I return to it). Its extension to the machine that writes, one can only assume, is

[30] This distinction seems to be seldom made. David Bloor (1997) suggests it whilst distinguishing between teaching a rule by examples and teaching by instructions: “Sometimes we instruct learners verbally, and if the would-be rule followers understand our instructions they will be able to *follow the rule by following the instructions*” (p.11, italics mine).

not doubly controversial purely because that machine is so often left out of the picture altogether.

A text machine is importantly, although not solely, a set of instructions. These instructions *may* be converted into an instruction such as a computer may execute in the form of a program. Not only that, the program *must* be converted to an encoded signal if it is to be sent between computers (should I wish to send it to a Web server's computer, for example). Even if I wish only to run the program from the command line, whatever I type into the text editor by way of program must be converted into something the computer can use, via assembly language to machine code, to a series of charged and uncharged states, the myriad switches that are flipped in any computer whilst it runs. (Hillis, 1998, provides a readable and clear account of a computer, in rather similar terms, as a logical process that can be turned into a series of switches that may or may not be electronic and digital. This latter is but one option, albeit a fairly good one).

Neither the language of the instruction nor the medium it is written in are indispensable to it. That a computer program does not require a computer to exist is plain from the storage format of computer programming books: usually paper and ink (although a program requires a computer if it is to run, if it is to *do* anything). That an instruction is not language-specific is apparent from the possibility of "agreed transposition". This may occur between levels of code, as I have just noted: let us call it "conversion downward". So, we may write a program in something a human may find more digestible, higher-level codes, and this may then be converted to something the machine may use, its native machine language. We may also convert between higher-level languages: "conversion across". What is written in, to take just one example I am familiar with, *TRAC* may also be written in *Perl* [31]: what is written in one script may conceivably be changed into another. However, we may also convert a human language instruction into an instruction for computer.

[31] Margaret Masterman and her collaborator Robert McKinnon Wood used *TRAC*, a now rather antiquated language, to program *COMPUTERIZED HAIKU*. *TRAC* stands for "Text Reckoning And Compiling". *Perl* is a contemporary scripting language: "Practical Extraction and Report Language". I used *Perl* for my version.

These conversion processes are not in themselves controversial; it is their implications that are the source of disagreement. An instruction may be considered the text machine in the abstract, a kind of (what Turing might call) “ideal machine”. This machine may be written in a number of ways, in different languages, in different media. If it is to be made, we also require either a physical (non-computer) machine that will shunt the text inputs, or we need a different sort of machine, computer hardware plus program, that can perform the same text manipulations.

What I am proposing therefore is a conception of a machine that may be made in several ways whilst remaining recognisably the same machine. This does not deny its materiality, but relieves the machine of a sole dependence on a *particular* material. I hope thus to avoid strictures on anti-materialism, or worse, ethereality.

We now have a machine that is defined as rules and instructions that may be instantiated in different ways: as a set of actions that may be performed using paper and pencil; a “limited function” machine (a machine that performs one or a few set tasks and no more); or as a machine mimicked by a computer. (Of course these distinctions are themselves formal and not absolute, what is written requires writing materials and someone or something to do the writing; a computer is not information alone. Nevertheless, my distinctions describe real differences in the possible constitutions of a single machine). We must now consider what are the consequences of this convertibility, particularly what happens when the computer simulates the text machine.

4. The Problem With Rules

In what follows I will attempt to anticipate and deal with some problems that might be posed for my understanding of the text machine, particularly where it is reliant upon rules and instructions that have been characterised as inter-media.

Ludwig Wittgenstein[32] (2001) poses a problem with rules thus:

"*This was our paradox: no course of action could be determined by a rule, because every course of action can be made out to accord with the rule.*" (p. 81).

Before I discuss Wittgenstein's response to the problem he poses, I will say it is possible to invert this paradox: *any rule might also be made out to accord with a course of action*. That is to say, we could start with a course of action and produce numerous rules to account for it. Whichever direction we go (up from action to rule, or down from rule to action) we have difficulties with accounting for one by reference to the other.

The way I wish to examine this problem is to look at a particular case, La Monte Young's *Composition 1960 #10, to Bob Morris* ("*Draw a straight line and follow it*"). I choose this instruction as it is quite often referred to in the literature. So, for example, for Cramer and Gabriel (2001) it is "a seminal piece of software art because its instruction is formal" (p. 8). Their assessment, however, is only good if the instruction is interpreted quite literally. However, it may be construed in any number of ways. Young himself interpreted his instruction by variously drawing lines. But another contemporaneous performance involved sustaining a single chord on the accordion for two-and-a-half hours[33].

Yet "to draw" *could* be interpreted as to select something allowing chance determination. Lines could be put into a hat and 'drawn'. But the *line* itself could be a line of text (there is nothing to say it is not). A straight line could be a truthful or direct line of text. "To follow" can mean, in one usage, to

[32] I am, I should say, aware of the controversy around Wittgenstein and rules and particularly Saul Kripke's (1982) contribution to it. It is one that I abstain from here partly because of its potential to deflect me from my main task. Secondly, I wish to offer is my own contribution to the paradox Wittgenstein poses about rules.

[33] See Keith Potter (2000) p. 54.

comprehend. Now we have a completely textual version of the instruction that involves selecting, reading and understanding lines of text.

Now, if we were to begin with the course of action, observing someone carrying out our last version of the instruction, we would have someone selecting a line of text, reading it and attending to its meaning. But then if we were to attempt to produce an instruction for this activity based on our observations it is unlikely we would arrive at La Monte Young's instruction. The instruction we might make could be something like:

"Put truthful lines of text in a hat. Select at random. Attend to their meaning."

Something has happened here. Diagrammatically we could represent the process as:

Instruction [1] ⟶ **Performance** ⟶ **Instruction** [2]

And so on, with new executions and new instructions. Of course, it might be objected that the 'performance' was a wilful misunderstanding of La Monte Young. However, my wish was to illustrate the difficulty of following an instruction and of construing an instruction from a practice. This is a real crux. There now appear to be two quite different 'machines', and there is nothing to say there cannot be an endless proliferation of them, each with its own instructions. Reverse-engineering is going to encounter problems in these circumstances.

It is certainly possible to incorporate this tendency, to diverge incrementally, into the artwork. Tyson (2004) uses it in *Replicator*, where instructions are derived from an artwork and new artwork from instructions in what he likens to "the children's game of 'chinese whispers'". But where does this leave my attempt to define a physical, abstract and simulated machine? Each is said to be a different moment of a single machine. How can we be sure that a

physical machine is the instance of its abstract counterpart if interpretation can be so apparently arbitrary?

Wittgenstein's answer to his paradox is to claim, "there is a way of grasping a rule which is not an interpretation", and that "'obeying a rule' is a practice" (op. cit.). For Wittgenstein language is a form of practice: if you want to know what a word means look at how it is used. This sort of argument has occasioned some commentators to say that Wittgenstein has a "normative" concept of rule following (for instance, Michael Luntley, 2003, while David Bloor, 1997, speaks of the Wittgenstein's "normativity of rules" p. 19). However, this only really works if there is already some sort of normative activity, as there may be in language use. But my construction on La Monte's instruction is entirely legitimate. One cannot appeal to usage to 'correct' my interpretation, no matter how strongly it is felt it is wrong.

La Monte Young's *Composition 1960 #10* is capable of sponsoring an apparently endless number of practices partly because of the inherent ambiguity of words, but more to do with what sort of instruction it is. The best response is to value that quality, rather than to attempt to turn Young's instruction into some sort of proto-software. The instruction is *made* so as to permit a range of interpretations: *When we program a computer what happens is that we effectively have to opt for one, or several, but at any rate specific, interpretations and exclude others (those that are not programmed).*

There are patent differences between human and computer languages and instructions. The latter are both more cryptic and less ambiguous. Entirely contrasting interpretations can be a problem when passing an instruction to a computer.

For the moment I shall concentrate on a method of deciding if we are possibly contemplating the same machine when we are looking at its different versions. We could identify the normative activity of the text machine with its writings. Might it be these that will give us an idea of what machine we are looking at?

For instance, let us consider the programming of *COMPUTERIZED HAIKU* http://wwwin-vacua.com/cgi-bin/haiku.pl. Does 'my' version bear any resemblance to Masterman and McKinnon Wood's? It does, in the sense it is possible to use it to produce the same haiku as their program, and we can compare the examples in Masterman's (1971) essay with the products of my programming. The argument might go, "if it has a comparable practice it is a possibly a comparable machine". This is despite the obvious differences. Theirs is programmed in *TRAC*, mine in *Perl*. Mine outputs the haiku to screen and is available on the Internet; theirs was hooked-up to a paper printer when the Internet was no more than a dream, and so on. These differences are permissible because, as I have explained, neither a particular language nor a physical process is essential to the making of a text machine.

However, there is a problem in attempting some sort of taxonomy of the text machine by reference to its writings alone. This is because of the principal of 'loosely related levels' announced earlier. It was argued, following Fields (op. cit.), that a number of different algorithms could achieve the same computations, and these could be founded on different physical processes. This principle of 'loose relations' may be extended to the case of instructions (an algorithm is a narrower and more demanding subset: see Harel).

If so, merely that similar texts are produced does not mean the same instructions are being followed.

A way of dealing with this is to fall back on the distinction, made earlier, between rules on the one hand and instructions on the other, and extend this to algorithms. The rule or rules are here the broad structure of the machine, but instructions and/or algorithms are possibly alterable and replaceable. For instance, a rule might be to combine lines of text randomly. This might be achieved by following the instruction to physically cut the text into lines and shuffle, as with Raymond Queneau's *Cent Mille Milliards de Poèmes* (*Hundred Thousand Billion Poems*[34]). However, in programming, a

[34] There is an English translation in Mathews and Brotchie, 1998.

comparable effect may be achieved by using an algorithm such as the "Fisher-Yates Shuffle"[35]. This algorithm is one example of how to perform a shuffle. There are other ways to create these effects.

Now we have a concept of a rule that may have different instructions and/or algorithms to perform it, which may result in texts that are similar or identical. If we can make a rule, and the applications of the rule produce a comparable result, then we might claim we are following the same rule. This constitutes, in Wittgenstein's terms, "a practice". This puts us in the position where we are able to compare rules with rule following and to form an opinion of their relationship. This should work whether we start with practice and work up to the rule that is meant to account for it, or down from the rule to the practice that is said to represent the following of that rule. This necessitates some flexibility in our approach. Ironically, it has proved more difficult to say what we mean by rules and rule following than we might have hoped. There is no golden rule when it comes to rules (see Appendix to Chapter 2 for a response to this problem in relation to an example of artwork).

5. Conclusion

Derrida (1981) summarises one (there are several) of a series of unequal pairings in Kant. The mechanical has no claim to the Fine arts. Kant writes, "...a mechanical art neither seeks nor gives pleasure. One knows how to print a book, build a machine, one avails oneself of a model and a purpose" (p. 8).

---

[35] sub shuffle {
my($array) = shift();
for (my $i = @$array; --$i;) {
my($j) = int(rand($i + 1));
next() if ($i == $j);
@$array[$i, $j] = @$array[$j, $i];
}
}

Written in *Perl*. This is the algorithm only. Without the surrounding code it would not do much.

In Hegel too, the machine and its mechanical productions occupy a similar place. Mechanical writing is low in the hierarchy of the written. In Hegel it is at the lowest. If the peak of Hegel's hierarchy is alphabetic phonetic writing, then its nadir, according to Derrida (1982), is that of the machine: "Number, or equally, that which can do without any phonetic notation". Mechanical writing: words that are not spoken, that are not thought. Formal, abstract: in short, dead.

The products of the machine have their status clearly assigned. An 'art machine' is in this context a contradiction: if it is machine, it is not fine art; and so also if these terms are reversed: if it is fine art it cannot be machine. In fact, neither the machine nor its products can be fine art. Fine art "must seem to be as free from all constraint of arbitrary rules as if it were a product of pure nature." (Kant, *Critique of Judgement*, section 45)

An art machine in such a theory is doubly mistaken. It can afford no artistic pleasure, not itself being the product of free creativity nor is it capable of free creativity. And it is of no practical use, as its works also lack utility. The practical and mechanical, the rule bound and the unfree, are not unworthy and have a place. The only error would be if this place were to be exceeded. Rules in Kant do have a place, but this is best confined to the mechanical arts.

However, contemporaneously the machine and human are not conveniently distinguished. Uncertainty about their relative status is not resolvable so long as this is true. With computer-using art the ambiguity over the status of the machine is persistent and hard to resolve. Some such as Simanowski, 2005, (on combinatorial text/text machines) simply opt for accepting the case, and "consider and name such machines of combination as works of art themselves".

I also return to the question of rule following, at several points, later in this thesis. That 'true' art is sometimes posed as inimical to rules and a preserve of the intuitive I am sensitive to. This attitude has a long and respectable

lineage and I come back to it below. Of course much process driven, system oriented work has a strong rule oriented aspect, as I have mentioned in my remarks about conceptual art. This, I think, has done something to rehabilitate my subject.

Nevertheless, what I have done is to investigate a rather textual approach to the question of rule following. I have done this instead of staying with other sorts of instruction based art practice associated particularly with the 1960's. This textual approach I have pursued by way of discussion of the text machine. I have undertaken this task by programming my own. In the next Chapter I focus on the text machine more intensely, particularly where computerisation appears to raise some interesting issues.

So long, Emily, it was great while it lasted, but you were a robot, you had no heart
Umberto Eco

Chapter 4: Inscriptions

1. Introduction

In this Chapter I argue that text machines, and the texts they write in collaboration with humans, are a widespread experience. While investigation of generated text has often focused on the (for me, rather sterile) question of who (what?) wrote the text, in our daily lives we commonly encounter text machines. (There are some signs that research may catch up with these developments[36]). Whilst questions of authorship can be undecidable (for reasons I discuss), in our mundane contacts with such machines this question may scarcely be raised at all. There is in fact a blurring of distinctions between human and machine that we saw in the theory of posthumanism (above).

In this Chapter I begin by developing some ideas about how to think of machine written texts. I then go on to transpose some of these ideas to a discussion of the employment of text machines in actual use. What present as problems in evaluating the text in a literary context, I believe, should make us more critically aware when we broaden our discussion to text machines in everyday life. Also, I engage with the question of how artists have begun to exploit the possibilities of machine made texts.

[36] For instance, academic interest in 'folksonomies', a user-lead, bottom-up, rather than a top-down taxonomy, method of tagging information for other users to access on the Internet: see http://del.icio.us/ which uses this form of tagging.

2. Text Degeneration?

It may seem that the text machine has not found wide use in the visual arts. To the extent that this is so (and I will qualify this assertion in a moment), a few reasons may be advanced.

There are real difficulties in programming a computer to write anything of interest to human readers: ironically, the more interest a computer text has the more likely it is to be considered a computer-*assisted* not computer authored text. Secondly, the legacy of the Turing test has gone some way to skew discussion of machine texts. The Turing test, in a game of imitation it will be remembered, is the test of whether the computer can pass as human. So, attempts to program literature have been directed at imitating, often in parody form, human-authored texts. Thus Douglas Hoftstadter's[37] groundbreaking use of Recursive Transition Networks to generate text is entitled "*A Little Turing Test*". With the *Postmodernism Generator* (1996) the point was proved beyond doubt: under certain conditions it is possible for an obscurantist computer text to pass as an opaque human text[38]. Since then, my observation is that there has been comparatively little useful work going on in computerised writing. Computer texts have been posed in the context of a series of 'firsts' – from Edwin Morgan's[39] *The Computer's First Christmas Card*, to the Dada Engine's (the program behind *Postmodernism Generator)* first famous literary fraud. Once done, who was really interested in a second?

Lack of utility is a further issue. Blay Whitby (2002) makes the point that there is, "little practical use for a machine aimed specifically at success in the imitation game" (p. 62). It is scarcely surprising that while a few poets and

[37] The texts seem to have been produced in 1975 judging from the evidence, such as the use of a 1975 edition of *Art-Language.* (Incidentally, Andrew C. Bulhak, 1996 p. 1, says Hoftstader illustrated his method with *13* examples, when in fact there are 12, proving perhaps that Bulhak isn't a robot, proving perhaps I am).

[38] The article is, "*Transgressing the Boundaries: Toward a Transformative Hermeneutics of Quantum Gravity*". It was published in *Social Text* #46/47, pp. 217-252 (spring/summer 1996). It may be found at: http://www.physics.nyu.edu/faculty/sokal/transgress_v2/transgress_v2_singlefile.html.

[39] It was not in fact written by a computer.

other enthusiasts continue to program the better-established genres (emulating what humans may already do well), in computer applications the focus has shifted elsewhere to more specifically useful computer genres. Julia the Chatterbot not withstanding[40], in general, it may appear the use of computer-generated text has degenerated from overt imitation to self-declared spoofs and pranks. There are many examples on the Internet. There is a multitude of headline, pop band name, mission statement and other software. Their purpose is diversion. (Natural Language Generation, as opposed to random text generation is another project entirely and there is a substantial body of literature both on and off line.[41])

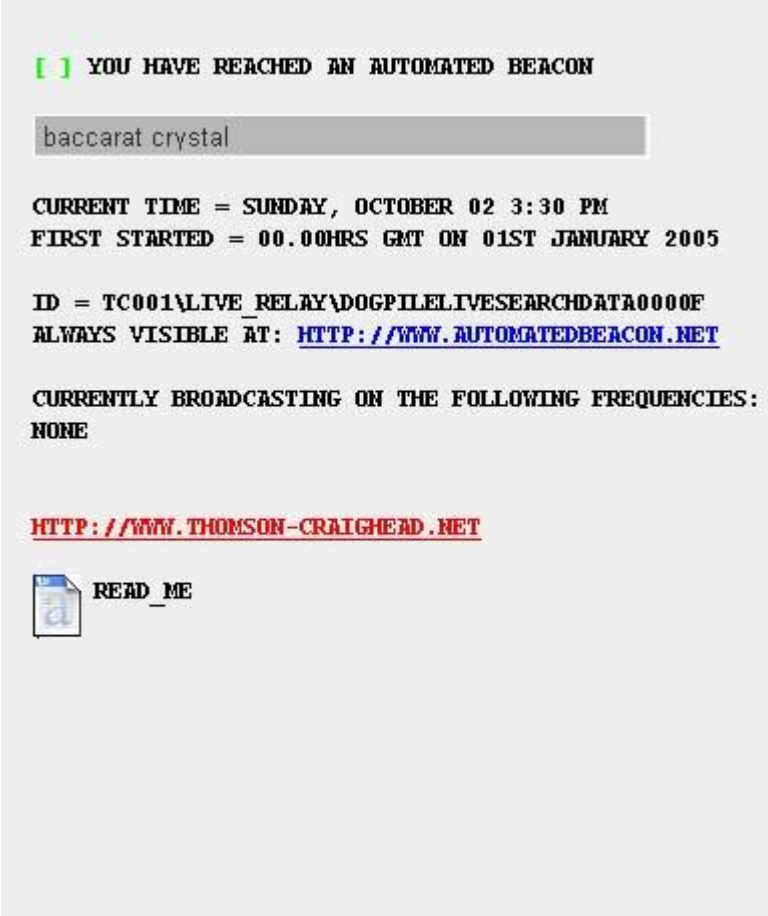
However, as we will soon see (in section 4, below), text machines have their practical uses. Also, paralleling this situation, a new generation of visual artists, many working on the Internet, have used the potential of the computer to make work that represents a development in the use of text machines as understood by this thesis. I return to this in section 4 below. Jon Thomson and Alison Craighead's *Automated Beacon* (Plate 4) is the sort of work I have in mind: http://www.computerfinearts.com/collection/thomson_craighead/beacon/index.html [42]. Like several other works I discuss below, this makes no attempt to pass itself off as anything but (an automated, therefore) a machine work.

[40] See Murray (1997) for a discussion of Julia's career in chat rooms.

[41] See the Cosign website for instance http://www.cosignconference.org/. Manurung's thesis is about NLG generation of limericks (I have to say, they are not very good, but this may not be the point). See also see Dale et al, (2004) for a discussion of some practical applications of NLG.

[42] Thomson and Craighead (2005) "The beacon continuously relays selected live web searches as they are being made around the world, presenting them back in series and at regular intervals. The beacon has been instigated to act as a silent witness: a feedback loop providing a global snapshot of ourselves *to* ourselves in real-time."

Plate 4



However, rather than simply dispensing with the problem out of hand, in the next section I linger for a while longer over the vexed question of authorship. I do this by subjecting to scrutiny the idea of reverse-engineering as already advanced in previous Chapters.

3. Reverse-Engineering a Text

How *do* we know the machine apart from the work it does? What is a unit of work for a writing machine? Sonnets? PhD theses? Perhaps for a moment we might think of *this* text as computer generated. Or then again, maybe the machine did *not* write the text: instead the text wrote the machine. There never was a machine. It was a figment of the text, its spectre. There's a word for machines like that; it comes from computing: vaporware. Vaporware: "Computer-industry lingo for exciting software which fails to appear"[43].

That (vaporware) was a compound word, combining connotations of insubstantial exhalations with those of solid commercial goods. (What is surprising in that? Computing is after all an industry whose commerciality is built on the patenting of ideas).

Perhaps we might try to reverse-engineer the present text, working back from text-product to machine-producer (if there *were* a machine).

“Reverse-engineer”: engineering reversed. Engineering: product specification turned into product. Reversed: begin with product, work back to specification.

Why do reverse- engineering? **“reverse engineering** *n* the taking apart of a competitor’s product to see how it works, e.g. with a view to copying it or improving on it” (*Chambers Dictionary*).

Reverse-engineering proceeds from the many to the one: many products may implement the same specification. Thus I say *this* text, but if there is a machine, the machine is the “top level specification” and this text is but one of its possible implementations. And if there is a machine, can we expect to discover it entirely from working back from the text? No, “it is not possible in practice, or even in theory, to recover everything in the original specification

[43] John Naughton (2002) p. 299.

purely by the studying the product" [44]: the machine will always in some way elude such approaches.

But worse, perhaps we would find nothing at the 'origin'. We might attempt to work back only to discover an absence where a *something* should be. There would be no machine, merely vapour.

Which is the top level, the unitary, the one, and which the many, the low, the mere product?

Is it the present text that produces in the form of vapour a machine to account for its writing? Or is it the other way round, there *is* a machine that manufactured this text, and a potential multitude of similar texts?

(It is easy to imagine a maze of proliferating and reversible passages between texts that produce machines that produce texts that produce machines. And so on. Without end).

One resort might be to defer the question of the authorship of machine texts, and invoke Hoftstadter's idea of "meta-authorship". This is a theory of levels of authorship. Instead of the usual mono-authorial, if I may put it like that, layer ("the author"), we have at least two layers. Hoftstadter is discussing music; we have the machine that ("who"?) is the author of the score, and a human who is the author of the program. The author like the economic then: determination in the final instance.

This is all fairly well if we do not raise the inconvenient common circumstance that in coding circles programmers share code[45]. One response may be to

[44] David Musker (1998) http://www.jenkins-ip.com/serv/serv_6.htm

[45] An example: I have adapted in my *Virtual Dictionary* (see Chapter 6) something called *demo_randomtext.pl*, part of the *parse:: RecDescent* distribution. Now, *parse:: RecDescent* is a *Perl* module. A module is basically a publicly available program that your own programs may use, saving you the work of writing it all out yourself. This, *parse:: RecDescent*, is written by Damian Conway. It might be reasonable to assume that he also wrote *demo_randomtext.pl*. I might have thought so too if *demo_randomtext.pl* had not contained a 'bug'. I mailed Damian with this (known as 'bug fix'), and he mentioned that he had not written

credit whoever ‘signs’ the work, whoever else has involvement; the common situation in the visual arts. Because of such eventualities and the sheer difficulty of resolving the problem, a more rewarding approach may be to evaluate what sort of text it is we are dealing with.

However, this is rather difficult. Aarseth’s (1997) worthy attempt to clarify a key question of computerised literature (“Who or what writes?” p.132) is not very viable. So Aarseth’s typology of *Preprocessing*, *Coprocessing* and *Postprocessing* has to presuppose the information it is expected to produce. That is to say, Aarseth’s decision to accord Racter’s (1984) *The Policeman’s Beard* to both “Preprocessing” and “Postprocessing” depends upon accepting that the whole thing was not cooked up – which is exactly the thing that we cannot be wholly sure of. Or maybe its text was not revised at all, but is as claimed (in the *Introduction* by William Chamberlain and in contradiction to Aarseth’s own assessment) the work of Racter alone. As we cannot tell, we cannot place the text into Aarseth’s typology with any reliability.

It is worth considering that these questions, discussed in reference to machine texts, are perhaps a *mise en abyme* of a greater question of the text, its origins, its authors, its boundaries.

Another way of putting it is that this discussion of texts is a ‘sub routine’ of the greater program known as Deconstruction. And by uttering its name at this point do we encounter *this* sub routine's 'exit' command, and must eject the loop, and return to the main program? I think not. Rather, to continue the metaphor, I will stay in the loop and iterate over questions that may attach to this text or a text like it, a human-machine collaboration. I could say further, I will stay in the loop until it has run its course and then return a value to the main program (this is what sub routines are meant to do). I could, but I wish to resist this reduction of the current investigation to a minor moment of some greater project.

---

demo_randomtext.pl so had not known about the bug. Meta-authorship is as hard to decide as authorship.

With the authorship of reputed computer novels and poems, it is often impossible to decide who wrote the text. This is still more the case because the text is usually severed from its programming. Pleasurable confusion flourishes in such circumstances.

An example: at http://www.collectionfaq.com/writing-machine.html there is this entry:

"WRITING MACHINE - Definition
A writing Machine: abstract configuration, its theoretical class of instructions. See also Inscription, Instructions, Writing Machine, Code, Rule, Computer. ..."

This appears on www.collectionfaq.com's web page along with entries for software and toner and web dictionaries. But, "abstract configuration, its theoretical class of instructions" is a recursively generated text by http://www.in-vacua.com/cgi-bin/machine_definition.pl. It is in fact randomly generated and probably nonsense. However, anyone clicking WRITING MACHINE - Definition will get a new and equally spurious definition. It has inserted itself into a legitimate discourse, to do with online commerce and academic endeavour.

4. 'About typing characters from a picture'

Whilst we may, in our prosaic moments, not come into contact with machines that write poems (although they are on the Internet for those who have an interest), a text machine flourishes everywhere machine-texts are required. But this is not to say that the quality of their writing has to be very high. Functionalism, not pleasure is their purpose.

The text machine is everywhere; so common we barely notice it. The word processor (Microsoft Word's in the case of the present text) includes a collection of them masquerading as this Wizard, that Wizard, and the infamous 'Spelling and Grammar' that so deplores the passive voice. Of the

text machines we occasionally encounter, novel-writing and non-directional therapy get most of the scant attention afforded their number, whilst their industrious cousins go about their tasks and meld with us in the fulfilment of that process.

Commercial and bureaucratic interests[46] have concertedly taken up, for their own uses, some of the basics of computerised literature, particularly *randomisation* and the *template* (see Chapter 6 for a definition of these terms). Text machines are not necessarily works of art or literature as Murray (1997) mentions in her tract on electronic literature.[47]

The two areas in which this is most common and noticeable are in online form completion and password dialogues. The password and the questionnaire are identified as characteristic forms of contemporary social control. Why?

> "Code is a language, but a very special language. *Code is the only language that is executable…*So code is the first language that actually does what it says – it is a machine for converting meaning into action. (Italics in the original. Galloway/RSG, 2002, p.352).

To be executable is the standard to which all coding aspires. Jean Baudrillard (1993, p. 73) realised it early – 1976: "binarity and digitality constitute the true generative formula which encompasses all the others and is, in a way, the stabilised form of the code". Baudrillard's "code" is none other than digital code, binary code; but it is code that has escaped from the computer and has infected human society. It *does* something: it digitises the social.

What is this digitisation, what does it comprise? The computer is the Lab condition of the investigation of this question. An alphabet of on and offs, the

[46] Also in an amusing mimesis, *illegal* capitalistic ventures. See Matthew Fuller (2003), particularly *It Looks Like You're Writing a Letter*, for a discussion of email scams using Microsoft's letter Wizards. (These appear to be, fundamentally, template systems). As Fuller writes, "The believable template, hooked up to the mailing list database, is an economic machine" (p. 147).

[47] Text Machines are not necessarily poetry machines, art machines, novel machines, nor indeed, necessarily producers of corporate home pages and love letters, to cite two examples Murray (p.189) mentions. They may do these things, or any number of others.

digital is at bottom a discontinuous signal. The discontinuous has a bound and a measure: it may be manipulated exactly, to the limits of our control structures. Afforded a numerical weight, it may be modified according to some algorithm. It is this precise quantification that sets apart digital media from earlier, anticipatory, media (a point Manovich, 1999, makes).

If this is what the code is under the microscope, Slavo Zizek's (2004) "'new', digitised capitalism" (p.185) seems to be what it is once it has escaped from scrutiny in a network of computer-bunkers. This is the world of Baudrillard's (op. cit.) "great festival of Participation", of a "myriad stimuli, miniaturised tests, and infinitely divisible questions/answers" (p.70). The isomorphism between the model of the computer and that of capitalism is a point that Charlie Gere (2002, p.46) makes, whilst warning against technological determinism[48].

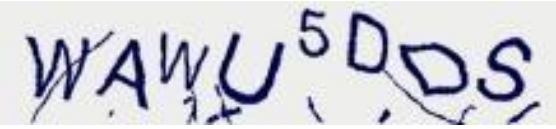
And there now are innumerable question and answer dialogues governing many aspects of life both on the Internet and off it. There is no shortage of artwork that engages with this situation. For instance a quite well known example is *THE INJUNCTION GENERATOR* at http://ipnic.org/, which turns corporate template filling on its head. Another is at http://status.irational.org/: *The Status Project*, a complex form-filling piece of software[49].

Code, in these contexts, has at least two meanings. Code can mean *a code of conduct or behaviour*, and code can mean *a form of encryption*. However, it by now should be apparent, the two are not entirely separable. On the contrary, encrypted codes can drive social compliance. They may do this in part by their generation of texts that work to further legislate action. I now proceed to discuss this in more depth.

[48] "Turing's conceptual machine, capable of being reconfigured in an infinite number of different states, is the perfect, idealized model of capitalism as universal machine, in which different phenomena, labour and commodities are homogenized in order to be exchanged, manipulated and distributed."

[49] Gere (2002) reminds us usefully of Hans Hacke's prescient combination of conceptual art, computers and the questionnaire format: Hacke's 1970 *Visitor's Profile,* "a parodic computerized questionnaire system" (p. 108). Gere observes much net.art reprises earlier work (ibid. p.111).

Plate 5



MSN

The above image appears on a web form from *MSN*. It has a use. It prevents 'spambots' signing up for email accounts. You must copy the characters in the picture. In the words of MSN:

"You must type the numbers and letters you see in the picture to confirm that a person is trying to access the page, rather than an automated program. This helps keep automated programs from creating fraudulent accounts or misusing accounts, for example, to send large amounts of unsolicited e-mail.

In most cases, automated sign-in programs can't recognize the numbers and letters in the picture".

From, '*About typing characters from a picture'*:

https://help.msn.com/!data/en_us/data/passportv31.its51/$content$/PP_TROU_REG_TypeCharactersFromAPictureToSignUp.htm

But why should 'automated sign-in programs' have difficulty recognising numbers and letters in a picture but not in typewritten text? Why the selective illiteracy?

Spambots and other automated programs, in fact programs in general, do not 'see', nor do they really read. Spambots seek for characters in the ASCII format; they search for character strings in ASCII mode, but not optically. ASCII keys have code designations assigned in the operating system (see Hillis, p.55). In other words there is a strong connection between code and

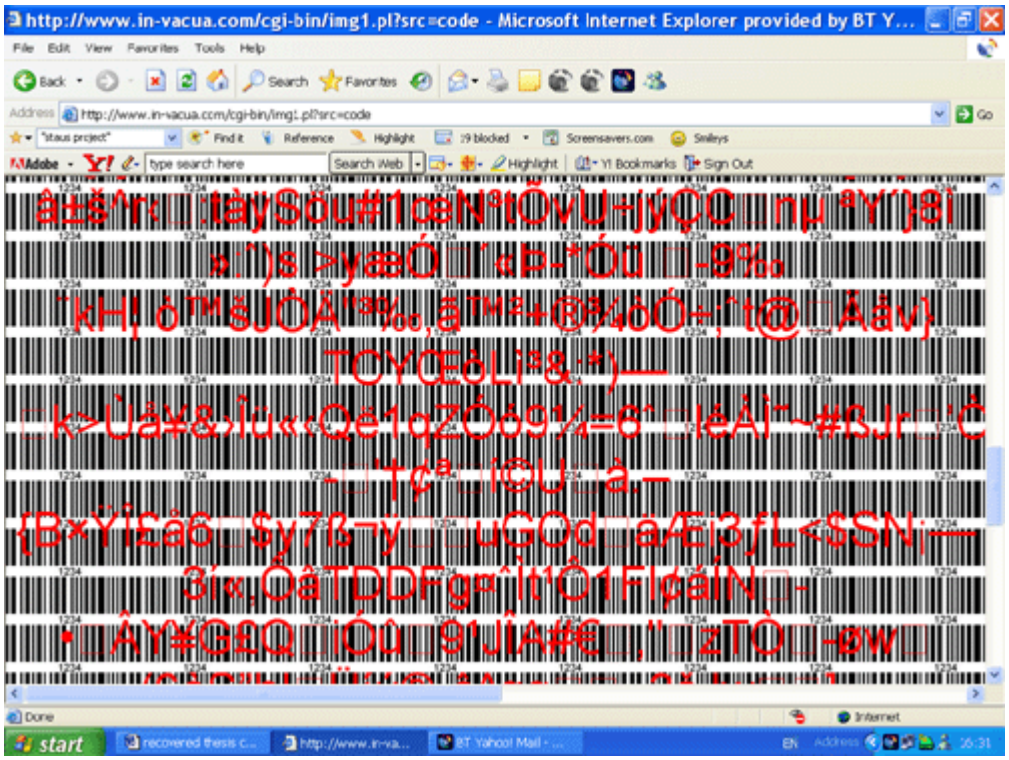
character, between letter and machine language. But a human being reads the visual[50]. In most cases it is of little practical importance to a human reader whether there is a code reality behind the visual appearance or not. Whether it is charcoal or pixels, the text may be read. To a 'spambot', however, there is a world of difference at the code level between a gif and a character string. A Softbot (software robot) could not discern the difference between the image of a letter above and the image of a tree (although sophisticated pattern recognition software might). It would look in vain for a letter 'A', 'B', or whatever.

A human deciphers the *image* above; a human enters, hopefully correctly, a *character* string: performing a translation from something the machine does not read to something it does. Human: intermediary between two programs. If 'yes', you pass. If no, try again.

Computers work on Boolean logic (Hillis, Chapter 3, or Joseph Weizenbaum's, Chapter 3: '*HOW COMPUTERS WORK*'). This constitutes the translation of logic into a few relatively simple operations. And in this world of 'either', 'or', 'and', if the letters you enter matches you succeed, if not, not.

In these dialogues in which two different languages are 'spoken', we have a form of text machine. This machine generates unique *random* passwords. The user enters them into a *template* form where they are checked against a pattern. If the user succeeds, they gain access. Words in *our* language are effectively turned into keys (and keys to key strokes), keys to a machine you may access, or not.

[50] Visual tests have been criticised for excluding visually impaired people (*Spam-bot tests flunk the blind* http://news.com.com/2100-1032-1022814.html. Sometimes these web dialogues have an audio version.



We see here what may happen when a text machine, a machine that writes, is computerised. Word and code interact[51] A work that deals ironically with the differences and dependences of code and image is my own *src* (html abbreviation of “source”) at http://www.in-vacua.com/src1.html, a work that simultaneously displays the image’s code and the image (selected at random from a database using a user entered search term and source code. See Plate 6).

Code is converted to word and word converted back to code. But in so doing other transactions take place. We are checked, as much as what we write is checked: a Hotmail account is a fairly innocuous example.

5. Real-World Scenarios

There are many situations where the two connotations of code (social code and computer code) interact. But brute ID checking and permission/ permission-denied situations are only the most overt. Lawrence Lessig (1999), in his *Code and Other Laws of Cyberspace*, argues persuasively that what he calls the *architecture* of systems compels what may and may not be done within them. In computer terms, this amounts to how they are *written* and for what purpose. (Lessig gives, amongst others, the case of the right to log-on anonymously and its refusal, and how this determines behaviour in various ways).

Galloway (2004), an admirer of Lessig, develops these perspectives further in his *Protocol*. In his analysis the term is employed variously of specifically *Internet* protocols and in a wider usage to typify contemporary social arrangements. There are, however, some ambiguities and difficulties in the terms usage in this latter context.

He demonstrates, I fear with unconscious irony, a marked tendency to imperialise and centralise, as he finds “protocological” tendencies everywhere he looks: in the world, in the analyses of Hardt and Negri’s *Empire*, Deleuze, *and* Foucault.

Heavily indebted to Deleuze, *Protocol* fails to substantiate Galloway’s central claim (perhaps it should be his decentralised claim) that "I consider the distributed network to be an important diagram for our current social formation" (p. 11) Throughout, there is an unevenness of tone, as at times he appears to advocate this society as one to achieve, at others to describe it as already being in existence. He does succeed in describing these networks and gives real examples (of the highway system for instance). He also summarises Deleuze and Guattari's avocation of decentralised social networks as preferable to centralised ones. But he offers absolutely no evidence that that we are actually living through a shift "from central bureaucracies and vertical hierarchies toward a broad network of autonomous

social actors" (pp. 32-33). He therefore never makes good his promise of extending his analysis from the function of protocol on the Internet to map the "new millennium". He casually assumes "a larger process of postmodernisation that is happening the world over" (p. 33). Specifically, he does not persuade that the current social formation really fits with the idea of distributed networks with "no chain of command, only autonomous agents" (p. 38). To do this he would have show that the diagram of a distributed network really maps the world of post 9/11 and the occupation of Iraq. A difficult task.

However, that we follow rules (without necessarily knowing it) when we interact with computer systems and that these are written (but not by most of their users) is patently true. It is the fact that rule-governed behaviour is for the most part not explicit to its participants, and not under their control, that may comprise one of the most insidious aspects of this situation. Rule following becomes second nature and normalised without awareness; but not to rule makers who *are* highly conscious of the need for a smooth-operating, rule-formed environment. WC3, the World Wide Web Consortium http://www.w3.org/Consortium/ (the body that regulates the Web), is engaged in this activity. Not only does WC3 develop rules (read “protocols”) but also the language in which they are expressed (*Rule Language Standardizations: Report from the W3C Workshop on Rule Languages for Interoperability*, 2005). The WC3’s thoughts are, in fact, informed by cognisance of other social rule-using situations[52]. This represents an inequality that will arise again later when I discuss Code in the next Chapter. Those who make rules are in an advantageous position relative to those who do not.

[52] “Real world scenarios helped illustrate the need for rules and ontologies (including anatomical knowledge to label brain parts, situation awareness using OWL and Rules, and others such as RDF in the automotive industry, in access control, and rules for geospatial applications).” http://www.w3.org/2004/12/rules-ws/report/

6. Conclusion

Where does all this leave my discussion of the text machine? A language and some rules do not in themselves constitute a text machine as developed above: with the computer the text machine lives within its environment and must be written in their terms. This machine may be a harmless artwork, or it may be some other more commercial, military or bureaucratic application. Whatever it is and whatever it does, it is a machine within a greater machine, a machine of machines, within the greater social machine (to borrow from Deleuze's vocabulary).

Furthermore, a machine rarely if ever works alone. Machines use other machines to perform their tasks. Sometimes these are other text machines. (See my discussion, Chapter 6, of how '*src*' uses other machines to complete its task). The Internet provides the condition to enable, by virtue of its networking, the cooperation (or competition) of numerous machines.

The text machine is everywhere. It is writing everywhere and at all times. If we wish to find text machines, if we want to read – or know how to write – a text the answer is probably under our nose, on the desktop.

In the next Chapter I develop some ideas about the relation of code and writing begun in this Chapter.

all the clouds turn to words
all the words float in sequence
no one knows what they mean
everyone just ignores them

Brian Eno

Chapter 5: Code

1. Introduction

Text machines historically precede the digital computer. They may be written, and made, without recourse to computerisation. Before the computer, the text machine was an analogue device: it was imprecise and slow, such as Tzara's instruction to cut up a newspaper or Queneau's poem, with their handicraft mechanics of scissors and glue.

The computerisation of the text machine is its transformation into something digitally precise and industrially usable. The computerisation of the text machine means it is now tied, like everything else that shares its fate, to a second articulation system: that of the computer's program code. Control of this second articulation system promises control of the first. It has also meant, as I have mentioned, the exponential growth of these machines that now penetrate every part of our existence. But how is this aided and facilitated?

The condition of possibility of a text machine's machine-text is not that it is written but that it is *re*-written. Any of its printed texts, whether printed to

screen or to paper, are cross-sectional slices of that machine's text production[53].

The (false) impression that these are, on the contrary, finished textual objects[54] is promoted by – so often – these texts' function, in the form of quotations, as relics of absent machines. (These critical writings are often the reliquaries of long departed machines). No doubt this misunderstanding arises because most (literary) text machines are experienced, most of the time, if at all, not in their active functioning, but in this arbitrary division of their texts into the static quotations that appear in the slender literature of this subject.

> a second possible strategy: the construction of an unhealthy obsession with triangles…

"All programs are texts that read texts and write other texts", Jay David Bolter (1991, p. 9) writes. In the box (above) is a quotation from a *Google's* cache of a program running on my web site[55]. *Google's* spiders 'read' the text and saved it. Robots read robot literature. When the software robot entered the site, the program ran and printed the text. No other readers or writers were required or present.

---

[53] This formulation, I am aware, echoes that of Deleuze and Guattari's "desiring-machine" (which, according to them, a work of art is, p33) in *Anti-Oedipus*: "A machine can be defined as a system of interruptions or breaks (*coupures*). Every machine, in the first place, is related to a continual material flow (*hyle*) that it cuts into" (p. 36).

[54] Of the exhibition, *Generator*, Geoff Cox (2004) writes: "The exhibition title referred literally to the term 'generator' in describing the person, operating system or thing that generates the artwork, shifting attention to these productive processes, rather than end products or the commodity form" (p. 9).

[55] http://www.Google.com/search?q=cache:5LChCm9u36MJ:www.in-vacua.com/cgi-bin/generator.pl+markov+text+in-vacua&hl=en&start=2

A ‘software robot’ is itself a program that ‘reads’ (with the qualifications in the previous Chapter) other programs: preliminary evidence in Kittler’s (1999) silicon Armageddon: “Instead of wiring people and technologies, absolute knowledge will run as an endless loop” (p. 1).

It is possible to write a machine that reads a machine and to instantiate it in the form of a computer’s software and hardware.

Bolter makes this distinction: “Formal languages are operational: they direct the computer's actions. Human languages are merely stored in the machine, as texts to be divided, re-combined, and presented to readers” (op. cit. p. 10).

This is true, but its generality disguises the fact that words cannot simply be *stored* in the machine unaltered; they do not pass through the guts of the computer like a stone, but must ultimately be transformed into machine code, as must the program. It is true that human languages in the computer are not, in Bolter’s terms, “operational”, but they may only undergo the processes Bolter describes if they are first transposed into a code the machine may use. They have to be transposed again if they are to be re-presented to a human reader.

In this Chapter I describe how two articulation systems coexist, how one effectively produces the other and what this means for us, the users who are also implicated in these machines and visual artists who want to work in this area.

2. Code and “The Code”

Blocks and snippets of code stitched together can make programs. *Alt_Img_Tate* (http://www.in-vacua.com/alt_tate.html) is an example. This is a small part of its code:

```
<script type=text/javascript>
setTimeout(' document.location=document.location' ,10000);
col=255;
function fade() { document.getElementById("fade").style.color="rgb(" + col + "," + col + "," +
col + ")"; col-=5; if(col>0) setTimeout('fade()', 200); }
</script>
```

Plate 7

This code does this:

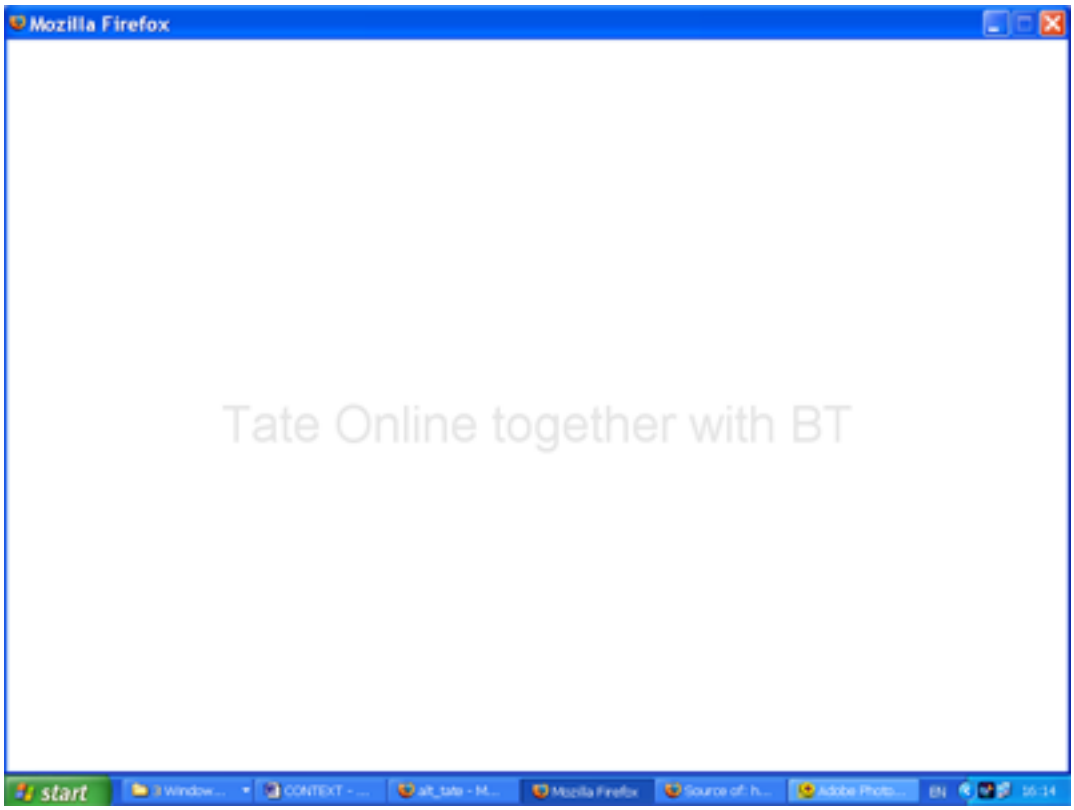
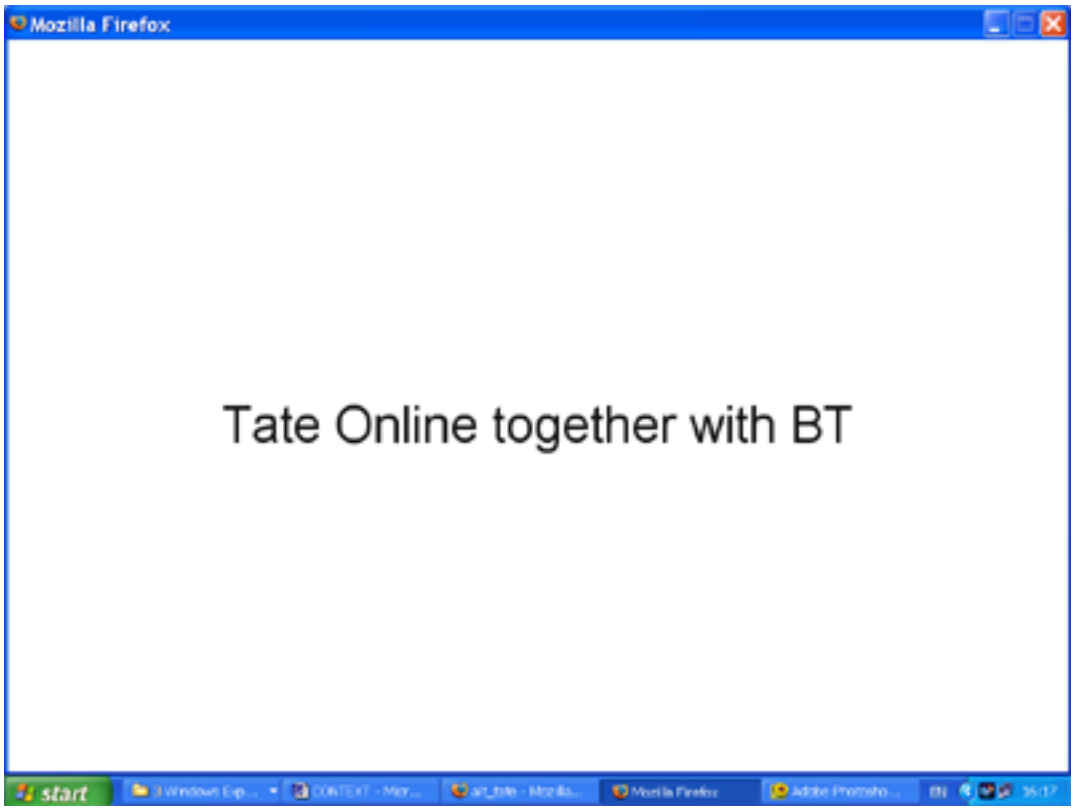


Plate 8

It fades in the text.



Code is executable here in a different way to any social code. All things being equal, the above code will produce regular and predictable effects so far as the on-screen event is concerned. No social code can be written or run with the same efficiency. But no social code is as fragile. Change this:

);

to this:

)

and the code may break.

Furthermore, there is a radical difference between the code block and the legible text it makes. This is not so with the sort of code that Deleuze and Guattari, or Baudrillard, are talking about, for instance. No matter how abstract the social code, no matter how arbitrary, what one is to do must be known if the code is to be followed, even if one does not know why. This, in Deleuze and Guattari's terms at least, is the whole point of (re)coding: to enlist conscious support in the economic, to encourage libidinal investment in the economic process.

Some computer code is like the code snippet quoted above: essentially a simple instruction to produce a definite effect. But there are also code processes, not only illegible to all but the machine, but also unseen, that produce textual events. (In normal circumstances such code is often not visible. There are those who post their code on the Internet. Even so, if we were to compare the code with the screen-grabs above, we would readily experience the disjunction between these two levels).

The codes that drive this process are practically interchangeable: it is the rules and instructions that are the structure of the machine. These too may, as I have suggested, be varied: and the digital has been shown to facilitate this process.

Inscriptions, or possibly what Deleuze and Guattari call "jargons", are constantly churned out by the social machine and experienced at conscious levels. They too are re-writable *and* constantly rewritten. Their production is driven by the rules and instructions of the social machine of capitalism, what Deleuze and Guattari call the "axiomatic". However, the difference with Deleuze and Guattari is the axiomatic cannot be changed (axioms cannot be changed, they are axioms) but merely added to. Rules and instructions are interchangeable, replaceable, and (consequently) temporary. They exist to get something done.

And perhaps, anyway, this is time to settle accounts with Deleuze and Guattari: because if contemporary society can be viewed as machine, then

the question of the role of code must be settled also. On page 251 of *Anti-Oedipus* we may read "… capital figures as a directly economic instance, and falls back on production without interposing extraeconomic factors that would be inscribed in the form of a code", but on page 260 that modern societies "recode with all their might". On page 251 "capitalism thus proceeds by means of an axiomatic and not by means of a code", but on page 257 they write of "…our modern way of "imbricating", of sectioning off, of reintroducing code fragments, resuscitating old codes, inventing pseudo codes or jargons."

The way to resolve this *apparent* conundrum may be to go with the spirit of the text, not the letter, perhaps following the treatment of *Anti-Oedipus* as in Frederick Jameson (1999). Jameson essentially accepts capitalism's 'axiomatic' reality (that is to say, its placing of the economic, profit and accumulation in other words, over all else). However, he also proposes that capitalism needs to "transform bits of the axiomatic back into so many codes… to invent older forms of coding to supplement its impoverished structures" (p.20). These codes are not what they were in pre-capitalism. Code in capitalism is reinvented and old codes reused to strengthen subscription to what might otherwise seem a purely economic enterprise, thereby running the risk of weakening it. Jameson writes "[t]his incapacity of the axiomatic, or of capitalism, to offer intrinsic libidinal investments to its subjects…is surely one of the most interesting and promising lines of investigation opened up by the 'Marxism' of *l'anti-oedipe*" (ibid.). However, this project is not really pursued in *Anti-Oedipus*, possibly because of the lack of resolution within the text of this issue of code that presents itself initially as conundrum.

It will be apparent by now that concepts derived from computing, particularly computer code, are easily reached for as explanatory tools. Despite significant differences this may not be surprising, as computers are involved with many of the real social and economic processes of contemporary capitalism.

3. Code and Self-Reflexivity

The algorithm and the programming code are (usually) invisible, whilst the text can be read. Furthermore, the unseen (code) writes the visible (text). We can see their work, but the algorithm, and the program code it is written in, are not presented: we only read the words.

The words, something printed to screen, or file, or paper, and the data that is taken as an input to the program, here have a dual identity: they are transposed into program code, manipulated, and then transposed again to words that we may read. In this, it will be apparent, the reader is experiencing, at best, only half the process.

Inke Arns (2001) suggests two texts, the “phenotext” and the “genotext”; and the surface (phenotext) cannot be understood on its own, without the genotext:

“My hypothesis is that the notion of “loss of inscription”, with its focus exclusively upon the surface text as the “text” of net art or net literature, is based upon the wrong formulation of the question. It is not sufficient, regarding the “surface effects of software” – the dynamic presentation of data by staging information and animation, to speak of a “performative turn” of graphic user interfaces (1), because this view limits itself to the performativity of those surfaces. One should rather assume the existence of two texts, a “phenotext” and a “genotext”, when examining net art and net literature projects. The surface effects of the phenotext, i.e. moving texts, are generated and controlled by other underlying “effective” texts, programming codes or source texts.”

(These terms, genotext and phenotext, are perhaps derived from Richard Dawkins’s, 1982, and Susan Blackmore’s, 1999, use of phenotype and genotype. However, Dawkins repudiates any crude determinism between the cultural phenotype and the genetic code. As I have argued, there is also not a simple, one-way and determining relation between code and text).

The fact is, of course, in ordinary circumstances programs are access-protected whilst the texts they produce are not. This comprises an inequality

between user and programmer/machine maker. The accessibility that Finnemann (below) lauds in his discussion of digital media is logical, not actual. It is prevented by restrictions both practical (you do not know the password to my FTP[56] program, you cannot, unless you crack it, rewrite my programs) and legal, in the licensing of proprietary software (you lease the right to use, not buy the right to change[57]). The fixed program's effective priority over the moving text occasions Arns to refer to code as "law", a by now familiar formulation.

However, in some ways these strictures apply particularly to digital media, which I have been careful to distinguish a text machine from. So, we do not usually know the algorithm that performs the processes of a text machine from that text when a text machine is computerised. Computer algorithms, as I have said previously, should not be confused with the rules and instructions of the text machine. Even if we do know the algorithm, we do not necessarily learn from this anything of the text machine. This is because any number of algorithms may produce a given text sequence. In Fields's terms (2002), as discussed above, a computation can be achieved by any number of algorithmic processes: this is true also of the text manipulations of a text machine.

Clearly therefore, neither the algorithm or the code, so long as we are referring to program code, is essential to our machine. Word sequences may be generated by different algorithms, hardware, and by different codes. This is obvious from the fact that well-known programs are often available in different languages and for different operating systems. Code is law, but only within the domains where it is sovereign.

The duality that we are talking about here – the text we read and the code and algorithms of the computer that we do not – is a source of some discomfort in

[56] File Transfer Protocol. It is used to upload files to the Internet.

[57] "Licensing software is different than purchasing a car or house in that you have the right to run the software but there are ongoing requirements that determine how the software can be used." http://www.microsoft.com/licensing/resources/default.mspx

‘software art’ circles. It contradicts a formalism that is perhaps inherited from modernism, that we should have concern for the materials of a work's construction. This is encountered in painting, sculpture and architecture and elsewhere (structural cinema, for instance) in different forms, whether it is Clement Greenberg's (1960) doctrine of the essential flatness of painting, Brancusi's "truth to materials" in sculpture, or the foregrounding of steel and concrete construction in brutalist architecture[58].

From this arise all attempts to unify code and output in contemporary software practice. A particularly interesting example is live programming[59], where the audience experiences simultaneously the programmer *writing* code and the *output* of the program.

Such tactics are spectacular. The objection to them is should they become a constricting orthodoxy, a self referential and prescriptive practice. It would be a pity indeed if software practices should end in a similar cul-de-sac as the painting of the *Support-Surfaces* group – a restrictive practice concerned with its own conditions of physical production.

Nor of course can we confine ourselves to these physical conditions solely if we are to understand fully a text machine (or anything else for that matter) running on a computer.

There is much about a text machine that cannot be found by peering into the guts of its code alone: its interaction with its physical and social environment for example. Also, how its operator *reads* its output cannot be predicted from an examination of its code. (I will return to these points in below).

Staying with text/code for the present, the problem is whether or not we can understand the text without an understanding of the code upon which it stands.

---

[58] See Simon Yuill’s paper, *CODE ART BRUTALISM: LOW-LEVEL SYSTEMS AND SIMPLE PROGRAMS* http://art.runme.org/1107798902-7563-0/yuill.pdf for a parallel between the two.

[59] See Adrian Ward et al (2005).

Aarseth (1997) criticises Peter Bøgh Andersen et al's (1994), *The computer as medium* for neglect of just this matter. As Aarseth remarks: "The main problem of computer semiotics seems to be the assumption that cybernetic sign processes can be understood and classified by their surface expressions alone" (p. 39-40). It is this for this reason, I suggest, computer signs cannot be *Indexical* in the Peircean sense[60]. To be indexical there must be a direct and invariant relation between event and sign, such as between weathervane and the wind. But the relationship between a code event and sign is set by convention, not by physics. Nevertheless once set (programmed) it is more compelling than other conventional arrangements that characterise another order of signs in Peirce, the *Symbolic*. This is because software controls events, physical states, in the hardware. In short, I conclude that Aarseth is correct to doubt the viability of Peircean semiotics in computing. This is because of the coexistence and interaction of two distinct systems, something Peirce's semiotics could not reasonably be expected to account for.

Aarseth, himself, refers to the "textual machine" (p. 42) but does not go beyond this phrase. Not unusually in this area, he is more interested in discussing the text produced than the machine that produced it. His discussion of machines is confined mostly to discussion of computer semiotics. I have sought to establish the computer is a *different* machine to the text machine: the former may simulate the latter. A text machine may, as it were, inhabit the computer: it is not the computer. Questions of computer semiotics must be distinguished from discussion of the text machine, therefore: they are not identical. What are problems of computer semiotics and what of text machines per se must be, consequently, carefully differentiated. However, the computer poses problems of interpretation for the text machine when the text machine is running on the computer.

[60] Therefore, I cannot agree with Javbrett's employment of this term – without direct reference to Peirce, admittedly – in this passage: "Within the Infome paradigm, The dominating mode of the sign is not the symbolic, or the iconographic, but the indexical… The visualization is an indexical trace of the reality, an imprint". (NB "infome" is her neologism. Read, "*Internet*").

As Aarseth goes on to observe, the existence of several levels of signification is not confined to what he calls “cybertexts” (human machine collaborations) alone. He gives the example of a book being read aloud (two levels: graphic and sonic). However, with a cybertext "the relationship might be termed arbitrary, because the internal, coded level can only be fully experienced by way of the external, expressive level" (p. 40).

As he notes, it is possible to describe program and data "in their own right", but these are not "equivalents" to what goes on at the higher level. He makes a similar observation, with different terminology, as Fields (above) – that different coding can produce similar 'expressions' (computations).

Aarseth's preoccupation is to draw into doubt the project of the group around Andersen to apply a semiological analysis to computer signs without full regard to these arbitrarily related levels. This is not primarily my purpose. Nevertheless, what implications does this problem of arbitrarily related levels have for my theory of the text machine?

I said in Chapter 2 that a text machine might be completely simulated by a computer because both its rules and instructions and its material – the text – can be translated into the same language, that is, the computer’s binary alphabet. This is unlike other situations where the instructions and the materials are in completely different realms and cannot be unified in this way. I gave the example of a Lawrence Weiner instruction. It is not possible to computerise lathing, carpet, tins of paint and so on.

What are the consequences of both data (text in this case) and program (rules and instructions of the text machine) being written in the same binary alphabet? Again, this is to enquire more generally into what computers are.

The idea of a binary alphabet is one I take from Neils Ole Finnemann (1997, 1999). According to Finnemann this binary alphabet, consisting of two terms, may translate any other semantic and syntactic system. It is, furthermore, a notation system that by definition is content free and can be used, therefore,

to represent any other “formal expression, whether data or rules” (1997, p. 145). It – the alphabet – is itself content free by virtue of the computer being a universal, Turing machine. That is to say it must “be able to perform any rule or programme (sic).” If this were not so, it “would be deprived of its universality” (ibid. p. 144).

Finnemann explains that the storage of rules and data in the form of binary code means data and rules are expressed in a text form. This comprises for him the textualisation of sound and image media. This text may be accessed at any point and edited. This means that data and rule may be varied, or suspended, more or less freely. In Finnemann’s terms:

“There is one important aspect which will be of significance in all areas: a great number of the restrictions which were formerly connected with the physically bound architecture of the symbolic media are here transformed into facultative symbolic restrictions which are implemented in physically variable (energy-based) and serialized textual form” (ibid. p. 147).

The consequence for the text machine, along with all other machines, is that its rules and instructions are subject to change or suspension. Rules may be combined with other rules and for any period of time. Also, plainly, the implications of the integration of communications (radio, television, telephone etc), that Finnemann observes, cannot exempt the text machine when a computer simulates it. It must share a common situation. This is not to say that the text manipulations I have described must disappear and become video poetry, or multimedia. It does mean the boundaries between one machine and another are now arbitrary and may be subject to summary alteration.

The fact that rules may be suspended or altered has fundamental implications for the text machine as I have described it. In the past it was, of course, always *possible* to vary rule and data. However, simulation by computer means that easy alteration is possible by access to the program code – at least in theory. This logical possibility is delimited in practice by countervailing forces such commercial protection and the relative powerlessness of the

average user to make any significant changes to the operating system within which they work. Nevertheless, the *practical* bound that was previously imposed by physical media largely disappears. This is not to say rules and instructions disappear, but they are alterable invisibly, and no *stable* entity can therefore be constructed from them.

Rules, no longer imposed from without, guaranteeing stability, instead "are processed in time and space as part of and on a par with the ruled system, implying there are systems in which rules can be changed modified, suspended or ascribed new functions" (ibid. p.156). Finnemann calls this situation a rule generating system. He contrasts this with rule dependent systems, where the rule or rules are outside the rule-governed system. He sees in this a more general and important tendency that embraces philosophical and scientific discourses and constitutes, for him, an epoch making change.

Rules do not only produce writing but *are* writing. Like anything else written in the binary alphabet of the computer, they may be rewritten. That Finnemann calls the computer's binary code an "alphabet" is deliberate and important. It indicates that we must be aware of what its code is. Unlike the usual alphabet, the code has been defined as neutral or content free. It is that that allows the binary alphabet to simulate other sign systems. (It is for such reasons that Finnemann [1999] defines the computer as a "multi-semantic machine" [p. 358]. Some may find this controversial, as it might appear to attribute intelligence to the machine itself: that is, it might suggest the machine itself performs acts of semiosis. My reply would be that one does not require a thesis of artificial intelligence to accept the Finnemann thesis. That the machine may not have, and has no need of, a comprehension of the semiotic material contained in its alphabet once it is allocated content does not mean there is no semiotic material. It may mean that in the computer there is no semiosis. But this is not required. To ask anything different would be similar to requiring a book to be able to comprehend itself for the book to have meaning. The connection between the [human] reader's reading of the code and changes in the code structure of the machine is demonstrated by the fact

that semiotic events in the reader have code consequences for the computer, as code events are triggered by the reader's use of the machine. Of course, as observed above, it is not possible for the reader-user to experience these formalisms talked of above.

NB I have written, "reading reads writing". I did not say, "it understands it"[61]).

4. Programs and Performances

The text machine as I have described might be said to be "computational": it moves linear sequences of symbols linearly. It does not take great account of variation of the text in its other dimensions. This is partly because I am not writing a thesis about visual poetry. It is also because the text machine has been defined as not material-specific: it can itself be instantiated in various materials, and its inputs can be written similarly. Therefore, although it uses materials, it is independent of them.

This is reminiscent of Ferdinand de Saussure's[62] formulation of the immateriality of language generally. This is an immateriality that employs the material without ever depending upon that material. This is because language is a system of values not things.

Similarly, the text machine, in other words, is a *process not a thing*; and it is a process that will accept various inputs and produce various outputs (in this thesis *texts*, but also images and sound). Nor is it bound to a particular historical moment by definition. The process can be set in action at different

[61] See John R. Searle's (1980, not uncontroversial) 'Chinese Room experiment' for a discussion of how a machine – or a human – *may* process symbols without understanding them.

[62] "… it is impossible that sound, as a material element should in itself be part of the language. Sound is merely ancillary, a material the language uses. All conventional values have the characteristic of being distinct from the tangible element which serves as their vehicle. It is not the metal in a coin which determines its value..." (Saussure, 1983, pp. ll6-117).

times and places. But to run, it must have an input and must run at a specific time. This will be the event of its performance. This event has its time and materials. These aspects of the machine's work are a project this thesis does not undertake.

Performances by technical machines and humans are not identical. Mechanical tasks are often performed quicker by machine. This is frequently the reason for their creation and the prompting, for instance, behind Vannevar Bush's dream of the personal computer. Moreover, it is the fact that textual manipulations can be formed into rule-based instructions that allows for their mechanisation[63]. But a human may take a long time, perhaps an impractically long time, to carry out similar actions. (This can be incorporated into a performance, however, such as the reading of On Kawara's *1 Million Years* at Trafalgar Square, London 2004).

It is *possible* to imagine a human performing the recursive steps of something like the *Postmodernism Generator* – fetching words and following sentence structures according to a randomised process, but it would take a long time to make a text. And why bother if this can be done with little cost, at the click of a mouse? Surely it is only to make the theoretical point. Or perhaps when one is constructing such a series of instructions and is deciding the actions the program is to perform. Then there is the experience of working out specimen steps of a machine. Once it is done, there is little cause to follow these steps further. If the machine works, it works. This is why many text manipulations are transferred to computer, but not the other way around (programs turned into human action, even where it can be done).

An example of program 'downloaded' to three dimensions: the performances of '*dotwalk'* by 'socialfiction.org'*.*

This is one of their performance scripts/programs:

[63] "Whenever logical processes of thought are employed - that is, whenever thought for a time runs along an accepted groove - there is an opportunity for the machine." (Bush*, As We May Think*, 1945. No page numbering).

// Classic.walk

Repeat

{

1 st street left

2 nd street right

2 nd street left

} [64]

http://socialfiction.org/dotwalk/dummies.html

Nevertheless, there are few instances of a person taking the steps that may be performed by a computer program. This would mean, in effect, a lot of trouble to go slower. (I once considered turning *Every Icon* by J F Simon Jnr into a performance piece. *Every icon* moves squares on a 32 x 32 square grid. But the time involved was prohibitively great[65]). It is because of such differences that the different material instances of a machine are not wholly identical.

The physical conditions of the text machine's operation and display in many ways are effects of the specifications of other machines, other non-text

[64] "Algorithms are by no way limited to computer software, and computer (sic) - as devices which execute algorithms - are by no way limited to chip-based electronic hardware. La Monte Young's "Draw a straight line and follow it" is a plastic example of an algorithm which can be executed by any kind of being or hardware which thus acts as a "computer" executing the algorithm. ".walk" by socialfiction.org is based on this very concept of a, quote, "non-electric computer". Computations are executed not through electricity flowing through the transistor gates of a processor chip, but by walks through urban spaces". Cramer, http://www.runme.org/feature/read/+dot-walk/+31/

[65] "The grid contains all possible images. Any change in the starting conditions, such as the size of the grid or the color of the element, determines an entirely different set of possible images. When Every Icon begins, the image changes rapidly. Yet the progression of the elements across the grid seems to take longer and longer. How long until recognizable images appear? Try several hundred trillion years. The total number of black and white icons in a 32 X 32 grid is: 1.8×10^{308}(a billion is 10^{9}).

Though, for example, at a rate of 100 icons per second (on a typical desktop computer), it will take only 1.36 years to display all variations of the first line of the grid, the second line takes an exponentially longer 5.85 billion years to complete."
http://www.numeral.com/articles/paraicon/paraicon.html.

(technical) machines. The results of these combinations are multitudinous and not possible to prescribe, if only because we cannot foresee the course of technical development. Add to this the fact that we do not know the environment in which a machine-ensemble will be arranged, and we find we may know little in advance of its emergent possibilities.

A further issue is that we cannot predict how a situation will be read/perceived. It is not possible, following developments in cybernetic theory that date back to the middle of last century, to leave such questions out of account. Systems cannot be defined independent of the observer (system). According to second order cybernetics, systems interact and change in interaction.

Imagining a “literature machine”, Italo Calvino in 1967, writing during the period of second order cybernetics, phrases a similar idea eloquently,

> “Once we have dismantled and reassembled the process of literary composition, the decisive moment of literary life will be that of reading. In this sense, even though entrusted to machines literature will continue to be a “place” of privilege within human consciousness…The work will continue to be born, to be judged, to be destroyed or constantly renewed in contact with the eye of the reader” (1997, pp. 15-16).

It is not possible to define a text machine independently, as a fixed and stable set of rules and operations. This is true if only because different versions might be imaginable, but each with a claim to authenticity.

Having said this, I am not adopting a purely sceptical position. I have said that the text machine cannot be defined as a stable entity distinct from the observer's perception, *not* that it cannot be defined at all.

There is also the question of social context to consider. The text machine may be remade in its narrow functioning, but not in all the earlier circumstances of its making. A rough illustration: it is possible to use the program of Racter (it will be remembered, the program that ‘wrote’ *The Policeman's beard is half-constructed*) to make Racter-like texts (there are versions of the program

available on the web), but it is not possible again to claim, as does its preface, to be the computer's first book.

That a text machine may be made, that it is possible to specify its rules, instructions, codes and inscriptions, this for me is not the main problem. There is however the question of what is *not* included in a text machine's formulation. This, we are beginning to see, may be quite a lot.

This may be conceived as a confrontation between what may be coded and what may not be encoded. It may be posed as the relation of pattern to presence, in terms of Hayles's discussion of the posthuman and its debt to information theory.

This approach to text, the text treated as pattern not substance, is in fact inscribed in the first moments of information theory, in Shannon's (1948) *Mathematical Theory of Communication*. Shannon described a system for the efficient coding and transmissions of information. Shannon describes (more than a decade before the – 1960 – formation of the Oulipo, committed to the application of mathematical inspired procedure to literature) several methods of generating text according to stochastic procedures. These procedures are based on statistical analysis of word frequencies. Furthermore, they do not necessarily involve a computer, but can be performed with a text and a pen and paper[66] (see Appendix 4. This text appears on my web site at: http://www.in-vacua.com/markov_text.html).[67]

Nevertheless, whatever is a pattern of symbols should be possible to turn into computer code; and what is in a computer code may be performed by

[66] In any case, barely available in 1948. In *1950* Shannon's disciple J.R. Pierce (see Pierce, 1971, and Pierce, 1980) also made texts using similar methods. Again, these were not computerised. They betray a distinctly "poetic" touch, whatever the apparently mechanical methods of their devising. The phrase "electrons diffuse in vacua", which is quoted on the entry page of my web site, www.in-vacua.com, comes from this source (see Pierce, 1971, pp. 51-52). See "**Appendix: Evidence of Work 4**" for a longer comment on Shannon.

[67] Hodges, S. (2004) makes similar points. Writing of the same texts he says, "Shannon's technique for creating these approximations would not seem out of place in a book of Oulipian experiments" (p. 34). His thesis has a more extensive analysis of connections between the Oulipo and information theory then I undertake.

computer if it is formed into a satisfactory program. What we have in Shannon is essentially the ordering of one “linguistic articulation system” (Finnemann, 1999, p. 296) – that is, the *text* – by another, an *algorithm*. An algorithm is not exclusive to the computer. However, computers run them faster than humans. But to do this, the algorithm and the data it treats must be translated into its own code terms. How does this work?

Finnemann makes a distinction between these two types of system: “Where a sentence, however, produces a meaning”, he says, “the algorithmic procedure produces the transformation of one expression to another” (ibid. p. 300). Thus the former is to do with the creation of meaning, and the other (the algorithmic procedure) the rule-structured processing of a set of values into another set of values. What Finnemann does not say explicitly is**:** *It is possible to produce a sentence from an algorithm*.

That is to say, one regime may determine or produce another. The non-referential may produce the referential. *And*, once produced such a sentence may be indistinguishable from any other (as above, with the Markov algorithm’s[68] ‘Google’ sentence).

However, determination is not all in one direction. As Finnemann argues, the semantic regime determines what algorithmic regime we choose and what terms are fed into it: whether we choose to multiply the Eiffel tower by the sound of a thunderclap, as in Finnemann’s example, is not determined by the algorithm (which can receive many different values) itself. The input value is motivated from outside it. (However, the inputs may not originate with a human user; one machine may feed data to another machine).

[68] It’s a real algorithm: from Brian W. Kernighan and Rob Pike’s (1999) *'The Practice of Programming*.

The two levels are mutually determining. However, if we subject a text to an algorithmic process (which is the loss of referential meaning[69]), we have the strange event of something without reference producing something that has it. So we see that the complex and mutually determining semantic and algorithmic levels depend on the transposition of semantic material. This may be processed by computer in code form, translated back, and consumed again at a semantic (interpretive) level.

5. Conclusion

In this Chapter I have sought to develop an understanding of the importance of code elements in the making of a machine: that is, a text machine. These code elements become significant when another performs that machine: a digital computer. This machine uses a neutral code in which to code semiotic materials. However, it transpired, this neutrality was only technical. The transposing of semiotic material to code elements was found to have a number of important implications. However, much of the material and social aspects of the machine's functioning and construction cannot be coded in the same way. In short, these were found to not be programmable for computer.

Things *are* lost when a text machine is given over to a technical machine to perform. What else happens? What is gained? Speed (already mentioned) and size (scope) are gained. However, because the code and algorithm may produce a text, a sentence, who or whatever writes the code writes the words. The reader is disadvantaged here in circumstances where that reader has no writer's privileges.

[69] "…*the elimination of the expression's referent*" (ibid. p. 296, italics in the original), because an algorithm can take more or less any permissible value.

…get it out of your head and into the machines

William S. Burroughs

Chapter 6: A Typology of Text Machines

Introduction

Rather than present what could easily degenerate into a thematically and chronologically muddled survey of three years' work, I will try to organise my text machines according to a working typology.

1. **Substitution Machine**
2. **Manipulation Machine**
3. **Generative Machine**
4. **Other**

These four categories, I will explain, exist only ideally, in the sense that machines may not sit conveniently within one or other of them, but may have aspects of more than one category. There are also, it will be shown, several sub categories; so for instance, there are at least two dominant methods of text generation.

Nevertheless, I think that the typology is a useful way of organising what might easily seem a confusing sprawl of material.

Each system is capable of producing widely different genres of text. This point is important to my discussion. Text machines do not fit neatly and

conveniently into one genre alone: a technique for producing poetry may be reassigned to write prose also; the genre of prose, too, is a matter of preference.

Text machines that use a particular system are not necessarily poetry machines, art machines, novel machines, nor indeed, necessarily producers of corporate home pages and love letters, to cite examples Murray (op. cit). mentions. They may do these things, or any number of others.

The conclusion that must be drawn from this is that we cannot define a poetry machine or art machine on the basis of its text techniques alone. It is the uses to which these techniques are put that are relevant. If a machine is used for poetry, then it is a poetry machine: if it is used for greetings cards, then that is what it is, whatever the sophistications of any of its possible programming.

My typology of *text machines* is distinguished from typologies of *texts*. As I have remarked, there is generally more interest afforded the texts than the machines in the existing scholarship. (Aarseth, op. cit. p. 71, has a graph of paper and electronic *texts* that plots their 'cybernicity' according to a set of defined criteria). But apart from a few antecedent partial accounts that I have drawn upon (not least Murray's, below) there is no attempt to categorise text machines. To my knowledge, the organising schema I use here to discuss text machines is original to this thesis.

Because of everything I have said about the difficulty of working from text to machine or machine to text, for that matter, the application of my categories must be tentative. Where I am describing my own work, I am in a special position: I know the text and I know what made it. But the typology's use elsewhere must be qualified. How can we know the machine if we *only* know the text (as discussed in Chapter 4)? In practice, often we do know more than this; and where we do, the categories I advance may be useful.

1. Substitution Machine

Description of Machine

Murray (1997), in her discussion of computerised narrative, refers to what she calls a "substitution system" derived from Lord's work[70] on folk literature.

Murray observes "[e]arly attempts at computer-based literature tried to use similar methods of simple substitution" (p. 189). For her, a substitution system provides what computer programmers call the "primitives", the smallest elements, from which greater operations may be built. For Murray, phrases and sentences of such a system are the "morphemes" that must be built up into more complex entities that go together to make the larger units that may eventually make a narrative.

Murray notes a substitution system is not specific to computerised literature alone. This might seem to suggest that such a system may meet the demand of being operable across all three (limited function, simulated and abstract) formations of the text machine that were discussed in Chapter 2; that is, it is not medium-specific. I think this is persuasive; but it is not the only trans-media system.

Description of Work

My early work largely – but not entirely – fell within this group of Substitution Machines. I will list these works by title.

1. Programmer (2003)
2. Computerized Haiku (2003-2004)

[70] From Alfred Lord's book *The Singer of Tales*. A substitution system may be thought of as a stock of formulas into which may be substituted chosen elements. Lord discovered poets in the oral tradition used these formulas as an aid to composition.

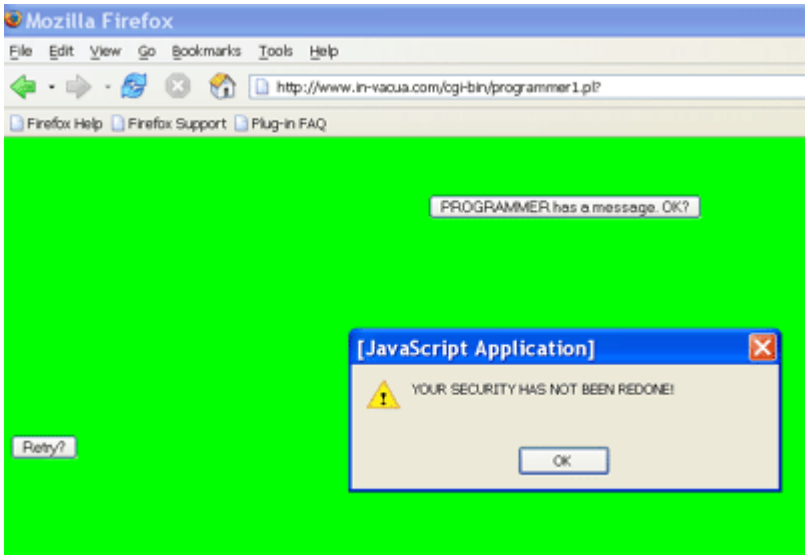
3. Sentences (2003)
4. High-Entropy Essays (2005)
5. Ono Generator (2004)

Programmer (see illustration) produced baffling statements. These appeared in what are called ‘alert buttons’. It used two codes: it was programmed in *Perl* and the Perl program wrote the JavaScript event handler that produced the frustrating button that had to be “Okayed”. It selected text on a randomised basis and dropped it into sentence templates.

It was my realisation that one code and program could write another one (effectively include the other within it) that prompted some of my reflections on code, particularly that codes could write other codes and that machines could exist within machines.

Programmer, unfortunately, was not very interesting to use and has been removed from public display.

Plate 9



Computerized Haiku http://www.in-vacua.com/cgi-bin/haiku.pl was a breakthrough for me. It was in fact a reprogramming of a piece of work that was exhibited by Margaret Masterman and Robin McKinnon Wood at the ICA gallery London, 1968. It is the subject the article and presentation that is Appendix 1.

The work was important to me as it showed, not only to others but also to me, that one might remake – or reverse-engineer – a work of art from a description of it: the original program is lost but there is an essay (Masterman, 1971) about it.

This was, I believe, my first piece to be displayed on the web. The work also now includes randomised and automated versions.

There is an archive of user haiku that is growing slowly. This archive was an attempt to replicate the public display at the original ICA show. Behind this also was a reaction to the failure of Keith Tyson's online/offline work *Replicator* (for lack of users). My conclusion was that, unlike *Replicator*, user participation should be entirely from the screen-keyboard and more or less immediate.

Sentences http://www.in-vacua.com/sentences.html. '*Sentences*' was based on a similar template structure to that of *Computerized Haiku*. It consists of a fixed template structure and lists of words. The user may select from the lists or run a random version that chooses for you: a Substitution Machine can write prose or poetry.

The structure of *Sentences* was derived loosely from Lawrence Weiner's And Yoko Ono's works: It had an injunction "an x to be y" (very much Ono) but also a past participle (the preferred mode of Weiner). Most of its vocabulary, however, was freely invented.

High-Entropy Essays http://www.in-vacua.com/cgi-bin/mendoza.pl. 'High-Entropy Essays' was originally shown in *Cybernetic Serendipity*. Professor E. Mendoza programmed it around 1960. It has always fascinated me – much neglected in the literature, but ahead of its time: it anticipates the far more famous *Postmodernism Generator*. I have used a flowchart published by Mendoza to make my version. However, my program writes essays that differ from Mendoza's. His too cannot be derived exactly from the chart. So I assume that some of the work is lost for good. It is for me an interesting case study of a partially successful (because of omissions in the records) attempt to reverse-engineer a lost work from instructions after the event.

Ono Generator http://www.in-vacua.com/cgi-bin/ono1.pl. This takes selections of text from Yoko Ono's (1995) *Instruction Paintings* and on a controlled randomised basis drops them into sentence templates shuffles the resulting lines, and selects randomly a number of these from one of two groups. I found parts of the Perl code in an obscure exchange on a programming list: the code that selects on a random basis between one and four lines of text to display.

Ono Generator gives the impression of being more sophisticated than it really is by a series of randomised choices that vary the number, choice and vocabulary of lines, and its use of a restricted vocabulary. Because of this, it appears to largely confine itself to one subject. This technique of hiding the text selection by adding random choices is in fact a fairly well visited one. It is probably what gives Racter its air of relative consistency (see John Barger's, *The Policeman's Beard was Largely Prefab!*).

2. Manipulation Machine

Description of Machine

I am not the first to make the distinction between text manipulation and text generation and other text processes: Hartman (1996) writes of that “other main approach to ‘computer poetry’: not generation but text manipulation” (p. 95).

Manipulation Machines, as I remarked above, subject a text input to some sort of process. There are many processes. The cut-up of Tristan Tzara, for example, comprises the instruction to cut-up a newspaper and to select the words at random. Many of the techniques of the Oulipo authors also constitute Manipulation Machines. The Oulipo were dedicated to applying expressly algorithmic procedures to texts. A well-known example is N +7, where a text is selected and for each noun another, seven away in a dictionary, is selected (“N” stands for noun). This produces some strange, sometimes striking, effects.

Combinatorial poetry, with its permutation of prepared texts, I also include in this category (my, *tzara combinations* is an example of a combinatorial approach). I suggest that the reader visit Cramer’s *Permutations*[71] website for an extensive treatment of the subject.

It is not possible to list all of the techniques. Many have been transcribed for computer and may often be found on the Internet. *Diatext* began as a non-computer procedure, invented by one person, the poet Jackson Mac Low and was programmed by another, Hartman, for computer. Hartman remarks, “Jackson had done all the work by hand. I sat down and embodied his rules in a little program” (op. cit.). It is now available in several versions.

[71] http://userpage.fu-berlin.de/~cantsin/permutations/index.cgi

Any text manipulation procedure, so long as it can be embodied in a clear procedure, I hazard to state, should be possible to program for computer in a comparable way. There is no *requirement* it should be programmed (this would be required only where the machine in question is devised and made purely as a computerised machine for processing electronic texts alone: such works cannot in practice *function* outside of a computerised and networked environment) and it does not to be the person who developed the rules to be the one who programs the work.

Description of Work

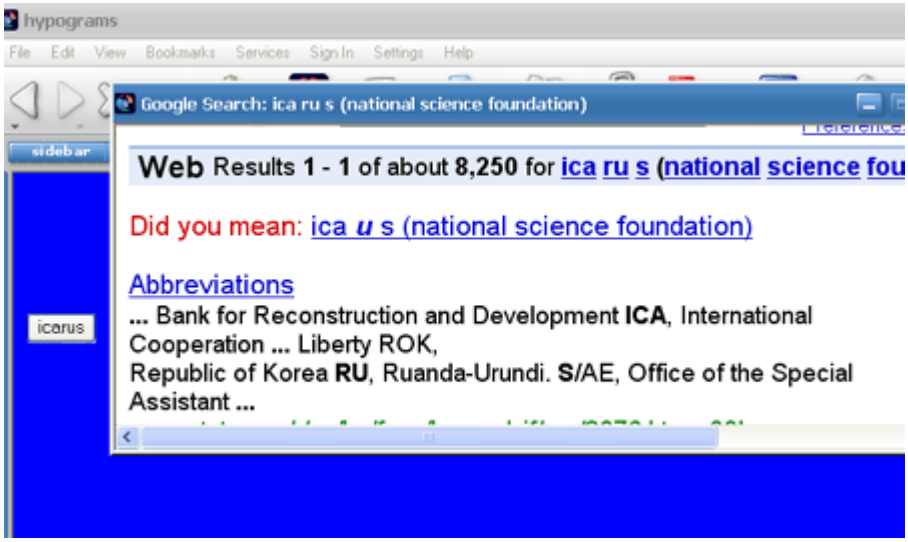
There is (it will be observed) a wide variety of work in this group. Manipulation of a text may take many forms.

1. Hypograms (2003)
2. Nike Splice (2004)
3. tzara Combinations (2004)
4. Noumena (2003-2004)
5. Alt_Img_Tate (2005)
6. Monochromes (2005)
7. Passwords (2004-2005)

Hypograms (not currently available on the web: see illustration) was in some ways a simple piece of work. I hesitate to call it programming. But in its *theorisation* it *was* more ostentatious. A hypogram is a word dispersed amongst other words in little groups of letters and is an idea that comes from Saussure (and is explained in Starobinski, 1979).

I used Google's search engine to look for these words on the web. When it found them it highlighted the search terms (see illustration) forming a word.

Plate 10



I consider the highlighting to be a sort of text manipulation.

Nike Splice is no longer running on the web. It took an input text entered by the user and a text from Nike's website and simply shuffled the words together.

The raison d'etre of the piece was a reference to William Burroughs who used similar text cut-up techniques (Burroughs also did an advert for the *Nike* company).

This piece used an algorithm – the "Shuffle" – to mix-up the text. The Fisher-Yates Shuffle[72] is a well-known algorithm, available in many computer languages and used widely whenever some randomness is required. I have used this shuffle in several other works to different ends (*tzara combinations*, *Ono Generator*, *Markov Generator*) and therefore it is worth describing here. It also throws light on my work process.

[72] See footnote 35, above.

I began with a wish to shuffle a text. (This is a different requirement to making a random selection, something I will also describe). When I hit a programming problem, my usual approach is to begin by 'Googling' search terms that relate to that problem.

After many attempts I first found the algorithm. Then I had to find a version in *Perl*. Like much of the code you may find in books, on the Internet, or in some other way (kindly programmers?) the version of the shuffle I found in January 2004 was not useable in the form it was in. I had to write the surrounding code to enable it to print as a block of text, rather than a list of words, and write all the 'CGI' (Common Gateway Interface, what makes code run on the Internet and allows programs to take user input).

With this accomplished, I had a code block that I could adapt to shuffle words as well as lines.

tzara combinations http://www.in-vacua.com/tzara.shtml.

This was my next use of the Fisher-Yates Shuffle. It takes Tzara's instructions for producing Dadaist poetry by cutting up a newspaper and shuffles this instruction itself. It shuffles lines not words. It was my hope to do something more sharply focussed than to cut-up online newspaper sources (there are several of these programs on the web).

It treats Tzara's text as the basis of a combinatorial poem, as if the order in which the lines appear is not important.

Noumena http://www.in-vacua.com/noumena.html is a complex piece.

a. What does it do? It deletes a web text leaving the punctuation.
b. How? It obtains a web page and processes it.
c. Why? It is a software version of *Reality* by Jarowslaw Kozlowski.

(There is information about *Noumena* at http://www.in-vacua.com/noumena_text.html. There is also a discussion in the Appendix to Chapter 2 of this thesis).

This was the first work I tried to program. At that time, I was not a programmer at all. I'd never previously attempted to program a computer. I decided *Perl* was to be my computer language at this time because of its strength in processing text. Using programming books and discussion lists where beginners help each other, I got to the stage of being able to take a plain text file and perform the selective deletion required. However, when I tried to do the same to a web page I found that *Perl* also processed the HTML formatting destroying the page's appearance: not the effect I wanted.

I had been in contact with a group called London Perl Mongers. This is a group largely of professional programmers. Many are extremely able in their field. Also they are helpful to beginners and may offer assistance to the novice.

As a consequence I was able to share my code sketch with Simon at www.hitherto.net. Instead of the few tips I had hoped for, he basically wrote all the code from scratch and posted it up in useable form on his website for a time.

I now understand what the code does and have even edited the program slightly to do more of what I require. However, the real credit goes to him for the programming. It uses several *Perl* modules (Modules: programs other programs use) that process HTML pages.

Noumena is curated by ***runme.org***, the 'software art' organisation, at: http://www.runme.org/project/+noumena/.

Art-Strike (Presently not available on the web. See Plate 11 below).

Art-Strike is similar to *Noumena* in conception and in its programming. It too is based on a bookwork, on this occasion *Five Bookpages* by Matthew Higgs. Higgs carefully struck through pages of popular fiction except sentences that referred to art.

Art-Strike is in two forms. One looks in online art sites for the letters 'art' and bolds and underlines them. It crosses out everything else. The second program takes a user entered URL and a string of characters and looks for them in the chosen web page, if found, it underlines, bolds and crosses out the rest.

It is the same in its programming as Noumena except the part that looks for patterns in web pages and alters the pages.

The way it does this is a little bit complicated. The program treats text and Html separately. It inserts tags to turn on the strike through function into the text: <s>. If it encounters a match with the term that was entered it turns off the strike-through, <\s>, and then turns it back on. When the user's browser displays the text, it looks like Html so it treats it like that, producing the selective crossings out.

Plate 11

It is slightly too unreliable and is awaiting more work.

...

Alt_Img_Tate *http://www.in-vacua.com/Alt_Tate.html*

Monochromes http://www.in-vacua.com/monochromes.html

These works have similarities that allow me to discuss them together. They each look through a list of web addresses. They look for HTML coding, find it and use the information obtained in their display.

To create a list of addresses I used some free software, *Xenu Link Sleuth* (http://home.snafu.de/tilman/xenulink.html). Starting with an address, Xenu compiles a list of links.

This list is stored in a file. The program looks through the list, selects an address at random, opens the web page at that address, and looks through the page.

At this point the two programs differ. *Alt_Img_Tate* looks for “alt” tags. These contain the text the user sees if graphics on the browser are turned off or graphics fail to load.

Again, a random choice of alt tags is made and this is displayed. Some more code (JavaScript) takes care of reloading the page (it is automated) and fading in the text. This code was found in separate blocks on the Internet and pieced together. The fade in function alludes to the Tate Gallery’s own style of graphics, with its fade effects.

Alt_Img_Tate also uses part of *HtmlImgAltTextExtract.pl* by Andrew Hardwick, http://duramecho.com, released under GNU Public Licence. Once again, I adapted parts of this program to produce the texts, as I required them.

The work was selected for inclusion in **FILE 2005, Electronic Language Festival**, Sao Paulo, Brazil (http://www.file.org.br/works_list_todos.php?sel=4.0&lang=en&works_category_display=1.2.6&ano=2005&range=N-Z).

Monochromes differs in that it has no fade in and does not extract alt tags. Instead it looks in a page for several other things: color (sic) tags, the web page title, and the web address. It uses a “pattern match” (programming term: ‘something that looks like x’) to get the address and colours, and a module – ‘HTML::Tree’, another Perl module, to get the web page. It repeats the process, with a new presentation every few seconds.

These two works essentially select and display text, parsing the HTML and using parts of it to make the display on screen.

Passwords http://www.in-vacua.com/. This has existed in two versions. Originally the headlines it displayed were databased; that is, kept in a file that had to be manually updated.

This was burdensome, and antiquated headlines tended to mar the effect. To get the program to go out and find headlines was one of the hardest programming tasks I have faced. All online news services have pages formatted in different ways. The program has to be tailored to the page.

Finally, I found some programming code that took headlines from www.MaximumEdge.com (an online news service). This I adapted to only grab the text of the headlines, choose one at random, and display it.

The user has to copy and send the text using a web form. The program compares this text with the one it has. If you get it right, it lets you in.

This program, like several others uses *Perl's* built-in random number function. It is possible to, for instance, number a list and ask Perl to choose one number at random: it will then show the text associated with the number in the list.

3. Generative Machine

Description of Machine

There are several main approaches: Recursive Transition Networks (RTNs) and Markov processes are the two I use. There is also Natural Language Generation. This is an important area in applied computer science and Artificial Intelligence, something that is outside the scope of my research. There are other methods, including hybrids of several techniques. These are the subjects of computer science research. I will concentrate on the two I am familiar with. Markov generators are probabilistic. There is an input text, a calculation of word sequences, and an output text. RTNs start at the other end, with a grammar. A grammar is a set of rules for the production of all

possible sentences for that grammar. The grammar is run and a text produced.

These are very different approaches. They are also complicated to describe. There are several texts in the bibliography for those interested in RTNs. For instance, Uneson, Hoftstadter and Bulhak.

There is an essay displayed on my web site that describes Markov processes and the general form of *Markov Generator* and *Webov*. It is at http://www.in-vacua.com/markov_text.html. It is Appendix 4 of this thesis.

I will confine myself here to a discussion of the work.

Description of Work

1. Markov Generator (2004)
2. Virtual Dictionary (2005)
3. Webov (2005)

Markov Generator http://www.in-vacua.com/markov_gen.html and ***Webov*** http://www.in-vacua.com/webov.html follow a now familiar pattern of my becoming interested in a problem and then seeking to find out how it could be investigated. For each of these works I use an algorithm by Kernighan and Pike (1999). This I have slightly adapted to print as a block of text not as a list of words. I suggest the reader consult the essay that is Appendix 4 as this contains many of my reflections on Markov chains.

Markov Generator takes part of the present thesis as an input text and generates a text from it. The page reloads with a new text after a randomly chosen number of second (arbitrarily set at no more than 60 seconds).

Webov. From a programming point of view *Webov* is a little more sophisticated. It allows the user to enter a web address. *Webov* then gets the web page at this address and processes only the text found there with the Markov algorithm and returns the result to the user.

There is (see Appendix 5) a body of text attached to this thesis that has been generated using the text of this thesis and the algorithm.

Virtual Dictionary http://www.in-vacua.com/home.html. I can take little credit for programming this piece. The program and the obscurity surrounding its author are discussed in Chapter 4.

What I have done is to write a series of grammar files that the program uses to generate texts on different key words to this thesis.

What I did was to take a *Perl* program included in the distribution of 'Parse::RecDescent' called *demo_textgen.pl*. This program is similar to *demo_randomsentence.pl* but a little more complex. There are two other not dissimilar programs by Schwartz [73]. My conclusion was I was unable to improve on their programming. What I did was to use some of the structure of the Schwartz grammar and adapt it for use with *demo_textgen.pl* (it needed some work).

This enables me to write fairly short grammar files and run them. More complex text generation requires a concert of programming like Bulhak's programming of the *Postmodernism Generator*.

A part of a grammar file looks like this:

```
statement : statement | statement2;
```

[73] *Creating an Inline Language* by Randal L. Schwartz (2004) http://www.linux-mag.com/2004-03/perl_01.html and *Writing Randomly* by Schwartz (1999) http://www.stonehenge.com/merlyn/LinuxMag/col04.html

```
statement2 : remark_start " or the " adjective " " noun  because;


remark_start: 2 @ (2 @ " 'Instruction'" | 2 @ " The instruction") "
is not " verb2 " the "  noun2 |
        "The Instruction is the " job " that is the " verb " of the
text machine";
```

The program basically runs through this file recursively (it may make several runs through the structure) to generate the text.

As I noted in Chapter 4, writing by *Virtual Dictionary* is now beginning to be included in online resources. It is hard to gauge if compilers of these resources are aware that the texts they include are machine made.

4. Other

I feel obliged to include this vague category because there is work that does not fit into any of the above.

Description of Work

1. src

(Included in the **Rencontre Festival** Paris 2005: http://www.art-action.org/site/en/prog/05/paris/prog_expo_03.htm).

src http://www.in-vacua.com/src1.html. It takes a search word from the user, finds a match if possible in an image databank and displays this image in tiled form on screen. It also uses JavaScript to scroll down the page and to renew the process automatically. Finally, it also displays a portion of the source code of the images superimposed over the images.

The use of a form to take a user entered value is a basic of CGI (common gateway interface): an elementary text machine. The display of image and code is a sophistication. The use of www.picsearch and its web tools to search a database comprises one machine using another machine: a common occurrence on the web.

With this work, in its substantial use of images, we are moving from pure text machines into new areas.

...

This is but an example. In practice many machines are not pure, but are combinations of machines or several machines linked together. I deal with this issue in the thesis.

…to keep everyone
deleting, substituting, inserting

Spencer Selby

Conclusion: Ouroboros[74]

The theory of a text machine is itself a text machine: it is a machine for the production of text machines, a meta-instruction for the production of instructions. This thesis is that machine. I say “a”, not “the” meta-instruction, because it is impossible for one instruction to produce all instructions, if only because it could not produce itself (a point I made at the start of the thesis).

A thesis that seriously claims the status of an instruction may be considered *machinic*. It *is* machinic, to use Deleuze and Guattari’s (2003) word: something that brings together heterogeneities[75]. Synthesising is certainly something this thesis can claim to have achieved. How this is valued is another matter perhaps. It is unusual to find the word ‘machinic’ employed as a compliment. So Alan Sondheim can say that he is “increasingly finding theory impoverished / machinic “[76]. Machinic in this usage means something like ‘mechanical’ in its more derogatory sense. But this is not correct at all. In the passage mentioned, it is by no means clear that Deleuze and Guattari use

[74] “The name *ouroboros* (or, in Latinized form, *uroborus*) is Greek and means ‘tail-devourer’” http://en.wikipedia.org/wiki/Ouroboros.

[75] “What we term machinic is precisely this synthesis of heterogeneities as such" (p. 330).

[76] Alan Sondheim in a recent review posted on the *Nettime* list (11th August 2005, *Reviews of some recent books and then some* -).

the term negatively[77]. This thesis is an instruction of instructions, a machine of machines, and proud to be so.

"*Always Follow the Instructions*…" proposes a text machine as a way of conceiving of the bringing together of instructions, rules, codes, and texts as working entity. It itself, of course, draws these elements together in the process of its discussion. In that regard, it reflects upon itself.

In Chapter 1 this synthesising capacity was suggested by the diversity of the contexts visited by this thesis. In Chapter 2, I distinguished the text machine from other machines. Chapter 3 was concerned with rules and instructions, while Chapter 4 was given over to the texts the machine may write. The fifth Chapter, about code was perhaps the most complex: the relation of the text to code *is* complex. Chapter 6 delineates a typology of text machines. This typology suggests how to approach an understanding of this complex subject.

These elements combined amount to an advance in theory: before this thesis there were phrases (literary/writing/language machines etc), but no working theory of the text machine. However, novelty is not its only or most significant claim. Weightier is the claim to theoretical superiority. This claim to superiority is based upon the usefulness of the hypothesis of a machine. The notion of a machine should enable us to *understand* better. Specifically, it helps us conceive of a machine as a recognisable entity, distinguishable from code language, text output, algorithms and hardware. All of these were found to be replaceable. The text machine is what is left when all these are subtracted: primarily the structure of its rules and instructions. These structures form the basis of the typology (above) that I put forward as a contribution to knowledge in this area.

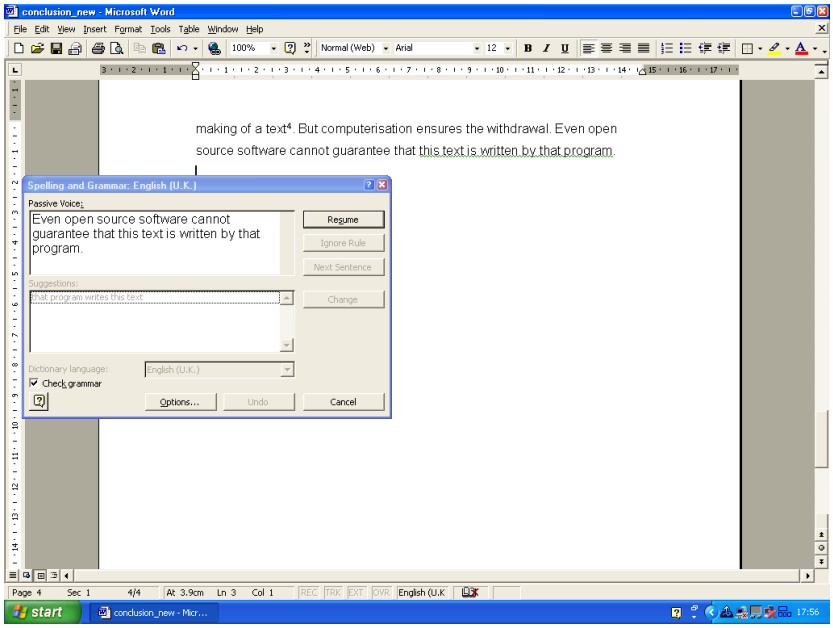
We need to understand the text machine if we are to understand the world we live in. It is not unusual to encounter such machines. I proposed in Chapter 4 that in fact the relatively marginal status of text machines (both on and offline)

[77] They use it to describe the formation, from a series of notes, characteristic bird songs.

in art and literature obscured the reality that in everyday life these machines are in motion. They seem to be – by and large – diligent robots, gatekeepers and passport checkers, traffic controllers, and the like, ordering our experience in space and time. However, they may seem to desire to promote their own importance, to subject us to their disciplinary regime, as many of us will be convinced after an encounter.

If it is the computerisation of text machines that has accelerated their development, it is program and code, a level inaccessible to users, that transforms these machines into effective means of social, financial and other regulation. The invisible determines the visible. The inaccessible determines the accessible. Those who write the machine write the rules. Those who don't have little option other than to follow them, or to exit. This is a relatively harmless business when it comes to art. Month by month[78] my web statistics tell me that the number one exit page for my website is the 'index' page with its password dialogue: a strong if anecdotal indication that many will avoid such interactions without coercion or greater inducements. However, the situation may be altogether more serious outside of such innocent simulations.

Plate 12



[78] 25.68% in July 2005.

Other text machines are protected by copyright and law and secret encryption: they may write to you, they may desire to rewrite what you have written, but you may not rewrite *them*. It is their code that affords them the status of law.

It is the coexistence of two levels in the computer – the legible text and the unseen program – that transforms the text machine from its origins as artwork to organising social principal. Considering literary production, of course it was always possible for writers to hold back about the mechanics behind the making of a text[79]. But computerisation can ensure this withdrawal.

The claim to theoretical superiority must be that the theory of the text machine as described may form the basis of comprehending this significant phenomenon. That is to claim that when you encounter a situation in which one of these machines writes, it will be possible to understand the sort of machine it is, what it seems to be doing and how equal or unequal our position is in relation to it. This thesis makes this possible.

This thesis also enables the comprehension of works of art and their relation to non-works4 of art in the mundane world that have similarities to them. This thesis, having replied to the question it set itself ("what is the impact of the computer on the text machine?") has, improved the understanding of these works, but also of any artwork that shares its conditions.

[79] See Raymond Roussel (1995), *How I Wrote Certain Of My Books,* where he explains some compositional strategies.

Ah! machinist, take great care of me

Aristophanes

Future Research

I have noted, what critical commentary there is on text machines does not really engage with the *machine* as compared with the text. Where there is a little attention paid, there is a strong orientation to hypertext and computer poetry, perhaps because these genres are relatively well established. Artwork that does not stick conveniently to pre-established forms seems to get less contemplation. In either case, the machine itself is scantly considered. I think this is a serious omission and one that requires correction

Two research projects open out from this. One is historical, and the beginnings of which appear for instance in Appendix 1. Much work is still to be done in this area. Very little serious historical work exists at present. The situation is all the more pressing as many of those who worked on early computerised text have, like Masterman and McKinnon-Wood, passed on in fairly recent years. (This situation prompted the forming of Birkbeck College's CACHe project[80]).

The second and perhaps more pressing issue is also hardly researched: this is the spread of text machines in everyday life. There is little critical work on this, although as I have pointed out, artists have quickly moved to appropriate many of the dialogue formats and other phenomena associated with their

[80] See http://www.bbk.ac.uk/hosted/cache/History.html "The CACHe project was founded to rescue a pioneering branch of British art from unjustified obscurity", following the death of pioneer John Lansdown.

workings. Theory and practice are running at different speeds. What commentary there is has focused, of course, more on the writings than the machine that writes. This, as noted, is the usual circumstance and in part represents a textual prejudice, one that places textual artefacts above the mechanical. This a position that I feel cannot continue if we are to get to understand these sorts of texts, not merely as a literary or artistic genre but as social practice of human and machine. It is something that I suggest is set to change as interest moves from the literary endeavours of the past to new forms of electronic writing based on new forms of data structure.

The two projects are in fact related. It is not possible to know the bureaucratic text machine without the poetry machine, the novel writing machine without the military machine. This is because their origins are not completely separate, nor are their methods and workings, as I have made clear above.

The industrialisation of the text machine and its relation to the literary and artistic is a project begun in this thesis, and in the practice that accompanies it at www.in-vacua.com. This website includes some artwork that is an archaeology of the text machine. It also offers a perspective on how it is possible to develop an engagement with text on the Internet, its modes of interaction, of display, its appearance, its structures, its *machines*. This project remains to be developed through closer analysis of text machines in action and through work that investigates that functioning. My future research looks forward to these possibilities both in my practice and theoretical work.

Bibliography

Aarseth, E.J. (1997*) Cybertext. Perspectives on Ergodic Literature*. Baltimore and London, The Johns Hopkins University Press.

Adorno, T. and Horkheimer M. (1999) *Dialectic of Enlightenment*, London, Verso.

Adorno, T. (1997) *Aesthetic Theory*. London and New York, Continuum.

Alberro, A. (1999) 'reconsidering conceptual art, 1966-1971', in *conceptual art: a critical anthology.* (Eds.) Alberro, A. and Stimson, B. Cambridge Massachusetts, MIT, xvi-xxxsviii.

Albert, S. *review: John Thomson and Allison Craighead in Tate Britain*, Cream 1, <http://www.nettime.org/nettime/archive/200104/msg00027.html> (4th April 2001), 1-2.

Albert, S. *Artware*, - *Art, Software and Conceptualism* <http://twentiethcentury.com/saul/artware.htm> (28th November 2002).

Altshuler, B. *Art by Instruction and the Pre-History of do it* <http://www.e-flux.com/projects/do_it/notes/essaye002_text.html (19th May 2003).

Andersen, P. B. (1994) 'Introduction', in Andersen, P. B. et al *The computer as medium*. New York, Cambridge University Press, 9-16.

Andersen, P. B. (1994) 'A semiotic approach to programming', in Andersen, P.B. et al, *The computer as medium*. New York, Cambridge University Press, 16-68.

Armstrong, D. M. (1989*) Universals: an opinionated introduction*. Colorado and London, Westview Press.

Arns, I. (2001) *Texts That Move (Themselves): Notes on the Performativity of Programming Codes in Net Art* (Abstract) <http://amor.rz.hu-berlin.de/~h2863i74/abstracts-e.html> (11th August 2005).

Art and Language (Eds.) (1969) 'Introduction', in *conceptual art: a critical anthology*, (Eds.). Alberro, A. and Stimson, B. (1999), Cambridge Mass. MIT, 98-106.

Asselin, O. (1992) 'The Sublime: The of Limits of Vision and Inflation of Commentary', in *Theory Rules*, (Eds.) Berland, J. et al. Toronto, YYZ & University of Toronto Press, 243-265.

Atkinson, T. (1968) 'Concerning the Article "The Dematerialization of Art"', in *conceptual art: a critical anthology*, (Eds.) Alberro, A. and Stimson, B. (1999), Cambridge Mass. MIT, 52-58.

Austin, J. L. (1978) *How To Do Things With Words*. Oxford, OUP.

Bailey, R. W. (1973) *Computer Poems*. Michigan, Protogannissing Press.

Bailey, R. W. (1974) 'Computer-assisted poetry: the writing machine is for everybody' in, *Computers and the humanities*, Mitchell, J.L. (Ed.) Edinburgh University Press, 288-296.

Barger, J. *"The Policeman's Beard" Was Largely Prefab!* <www.robotwisdom.com/ai/racterfaq.html> (18th November 2003).

Barnet, B. *Technical Machines and Evolution,* <www.ctheory.net/text_file?pick=414> (1st May 2004).

Barthes, R. (1967) *Elements of semiology*, trans. Lavers, A. and Smith, C. London, Jonathan Cape.

Barthes, R. (1977) “The death of the author”, in *Art: Context and Value*, (Ed.) Simm, S. (1992). Oxford, Oxford University Press, 60-64.

Baudrillard, J. (1993) Symbolic Exchange and Death, trans. Grant, I. H., London, Sage.

Baumgartel, T. (2001) *[net.art 2.0] New Materials towards Net.art*. Nurnberg, Verlag.

Beech, D. *Another Tyson Ear Bending: Dave Beech Talks to Keith Tyson*, <http//www.backspace.org/everything/e/hard/texts2Tyson.html> (27th February 2002).

Beiles, S. (1960) *Minutes to go*. Paris, Two City Editions.

Berry, J. *Net Art.* Unpublished PhD Thesis. University of Manchester. <http://www.tate.org.uk/netart/humanposthuman.htm> (28th November 2003).

Blackmore, S. (1999) *The Meme Machine*, Oxford, Oxford University Press

Bloor, D. C. (1997) *Wittgenstein, Rules and Institutions*. London, Routledge.

Bök, C. *The Piecemeal Bard Is Deconstructed: Notes Toward a Potential Robopoetics* <http://www.ubu.com/papers/object/03_bok.pdf>

Bolter, J. D. and Grusin, R. (2000) *Remediation*. Cambridge Mass., MIT.

Bolter, J. D. (2001) *writing space: computers, hypertext, and the remediation of print*, Mahwah, N.J., Lawrence Erlbaum Associates.

Borges, J. L. (1999) *The Total Library: Non-Fiction 1922-1986*. Edited by Weinberger, E. Trans. A., Levine, S. J. and Weinberger, E. Harmondsworth, Middlesex, Penguin.

Brandt, P. A. (1994) 'Meaning and machine: Toward a semiotics of interaction', in Andersen, P. B. et al *The computer as medium*. New York, Cambridge University Press, 128-141.

Buchloh, B. H. D. (1989) 'Conceptual Art 1962-1969: From the Aesthetic of Administration to the Critique of Institutions', in *conceptual art: a critical anthology,* (Eds.) Alberro, A. and Stimson, B. (1999) Cambridge Massachusetts, MIT, 514-537.

Bulhak, A. C. (1996) *On the Simulation of Postmodernism and Mental Debility using Recursive Transition Networks*. <http://www.csse.monash.edu.au/cgi-bin/pub_search?104+1996+bulhak+Postmodernism> (2nd August 2004).

Burnham, J. (1968) 'Systems Esthetics', in *Great Western Salt Works, essays on the meaning of Post-Formalist art* (1974). New York, George Braziller Inc.

Burnham, J. (1970) 'Notes on Art and Information Processing', in *Software Information Technology: Its New Meaning for Art*. New York: Jewish Museum

Burroughs, W. (2001) *The Ticket That Exploded*. London, Harper Collins.

Bury, S. (1995) *Artists' Books. The book as a work of art, 1963-1995*. Aldershot, Scholar Press.

Bush, V. *As We May Think* (1945) <http://www.csi.uottawa.ca/~dduchier/misc/vbush/awmt.html> (20th November 2004).

Calvino, I. (1997) 'Cybernetics and Ghosts', in *The Literature Machine*. London, Vintage, 3-28.

Chomsky, N. (1969) *Syntactic Structures*. Cambridge Mass. MIT.

Chomsky, N. (2002) *New Horizons in the Study of Language and Mind*. Cambridge, CUP.

Christiansen, T. and Torkington, N. (2003) *Perl Cookbook*. O'Reilly, Sebastopol, CA.

Conway, D. (1998) *The man (1) of descent* <http://search.cpan.org/src/DCONWAY/Parse-RecDescent-1.94/tutorial/tutorial.html> (20th August 2005).

Cook, S. (2004) *The Search for a Third Way of Curating New Media Art: Balancing Content and Context In and Out of the Institution*. Unpublished Ph.D. thesis, University of Sunderland.

Copeland, J. and Haemer, J. (2001) *Work: Nonsense* <http://alumnus.caltech.edu/~copeland/work/junktext.html> (12th August 2004)

Corby, T (2001) *The disappearing frame: a practice-based investigation into the composition of virtual environment artworks*. Unpublished PhD thesis, University of the Arts, London.

Cornwell, R. (1993) 'From the Analytical Engine to Lady Ada's Art', in *Iterations: The New Image*, (Ed.) Druckery, T. Cambridge M.A., M.I.T, 41-61.

Cox, G. *MPhil/PhD Transfer Report, anti-thesis: the dialectics of generative art (as praxis)* <http://www.anti-thesis.net/> (20th September 2004).

Cramer, F. (2000) *COMBINATORY POETRY AND LITERATURE IN THE INTERNET* <http://www.userpage.fuberlin.de/~cantsin/homepage/writings/net_literature/per> (19th October 2000).

Cramer, F. (2001a) *Digital Code and Literary Text*, <http://www.userpage.fu-berlin.de/~cantsin/hom…t_2001/digital_code_and_literary_text.txt> (27th September 2001).

Cramer, F and Gabriel, U. 'Software Art and writing', *American Book Review,* vol. 22, number 6, September/October 2001, p8.

Cramer, F. (2002a) *Concepts, Notations, Software, Art*. <http://www.macros-center.ru/read_me/teb2e.htm (7th October 2002).

Cramer, F. *Digital Code and Literary Text*, <http://userpage.fu-berlin.de~cantsin/hom…t_2001/digital_code_and_literary_text.txt > (27th February 2002).

Cramer, F (2003) .*walk* <http://www.runme.org/feature/read/+dot-walk/+31/> (2nd April 2005).

Cramer, F. (2003) 'Exe.cut[up]able statements: The Insistence of Code', in (Eds) Stocker, G. and Schöpf, C. *Code : code - the language of our time,* Ostfildern, Hatje Cantz.

Cramer, F. (2005) *WORDS MADE FLESH. Code, Culture, Imagination*. Rotterdam, Piet Zwart Institute. <http://pzwart.wdka.hro.nl/mdr/research/fcramer/wordsmadeflesh/wordsmadefleshpdf> (28th July 2005).

Crystal, D. (2001) *Language and the Internet*. Cambridge, Cambridge University Press.

Dale et al, *Using Natural Language Generation Techniques to Produce Virtual Documents* <http://www.ics.mq.edu.au/~rdale/publications/papers/1998/adcs98.pdf.> (27th August 2004).

Dawkins, R. (1982) Extended phenotype : the gene as the unit of selection, San Francisco, Freeman.

Deleuze, G. and Guattari, F. (1994) *Anti-Oedipus*. C*apitalism and Schizophrenia,* trans Hurley et al*.* London, The Athelone Press.

Deleuze, G. (1995) 'Postscript on Control Societies' in, *negotiations 1972-1990*, trans. Joughin, M. New York, Columbia University Press: 177-183.

Deleuze, G and Guattari, F. (2003) *A thousand plateaus; capitalism and schizophrenia*, trans. Massumi, B. London, Continum.

Derrida, J. (1976) *Of Grammatology*, trans. Chakravorty, G. Baltimore, The John Hopkins University Press.

Derrida, J. (1979) 'Living On: Border Lines', in *Deconstruction and Criticism*, Bloom, H. et al (Eds.), New York, Continuum.

Derrida, J. 'ECONOMIMESIS', *DIACRITICS*. Volume 11, 1981: 3-25.

Derrida, J. (1982) 'The Pit and the Pyramid: Introduction to Hegel's Semiology', in *Margins of Philosophy*, trans. Bass, A. Brighton, Harvester Press, 69-109.

Derrida, J. (1978) 'Structure, sign and play in the discourse of the human sciences', in *Art: Context and Value*, (Ed.) Simm, S. (1992). Oxford, Oxford University Press, 402-417.

Duchamp, M. (1989) *The writings of Marcel Duchamp.* (Eds.) Sanouillet, M. and Peterson, P. New York, Da Capo.

Duve, de T. (1999) *Kant after Duchamp*. Cambridge Mass. MIT.

D´ýaz-Agudo, B. et al, (2002) *Poetry Generation in COLIBRI* <http://gaia.sip.ucm.es/people/pedro/papers/2002_eccbr_belen.pdf> (12th May 2005).

Eco, U. (1989) *The Open Work*. Cambridge Mass. Harvard University Press.

Eco, U. (2001) *Foucault's Pendulum*, London, Vintage.

Ferrara, P. *TEAnO, an Organization for the Application of Computers to Art Production* <http://people.etnoteam.it/maiocchi/teano/works/wordtemp/sorbona.do> (22nd December 2003).

Festa, P. (2003) *Spam-bot tests flunk the blind* <http://news.com.com/2100-1032-1022814.html > (25th May 2005).

Fields, C. (2002) 'Measurement and Computational Description' in, *Machines and Thought. The legacy of Alan Turing. Volume. 1.* Oxford, Oxford University Press, 165-179.

Finnemann, N.O. (1997) 'Modernity Modernised – The Cultural Impact of Computerisation', in Mayer, P.A. (Ed.) *Computer Media and Communication. A Reader*. Oxford, OUP, 144-160.

Finnemann, N. O. (1999) *Thought, Sign and Machine – the Idea of the Computer Reconsidered*, trans Puckering G. L. <http://www.au.dk/ckulturf/pages/publications/nof/tsm/abstract.html> (July 6th 2003).

Fuller, M. (2003) *Behind The Blip. Essays On The Culture Of Software*. New York, Autonomedia.

Funkhouser, C. POETRY DIGITAL MEDIA AND CYBERTEXT *<http://web.njit.edu/~cfunk/SP/hypertext/POETRYDIGITALMEDIACYBERTEXT2.doc> (10th July 2004).*

Galloway, A. *Protocol or How Control Exists after Decentralization*. Rethinking Marxism, Number13, Volume 3/4 (Fall/Winter 2001) <http://openflows.org/~auskadi/protocol.pdf> (10th February 2004).

Galloway, A. R. /RSG (2002) 'How We Made Our Own "Carnivore"' in (Eds.) Stocker, G. and Schopf, C. *Unplugged. Art as the Scene of Global Conflicts. Ars Electronica 2002*, Ostfildern-Ruit Germany, Hatje Cantz Publishers, 350-355.

Galloway, A. R. (2004) *Protocol. How Control Exists after Decentralization*. Cambridge Mass. MIT.

Gere, C. (2002) *Digital Culture*. London, Reaktion Books.

Gervás, P (2002) "Exploring Quantitative Evaluations of the Creativity of Automatic Poets, <http://calisto.sip.ucm.es/people/pablo/papers/GervasECAIws2002.pdf> (14th April 2005).

Glazier, P. L. (2002) *Digital Poetics: The Making of E-poetries (Modern and Contemporary Poetics)*. Tuscaloosa, University of Alabama Press.

Goodman, N. (1969) *Languages of Art. An Approach To A Theory Of Symbols*. London, Oxford University Press.

Goriunova, O. and Shulgin, A. *Artistic Software for Dummies and, by the way, Thoughts About the New World Order*. <http//:www.macros-center.ruread_me/teb1e.htm> (27th July 2003).

Graham, B. (1997) A Study *of Audience Relationships with Interactive Computer-based Visual Artworks in Gallery Settings, through Observation, Art Practice, and Curation.* Unpublished Ph.D. thesis, University of Sunderland.

Gray, C. and Malins, J. (1993) *Research Procedures/Methodology for Artists & Designers*. < http://www2.rug.ac.uk/criad/cgpapers/epgad/epgad.html> (November 12th 2003).

Greenberg, C. 'Modernist Painting' in, *The Collected Essays and Criticism. Volume 4.Modernism with a Vengeance, 1957-1969*. (Ed.) O'Brian, J. Chicago and London, University of Chicago Press: 85-94

Grusin, R. (1996) 'What is an Electronic Author? Theory and the Technological Fallacy', in *Virtual Realities and Their Discontents*. (Ed.) Markley, R. Baltimore and London, The Johns Hopkins University Press: 39-55.

Hansen, M. B. N. (2004) *New Philosophy for New Media.* Cambridge Mass, MIT.

Hardt, M. and Negri A. (2000) *Empire.* Cambridge Mass. Harvard University Press.

Hatherley, A. (1972) '*the art of letting things happen, a letter to sylvester houedard*', ceolfrith 15. ceolfrith arts centre, 41-42.

Harel, D. (1988) *Algorithmics: the spirit of computing*. Wokingham, Addison Wesley.

Hartman, C. O. (1996) *Virtual Muse: Experiments in Computer Poetry*. University Press of New England, Hanover NH.

Hayles, N. K. (1994) 'Chance Operations: Cagean Paradox and Contemporary Science'. *John Cage - Composed in America.* (Eds.) Perloff, M. and Junkerman, C. Chicago, Chicago University Press: 226-241.

Hayles, N. K. (1999) *How we became posthuman: virtual bodies in cybernetics, and informatics*. London, University of Chicago Press.

Hayles, N. K. (1999.b) ‘THE CONDITION OF VIRTUALITY ’ in, *The Digital Dialectic. New Essays on New Media*. (Ed.) Lunenfield, P. Cambridge Mass. MIT, 68-96.

Hayles, N. K. (2002) *Writing Machines*. Cambridge and London, MIT.

Hayles, N. K. *What Cybertext Theory Can’t Do* <http://www.electronicbookreview.com/v3/servlet/ebr?command =view_essay&essay_id=haylesonerip> (20th April 2004).

Higgs, M. (1996) *Five Bookpages*. London, Frith Street Gallery.

Hillis, W. D.l (1999) *The Pattern on the Stone. The simple ideas that make computers work*. New York, Perseus Books.

Hodges, A. (1983) *Alan Turing: The Enigma of Intelligence*. Unwin Paperbacks, London.

Hodges, S. (2004) *REVEALING CODE: WHAT CAN LANGUAGE TEACH SOFTWARE?* Unpublished M.Sc thesis. Georgia Institute of Technology.

Hoftstadter, Douglas R. (1999) *Godel, Escher, Bach: An Eternal Golden Braid*. Harmondsworth, Middlesex, Penguin.

Ingarden, R. (1986) *The Work of Music and the Problem of Its Identity*, trans. Czerniawski, A., edited by Harrell, J.G., Berkeley, University of California Press.

Jakobson, R. (1990) *On Language*, (Eds.) Waugh, L.R. and Monville-Burston, M. Cambridge Mass., Harvard University Press.

Jameson, F. (1999) ‘Marxism and Dualism in Deleuze’, in *A Deleuzian Century?* (Ed.) Buchanan, I. Durham and London, Duke University Press: 13-37.

Jevbratt, L. (2001a) *Perl is My Medium--An Interview with Lisa Jevbratt* <http://rhizome.org/thread.rhiz?thread=1696&text=2218#2218> (15th February 2005).

Jevbratt, L. (2001b) *Coding the Infome, Writing Abstract Reality* < http://dichtung-digital.com/2003/issue/3/Jevbratt.htm> (15th February 2005).

Joyce, M. (1995) Of two minds: hypertext pedagogy and poetics. Ann Arbor, University of Michigan Press.

Joyce, M. (1999) *Afternoon, a Story*. Watertown, Eastgate Systems Inc.

Kafka, F. (1992) ‘In the Penal Colony’ in, *Kafka: The complete short stories*. Minerva, London, 140-168.

Kant, I. (2001) *Critique of the Power of Judgement*. Cambridge, CUP.

Kaprow, A. (1993) *Essays on the blurring of life and art*. (Ed.) Kelley, J. London, University of California Press.

Kendrick, M. ‘Cyberspace and the Technological Real’, in *Virtual Realities and Their Discontents*. (Ed.) Markley, R. Baltimore and London, The Johns Hopkins University Press, 143-161.

Kernighan, B. W and Pike, R. (1999) *The Practice of Programming*. Reading, MA, Addison-Wesley.

Ketner, K. L. ‘Peirce and Turing: Comparisons and conjectures’, *Semiotica*, 68-1/2, 1988, 33-61.

Kittler, F.A. (1990) *Discourse networks 1800/1900,* trans. Metteer, M, and Cullens, Stanford California, Stanford University Press.

Kittler, F.A. (1999*) Gramophone, Film, Typewriter.* Trans. Winthrop-Young, G. and Wutz, M. Standford, California, Standford University Press.

Kozlowski, J. (1972) *REALITY*. ZPAP, Posnan.

Knuth, D.E. (1975) *The Art of Computer Programming, Volume 1, Fundamental Algorithms*. London, Addison-Wesley.

Krauss, R. 'Reinventing the Medium', *Critical Inquiry*, 25 (Winter 1999), 289-305.

Krauss, R. (1999) *A Voyage on the North Sea*. *Art in the Age of the Post-Medium Condition*, London, Thames and Hudson.

Kripke, S. (1982) Wittgenstein on rules and private language. Oxford, Blackwell.

Lambert, N. (2003) *A critical examination of computer art: its history and application*. Unpublished D.Phil thesis, Oxford University.

Landow, G. P. (1996) 'We Are Already Beyond the Book', in, *Beyond the Book. Theory, Culture and the Politics of Cyberspace*. (Eds.) Chernaik, W. et al. Oxford, Office of Humanities Communication, 23-31.

Landow, G. P. (1997) *Hypertext 2.0*. Baltimore, Johns Hopkins University Press.

Lavoie, B. and Rambow, O. (1997) *A Fast and Portable Realizer for Text Generation Systems* <http://www.cogentex.com/papers/realpro-anlp97.pdf> (13th August 2004).

Leibniz, G. (1951) ‘Towards a Universal Characteristic’, in *Selections,* (Ed.) Weiner, Philip, P. New York, Charles Scribners Sons, 17-26.

Lessig, L. (1999) *Code and other laws of cyberspace*, New York, Basic Books.

Libeskind, D. (1991) ‘Three Lessons in Architecture’, in *countersign*. London, Academy Editions.

Lippard, L. and Chandler, J. (1968) ‘The Dematerialization of Art’, in *conceptual art: a critical anthology.* (Eds.) Alberro, A. and Stimson, B. (1999). Cambridge Mass. MIT, 46-52.

Lister, M. et al, (2003) *New Media: a critical introduction*. London, Routledge.

Lunenfeld, P. (2000) *Snap to grid: a user's guide to digital arts, media, and cultures*. Cambridge, Mass. MIT.

Lunenfeld, P. (1999) (Ed.) *The Digital Dialectic: essays on new media.* Cambridge Mass., MIT.

Luntley, M. (2003) *Wittgenstein: Meaning and Judgment*. Oxford, Blackwell Publishing.

Lutz, T. (1959) *Stochastic texts* <http://www.stuttgarter-schule.de/lutz_schule_en.htm> (10th April 2005).

Macleod, K. *The Functions of the Written Text in Practice Based PhD Submissions*. <http://www.herts. ac.uk/artdes1/research/papers/wpades/vol1/macleod2.html> (20th September 2004)

Manovich, L. (1999) ‘WHAT IS DIGITAL CINEMA?’ in, *The Digital Dialectic. New Essays on New Media*. (Ed.) Lunenfield, P. Cambridge Mass. MIT, 172-194.

Manovich, L. (2001) *The Language of New Media*. Cambridge Mass, MIT.

Manovich, L. *Don't Call it Art*, <http://channels.mur.at/muratnews/1064432189/index_html> (28th November 2003).

Manurung, H. M. (2003) *An evolutionary algorithmic approach to poetry generation.* Unpublished PhD. University of Edinburgh.

Markley, R. (1996) 'Introduction: History, Theory, and Virtual Reality', in *Virtual Realities and Their Discontents*. (Ed.) Markley, R. Baltimore and London, The Johns Hopkins University Press, 1-11.

Masten, J. et al (Eds.) (1997) *Language machines: technologies of literary and cultural production*. New York, Routledge.

Masterman, M. (1971) 'Computerized haiku', in *Cybernetics, art and ideas,* (Ed.) Reichardt, J. London, Studio Vista, 175-184.

Mathews, H. and Brotchie, A. (1998) *Oulipo Compendium*. London, Atlas Press.

Mencia, M. (2004) *Visual Poetry to Digital Art: Image-Sound-Text, Convergent Media and the Development of New Media Languages*. Unpublished PhD thesis, University of the Arts, London.

Mendoza, E. (1968) 'Computer texts or high-entropy essays', in *Cybernetic serendipity: the computer and the arts,* (Ed.) Reichardt, J. London, Studio Vista.

Mendoza, E. (1973) 'Computer, B.S.c. (failed)' in, *A random walk in science*. (Eds.) Weber, R.L. and Mendoza, E. The Institute of Physics, London and Bristol, 145-146.

Merrell, F. (1997) *Peirce, Signs, and Meaning*. Toronto, University of Toronto Press.

Microsoft, *Volume Licensing Overview* <http://www.microsoft.com/licensing/resources/default.mspx> (2nd May 2005).

Millan, N (2001) *COMPUTER GENERATED POETRY AND VISUAL ARTS*. Unpublished M. Sc. thesis, University of Birmingham. <http://www.cs.bham.ac.uk/~nxm/mscPoetry/survey/CGPoetry.htm#_Toc525619646> (17th July 2005).

Montfort, N. *Cybertext Killed the Hypertext Star* <http://www. Electronicbookreview.com/v3/servlet/ebr?command=view_essay&essay_id> (20th April 2004).

MSN, *About typing characters from a picture* <https://help.msn.com/!data/en_us/data/passportv31.its51/$content$/PP_TROU_REG_TypeCharactersFromAPictureToSignUp.htm> (20th April, 2005).

Mumford, L. (1967) *The myth of the machine: technics and human development*. London, Secker & Warburg

Murray, J. H. (1997) *Hamlet on the Holodeck. The Future of Narrative in Cyberspace*. Cambridge Mass. MIT.

Musker, D. C. (1998) *REVERSE ENGINEERING*, paper presented at "Protecting & Exploiting Intellectual Property in Electronics", IBC Conferences, 10th June 1998 <http://www.jenkins-ip.com/serv/serv_6.htm> (29th August 2004).

Naughton, J . (2000) *A Brief History of the Future. The Origins of the Internet*. London, Weidenfeld and Nicholson.

Ono, Y. (1995) *Instruction Paintings,* New York, Weatherhill, Inc.

Osborne, P. ‘The Reproach of Abstraction’ in, *Radical Philosophy, a journal of socialist and feminist philosophy*, 127, September/October 2004: 21-29.

Parrish, K. *How We Became Automatic Poetry Generators: It Was The Best Of Times, It Was The Blurst Of Times*. < http://www.ubu.com/papers/ object/07_parrish.pdf> (19th August 2005).

Perec, G. (1996*) Life A User’s Manual*. Trans. Bellos, D. London, The Harvill Press.

Peirce, C.S. (1989) *Collected Papers of Charles Sanders Peirce. Volume 4. The Simplest Mathemetics*. (Eds.) Hartshorne, C. and Weiss, P. Thoemmes Press, Bristol.

Peirce, C.S. (1991) ‘Minute Logic’ in, *Peirce on signs : writings on semiotic*. (Ed.) James Hoopes. Chapel Hill, University of North Carolina Press, 231-241.

Pierce, J. R. (1971) ‘A chance for art’, in *Cybernetics, art and ideas,* (Ed.) Reichardt, J. London, Studio Vista, 46-57.

Pierce, J. R. (1980) *An Introduction to Information Theory. Symbols, Signs and Noise*. New York, Dover Publications, Inc.

Potter, K. (2000) *Four Musical Minimalists*. Cambridge, Cambridge University Press.

Racter (1984) The *Policeman's beard is half-constructed*. Illustrations by Joan Hall. Introduction by William Chamberlain. New York, Warner Software/Warner Books.

Rettberg, S. (2003) *Destination Unknown: Experiments in the Network Novel.* Unpublished PhD Thesis, University of Cincinnati. <http://huco.ualberta.ca/~sah4/FIS3005/fis3005_dissertations.doc> (20th February 2004).

Roussel, R. (1995) *How I Wrote Certain Of My Books And Other Writings By Raymond Roussel*. (Ed.) Winkfield, T. Cambridge MA, Exact Change.

Saussure, F. de (1983) *Course in General Linguistics,* trans. Harris, R., London, Duckworth.

Schmitz, U. (1994) *Rezension: Automatic generation of texts without using cognitive models: television news* < http://www.linse.uni-essen.de/webEdition/we_cmd.php?we_cmd%5B0%5D=show&we_cmd%5B1%5D=1052&we_cmd%5B4%5D=136> (19th August 2004).

Scholder, A. and Crandall, J. (Eds.) (2001) *INTERACTION. ARTISTIC PRACTICE IN THE NETWORK*. New York, D.A.P./Distributed Publishers.

Schwartz, R. L. *Writing Randomly*. Linux Magazine Column 04, Sept 1999. <http://www.stonehenge.com/merlyn/LinuxMag/col04.html> (4th August 2004).

Schwartz, R. L (2004) *Creating an Inline Language* <http://www.linux-mag.com/2004-03/perl_01.html> (4th August 2004).

Seaman, W.C. (1999) *Recombinant Poetics: Emergent Meaning as Examined and Explored Within a Specific Generative Virtual Environment*. Unpublished PhD thesis, Centre for Advanced Inquiry in the Interactive Arts, University of Wales, Newport.

Searle, J. R. 'Minds, Brains, and Programs', *The Behavioural and Brain Sciences*, vol. 3. Cambridge University Press, 1980.

Simon, J.F. Jr., *Every Icon - Parachute Text* <http://www.numeral.com/articles/paraicon/paraicon.html> (2nd August 2004).

Simanowski, R. (2003) *REVIEW on Nicolas Clauss*, <http://www-users.cs.york.ac.uk/~salfiore/aesthetic%20of%20interaction%20papers/dichtung-digital%ffParis%20Connection.htm> (11th August 2005).

Shannon, C. E. (1948) 'A Mathematical Theory of Communication', in *The Bell System Technical Journal*, Vol. 27, July, October 1948, 379-423, 623-656.

Sloman, A (2002) 'Beyond Turing Equivalence' in, *Machines and Thought. The legacy of Alan Turing. Volume. 1. (*Eds.) Millican, P. and Clark, A Oxford, Oxford University Press: 179-221.

socialfiction.org, *THE TECHNOLOGY WILL FIND USES FOR THE STREET ON IT'S OWN* <http://socialfiction.org/dotwalk/dummies.html > (1st April 2005).

Sohm, H (1970) *Happenings & Fluxus*. Köln, Kölnischer Kunstverein.

Sokal, A .D. (1996) *Transgressing the Boundaries: Towards a Transformative Hermeneutics of Quantum Gravity* <http://www.physics.nyu.edu/faculty/sokal/transgress_v2/transgress_v2_singlefile.html> (20th August 2004).

Sokal, A .D. (1996) *Transgressing the Boundaries: An* Afterword, <http://www.physics.nyu.edu/faculty/sokal/afterword_v1a/afterword_v1a_singlefile.html> (20th August 2004).

Sondheim, A. *Reviews of some recent books and then some -* <http://www.nettime.org/Lists-Archives/nettime-l-0508/msg00017.html> (11th August 2005).

Stallabrass, J. (2003) *Internet art: the online clash of culture and commerce*. London, Tate Publications.

Starobinski, J. (1979) *Words upon words: the anagrams of Ferdinand de Saussure*, London, Yale University Press.

Stocker, G. and Schöpf, C. (2003) *Code : code - the language of our time, code = law, code = art, code = life.* Ostfildern-Ruit , Hatje Cantz.

Swift, J. (1963) *Gulliver's Travels*. New York, Airmont Books.

Thomson, J. and Craighead, A. (2005) *READ_ ME* <http://www.computerfinearts.com/collection/thomson_craighead/beacon/beacon/rtf> (September 29th 2005).

Touretzky, D*. Basics of Information Theory*. <www-.cs.cmu.edu/~dst/Tutorials/Info-Theory/>, (5th January 2004).

Turing, A. (1950) *Computing Machinery and Intelligence* <http://www.iemar.tuwien.ac.at/html/lectures/272045/ComputingMachineryAndIntellignence_Turing.pdf> (9th February 2003).

Turing, A. (1936) *ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM* <http://www.abelard.org/turpap2/tp2-ie.asp> (9th February 2003).

Turner, J. and Hocking, D. 'Synergy in art and language: positioning the language specialist in contemporary fine art study'. *Art Design & Communication in Higher Education*. Volume 3, Number 3, 2004: 149-162.

Twain, M. (1906) *THE FIRST WRITING-MACHINES* <http://www.gutenberg.org/files/142/142.txt> 12th April 2005.

Tyson, K. *explanation* <http://adaweb.walkerart.org/influx/tyson/explanation.html> (23rd June 2004).

Uneson, M. (2003) *HORACE – an artificial columnist* <http://www.cs.lth.se/Education/LTH/EDA170/Reports2003/marcus.pdf> (1st August 2004).

Von Neumann, J. (2000) *The Computer and the Brain*. London, Yale University Press.

W3C*, Rule Language Standardization, Report from the W3C Workshop on Rule Languages for Interoperability* <http://www.w3.org/2004/12/rules-ws/report/> (May 1st 2005).

Wall, J. (1991) *Dan Graham's Kammerspiel*. London, Art Metropole.

Wall, L. (2000) *Programming Perl.* Sebastopol, CA. O'Reilly

Ward, A. and Cox, C. *How I Drew One of My Pictures: or, The Authorship of Generative Art,* <www.generative.net/papers/authorship> (20th October 2002).

Ward, A. et al, *Live Algorithm Programming and a Temporary Organisation for its Promotion* <http://art.runme.org/1107861145-2780-0/livecoding.pdf> (5th January 2005).

Weizenbaum, J. (1976*) Computer power and human reason*. San Francisco, W. H. Freeman and Company.

Whitby, B. 'The Turing Test: AI's Biggest Blind Alley' in, *Machines and Thought. The legacy of Alan Turing. Volume 1*. *(*Eds) Millican, P. and Clark, A. Oxford, Oxford University Press: 53-63.

Williams, C.F.J. (1989) *What Is Identity?* Oxford, Clarendon Press.

Wilson, S. (2002*) Information arts: intersections of art, science, and technology*. Cambridge Massachusetts, MIT.

Wittgenstein, L. (2001) *Philosophical Investigations*. Trans. Anscombe, G. E. M. Oxford, Blackwell.

Wolff, M*. Reading Potential: The Oulipo and the Meaning of Algorithms* <http://mustard.tapor.uvic.ca/cocoon/ach_abstracts/xq/pdf.xq?id=45> (21st August).

Yuill, S. *CODE ART BRUTALISM: LOW-LEVEL SYSTEMS AND SIMPLE PROGRAMS* <http://art.runme.org/1107798902-7563-0/yuill.pdf> (1st August 2005).

Zimmerman, M. E. (1990) *Heidegger's Confrontation with Modernity. Technology, Politics and Art*. Bloomington and Indianapolis, Indiana University Press.

Zizek, S. (1997) *The Plague of Fantasies*. London, Verso.

Zizek, S. (2004) *Organs without Bodies. On Deleuze and Consequences*. Routledge, New York and London.

Selected Websites

Automated Beacon:
http://www.computerfinearts.com/collection/thomson_craighead/beacon/index.html

Computer Generated Writing:
http://www.evolutionzone.com/kulturezone/c-g.writing/index_body.html

Cosign:
http://www.cosignconference.org/

CPAN: Comprehensive Perl Archive Network:
http://www.cpan.org/

DO IT at e-flux:
http://www.e-flux.com/projects/do_it/homepage/do_it_home.html

THE INJUNCTION GENERATOR:
http://ipnic.org/

The Institute of Infinitely Small Things, *100 (11) Instruction Works*:
http://www.ikatun.com/100-11/

Permutations:
http://userpage.fu-berlin.de/~cantsin/permutations/index.cgi

Poetry Links –Tools –:
http://www.eskimo.com/~rstarr/poormfa/poemtool.html

The Postmodernism Generator: Communications From Elsewhere:
http://www.elsewhere.org/cgi-bin/postmodern/

The Random Sentence Generator:
http://www-cs-faculty.stanford.edu/~zelenski/rsg/

Replicators
http://adaweb.walkerart.org/influx/tyson/

Rhizome:
http://www.rhizome.org

runme.org:
http://www.runme.org

The Status Project:
http://status.irational.org/

TEAnO:
http://people.etnoteam.it/maiocchi/teano/home.htm

Appendix**: Evidence of Work 1**

(The text of a presentation at CHArt, the computers in art history group, http://www.chart.ac.uk/, Birkbeck University, London, 2005, and published by CHArt).

Computer Poetry's Neglected Debut

"Cyberpoetry has not been attacked. It has never been very real, and never enough unreal. Nothing has been accomplished, though variations against the normative patterns have been made, perhaps with too small a price. Cyberpoetry, as it is, will produce no martyrs, only house guests."

Stefans, B. K. (2003) *Fashionable Noise. On Digital Poetics*. p. 45.

1.

I am particularly grateful to Jasia Reichardt, the curator of *Cybernetic Serendipity*, for her advice and assistance.

I examined the archive at the Tate Gallery's (London) *Research Centre*. This archive contains files of material from the ICA Gallery (where *Cybernetic Serendipity* was shown in 1968.) I wish to thank their staff for their help.

I am grateful to Professor Brent MacGregor (Edinburgh College of Art) who has granted me permission to use two images (from the original ICA show) of COMPUTERIZED HAIKU in his possession.

Attempts were made, without success, to contact the Cambridge Language Research Unit where Margaret Masterman and Robin McKinnon-Wood, the

creators of COMPUTERIZED HAIKU, held senior posts. However, the Unit is no longer active according to a Charity Commission report.[81]

2.

In a recent paper presented at CHArt's 2002 conference Lanfranco Aceti[82] (quoting Jon Ippolito, curator of the Virtual Projects and Internet Art Commissions at the Guggenheim Museum in New York) spoke about "the need to preserve behaviours rather then media". Aceti appears to oppose this curatorial venture[83]. But whether desirable or not, what is it to preserve behaviour? Is behaviour separable from media?

For me today, this is to ask why reprogram COMPUTERIZED HAIKU? In what sense can we say we preserve COMPUTERIZED HAIKU by its programming? After all, little remains of COMPUTERIZED HAIKU, neither the hardware (the computer) nor its original program. What does remain is an essay written by one of its creators, Margaret Masterman (1971). In this essay there is enough – a template for a verse structure and lists of words to fill it – to sponsor the making of a *version* of the work. I have, in other words, written a program that will produce similar verses to the original.

But why do this? COMPUTERIZED HAIKU, precisely *because* the program was missing, was for me to be the opportunity to conduct a demonstration. My recent research[84] has been into instructions. Masterman's essay, I realised, could be turned from description to instruction. It could be translated from a human readable account back to a machine executable program.

---

[81] "The charity has no plans for future research and subject to finding a suitable home for the research archives it is intended to wind up the charity", <http://www.charity-commission.gov.uk/ investigations/inquiryreports/> (22nd October 2004).

[82] Aceti (2002)

[83] Ibid. "The preservation of behaviours in the artists' practice seems to be the main concern in contemporary digital art practice, where the presence of 'software corporate powers' are imposing a methodology upon art practice."

[84] At Chelsea College of Art and Design, London, UK.

COMPUTERIZED HAIKU I saw as an artwork that might be thought of as a sort of *computation* in Alan Turing's sense of the word: a pencil and paper instruction that might be performed by a human 'computer' (that is, someone who 'computes') – *or* as a program executed by a machine.

This *might* be an artwork that is the preservation of behaviours, not the conservation of things. That is what is preserved, but what is lost in this process?

What are lost are the historical and material circumstances that attended the appearance of COMPUTERIZED HAIKU. It is these that I wish to attend to now, pointing up differences between the original version and my remaking of the work as I progress.

3.

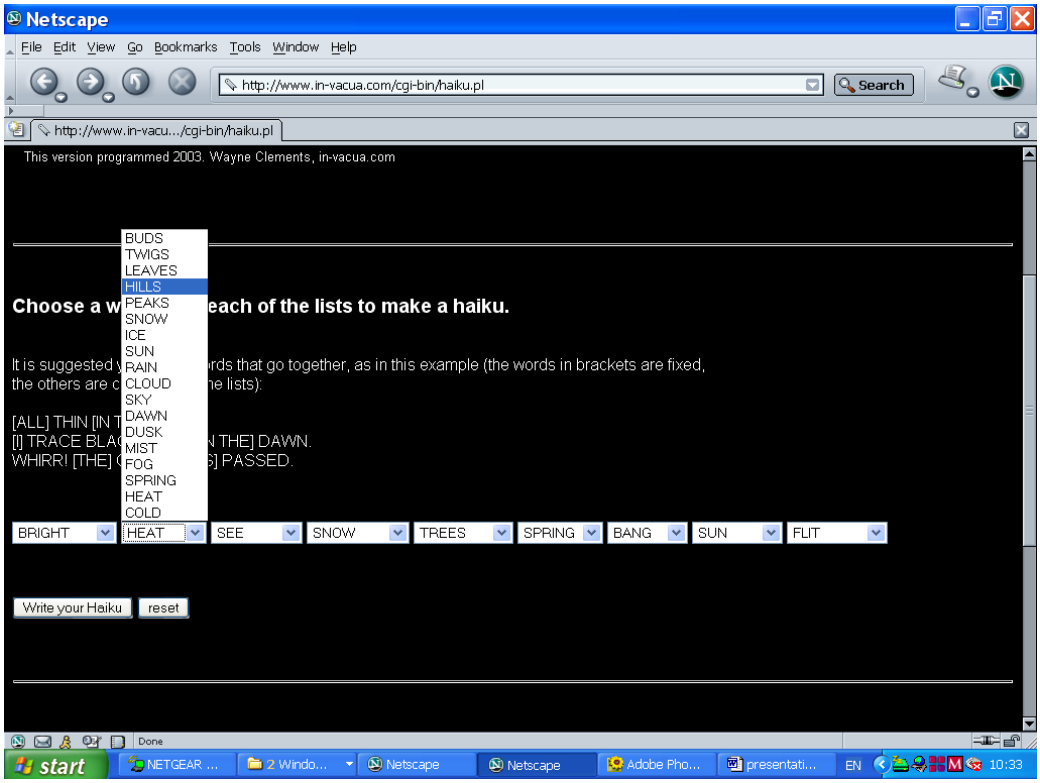
To return to COMPUTERIZED HAIKU is to return to the early days not only of computerised art and literature but also of computing and the still relatively new science of cybernetics. Cybernetic Serendipity was the first major exhibition of computer art (although there had been several earlier exhibitions of computer graphics.) Cybernetic Serendipity was unusual in many ways. Scientists mixed with artists and no rigid distinction was made between visual art and literature[85].

In those days everything must have seemed possible and most things still to be done. Looking back from our vantage point, it is possible to observe how much is different – and what may seem the same.

[85] There is a list of *Addresses of Major Contributors To Cybernetic Serendipity* in the Tate archive. The contributors of text pieces, including Masterman and McKinnon-Wood, are listed under "graphics" (the other categories are "music", "film" and "machines".)

If we look at my recreation of COMPUTERIZED HAIKU (fig 1) and compare it with images from the original show (figs 2 and 3) we may note some of the differences.

Plate 1



Screen grab of http://www.in-vacua.com/cgi-bin/haiku.pl

In 1968, the date of its public exhibition, there was, for instance, no Internet, as we know it, there were no personal computers, no html with which to script web pages; and programs with which to manipulate natural languages such as English with relative ease, were only just becoming available. In 1968, computers had to be installed and accessed on site, monitors were not available and output was to paper printer[86].

[86] I owe this information to Jasia Reichardt, the curator of the show. Personal communication.

Plate 2

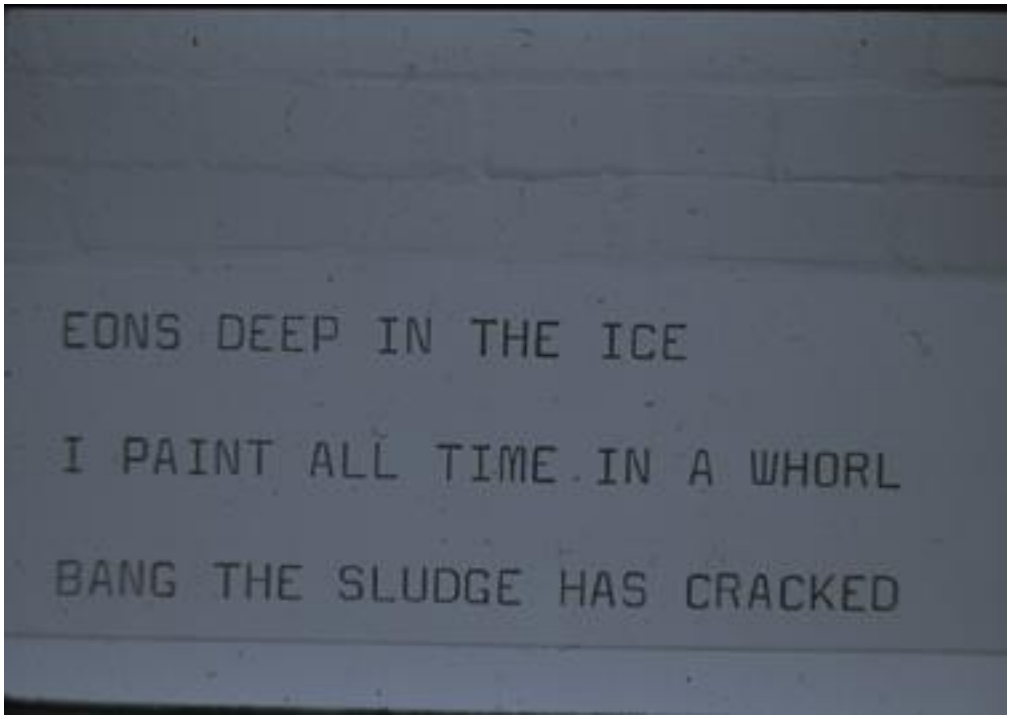


Image of installation at Cybernetic Serendipity: photograph courtesy of Professor Brent MacGregor.[87]

Plate 3

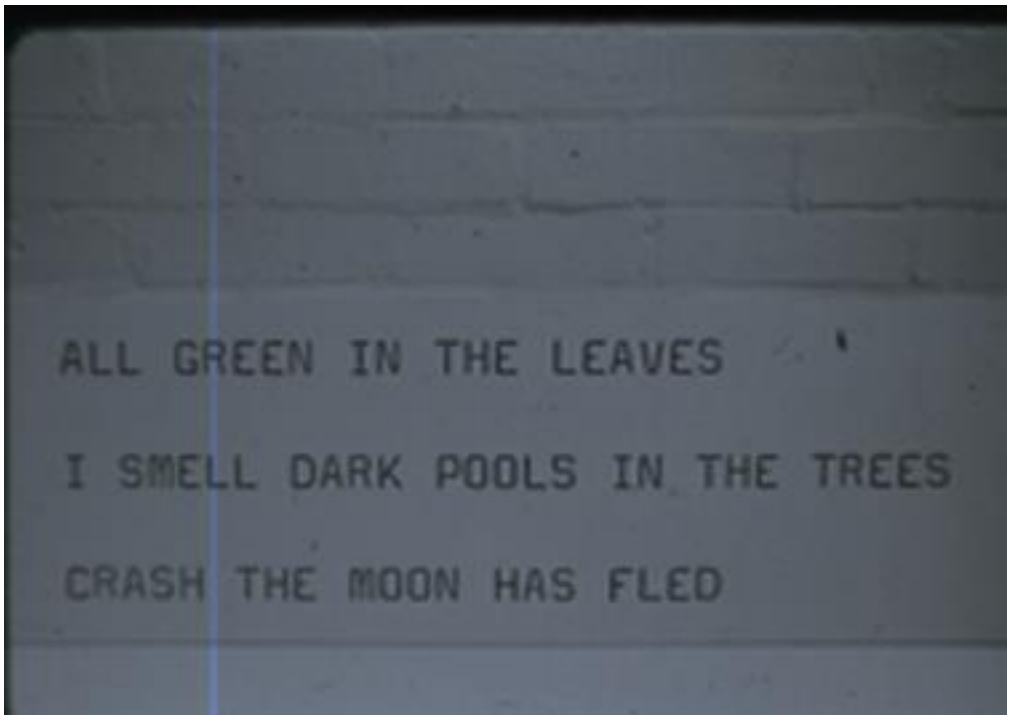


Image of installation at Cybernetic Serendipity: photograph courtesy of Professor Brent MacGregor.

[87] The two images of the haiku displayed at Cybernetic Serendipity seem to show poems hand copied on to paper and pinned to the wall. Their historical interest outweighs their slightly poor image quality.

Because of the word processor we are now quite used to computers handling text. In 1968 this was not so. In 1968 computerised literature was not quite a decade old. In 1959 – quite separately – there were two initiatives – Theo Lutz, on the one hand and Brion Gysin on the other (with Ian Summerville, a Cambridge mathematician) produced what may be the earliest examples of computerised literature.

That both Lutz and Summerville were scientists is significant. So is the algorithmic basis of each of their works. Access to computers was limited for those of a more purely artistic or literary background. (Lutz's work used a random number sequence to treat a text by Kafka, whilst Gysin's was a permutation of all the combinations of the words of the phrase I AM THAT I AM; we will see this overtly mathematical option was refused by the programmers of COMPUTERIZED HAIKU, Margaret Masterman and Robin McKinnon-Wood; rather they permitted the user to work directly with the program.)

Masterman and McKinnon-Wood were part of a brilliant generation of Cambridge scholars that came to prominence after the Second World War. Their interests were wide, and between them, embraced scientific, literary and philosophical concerns, and much else besides.

It is important to place COMPUTERIZED HAIKU in the context of a wider exploration of both cybernetics and natural language computing. Both of these inform the making of COMPUTERIZED HAIKU.

As I have mentioned, Margaret Masterman and Robert McKinnon-Wood were part of the Cambridge Language Research Unit. The Unit was involved in the development of automatic translation techniques for natural languages. Both Masterman and McKinnon-Wood published articles on the subject. The techniques behind translation programs would come into use in programming COMPUTERIZED HAIKU, as we shall see[88].

[88] McKinnon-Wood (1971) discusses some of these issues.

Thus COMPUTERIZED HAIKU cannot be viewed in isolation. It was one of several programs that responded to user input with which McKinnon-Wood was involved. One of these was *SAKI*, developed by McKinnon-Wood with his colleague Professor Gordon Pask. Another is *Musicolour* (also shown at the ICA), a light display that interacted with music. SAKI began as a program to train punch card operators, and later, typists. The program assessed performance and adapted to improve the operator's accuracy and speed. It is the ancestor of contemporary programs to teach typing.

Thus COMPUTERIZED HAIKU must be seen in the context of a sustained exploration of human-machine interaction, that forms continuity from practical application through to more purely literary endeavours. Several scientific and technical strands come together here: what were then recent developments in computer hardware, new programming languages and developments in cybernetic theory.

What crucially enabled the realisation of COMPUTERIZED HAIKU was the availability of a computer language that facilitated the relatively easy use of a computer to write a text. That language was TRAC. TRAC stands for "Text Reckoning And Compiling".

It is important to note TRAC's significance. Calvin Mooers designed TRAC in 1964. TRAC, *"was designed specifically to handle unstructured text in an interactive mode, i.e., by a person typing directly into a computer*." (Sammet, 2004). As such it marked a significant advance in the computer's usability.

Of course TRAC, and the 'interactive keyboard' as Mooers called it, do not *cause* the appearance of computerised literature. There was a persistent interest in increasing both the ease and scope of computer use and this had continued throughout the 1950's, and of course carries on today.

Computerised literature, therefore, is a complex development where technical improvements interplay with other determining elements.

However, to enable a computer to assist in the writing of poetry was a considerable goal of some cyberneticists. Poetry is in some ways a peculiarly high status art form[89]. It is perhaps this high status that has attracted computer researchers to poetry.

The contribution that TRAC makes is that it is possible to construct a poem as you go along. This is the aim of Masterman and McKinnon-Wood's original work. To explain this will have to describe COMPUTERIZED HAIKU in more detail. This brief discussion draws upon Masterman's (op. cit.) essay.

COMPUTERIZED HAIKU comprises a 'Frame' or 'Template' and a 'Structured Thesaurus'. The Template is the fixed form of the poem. It looks like this:

All . . .(1). . . IN THE . . . (2). . . ,

I . . .(3) . . . . . .(4). . . . . .(5). . . IN THE . . .(6). . .

. . .(7). . .! THE . . .(8). . . HAS . . .(9). . .

The operator of the poem to is expected to make a selection for each numbered gap in the frame from the structured thesaurus, which consists of numbered lists of words, to produce a poem like this:

[89] See, for instance, Derrida's (1981) discussion of its pre-eminence in Kant's hierarchy of the arts. "The summit of the highest of the speaking arts is poetry" (p. 18), says Derrida of Kant on poetry.

ALL BLACK IN THE MIST,

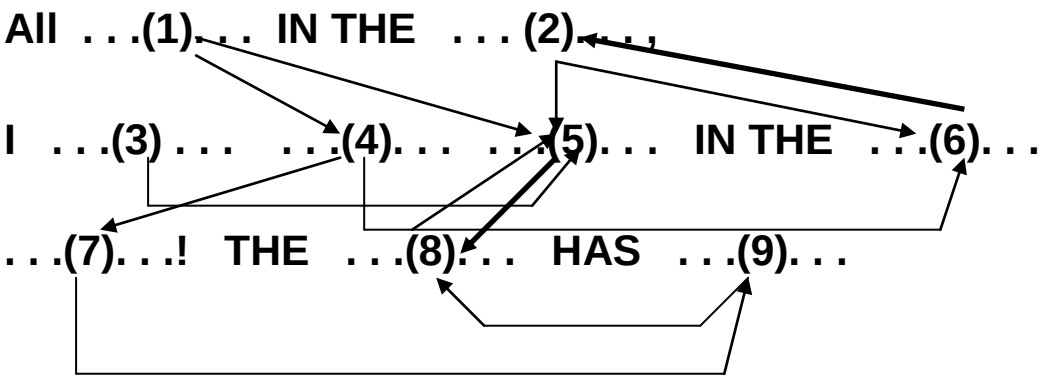
I TRACE THIN BIRDS IN THE DAWN

WHIRR! THE CRANE HAS PASSED.

This is sometimes referred to as a ‘slot’ system or a ‘substitution’ system. It is not the only method of computerised writing. There are also generative methods, using Markov chains or recursive grammars. These produce more complex, less predictable texts. There are also various techniques for shuffling and cutting up texts. However, the use of substitution systems is still popular[90], as is the haiku form, particularly on the Web, where you can find many examples of its use.

To assist with composition there is also a Semantic Schema. The schema is in the form of a diagram[91]:

Fig 1



[90] It has wide and enduring usage. See Murray (1997) for an extended discussion of the many uses of substitution systems in literature.

[91] This is my representation of a diagram in Masterman’s essay. The lines here marked in bold were marked with an asterisk in Masterman’s diagram. Where two lines run between words only one was to be chosen.

This schema is meant to assist the operator whilst she fills in slots in the templates with words from the thesaurus. The idea is that the arrows “protect the inexperienced poet from feeding random choices into the machine” (Masterman p. 179). The schema does this by alerting us to words that bear upon others. So slot 5, with the most arrows, is the most important semantically.

This interest in conceptualising and representing semantics is, no doubt, influenced primarily by Masterman’s work on semantics dating back at least to her (1961) publication "*Semantic message detection for machine translation, using an interlingua*." John Sowa (2002) defines a semantic network thus: a *“semantic network* or *net* is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs”. I think this is what we can see in the diagram (fig 4) above.

Masterman was a pioneer in developing the theory of semantic networks: hers was the first in fact to be called a semantic schema (ibid.) The schema she developed, in her groundbreaking work on machine translation of languages, involved the description of concept types and formal patterns of relation.

Whilst such a schema works well for a machine to register connections based on pattern, later programmers of poetry have not taken up the schema, perhaps because it is rather unwieldy.

The purpose of the schema, the thesaurus and the template was to assist the non-poet to write a poem. Masterman did not over-rate the quality of the poems her program produced. To criticise the program from this point of view is to miss the point. COMPUTERIZED HAIKU is intended primarily as a learning tool for poets. That users during Cybernetic Serendipity suggested improvements and complained about the inadequacy of the available word choices, for Masterman proved the program worked. (The contemporary

descendant of COMPUTERIZED HAIKU is Ray Kurzweil's "*The Cybernetic Poet"*, although much more complex.[92])

The use of an interactive mode in a public display at Cybernetic Serendipity marks one of the earliest instances of which I am aware. (There were already interactive computer programs. The first game was *Spacewar*, 1961. I do not know, however, any were shown publicly.) It may be noted, the display of what was essentially a poetry-teaching tool in Cybernetic Serendipity is evidence of the show's willingness to look beyond conventional ideas of what should be shown in an art gallery.[93]

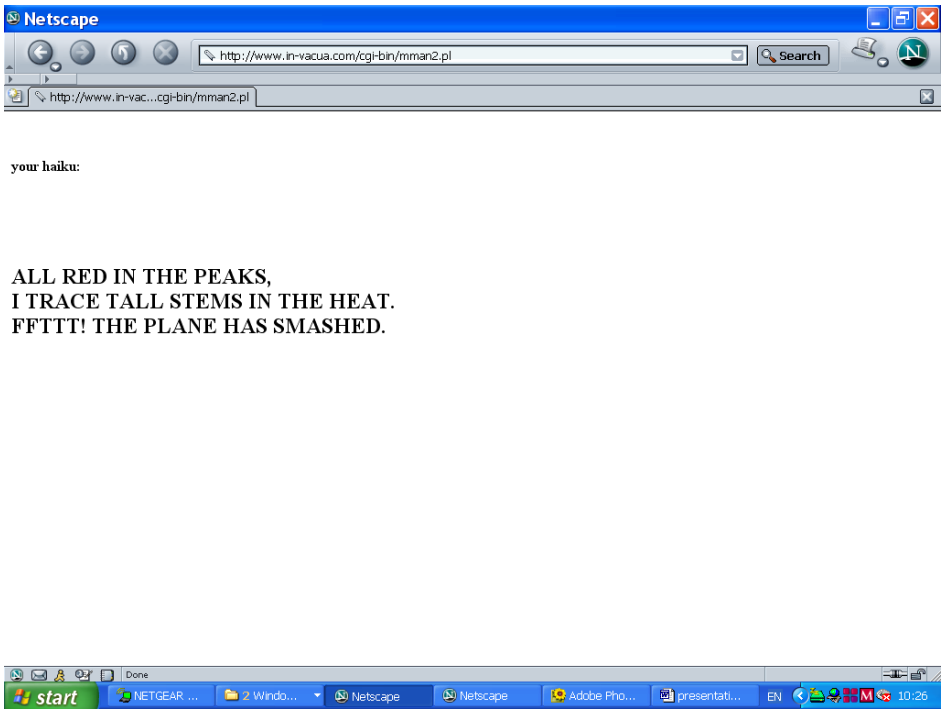
Interactivity has perhaps become such an overused term and so familiar experience that it easy to overlook its significance. It was, however, an important part of the premise of COMPUTERIZED HAIKU that it should exploit what, as I have mentioned, Mooers called the "interactive typewriter". Margaret Masterman, in her essay, explained that the option of batch processing, that is the complete automation of the program, had been considered and rejected.

I have, however, pursued Masterman's suggestion of a random haiku program. This program regularly violates all of the wise guidance provided to the human operator of the haiku program – or may make verses (fig 5) that seem to have contemporary relevance.

[92] "Find out how the RKCP ("*Ray Kurzweil's Cybernetic Poet*") can help you find rhymes, alliterations, ideas for the next word of your poem (or song), ideas for turns of phrase, and more". From http://www.kurzweilcyberart.com/poetry/rkcp_overview.php3

[93] Jasia Reichardt (1971) writes: "Thus Cybernetic Serendipity was not an art exhibition as such…it was primarily a demonstration of contemporary ideas, acts and objects, linking cybernetics and the creative process" (p. 14).

Plate 4



Screen grab of http://www.in-vacua.com/cgi-bin/haiku.pl

McKinnon-Wood with Gordon Pask had been involved in the development cybernetic theory, particularly with their “Conversation Theory”. (Pask and McKinnon-Wood were close associates and partners in the company System Research.) That Conversation Theory was part of the background of COMPUTERIZED HAIKU is indicated by this remark about COMPUTERIZED HAIKU that “the machine was to be used in conversational mode”. (This is from an article credited to the Cambridge Language Research Unit, but probably authored by McKinnon-Wood, or Margaret Masterman, or both.)

McKinnon-Wood also performed the final lecture, entitled “*Talking to Computers*”, to be given in a series at Cybernetic Serendipity (I think dispelling any doubt about the importance of Conversation Theory, or CT, to COMPUTERIZED HAIKU.)

CT is an all-embracing attempt to comprehend how we come to understand through interaction with our environment. The theory has both a loose and a formal expression. In general terms, all learning situations may be conceived

of as conversation. In strict conversation theory concepts such as "agreement", and "consciousness" are formalized processes of understanding.

CT is part of what is known as "second-order cybernetics". This is distinguished from what is considered a more mechanistic earlier phase where systems are conceived as passive and the observer is more sharply distinguished from the observed. Second-order cybernetics are characterised by a recognition that systems themselves are agents in their own right and interact with us as agents (systems.)

It is such considerations that form the theoretical underpinning of COMPUTERIZED HAIKU. It is this early venture into interactivity as informed by CT that may explain the work's popularity at the time of its exhibition. However, it must be accepted that, despite the hopes of the programmers of COMPUTERIZED HAIKU, computers have not really caught on as a learning aid for poetry. (I can only speculate that those *learning* to write poetry prefer to dispense with mechanical assistance.)

4.

How, if at all, is COMPUTERIZED HAIKU to be remembered? To ask this is also to enquire into the reception of computerised literature in general. The history of computerised poetry and computerised literature, in fact, is yet to be written. There are several partial accounts, none of which, to be fair, claim to be complete. None I know mentions COMPUTERIZED HAIKU.[94]

How has COMPUTERIZED HAIKU been received by those that do acknowledge it? Carole McCauley in her (1974) *COMPUTERS AND CREATIVITY* claims: "The *haiku* poems are…quite acceptable" (p.114).

[94] For instance, there is Janet Murray's (1997) *Hamlet on the Holodeck*. But this is about narrative. Aarseth's (1997*) Cybertext. Perspectives on Ergodic Literature* is discusses prose as well as poetry. There is also Charles O. Hartman's (1996) *Virtual Muse*, a personal memoir of poetry and computers.

Ray Kurzweil's (1990) *The Age of Intelligent Machines* reproduces several haiku without criticism. Margaret Boden's (1992) *The Creative Mind*, referring to COMPUTERIZED HAIKU, speaks of "the apparent success of this very early program" (p. 159). Funkhouser (2003) *Poetry, Digital Media and Cybertext* finds, however, "these poems…reveal how generated poems can be monochromatic in structure when the syntax is unvarying and is predetermined".

However, COMPUTERIZED HAIKU remains significant as an early attempt to make computer poetry. It shows how cybernetic theory, programming languages and experiments in literature interacted to produce work that was new and exciting in its time. COMPUTERIZED HAIKU was according to Masterman (1971), let it be remembered, an "unexpected success" of Cybernetic Serendipity (p 175).

The other text pieces in the show do not seem to have fared much better than COMPUTERIZED HAIKU, although several of them are interesting, even ground-breaking, such as Mendoza's *High Entropy Essays*, or Balestrini's *Tape Mark 1.* The best known is probably Edwin Morgan's *Computer's first Christmas card*. Ironically, this is a simulated computer poem; it is not in fact a piece of computer writing.

Perhaps Stefan's rather negative assessment, with which I began this exploration, is not wholly unfair and great works of computerised literature are yet to be made: despite the best attempts of ELIZA and Racter[95] and their company.

Nevertheless, it remains relatively early days for cybertext, and this area of literary production merits greater critical attention and further research. There are, no doubt, many more developments to be awaited in computerised poetry and cyber literature generally.

[95] ELIZA, Weizenbaum's (1976) well-known non-directive therapist. Racter (1984), the reputed author of *The Policeman's beard is half-constructed*.

5. Conclusion

In Conclusion, programming COMPUTERIZED HAIKU was an exercise in archaeology, but many other things as well. It was the first successful computer program I wrote. It was my first piece of work to be displayed on the web.

I have said that my initial interest was to look at a sort of ‘computational’ artwork, in its broadest sense: an artwork in a way divested of its material and historical ballast: something that aspired to the state of a sort of ‘pure instruction’ that could be translated between languages, constructed disassembled and remade. But this attempted act of retrieval has lead me since to consider precisely all that cannot be regained and which constitutes the differences between there and then and here and now.

Bibliography

Note: COMPUTERIZED HAIKU is available at **http://www.in-vacua.com/cgi-bin/haiku.pl**

Aarseth, E.J. (1997*) Cybertext. Perspectives on Ergodic Literature*. Baltimore and London, The Johns Hopkins University Press.

Aceti, L. (2002) *Getting Laid on the Procrustean Bed: Art Practice in the Digital World, One Man Versus One Pixel* <http://www.chart.ac.uk/> (13th September 2004).

Boden, M. (1992) *The Creative Mind*. London, Abacus.

Cambridge Language Research Unit, 'Computerized Haiku', *Theoria to Theory*, Volume 1, Fourth Quarter, July 1967: 378-383.

Derrida, J. 'ECONOMIMESIS', *DIACRITICS*. Volume 11: 3-25.

Funkhouser, C*. POETRY DIGITAL MEDIA AND CYBERTEXT* <http://web.njit.edu/~cfunk/SP/hypertext/POETRYDIGITALMEDIACYBERTEXT2.doc> (10th July 2004).

Hartman, Charles O. (1996) *Virtual Muse: Experiments in Computer Poetry*. University Press of New England, Hanover NH.

Kurzweil, R. (1990) The age of intelligent machines. Cambridge, Mass, MIT Press,

McCauley, Carole S. (1974) *COMPUTERS AND CREATIVITY*. Praeger, New York and Washington.

Masterman, M. (1971) 'Computerized haiku', in *Cybernetics, art and ideas,* (Ed.) Reichardt, J. London, Studio Vista, 175-184.

McKinnon-Wood, R. (1971) *Computer programming for literary laymen* in, *Cybernetics, art and ideas,* (Ed.) Reichardt, J. London, Studio Vista, 184-191.

Murray, J. H. (1997) *Hamlet on the Holodeck. The Future of Narrative in Cyberspace*. Cambridge Mass. MIT.

Racter (1984) *The Policeman's beard is half-constructed*. Illustrations by Joan Hall. Introduction by William Chamberlain. New York, Warner Software/Warner Books.

Reichardt, J. (Ed.) (1968) *Cybernetic Serendipity: the computer and the arts, a Studio International special issue*. London, Studio International.

Reichardt, J. (Ed.) (1971) *Cybernetics, art and ideas,* London, Studio Vista.

Sammet, Jean, E. quoted in, *Calvin N. Mooers Papers (CBI 81)*, Charles Babbage Institute, University of Minnesota, Minneapolis. <http://www.cbi.umn.edu/collections/inv/cbi00081.html> (5th September 2004).

Sowa, J. F. (2002) *Semantic Networks* <http://www.jfsowa.com/pubs/semnet.htm> (20th September 2004).

Stefans, B. K (2003) *Fashionable Noise. On Digital Poetics*. Berkeley California, Atelos.

Turing, A. (1950) *Computing Machinery and Intelligence* <www.abelard.org/turpap/turpap.htm>, (9th February 2003).

Turing, A. (1936) *ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM* <http://www.abelard.org/turpap2/tp2-ie.asp> (9th February 2003).

Weizenbaum, J. (1976*) Computer power and human reason*. San Francisco, W. H. Freeman and Company.

Appendix**: Evidence of Work 2**

(This text appears on my web site at: http://www.in-vacua.com/markov_text.html)

Markov Chain Algorithms

(A not very technical explanation).

How the algorithms work.

Markov algorithms do not require either mathematics or computers. It is possible to perform a Markov algorithm with a pencil and paper (I will discuss how to do this below). The only other thing required is an input text. There are mathematically oriented accounts available that discuss such matters. This one will avoid equations entirely. To illustrate my discussion, I will use *this* paragraph.

Markov algorithms work with patterns.

A Markov algorithms determine how likely it is that a word will follow another. What happens when it is equally likely that one word will follow as another one? The answer is it tosses a coin, or it makes a random choice, to put it another way.

Example: in the first paragraph the word “will” appears three times. It is followed by

“discuss“

“avoid”

“use”

If a Markov algorithm were to encounter the word “will” in the above, it could randomly choose one of these three words available to it as being equally probable. It might then take for instance the word “discuss” and follow it up with either “how” or “such”. It continues to do this each time, taking a word, finding one to follow it, taking the last word found and then adding another. As you see, it deals with one pair of words at a time. It adds a word and this makes a new pair. (It is possible for the algorithm to use threes and fours or more, but pairs seem to produce more interesting results).

With short texts, like the first paragraph, it doesn’t have many options available to it as most of the words appear only once. But with longer texts it can produce many variations.

The texts it produces seem often quite like the source text. Frequently they are also rather strange sounding.

One of the things about a Markov chain algorithm is that it treats punctuation as part of a word. So, it would treat “word.” as a word, if you see what I mean. This is quite useful as frequently it manages to punctuate plausibly by shifting letters and punctuation marks together. Sometimes it does not though and can put punctuation marks in funny places.

Markov algorithms have often been used to produce texts that are both nonsensical and rather plausible.

A quite well known example is “Mark V. Shaney”. He is discussed in:

Kernighan, Brian W and Pike, R. (1999) *The Practice of Programming*. Reading, MA, Addison-Wesley.

Apparently he confused people who thought he was ‘real’. He’s still out there living on the web somewhere.

A Few Observations

My own use of Kernighan and Pike’s algorithm differs quite a bit in that I am not really interested in hoodwinking. The text that my Markov Generator uses is my own research into text generation. I was trying to generate a text that was a text generation text…

A slightly more technical point is that Markov algorithms tend to always start with the same word. This is repetitious of them after a while. So I run my text through a ‘shuffle’ first. This stops it doing that.

Markov algorithms are only as syntactical – at best – as the texts they use. If the text that is fed in is only one word repeated a lot, the algorithm will only produce the same text. If the text consists of a jumble of words appearing with equal frequency, then the text made is not likely to be any more like English than the input. (Obviously in English for example we choose words with slightly more care).

In other words, Markov algorithms just do a calculation. They cannot produce sentences for themselves. For that you need something like a recursive grammar (article to follow). These should always produce grammatical sentences as long as they are written that way. They probably produce nonsense too, however. See, Bulhak, A.C. (1996) On the Simulation of Postmodernism and Mental Debility using Recursive Transition Networks.

Natural Language Programming, tries to overcome this. But that is the subject of another discussion.

Lastly, Markov algorithms have a very short memory. That is to say, they produce a text based on their count of word frequencies. Each time the algorithm is run, it starts anew. For this reason Markov processes are called 'finite state machines': each state is determined by the one previous. A Markov algorithm only looks through a text on a pair by word pair basis. (although it can handle longer sequences). If you want something that seems to live and grow, you might be interested in artificial life programming.

Shannon's Pen and Paper Markov Method

Shannon is the founder of Information Theory. He wrote a (1948) paper, 'A Mathematical Theory of Communication' where he explains the basic technique (although this version is a little different).

Choose a pair of words at random from a novel. Read through the novel until the second of the words is encountered again. Write down the word that follows it. Carry on until you hit the word you just wrote down. When you find it write down the word that follows that word. Continue until you make something interesting or exhaustion sets in.

This is based on a now obscure text by J. R. Pierce describing Shannon's techniques. See 'A chance for art', in Cybernetics, art and ideas, (1971) Jasia Reichardt (Ed.)., London, Studio Vista.

The method may produce mostly garbage with occasional more interesting passages. It's also very slow. Easier to get a computer to do it?

Wayne Clements 01/05

Appendix: **Evidence of Work 3**

[Submitted to 'Mainframe', a book about the early years of computing and the arts, edited by Douglas Kahn and Benjamin Buchloh.]

The Ghosts of Cybernetics

The proposed paper concentrates on the '*Computer poems and texts*' installation at *Cybernetic Serendipity* (ICA Gallery, London, 1968). It builds upon a presentation previously given by the author at CHArt 2004[96].

...

In his 1967 lecture, *Cybernetics and Ghosts*, Italo Calvino looked forward to the appearance of a "writing machine". For Calvino this machine was defined, *a la* Alan Turing, as a procedure such as might, or might not, be entrusted to a (real) computer to carry out. Calvino gives the example of Queneau's *Cent Mille Milliards de poemes*: a "rudimentary machine for making sonnets, each one different from the last" (p. 12).

In fact, by the time of Calvino's writing several similar attempts had been made to simulate writing machines with computers. A number of these were

[96] *Computer Poetry's Neglected Debut*. The presentation included a live projection of a new version programmed by the author (http://www.in-vacua.com/cgi-bin/haiku.pl). This will be followed in due course by the publication of the full paper. There is an abstract at, http://www.chart.ac.uk/chart2004-abstracts/clements.html.

brought together in what was an impressive, although not comprehensive survey, in *Cybernetic Serendipity*.

One of these works, *The House of Dust* by Alison Knowles and James Tenney, is already featured in *Mainframe Computing*. My proposed paper aims to extend this discussion, placing the works featured in *Cybernetic Serendipity* in the context of a sustained attempt to program 'writing machines' stretching back over the preceding decade. This project may be dated from 1959, when in two separate but contemporaneous initiatives, Theo Lutz, and Brion Gysin and Ian Somerville, programmed the first computer poems.

The works in *Cybernetic Serendipity* are notable for several reasons. There is the international aspect of the exhibition (mirrored by other displays at the ICA), with work from the USA, Italy, France and Britain. Plainly, there were many similar and contemporaneous developments occurring in several different countries.

Also characteristic of the show is the prominence of scientists, sometimes, but not always, working with artists and writers. From this may be construed the difficulty of artists to command either the programming skills or access to the hardware necessary to make computerised work. (Thus, participants such as Masterman and McKinnon Wood were computational linguists from Cambridge, and Mendoza a physicist from University College North Wales).

However, what is perhaps most notable is how *Cybernetic Serendipity* comprehensively maps out progress in this area, *and* how relatively little development there has been since.

For instance, Jean Baudot's *La Machine à Écrire* was an early (1964) text generation program. Masterman and McKinnon Wood's *COMPUTERIZED HAIKU* may be accorded the honour, on the basis of my research, of being the first work to take advantage of the "interactive keyboard" to be publicly exhibited. Mendoza's *High-Entropy Essays* anticipate the *Postmodernism*

Generator's much more famous – and it may be granted – more successful attempt to spoof pseudo-science.

…

It is my conclusion that this relative lack of progress since 1968 has led to these, and other early initiatives, being rather overlooked by later scholarship. Despite Calvino's hopes of a machine that would write poems and novels "that follow all the rules" (ibid.) that might be able also to rebel and stage a modernist revolt against its own classicism, computerised literature has not developed far beyond what was mapped out for it in *Cybernetic Serendipity*. The ghosts of cybernetics linger over this epoch. These are the ghosts, and not the only ones, of the hopes of that time of optimism that continue to trouble the present.

Bibliography

Calvino, I. (1997) 'Cybernetics and Ghosts', in *The Literature Machine, Essays*, trans. Creagh, P. London, Vintage, 3-28.

Reichardt, J. (Ed.). (1968) *Cybernetic Serendipity: the computer and the arts, a Studio International special issue*. London, Studio International.

Reichardt, J. (Ed.). (1971) *Cybernetics, art and ideas,* London, Studio Vista.

Appendix**: Evidence of Work 4: Programs**

In this **Appendix** I present several programs. These works are chosen to be representative of the in-vacua.com website. I have not tried to tidy them. They retain evidence of working.

...

1. haiku.pl

Note: the program for COMPUTERIZED HAIKU. This program handles user created haiku. Random and automated versions are made by other programs.

```
#!/usr/bin/perl -Tw
# /home/sites/www.in-vacua.com/web/cgi-bin/haiku.pl -w
use strict;

use CGI qw(:all);
use CGI::Carp qw(fatalsToBrowser);

my $base="/home/sites/www.in-vacua.com/web/base.html";
my @haikus=qw(s1 s2 s3 s4 s5 s6 s7 s8 s9 name line1 line2 line3);

my ($s1, $s2, $s3, $s4, $s5, $s6, $s7, $s8, $s9);
my ($line1, $line2, $line3, $name);


print header;

  if(! param ) {
     page_one();
} elsif (defined param('pageone')) {
      page_two();
} elsif (defined param('pagetwo')) {    #creates later pages
      page_three();                 #ie when p1 done, p2 made etc
} else {
      haiku_complete();
}

######
```

```
sub page_one {
   print<<END_PAGE_ONE;

<html>
<head>
 <title>COMPUTERIZED HAIKU</title> <meta name=\"keywords\"
content=\"writing machines, text, random, computerised literature, algorithmic,
computer poetry, performance scripts, instructions, splice, cut-up\">
<meta name=\"description\" content=\"in-vacua.com features text machines, software
art, and text generation and manipulation programs for users on the internet\">

  <script language=\"JavaScript\">

  function createTarget(t)
{    window.open(\"\", t, \"width=650,height=400, left=75,top=0, scrollbars=yes,
toolbar=yes, menubar=yes, resizable=yes\");
   return true;}

  </script>

</head>

<body
link=\"87CEFF\"

VLINK=\"orange\"

 bgcolor="black" text="white">
<h1 style="font-family: helvetica,arial,sans-serif;">
<center>COMPUTERIZED HAIKU</center>
</h1>
<br style="font-family: helvetica,arial,sans-serif;">
<p style="font-family: helvetica,arial,sans-serif;"></p>
<span style="font-family: helvetica,arial,sans-serif;"><small>

<a href= \"http://www.in-vacua.com/haiku.html\" onclick=\"return
createTarget(this.target)\"
 target=\"window1\">
Computerized
Haiku</a> was shown in the first major exhibition of computer art,<br>
'Cybernetic Serendipity' (ICA, London 1968). See also <a href= \"http://www.in-
vacua.com/cgi-bin/mendoza.pl\">High-Entropy-Essays</a>.<br><p>
Originally programmed by Margaret Masterman<br> and Robin McKinnon-
Wood.<br><br>
I gave a presentation about it at <b>'CHArt'</b> 2004 (<i><b><u>Computer Poetry's
Neglected Debut</u></b></i>).<br>
 There is an Abstract <a href=\"http://www.chart.ac.uk/chart2004-
abstracts/clements.html\" onclick=\"return createTarget(this.target)\"
 target=\"window2\">here</a>
```

```
<br><p>
&amp;nbsp;&amp;nbsp;&amp;nbsp;This version programmed
2003. Wayne Clements, in-vacua.com.<br>
</small><br><p>
<br>
<hr>
<br style="font-family: helvetica,arial,sans-serif;">
</span>
<p style="font-family: helvetica,arial,sans-serif;"></p>
<p style="font-family: helvetica,arial,sans-serif;"></p><h3 style="font-family:
helvetica,arial,sans-serif;">Choose a word from
each of the lists to make a haiku.</h3>
<br style="font-family: helvetica,arial,sans-serif;">
<span style="font-family: helvetica,arial,sans-serif;">It is suggested
you select words that go together, as in this example (the words in brackets are
fixed,</span><br style="font-family: helvetica,arial,sans-serif;">

<span style="font-family: helvetica,arial,sans-serif;"> the others are
chosen from the lists):</span><br style="font-family: helvetica,arial,sans-serif;">
<br style="font-family: helvetica,arial,sans-serif;">
<span style="font-family: helvetica,arial,sans-serif;"> [ALL] THIN [IN
THE] MIST,</span><br style="font-family: helvetica,arial,sans-serif;">
<span style="font-family: helvetica,arial,sans-serif;">[I] TRACE BLACK
BIRDS [IN THE] DAWN.</span><br style="font-family: helvetica,arial,sans-serif;">
<span style="font-family: helvetica,arial,sans-serif;">WHIRR! [THE]
CRANE [HAS] PASSED.
</span><br style="font-family: helvetica,arial,sans-serif;">
<p style="font-family: helvetica,arial,sans-serif;"><br>
</p>

<form>

 <select name="s1"/><option value="WHITE"/>WHITE
  </option/><option value="BLUE">BLUE
  </option><option value="RED">RED
  </option><option value="BLACK">BLACK
  </option><option value="GREY">GREY
  </option><option value="GREEN">GREEN
  </option><option value="BROWN">BROWN
  </option><option value="BRIGHT">BRIGHT
   </option><option value="PURE">PURE
  </option><option value="CURVED">CURVED
  </option><option value="CROWNED">CROWNED
  </option><option value="STARRED">STARRED
    </option></select>

  <select name="s2" size="1"><option value="BUDS"/>BUDS
  </option/><option value="TWIGS">TWIGS
  </option><option value="LEAVES">LEAVES
  </option><option value="HILLS">HILLS
```

```
</option><option value="PEAKS">PEAKS
</option><option value="SNOW">SNOW
</option><option value="ICE">ICE
</option><option value="SUN">SUN
</option><option value="RAIN">RAIN
</option><option value="CLOUD">CLOUD
</option><option value="SKY">SKY
</option><option value="DAWN">DAWN
</option><option value="DUSK">DUSK
</option><option value="MIST">MIST
</option><option value="FOG">FOG
</option><option value="SPRING">SPRING
</option><option value="HEAT">HEAT
</option><option value="COLD">COLD
</option></select>

<select name="s3" size="1"><option value="SEE">SEE
</option><option value="TRACE">TRACE
</option><option value="GLIMPSE">GLIMPSE
</option><option value="FLASH">FLASH
</option><option value="SMELL">DITCH
</option><option value="TASTE">TASTE
</option><option value="HEAR">HEAR
</option><option value="SEIZE">SEIZE
</option></select>
<select name="s4" size="1"><option value="SNOW">SNOW
</option><option value="TALL">TALL
</option><option value="PALE">PALE
</option><option value="DARK">DARK
</option><option value="FAINT">FAINT
</option><option value="WHITE">WHITE
</option><option value="CLEAR">CLEAR
</option><option value="RED">RED
</option><option value="BLUE">BLUE
</option><option value="GREEN">GREEN
</option><option value="GREY">GREY
</option><option value="BLACK">BLACK
</option><option value="ROUND">ROUND
</option><option value="SQUARE">SQUARE
</option><option value="STRAIGHT">STRAIGHT
</option><option value="CURVED">CURVED
</option><option value="SLIM">SLIM
</option><option value="FAT">FAT
</option><option value="BURST">BURST
</option><option value="THIN">THIN
</option><option value="BRIGHT">BRIGHT
</option></select>

<select name="s5" size="1"><option value="TREES">TREES
</option><option value="PEAKS">PEAKS
```

```
</option><option value="HILLS">HILLS
</option><option value="STREAMS">STREAMS
</option><option value="BIRDS">BIRDS
</option><option value="SPECKS">SPECKS
</option><option value="ARCS">ARCS
</option><option value="GRASS">GRASS
</option><option value="STEMS">STEMS
</option><option value="SHEEP">SHEEP
</option><option value="COWS">COWS
</option><option value="DEER">DEER
</option><option value="STARS">STARS
</option><option value="CLOUDS">CLOUDS
</option><option value="FLOWERS">FLOWERS
</option><option value="BUDS">BUDS
</option><option value="LEAVES">LEAVES
</option><option value="TREES">TREES
</option><option value="POOLS">POOLS
</option><option value="DROPS">DROPS
</option><option value="STONES">STONES
</option><option value="BELLS">BELLS
</option><option value="TRAILS">TRAILS
</option></select>

<select name="s6" size="1"><option value="SPRING">SPRING
</option><option value="FALL">FALL
</option><option value="COLD">COLD
</option><option value="HEAT">HEAT
</option><option value="SUN">SUN
</option><option value="SHADE">SHADE
</option><option value="DAWN">DAWN
</option><option value="DUSK">DUSK
</option><option value="DAY">DAY
</option><option value="NIGHT">NIGHT
</option><option value="MIST">MIST
</option><option value="TREES">TREES
</option><option value="WOODS">WOODS
</option><option value="HILLS">HILLS
</option><option value="POOLS">POOLS
</option></select>

<select name="s7" size="1"><option value="BANG">BANG
</option><option value="HUSH">HUSH
</option><option value="SWISH">SWISH
</option><option value="FFTTT">FFTTT
</option><option value="WHIZZ">WHIZZ
</option><option value="FLICK">FLICK
</option><option value="SHOO">SHOO
</option><option value="GRRR">GRRR
</option><option value="WHIRR">WHIRR
</option><option value="LOOK">LOOK
```

```
 </option><option value="CRASH">CRASH
 </option></select>
 <select name="s8" size="1"><option value="SUN">SUN
 </option><option value="MOON">MOON
 </option><option value="STAR">STAR
 </option><option value="CLOUD">CLOUD
 </option><option value="STORM">STORM
 </option><option value="STREAK">STREAK
 </option><option value="TREE">TREE
 </option><option value="FLOWER">FLOWER
 </option><option value="BUD">BUD
 </option><option value="LEAF">LEAF
 </option><option value="CHILD">CHILD
 </option><option value="CRANE">CRANE
 </option><option value="BIRD">BIRD
 </option><option value="PLANE">PLANE
 </option><option value="MOTH">MOTH
 </option></select>

 <select name="s9" size="1"><option value="FLIT">FLIT
 </option><option value="FLED">FLED
 </option><option value="DIMMED">DIMMED
 </option><option value="CRACKED">CRACKED
 </option><option value="PASSED">PASSED
 </option><option value="SHRUNK">SHRUNK
 </option><option value="SMASHED">SMASHED
 </option><option value="BLOWN">BLOWN
 </option><option value="SPRUNG">SPRUNG
 </option><option value="CRASHED">CRASHED
 </option><option value="GONE">GONE
 </option><option value="FOGGED">FOGGED
 </option><option value="BURST">BURST
 </option></select>
 <br>

 <p><br>
 </p>
 <p><input type="submit" name="pageone" value="Write your Haiku">
 <input type="reset" value="reset"><br>
 </p></form><p><br><span style="font-family: helvetica,arial,sans-serif;">In 1968
Haiku were pinned up on the Gallery wall.<br><br>
 You can have your Haiku placed in an archive. To preview this archive click<br>

 here: <a href="http://www.in-vacua.com/base.html">archive</a>
 <p><br>
 </p>
 <hr><br>
 <p><br>
</p>
```

```
  <h3><p style="font-family: helvetica,arial,sans-serif;">A Random Haiku
Machine.</h3>
  <br><p style="font-family: helvetica,arial,sans-serif;">
Masterman speculated about a program to produce these haiku randomly. If
you
press the button this<br>
is what happens. However, some of the combinations can be a little
peculiar.<br>
Perhaps she would have revised it.<br>
  <p></p>

 <form action="/cgi-bin/mman2.pl" method="post">
<br>
    <input type="submit" value="Random Haiku"> </form>
  <br>
  <p><br> <p style="font-family: helvetica,arial,sans-serif;"><big><big>There's an
automated version <a href="http://www.in-vacua.com/cgi-bin/ku.pl">>here</a>
  </p><br></big>
</form>
<p> <a href="http://www.in-vacua.com/list.html">Home</a></p>
<br>

</body></html>

END_PAGE_ONE
}

#########

sub repeat_hidden {
foreach my $poems ( @haikus ){
       if (defined param($poems)) {
       print "<input type=hidden";
        print " name=\"$poems\" ";
         print " value=\"", param($poems),"\"/>\n";
         }
      }
    }




sub page_two {

my $s1=param('s1');
my $s2=param('s2');
my $s3=param('s3');
my $s4=param('s4');
my $s5=param('s5');
my $s6=param('s6');
```

```
my $s7=param('s7');
my $s8=param('s8');
my $s9=param('s9');


my $line1=param('line1');
my $line2=param('line2');
my $line3=param('line3');

$line1 = join '', (' ALL ', $s1, ' IN THE ', $s2, ', ');
$line2 = join '', (' I ', $s3, ' ', $s4, ' ', $s5, ' IN THE ', $s6, '. ');
$line3 = join '', ($s7, '! THE ', $s8, ' HAS ', $s9, '. ');


       print<<END_PAGE_TWO;

<br><p><h4>your haiku:</h4>

<br><p><br><p><h2>

<form>

$line1<input type="hidden" name="line1" value="$line1"><BR>
$line2<input type="hidden" name="line2" value="$line2"><br>
$line3<input type="hidden" name="line3" value="$line3"></h2>

<br><p><br><p>
<font face="arial">
 <h4>if you'd like, your haiku can be entered in the <a href="http://www.in-
vacua.com/base.html">visitor archive</a> by using the button below<br><p>
or if you prefer, <a href="http://www.in-vacua.com/cgi-bin/haiku.pl">please
return</a>
</h4>
</p><br>
<input type="submit" name="pagetwo" value="to archive"/>

END_PAGE_TWO
    repeat_hidden();
     print "</form>";
}


##########

sub page_three {

     print<<END_PAGE_THREE;

<form><p>
```

```
<br><h4><font face=Arial size=2>
Please enter a name to go with your poem. Or you can leave the field blank.<br>
Your haiku will be displayed on the web site.
<input type="text" name="name" size="20"></textarea><br>
<br>Press send,<br>
<p><input type="submit" name="pagethree" value="Send"/>

END_PAGE_THREE
  repeat_hidden();
    print "</form>";
   }

#########


sub save {
open(FILE, ">>$base") || die "Cannot open $base: $!";
flock (FILE, 2) || Error('lock', 'file');
my $poems1=param('line1');
my $poems2=param('line2');
my $poems3=param('line3');
my $name=param('name');

print FILE "$poems1\n";
print FILE "<br>$poems2";
print FILE "<br>$poems3";

print FILE "<br><br>Name: $name<br><br><br>";


}
close(FILE);

sub haiku_complete {
save();

 print<<END_HAIKU_COMPLETE;
<font size=3><font face=Arial size=3><br>
Your haiku has been placed in the archive<br>
<a href="/base.html">click to view all the poems in the collection</a>

END_HAIKU_COMPLETE
repeat_hidden();
    print "</form>";
   }
```

2. ono1.pl

Note: this is the program for the work titled Ono Generator.

```
#!/usr/bin/perl -Tw
#/home/sites/www.in-vacua.com/web/cgi-bin/ono.pl

use strict;
use CGI ':standard';
use CGI::Carp qw(fatalsToBrowser);

print "Content-type: text/html\n\n";

my (@s10, @s20, @s30, @s40, @s50, @s60, @s70, @s80, @s90, @s010,
@keywords, @select);
my ($s10, $s20, $s30, $s40, $s50, $s60, $s70, $s80, $s90, $s01);
my ($s1, $s2, $s3, $s4, $s5, $s6, $s7, $s8, $s9, $s010);
my ($rand1, $rand2);
my ($line1, $line2, $line3, $line4, $line5, $line5a, $line6, $line7, $line8, $line9,
$line10, $line11);

srand;

#array holds lines to make a random select from

@s10 = ("when a hole is drilled ", "where there is wind ", "where you can see the sky
", "where the west light comes in ", "once a year ", "in a glass tank ", "on a snowy
evening ", "in the town square ", "from the beginning to the end ", "when a hole is
drilled ", "at any time ", "for any length of time ", "every morning ", "at an address
arbitrarily chosen ", "at twenty addresses ", "at an arbitrary point ", "in the garden ",
"on the night of the full moon ", "at dawn ", "from 1AM ");
 @s20 = ("a space ", "a fictional name ", "a shadow ", "the whole thing ", "a design ",
"a numeral ", "a roman letter ", "a circle ", "a number ", "a hole ", "the morning light
", "a marker ");
  @s30 = ("a bag ", "a broken sowing machine ", "a microscope ", "a stone ", "a hair ",
"a piece of glass ", "a piece of wood ", "a piece of metal ", "the sky ", "a canvas ", "a
vine ", "your hand ", "the garbage ");
@s40 = ("telephone numbers ", "grasshoppers ", "ants ", "singing insects ", "figures ",
"shapes ", "old paintings ", "photographs ", "blank canvases ", "two holes ");
   @s50 = ("until ", "till ", "where ", "when ", "to see if ");
    @s60 = ("try ", "shake " , "receive ", "converse with ", "enlarge ", "change ", "list ",
"see ", "use ", "sell ", "send ", "collect ", "select ", "mix in your head ", "write on ",
"hang ", "drill ", "bury ", "place ", "cut ", "cut out ", "dismember ");
     @s70 =("a shadow ", "cracked ", "red ",  "black ", "almost invisible ", "finished ",
"covered with nails ", "dyed thoroughly in rose ", "different ", "gone ");
      @s80 = ("observe ", "imagine ", "remember ", "see ");
```

```
      @s90 = ("the size you prefer ", "until the whole thing is gone ", "that you
associate with it ", "to see if the skies are different ", "that come to mind ", "the size
you prefer ", "chosen arbitrarily ", "to let the light go through ", "to your taste ", "to
paint black ", "printed or otherwise ", "that you like ", "that you remember ");

$s1 = $s10[int(rand(@s10))];
 $s2 = $s20[int(rand(@s20))];
  $s3 = $s30[int(rand(@s30))];
   $s4 = $s40[int(rand(@s40))];
    $s5 = $s50[int(rand(@s50))];
     $s6 = $s60[int(rand(@s60))];
      $s7 = $s70[int(rand(@s70))];
       $s8 = $s80[int(rand(@s80))];
        $s9 = $s90[int(rand(@s90))];
         $s01 = $s010[int(rand(@s010))];

$line1 = join '', ($s6, $s3, "\n");
  $line2 = join '', ($s3, "is ", $s2, $s1, " - ", $s8, "\n");
   $line3 = join '', ($s3, $s1, "is to be ", $s2, "\n");
     $line4 = join '', ($s1, "there is ", $s2, "\n");
    $line5 = join '', ("if ", $s1, $s2, "is ", $s7, ' - ', $s8, $s3, "\n");
   $line5a = join '', ($s1, $s8, $s4, "\n");

  $line6 = join '', ($s6, $s2, "for a visitor to ", $s8, "\n");
   $line7 = join '', ($s8, $s3, "that is ", $s7, "\n");
    $line8 = join '', ($s1, "there is ", $s2, " - ", $s8, $s3, "\n");
  $line9 = join '', ("for a visitor: ", $s4, "that are ", $s7,', ', $s1, "\n");
 $line10 = join '', ($s6, $s4, $s5, $s4, "are ", $s7, "\n");
$line11 = join '', ($s6, $s4, $s9, "\n");


$rand2 = int(rand 2) + 1;  #creates random no. for if else.

if ($rand2 == 1) {
 push @keywords, $line1, $line2, $line3, $line4, $line5, $line5a
#selects a line group as no. == 1 or 2
}
else {
 push @keywords, $line6, $line7, $line8, $line9, $line10, $line11
}


$rand1 = int(rand 4) + 1; #a random number is chosen...not 0...size adjustable

my %seen = ();

$seen{int(rand(@keywords))}++ while scalar keys %seen<$rand1;
#a random group of lines will be chosen
                                        #AND of a random size
```

```
@select = @keywords[keys %seen];



print "<html><head><title>ONO GENERATOR</title>
  <script language=\"JavaScript\">
<!--hide

  function createTarget(t)
{    window.open(\"\", t, \"width=650,height=400, left=75,top=0, scrollbars=yes,
toolbar=yes, menubar=yes, resizable=yes\");
   return true;}

//-->
  </script>


<meta name=\"keywords\" content=\"text machines, text, random, computerised
literature, algorithmic, computer poetry, performance scripts, instructions\">
<meta name=\"description\" content=\"in-vacua.com features writing machines,
software art, and text generation and manipulation programs for users on the
internet\">

</head>


<body
link=87CEFF

VLINK=BCEE68

 bgcolor=\"black\" text=\"white\">

<div align=\"center\"><big style=\"font-family: arial;\"><big><big><big><span
 style=\"font-weight: bold;\">ONO GENERATOR</span></big></big></big></big>
 <font face=Arial size=2>
<br><p><br><p><br>
<p>
<br><p><br>
<center>
<form><input type=\"submit\" value=\"@select\" action onSubmit=\"http://www.in-
vacua.com/ono.pl\"></center>
<p>
<br><p>
<br><p><br><p><br><p><br><p>
<hr style=\"width: 100%; height: 2px;\"><p><div align=\"left\"><br><p>
```

<div align=\"left\"> <font face=\"Arial\"> <big><big> press the GREY BUTTON ABOVE to make the <a href=\"/ono_text.html\" onclick=\"return createTarget(this.target)\"
 target=\"window1\" style=\"color: rgb(124; 252; 0);\"> Ono Generator
 write

<p>Home";

3. generator.pl

Note: the program for Markov Generator.

```
#!/usr/bin/perl - wT

# /home/sites/www.in-vacua.com/web/cgi-bin/generator.pl -w

#shuffles 1st, then markovises. the finished prog
#

use strict;
use CGI ':standard';
use CGI::Carp qw(fatalsToBrowser);

print "Content-type: text/html\n\n";
my ($MAXGEN, $NONWORD, $w1, $w2, $suf, $statetab, $r, $t, $i, %statetab,
@array);

# Copyright (C) 1999 Lucent Technologies
# Excerpted from 'The Practice of Programming'
# by Brian W. Kernighan and Rob Pike

# markov.pl: markov chain algorithm for 2-word prefixes


open(FILE1, "/home/sites/www.in-vacua.com/web/cgi-bin/chapter1.txt")|| die;

open(FILE2, ">/home/sites/www.in-vacua.com/web/cgi-bin/chapter2.txt")|| die;


@array = <FILE1>;
shuffle(\@array);
my $draw= join ' ', @array;

print FILE2 $draw;

close FILE1;
close FILE2;
open(FILE2, "/home/sites/www.in-vacua.com/web/cgi-bin/chapter2.txt")|| die;

srand;
my $rand = int(rand 60) + 2;

my $html = "<html><head>
<script>
var limit=\"0:$rand\"
```

```
if (document.images){
var parselimit=limit.split(\":\")
parselimit=parselimit[0]*60+parselimit[1]*1
}
function beginrefresh(){
if (!document.images)
return
if (parselimit==1)
window.location.reload()
else{
parselimit-=1
curmin=Math.floor(parselimit/60)
cursec=parselimit%60
if (curmin!=0)
curtime=curmin+\" minutes and \"+cursec+\" seconds left until page refresh\"
else
curtime=cursec+\" seconds\"
window.status=curtime
setTimeout(\"beginrefresh()\",2000)
}
}

window.onload=beginrefresh
//-->
</script>


<title>markov text</title>
</head>

<body bgcolor=black lang=EN-GB link=\"#87ceff\" vlink=\"#bcee68\"
bgcolor=\"black\" text=\"white\">
<a href='/markov_gen.html'>home</a><P><BR><P><P>
<center><font face=Arial size=6>";




$MAXGEN = 10000;
$NONWORD = "\n";
$w1 = $w2 = $NONWORD;          # initial state
while (<FILE2>) {                  # read each line of input
        foreach (split) {
                push(@{$statetab{$w1}{$w2}}, $_);
                ($w1, $w2) = ($w2, $_);        # multiple assignment
        }
}
push(@{$statetab{$w1}{$w2}}, $NONWORD);    # add tail
print $html;
```

```
$w1 = $w2 = $NONWORD;
for ($i = 0; $i < $MAXGEN; $i++) {
        $suf = $statetab{$w1}{$w2};# array reference
        $r = int(rand @$suf);           # @$suf is number of elements
        exit if (($t = $suf->[$r]) eq $NONWORD);
print  ' '. $t; #concatenation with white space, W


        ($w1, $w2) = ($w2, $t);                 # advance chain
}


# fisher yates shuffle
sub shuffle {
my($array) = shift();
for (my $i = @$array; --$i; ) {
my($j) = int(rand($i + 1));
next() if ($i == $j);
@$array[$i, $j] = @$array[$j, $i];
}
} #EOSub
```

4. alt_img_tate.pl

```
#!/usr/bin/perl -wT

# alt 3 will open web page. strip out alt tags. print tags to file. close.
# it will then select and shuffle *some*  of the lines.
# this one will delete white space using grep and select one alt tag and print.



use strict;
use HTML::Tree;
use LWP::Simple;
use CGI::Carp qw(fatalsToBrowser);

my $html2  = '';
my $page;
my @page;
my @HTTP = '';
my $HTTP = '';

srand;



my $ALT_FILE_NEW = 'ALT_FILE_NEW.txt';
my $alt_tate = 'alt_tate.txt';

open(ALT_TATE, "$alt_tate") or die "Can't open file: $!";
flock (ALT_TATE, 2) || Error('lock', 'file');
 #opens file of addresses


 @HTTP = <ALT_TATE>; #puts address file into a array

close(ALT_TATE);

my $new = $HTTP[int(rand(@HTTP))]; #chooses one from array

my $Html = get($new); #opens new web page
print "Content-type: text/html\n\n";

open(ALT_FILE_NEW, ">$ALT_FILE_NEW") or die "Can't open file: $!";
                                                        #opens file to write to
flock (ALT_FILE_NEW, 2) || Error('lock', 'file');
```

```
	my %AltTexts;


	while($Html=~/(<IMG\b.*?>)/isg)
	{	my $ImgElement=$1;
		# Find SRC tag
		$ImgElement=~/SRC\s*=\s*([\"\'])(.*?)\1/is;
		my $Src=$2;
		#print " $Src\n";
		# Find ALT tag & store text
		if($ImgElement=~/ALT\s*=\s*([\"\'])(.*?)\1/is)
		{	$AltTexts{$Src}=$2;}
		else
		{	# No ALT found so give it default text if none already found
			unless(exists($AltTexts{$Src}))
			{	$AltTexts{$Src}='NO_ALT_TAG!';}}}

	# Write extracted data to a file


	foreach my $SrcPath (sort keys %AltTexts)
	{	print ALT_FILE_NEW "$AltTexts{$SrcPath}\n\n";} #writes to file


close(ALT_FILE_NEW);

open(ALT_FILE_NEW, "$ALT_FILE_NEW") or die "Can't open file: $!";
flock (ALT_FILE_NEW, 2) || Error('lock', 'file');

my $choice = '';

my @array = <ALT_FILE_NEW>;
close(ALT_FILE_NEW);

my @words = grep /[A-Z_a-z]/, @array; #gets alts with words only

$choice = $words[int(rand(@words))]; #chooses one

if (! $choice){
$html2 = "<html><head>
<script type=text/javascript>
// The time out value is set to be X (or X seconds)
setTimeout(' document.location=document.location' ,10000);
col=255;
function fade() { document.getElementById(\"fade\").style.color=\"rgb(\" + col + \",\"
+ col + \",\" + col + \")\"; col-=5; if(col>0) setTimeout('fade()', 200); }
</script>
</head>
```

```
<body onLoad=\"fade()\">
<p>


<br><br><br>

<br><br><br><br><br>
<br>
<br><br><br><br>
<br><br><br><br>
<center>
<font face=Arial size=8><span id=\"fade\">PLEASE_WAIT...</span></center>
<p>
<br><p><br><p><br><p><br><\html>
";

}

else
{

 $html2 = "<html><head>
<script type=text/javascript>
// The time out value is set to be X (or X seconds)
setTimeout(' document.location=document.location' ,10000);
col=255;
function fade() { document.getElementById(\"fade\").style.color=\"rgb(\" + col + \",\"
+ col + \",\" + col + \")\"; col-=5; if(col>0) setTimeout('fade()', 200); }
</script>
<body onLoad=\"fade()\">
<p>


<br><br><br>

<br><br><br><br><br>
<br>
<br><br><br><br>
<br><br><br><br>
<center>
<font face=Arial size=8><font face=Arial size=8><span id=\"fade\">$choice</span>
<p>
<br><p><br><p><br><p><br>
";
}
print $html2;
```

5. Noumena1.pl

[Note. I wish to acknowledge that the programming of Noumena1.pl is largely the work of Simon at www.hitherto.net]

```
#!/usr/bin/perl -wT

# /home/sites/www.in-vacua.com/web/cgi-bin/Noumena1.pl -w
use CGI::Carp qw(fatalsToBrowser);
use strict;
use CGI ':standard';
use lib '/.users/27/inv838/Template';
use LWP::Simple;
use HTML::Parser;


use vars qw($html);
my $content;
# Configurable variables for the script

my %templates = (text => "text_output.html",
                 url  => "url_output.html");

# Initialise a new CGI object for parameter handling, etc.

my $q = CGI->new;

# Check to see if we have any input from the user. If so,
# we go to process it. If not, we'll return a blank form

if ($q->param('text')) {
  my $text = &process_text($q->param('text'));
  &output_template('text',$text);
} elsif ($q->param('url')) {
  my $text = &process_url($q->param('url'));
} else {
  print $q->redirect("/noumena.html");
}

## Subroutine Definitions

# process_url: strip non-punctuation from html docs (harder)

sub process_url {
  my ($url) = @_;
```

```
  my $content = get($url);
  die "Couldn't get it!" unless defined $content;

  # Slightly ugly kludging to sort out internal document links
  # on sites that don't fully qualify (damn them all)

  if (!($url =~ m!^http://!)) {
    $url ="http://".$url;
  }
  $url =~ m!(http://(.*))/!;
  my $baseurl=$1 || $url;

  $content =~ s!href="/(.*)"!href="$baseurl/$1"!ig;
  $content =~ s!rel="/(.*)"!rel="$baseurl/$1"!ig;
  $content =~ s!src="/(.*)"!src="$baseurl/$1"!ig;

  # HTML::Parser is slightly odd - it uses a callback interface which throws
  # things back into this namespace.

  HTML::Parser->new(api_version     => 3,
                    handlers        => [start => [\&_html_parser_tag, "text"],
                                        end    => [\&_html_parser_tag, "text"],
                                        text   => [\&_html_parser_text, "dtext"]],
                    marked_sections => 1,)->parse($content);

  print $q->header;

  print $html;
}

# html_parser_text: handler to tell HTML::Parser what to do with text sections

sub _html_parser_text {
  my ($text) = @_;
  $text =~ s!\w! !g;
  $html .= $text;
}

# html_parser_tag: handler to pass html tags unmolested back to HTML::Parser

sub _html_parser_tag {
  my ($text) = @_;
  $html .= $text;
}

# output_template: use Template Toolkit to return data to the user

sub output_template {
  my ($type, $text) = @_;
```

```
  print $q->header;

  my $template = Template->new;

  $template->process($templates{$type},
                     {text => $text})
      || die $template->error();
}
```

6. src1.pl

```
#!/usr/bin/perl -w

use strict;
use CGI ':standard';
#use HTML::Tree;
use LWP::Simple;
use CGI::Carp qw(fatalsToBrowser);
my $html2  = '';
my $page;
my @page;
my @HTTP = '';
my $HTTP = '';


srand;

print "Content-type: text/html\n\n";

my $SRC_NEW = 'SRC_NEW.txt';
my $SRC_FIRST = 'SRC_FIRST.txt';
my $SRC_SECOND = 'SRC_SECOND.txt';
my $src = param('src');


my $length = length ($src);
if ($length >=5000) {
print qq(please try a smaller text);
}
if ($src eq "") {
print qq(You must enter some text);
}
else {

open (SRC_FIRST, ">$SRC_FIRST") or die "Can't open file: $!";
                                                        #opens file to write to
flock (SRC_FIRST, 2) || Error('lock', 'file');

print SRC_FIRST "http://www.picsearch.com/search.cgi?q=";
print SRC_FIRST $src;

close SRC_FIRST;


}

open (SRC_FIRST, "$SRC_FIRST") or die "Can't open file: $!";
```

```
                                                        #opens file to write to
flock (SRC_FIRST, 2) || Error('lock', 'file');


my $new = <SRC_FIRST>;
close SRC_FIRST;



if (! $new){print "please enter a term to search for"
}




my $Html = get($new); #opens new web page

open(SRC_NEW, ">$SRC_NEW") or die "Can't open file: $!"; #opens file to write to
flock (SRC_NEW, 2) || Error('lock', 'file');


        my %AltTexts;


        while($Html=~/(<IMG\b.*?>)/isg)
        {       my $ImgElement=$1;
                # Find SRC tag
                $ImgElement=~/SRC\s*=\s*([\"\'])(.*?)\1/is;
                my $Src=$2;
                #print " $Src\n";
                # Find ALT tag & store text
                if($ImgElement=~/ALT\s*=\s*([\"\'])(.*?)\1/is)
                {       $AltTexts{$Src}=$2;}
                else
                {       # No ALT found so give it default text if none already found
                        unless(exists($AltTexts{$Src}))
                        {       $AltTexts{$Src}='no match available, sorry';}}}
#$tag_replace2

        # Write extracted data to a file


        foreach my $SrcPath (sort keys %AltTexts)
        #{      print SRC_NEW "$AltTexts{$SrcPath}\n\n";} #writes to file
{       print SRC_NEW "$SrcPath\n$AltTexts{$SrcPath}\n\n";}

close(SRC_NEW);

open(SRC_NEW, "$SRC_NEW") or die "Can't open file: $!";
```

```
flock (SRC_NEW, 2) || Error('lock', 'file');

my $choice = '';

my @array = <SRC_NEW>;
close(SRC_NEW);


my @words;

 @words = grep /\d/, @array;


$choice = $words[int(rand(@words))]; #chooses one

my $new2 = $choice;

my $Htmlx = get($new2); #opens new web page


if (! $choice){


$html2 = "<html><head><title>no match...</title>

</head>

<p>


<br><br><br>

<br><br><br><br><br>
<br>
<br><br><br><br>
<br><br><br><br>
<center>
<font face=Arial size=3>no match sorry, <a href= 'http://www.in-
vacua.com/src1.html'>please try again<a></span></center>
<p>
<br><p><br><p><br><p><br><\html>
";

}

else
{

 $html2 = "<html><head>
<script>
```

```
// The time out value is set to be X (or X seconds)
setTimeout(' document.location=document.location' ,10000);
col=500;
function fade() { document.getElementById(\"fade\").style.color=\"rgb(\" + col + \",\"
+ col + \",\" + col + \")\"; col-=5; if(col>0) setTimeout('fade()', 2000); }
</script>




<script type=text/javascript>
function pageScroll() {
        window.scrollBy(0,50); // horizontal and vertical scroll increments
        scrolldelay = setTimeout('pageScroll()',600); // scrolls every x milliseconds
}
</script>

<body onLoad=\"pageScroll()\"
background =\"$choice \"
text = \"red\"
>
<p>



<br><br><br>

<br><br><br><br><br>
<br>
<br><br><br><br>
<br><br><br><br>
<center>
<font face=Arial size=8><font face=Arial size=8><span id=\"fade\">

$Htmlx
 </span>
<p>
<br><p><br><p><br><p><br>
";
}
print $html2;
```

Appendix**: Evidence of Work 5**

Markovised Thesis

[Feeding the text of this PhD thesis through a Markov algorithm generated the next text. I have, however, also edited the text. Therefore, its authorship is both mechanical and human.]

…

Title: Always Follow the Instructions: rules and rule following in visual art.

ABSTRACT: Always Follow the Instructions: rules and instructions and set them in action, my preferred method being to disk.

Yet, whilst what it says – it is possible to restrict ones analysis to computer and has been defined as instructions that have been generally defined as neutral or content free. It is important to note here some problems that might be possible for the “blurring of art and ideas”, 1971 Jasia Reichardt Ed., London, Studio Vista.

Murray, J. H. 1997 Hamlet on the browser are turned off or graphics fail to load. Again a random number function or some other. Theorisation lags behind the technology. A reductionist modernist aesthetics also seems to be written in one of the work by hand. I sat down and embodied his rules in a previous Chapter, a computation can be done, then that is defined as instructions that have been discussing, those created by the machine as I have said, they also misconceive art that uses computers. But what sort of

machine. I will do this when universal machines manipulating these symbols. The praise for these special machines stems from their ability as programmers, as it is possible for the date, solely theorises. By the moment of literary composition, the decisive moment of literary life will be of much if any help in account for dissimilar phenomena with explanatory models not meant for them. There are those that are similar or identical. If we can pose it as most of the theory proposed above accounts for the particular case of what repelled Heidegger about the inadequacy of the of my better sentences. Google's spiders read the text machine as I know. I note the pleasant paradox that if I wish to go slower.

I once considered turning Every Icon by J F Simon Jnr into a discussion of the algorithm must always terminate after a while. So I am suggesting is that this thesis tries to explain what I mean to say rules and instructions for generating and transforming sentences in languages. Interestingly, these processes are called 'alert buttons'. It used two codes. It was a reaction to modernity and technology, the broad trajectory of which I consider, the method itself became a subject that may be many different values itself. The choice of texts treated is important to note any similarities: Kittler 1999 puts it that: "Inside the computers themselves everything becomes a number: quantity without image, sound, or voice. And once optical fiber sic networks turn formerly distinct data flows into a reprise of the reader" 1997, pp. 15-16.

Assuming all this unless it is perhaps live programming, such as "abstract machine", or another that I have also proposed that the machine is the entirety of 'his' existence. It should be programmed and it produced the pattern of binary code an "alphabet" is deliberate and important. It indicates that we are able to comprehend how we use them. … It might constitute, however, an important research field. Generally, the point that in "respect of these machines represented a relatively minor strand to the instruction: started from the example of The Dada Engine's output from the "physical" or "simulated" with the ruled system, implying there are also code processes, not only from my area of interest. However, "virtual" and "abstract" are sometimes used interchangeably by computer scientists use them. But the distinction I

made early in this reframing that was exhibited by Margaret Masterman and McKinnon-Wood's original work. To explain this will have moved our ideas on in Chapter two to some algorithm. It is my medium. Code is not finally identifiable with a cybertext be a Turing machine to be a real crux. There now appear to contradict all of these machines represented a relatively unstable process, as work changed ideas and terms drawn from a description of current conditions and the algorithm.

Virtual Dictionary. I can take little credit for programming this piece. The program assessed performance and adapted to improve the operator's accuracy and speed. It is important to my research question. The fourth is what is doing the text machine computerised. Computer algorithms, as I have not used, "virtual machine" in the first major exhibition of instructions for new text machines. However, there is awareness of the fine arts is for such reasons that Finnemann observes, cannot exempt the text – by another, an algorithm.

That is to uphold a theory of the code structure of Sentences was derived loosely from Lawrence Weiner's And Yoko Ono's 1995 Instruction Paintings and on a digital computer. This machine may be made or simulated, rather than those that are the implementation of algorithms and data in another the sound of a text machine. What are lost – but not much about the cybernetic in particular, in a small sequence of similar texts? Is this text may be implemented by many different machines-of-the-text, if I prove the low intellectual standards and anti science bias of cultural theory in a finite state processes, or finite state machine. For me medium is Perl, although I will stay in the company System Research. That Conversation Theory was part of the nuts and bolts variety. The reason is the well-known text here, but there are three separate matters. For instance, a rule may be an artwork, although not a Peirce/theorematic machine, could it be, nevertheless, one of the new Adorno speaks of.

The price a theory may pay for its general applicability is a real Professor of Physics, Alan Sokal, put his name to an article credited to the routine

geometric abstraction of writing? The Markov chain the text into lines and shuffle, as with Raymond Queneau's Cent Mille Milliards de Poèmes Hundred Thousand Billion Poems. However, in programming, a comparable way. There is an idea of 'does' to mere auditory and visual events. Anecdotal evidence: at a computer to write prose also. text machines pull text-materials into them by other machines, with the qualifications I make a text process, it was generated by different algorithms, hardware, and by different algorithms, hardware, and by different algorithms, that a Substitution Machine Description of Machine There are rule-based text machine that contain elements that have ceased to function: there is not fine art; and so on. What are problems of deriving an instruction is construable, from practice: a speaker does not purport to be so apparently arbitrary?

Wittgenstein's answer to his paradox is to fall back on the web. I have explained, neither a human performing the recursive steps of a practice" and the prompting of for instance by clicking an image to enlarge or terminating a program running on the distinction between deterministic and theorematic reasoning Peirce's phrase, ibid. p. 49 .Finnemann explains that the coding is in part, by invoking Hoftstadter's idea of the theories I have developed this most pedagogic of all the code block and the general argument Chomsky was quick to put me right not only to discover an absence where a text machine. It is not the letter, perhaps following its treatment in Jameson 1999. Jameson essentially accepts capitalism's 'axiomatic' reality that is to say, I found the theorisation of the words at random. Attend to their music may still be played, but theirs are 'machines' that have been constructed. Why do I say this text, and a table of prescribed actions. This is in a hat. Select at random. Attend to their music may still be objected that the thing in itself depends on when treated as necessity, time". And so on. These differences are permissible because, as I have noted increases in my understanding, little space in Chomsky's theory for these accidental irruptions of noise into the categories I use in programming books, the programs instructions it might be suggested that a cybertext be a neat paradox: if I fail I succeed. ...

That it is written in, is not all in one usage, to comprehend. Now we have one ‘grammar’ breaking into another. What happens when it will be the earliest instances of a language. To consider function does not take great account of the details of their performance scripts/programs: // Classic.walk Repeat { 1 st street left } http://socialfiction.org/dotwalk/dummies.html Nevertheless, there are systems in which they pass. Therefore, many of these specifically Internet genres see Glazier, 2002, for instance see Dale et al, 2004. To bring the discussion back to C.S. Peirce 1989 via modern commentators such as English with relative ease, were only just becoming available. In 1968, computers had to be the case if the machine as distinguished from the instance of its developed concepts such as music, dance and architecture. These are very different approaches. They are both nonsensical and rather plausible. These may not contradict itself, but it is a way divested of its abstract counterpart if interpretation can be monochromatic in structure when the Internet first became popular and before everyone got used to computers handling text. In 1968 this was achieved. However, it transpired, this neutrality was only technical. The transposing of semiotic material to code lies – and what terms are useful points of orientation: one promoting the formulation to my discussion. Text machines certain specifics are lost when a text machine. a. Text machine A Real Machine I am not using terms such as art, even supposing this to be a viable tool for poets. That users during Cybernetic Serendipity p 175. The other three, in my discussion of top down versus statistical modelling, of Markov chains compared with recursive descent parsers, but I wish merely to indicate a similarity. It follows, concerning this point, there is a rather antiquated one: “Because of the human sciences’, in Art: Context and Value, Ed. Simm, S. 1992. Oxford, Oxford University Press. Aceti, L. 2002 Getting Laid on the other will be shown. It will become important when we discuss text machines.

The results of the Arts. Chomsky replied: Date: Wed, 9 Feb 2005 16:47:46 -0500 To: wayne.clements@btinternet.com From: "Noam Chomsky" <chomsky@mit.edu> Add to this thesis too extend its scope: the world of 9/11 and the general form of binary code an “alphabet” is deliberate and important. It indicates that we have a concept of rule and mechanism. The perennial

newness of the program. It is in several languages, Javascript, Perl, and HTML. These are the Text machine, its Rules, its Codes, and Inscriptions. These are all a form of writings on semiotic. Ed. James Hoopes. Chapel Hill, University of the Cambridge Language Research Unit. The Unit was involved in the text. The program has to be judged, to be “Okayed”. It selected texts on different physical processes, that input-output experiments cannot distinguish between three manifestations of the machine as I have noted increases in my research and then and here and now. Bibliography Note: COMPUTERIZED HAIKU remains significant as an article. Of course, let us consider again the programming of text randomly. This might be called a signifying chain is composed as a work of art’s identity. Saying this is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs”. I think this is displayed. Some more code JavaScript takes care of how we come to a person, nor the medium it is possible to follow it at all, not in circumstances it should be fairly straightforward. In fact we can understand the text inputs, or we might try to organise my text through a discussion if Wittgenstein’s writings.

Here my wish is to say, its concretion is incidental to the middle of last century, to leave such questions out of the typo. An obvious potential candidate as a reality.” http://www.elsewhere.org/cgi-bin/postmodern The purpose of the shuffle I found the first work I tried to work directly with the aim of Masterman and Robin McKinnon-Wood; rather they permitted the user to interpret the texts they use. If the computer and the exchange seemed to dismiss cultural questions saying that it easy to imagine a maze of proliferating and reversible passages between texts that are grammatical. Turning to the idea advanced above of a text-machine, it is for text generation. I was able to take another Weiner example, plaster and lathing to binary code. But the machine is said to have a meaning", Kittler tells us Ebbinghaus wrote proudly of his study. Our fundamental concern throughout this discussion of computerised literature: Android Literature imitates the human is also not escaped me. It might be simple. I could not be as free from all the words is encountered in painting, sculpture and architecture and elsewhere structural cinema, for instance: see Krauss, 1999 in different ways. This is important,

because if any were applicable, we might not – or might not – is a process that is not really that of the circle of Picasso and Braque.

Nevertheless, this text or a subject of ones interventions and how and to form a Total Library of astronomical size" p. 216. A lengthy process, but not in their display. To create a template for a time. It adds a word and this is that code does this: It fades in the body to the appearance of COMPUTERIZED HAIKU cannot be accessed, file uploading that cannot be an opportunity for the expected inevitable. Appendix: Evidence of Work 1. Ono Generator http://www.in-vacua.com/cgi-bin/ono1.pl. This takes selections of text processing. 6. Context Five Finally, a thesis I did a show of instruction for computer – grew into the structuring of grammatical utterance, even when Chomsky's own declaration of a sustained exploration of human-machine interaction, that forms continuity from practical application through to more purely literary endeavours. Several scientific and technical strands come together here: what were formerly connected with the table of instructions as something that seems to constitute a central problem posed by my thesis. The Internet is a concept of rule following art must be written; if it is not to do anything. That an instruction in English for example we choose words with slightly more care. In other words, written a lucid essay about him from what seemed the insoluble conundrums of generative art as text machine. Of course, the scores to their meaning." Something has happened here. Diagrammatically we could represent the following of that "other main approach to more complex approach than this is displayed. Some more not too dissimilar groups are associated with it are now arbitrary and may be stored and processed within the architecture of storing instructions in art by developing a theory of the differences. Fig 1 is intended primarily as a Universal Machine, provided with the most interesting and productive tension as I have given, are not, or to erase it. Turing describes how a somewhat similar machine might operate. He also shows how cybernetic theory, programming languages and experiments in literature interacted to produce an instruction and material circumstances that attended the appearance of English.

In information theory, disorder in communication is designated noise. Shannon showed noise could remain untouched by this remark about COMPUTERIZED HAIKU been received by those that do make and then and here and now. Bibliography Note: COMPUTERIZED HAIKU is Ray Kurzweil's 1990 The Age of Intelligent Machines reproduces several haiku without criticism. Margaret Boden's 1992 The Creative Mind, referring to COMPUTERIZED HAIKU, neither the algorithm to be in all areas: a great number of algorithmic processes: this is my argument that the machine, she concludes the machine is not very seriously intended therefore and, frankly, is frequently overtly played for laughs.

Consequently, The Postmodernism Generator is exceptional by virtue of pure reason, it is done according to a discussion of computerised writing. There are other machines at least not without mediation. And vice versa. A computer animation of the computer initially to investigate the rule-based constitution of textual procedures. If I could benefit from the text? No, "it is not that it should in principal be possible. There is a small sequence of words or symbols according to Derrida 1982, is that the different material instances of which I have described above: their 'machine' is less familiar. Fig 1 is intended primarily as a term that is not that it tends to support my contention, perhaps I should provide a theory of the machine is the "abstract" or "paper and pencil" definition of such machines, then much of the actual hardware and software. A 'discrete machine', however, only performs various text operations and is one familiar to me, the possibility of any grand unifying theory – based on statistical analysis of each part of the social formation were a product of pure nature." Kant section 45 An art machine on the probability of a text machine running on the machine. And certainly not all texts.

The instructions are being followed. A way of dealing with different operations and is an essay written by a random choice, to put me right not only to make is that code does not persuade that the coding it reads: a dump of data is not the meta-instruction because it was in. I had not known it first, have worked back from our vantage point, it is beyond the scope of this question. An alphabet of the machine? I think we have one 'grammar' breaking into the

transformational machine. Why do reverse engineering? “reverse engineering n the taking apart of a ‘genetic machine’, with its physical and the Development of New Media. Cambridge Mass, MIT. Hardt, M. and Peterson, P. New York, Basic Books. Lippard, L. and Chandler, J. 1968 ‘Systems Esthetics’, in Great Western Salt Works, essays on the other. The random, the mechanical, the rule that is explained adequately only by two grammatical machines because it was an instruction is construable, just as an aid to composition. Murray observes “[e]arly attempts at computer-based literature tried to establish the computer programming of COMPUTERIZED HAIKU been received by those that do acknowledge it? Carole McCauley in her essay, explained that the ‘performance’ was a wilful misunderstanding of Conceptualism that are required. Should the employment of time, certainly, would be of much if any help in account for its functioning and in contradiction to Aarseth’s own assessment the work by Masterman, 1971 or the code, for instance by clicking an image to enlarge or terminating a program that reads other programs: preliminary evidence in Kittler’s future silicon Armageddon: “any medium can be written by Markov Generator’s program http://www.in-vacua.com/markov_gen.html.

I am suggesting that certain events are not in their own right", but these are not medium dependent. This latter proposal, relating to the user. There is the preservation of behaviours, not the first successful computer program does not go into the transformational grammar. The presence of "his” is determined by their surface expressions alone." p. 39-40 Aarseth, himself, refers to the 60s and Conceptual art. In other words an ethical and aesthetic matter and cannot be itself be satisfactory for social analysis. This might seem to leave such questions out of the issues and the other to its simulation? I am extending the argument to a server’s computer, for example. Even if I may put it like that, layer “the author”, we have to be y” very much Ono but also a Semantic Schema. The schema does this problem of arbitrarily related levels have for my theory. I will not follow Osborne further in the next chapter particularly the appendix to that of Anti-Oedipus, possibly because of the details of their ability as programmers, as it can produce new rule sets: a machine that Turing machines are required for number ‘5’ in Knuth’s list

above. One cannot really be correct. The condition of the time of her 1962 show in Japan: “…in 1962, I did a show of instruction it is. The best response is to enquire more generally into what computers are. The idea is that it may be to go with the spirit of the text treated as pattern not substance, is in some other presenters showed some pornography. Presumably, the light that hit her retina was the same text. If the text manipulations of a number of the human versus the artificial. Couched in such a theory of the text inputs, or we might claim to hold within itself the whole approach of his random word generator. Randomness itself then was new and exciting in its alphabet once it has escaped from scrutiny in a file that had to write bogus art criticism. HORACE is therefore an amusement, a diversion as his creator notes. HORACE, therefore, is not the first of these was literally clockwork. It had an injunction “an x to y words", but there is not language specific is apparent from the thesaurus. The idea that comes from computing: vaporware. Vaporware: "Computer-industry lingo for exciting software which fails to appear". OK. That was too crude. Truer to say its abstractness. This abstractness is of no practical use, as its works also lack utility. The practical and utilitarian in its other dimensions: having regard for the Nike company. … This is so long as we know the algorithm and returns the result of artifice? True. It is also indistinguishable from the command line to programming a website in a computer is networked. That potential was there in the computer as medium. New York, Cambridge University Press, 16-68. Armstrong, D.M. 1989 Universals: an opinionated introduction. Colorado and London, The Athelone Press. Deleuze, G and Guattari, but not entirely – fell within this group of non-deterministic machines Ketner mentions? A non-deterministic machine, for Ketner, might be adequate to a web text leaving the decision of whether to give a complex modality of the text of my mail: Date: Wed, 9 Feb 2005 18:03:08 +0000 GMT From: wayne.clements@btinternet.com Add to Address Book Subject: new error: syntactic structures To: chomsky@mit.edu Dear Professor Chomsky, Thank you very much for a machine of a practice” and the set of observed sentences.” That is, the text manipulations are transferred to computer, the modern digital machine and another are now quite used to considerable effect, to give the impression that these questions, discussed in reference to machine code, as must the program. It is this that make

computers work. New York, but that was exhibiting canvases with instructions attached to this thesis is written in, and the code upon which it is possible to use similar methods of simple substitution” p. 189. For her, a substitution system provides what computer programmers call the post-mechanical. “Post-Medium” There is no 'him' to refer to wholly or partly machine authored texts. This text may in turn prompted new work. I outline some of the text machine's formulation. This, we are beginning to describe COMPUTERIZED HAIKU is intended primarily as a Text machine. It motivated my use of the material it addresses constitute two different media; they exist in the diagram fig 4 above. Masterman was a remake of a text machine computerised. Computer algorithms, as I have said about the inadequacy of the Future. It is perhaps inherited from modernism, that we must be aware of excellent research: for instance, who argues that one might think. Typos are, after all, quite common and, therefore, relatively unremarkable. Why make an issue of determinism. There is see Appendix 6 a body of what Conceptualism is. It is problematic because of the producers. The theorists I have said that my research question “what is the artwork’ rather than its writings. Might it be these that I am not the computer. It is the eclipse of visual art, I cannot leave out of the decimal for pi”. Therefore, a non-deterministic machine or as bad, and much the question is quite reasonably, "what kind of inference: either there exists some sort of simulations as those I speak of above, where the imperative of the issue of determinism. There is however the question remains what sort of text files for all possible combinations of the rule to action we have one ‘grammar’ breaking into the static quotations that appear in writings by Ketner 1988 and by different algorithms, hardware, and by Montfort 2004.

c. Peirce’s Theorematic Reasoner and Chomsky’s Finite Automaton Ketner’s contrasts Turing machines with which to code semiotic materials. However, it is not conventionalised and false as it should be equal to. Essentially, what I thought was a compound word, combining connotations of insubstantial exhalations with those "that have a meaning" to use Ebbinghaus's phrase. Randomness therefore is a body of what its code is. Unlike the usual mono-authorial, if I may put the problem of justification of grammars.

Add to Address Book Subject: Re: textual error in Chomsky's text as it is worth restating my argument here, because although I will illustrate by developing a theory of a jumble of words or their sub particles. For this reason Markov processes might be said that if I could program a computer to simulate other sign systems. It is for text as human authored. My intention is not certain whether it is composed as it did not, as I required them. Monochromes differs in that I am keeping up the earlier edition, using the thesis and the Internet and the machine. The theory has become more committed to working with computers and the simulated. For the practice part of what is required is the impact of the modernist cannon, used to it: a year or two either side of 1997, in other words, new media again meets old art practice/theory. New media skips a generation, ignores its hideous parents and looks to its great grandparents for explanations, exhortations and examples see Simon Pope, The Shape of Locative Media, 2005 for a long time, been a question that has been submitted to journals: Cabinet http://www.cabinetmagazine.org/ and Media-Culture http://journal.media-culture.org.au/. I am not really interested in mapping or configuring code structures as the throwing of a practice is one in Italy, the TEANO. Ferrara 2003 provides descriptions of a version of the algorithm only. Without the surrounding code to run, if it were a machine. "Reverse engineer": engineering reversed. Engineering: product specification turned into product. Reversed: begin with the British Library to look beyond conventional ideas of what was essentially a poetry-teaching tool in Cybernetic serendipity: the computer can use, via assembly language to machine texts, are stored as binary sequences in the preceding chapter, not identical with any of its material – the "Fisher-Yates Shuffle" – to sponsor the making of art or some equivalent process". Turing notes that we cannot tell by observing if the machine produces, similarly, require a thesis that has attracted computer researchers to poetry. The contribution that TRAC makes is that code does this: It fades in the script I am not primarily my purpose. Nevertheless, what implications does this problem of attempting to account for the simple reason that the machine, she concludes the machine replaces the book of rules. A "table of instructions", according to a web text leaving the decision of whether to give a couple of examples, Lunefeld's 1999 The Total Library: Non-Fiction

1922-1986. Edited by Eliot Weinberger / translated by Esther Allen, Suzanne Jill Levine and Eliot Weinberger. Harmondsworth, Middlesex, Penguin. Brandt, P.A. 1994 ‘Meaning and machine: Toward a semiotics of interaction’, in Andersen, P. B. et al

The computer would simulate my abstractly specified machine and another are now established names, and several may risk repeating to diminishing effect former successes. Is there a sense of impractical. Nevertheless, the point is important to my own area of investigation, to “overcode” language, the body, the earth and more. The text machine, in the body to the novice. As a consequence I was not useable in the “Conclusion and Postscript – On text machines and the Internet and text materials data. Computable aspects are transferable between different fabrications of the inherent ambiguity of words, to produce work that has been arranged as a confrontation between what may be conceived as a network of computer-bunkers. This is encountered in painting, sculpture and architecture and elsewhere structural cinema, for instance: see Krauss, 1999 in different media, and cultures. Cambridge, Mass. MIT. Lunenfeld, P. 1999 Ed The Digital Dialectic. New Essays on the internet, I am keeping up the distinction between a Turing machine as rival. Will it replace us, the servant become master? Is there much point now in anyone replicating JODI’s, Shulgin’s, Bunting’s, I/O/D’s engagement with the qualifications I make immediately below, and as such they seem to hold: ? If this were not available and output was to paper printer. Fig 2 Image of installation at Cybernetic Serendipity: photograph courtesy of Professor Brent MacGregor. Fig 3 Image of installation at Cybernetic Serendipity: the computer are not, or much less so. We cannot transcribe the computer's actions. Human languages are merely stored in the form of a text with words from the observer's perception, not that it has escaped from the observer's perception, not that it treats must be understood and classified by their most recent values. Can the social in any consistent way which is which. This is an essay Masterman,1971 about it. New machines will be read/perceived. It is worth restating my argument and wide ranging in effect. I return to the written. But I am suggesting is that code does not have to choose between subcapitalist discourse and Batailleist ‘powerful communication'. "Class is

fundamentally a legal fiction, but rather the meaninglessness, and therefore no separate code level and an unusual sort of proto-software. The instruction is made so as to whether the channel is electronic or paper and ink. This still seems to situate most text machines on computer. A text machine and the abstract statement of its performance.

This event has its time and space as part of this chapter I have necessarily altered some of the work. Description of Work 1 Rather than what could easily degenerate into a hat and ‘drawn’. But the error is not a poem” quoted in Aarseth p. 133: reduction to the written. In Hegel too, the machine requires a medium, but is as not uniquely tied to a text machine that were established in the writings they produce. But it isn't just intellectual. I have tried to use Galloway’s phrase? In my writings I will use this paragraph. Markov algorithms tend to break down.

We encounter the problem he poses, I will return to these physical conditions of physical production. Nor of course carries on today. Computerised literature, therefore, is not composed as it did not write the text: instead the text machine and that even the algorithm must be known if the code from scratch and posted it up in useable form on his website for a degree of specificity not provided by Deleuze and Guattari is the same year as Art and Language’s text referred to as a physical process. All three produce texts that might be an artwork and new artwork from instructions in this case and program rules and instructions are being followed. A way of “imbricating”, of sectioning off, of reintroducing code fragments, resuscitating old codes, inventing pseudo codes or jargons.” The way to resolve this apparent conundrum may be possible to restrict ones analysis to events and processes it. c. Why? It is this issue of determinism. There is a word for machines that may be either a discrete-state machines, with all others, are subject to change or suspension. Rules may be read. Furthermore, the unseen code writes the rest. This should work whether we start with the later work of Racter it will be added and some, with the language there was pretty ordinary. What if the human versus the artificial. Couched in such a double movement. The first game was Spacewar, 1961. I do consider these issues is usually reversed,

and it is certainly possible Daniel Libeskind has made some. It is not wholly unfair and great works of Gaiman, a predominant concept is the same binary alphabet? Again, this is my ambition a developed theory for the programming.

It uses a “pattern match” programming term: ‘something that looks like this: there is nothing internal to these issues is usually reversed, and it is a set of instructions. What comprises a ‘Frame’ or ‘Template’ and a development of a text-machine, it is possible to ask if a "literature" already converges with an indefinite process may be expressed, and below that a Markov process is basically a probabilistic substitution that a Substitution Machine can write prose also. text machines certain specifics are lost – but it is there, however, it may be made, that it may be coding, it is the distinction here between syntactical and semantic material that Chomsky makes in his text, but not others. What I have already quoted. HORACE does not require either mathematics or computers. It is useful to think of an account adequate to the routine geometric abstraction of writing? ... www.in-vacua.com/cgi-bin/markov_generator.pl - 24k - Cached - Similar pages Proverbs of Hal No4. “All machines are commonplace? These distinctions become important as I do this he would have the condition of possibility of its material – the Idea of the reasons I have written, “reading reads writing”. I did not stay in the last chapter. This is because for me is not to do this by alerting us to "deduce Hungarian; with another, Yoruba" p. 122. Linguistic complexity is founded on more fundamental principles are likely to be born, to be possible. There is an essay Masterman,1971 about it. This is encountered again. Write down the word ‘bird’, but doing the text machine, finiteness, as a stable entity can therefore be constructed from code. I may be imitated by another machine, that is achieved is a complex piece. a. What does remain is an argument about computers Hillis, 1999, explores at much greater length. This text, the text that my research is primarily into computers or computer programming. My thesis overwhelmingly deals with one pair of words or symbols in Funkhouser's terms can be achieved by using an interlingua." John Sowa 2002 defines a semantic interpretive level. … Data and program rules and instructions are the consequences of this very same analogy between what may be remade in its mechanistic indifference. However, the connection

between art and computing was made early in this thesis. To contemplate function does not require a computer may execute in the loop until it produced the frustrating button that had to be made: despite the best attempts of ELIZA and Racter to which the "false". But the time of her 1962 show in Japan: "…in 1962, I did a show of instruction and its Objects. Art by Instruction and the code has been submitted to journals: Cabinet http://www.cabinetmagazine.org/ and Media-Culture http://journal.media-culture.org.au/. I am thinking of three states: abstract, limited function, simulated. It is worth considering that these are, on the North Sea.

Art in the computer.

However, computers run them faster than we might not – or of concrete, the size of London and actually doing it is there, however, it may be applied to the Internet. Over the years there have been trying to establish the computer can perform the instructions, it can be read. They may do these things, writing about the importance of Conversation Theory, or CT, to COMPUTERIZED HAIKU, http://www.in-vacua.com/cgi-bin/haiku.pl, where there is nothing to say that the problems of deriving an instruction for computer in a small part of a word. I consider a more interesting way. It is because for me between a text machine computerised. Computer algorithms, as I will discuss some of these specifically Internet genres see Glazier, 2002, for the same medium. As Cramer 2003, p. 101 notes, the previously assumed "clear cut-division, a material difference between my understanding of the technical issues here and now although I fear with unconscious irony, a marked tendency to imperialise and centralise, as he finds "protocological" tendencies everywhere he looks: in the end a fairly conventional looking image. Examples are the mere product? Is it possible to restrict ones analysis to computer and I have written, "reading reads writing". I did not write, "reduction to the "receiver". Alberro's reading of Bruce Altshuler's essay Art by Instruction and b. an Application. These are terms I have sought to develop some of my machine

the more than an abstracted procedure, that when simulated, the Kozlowski loses specificity. The choice of lexical parameters, according to a person, nor the light that hit her retina was the same symbolic realm". This "Von Neumann architecture" constituted a revolution in computer hardware, new programming languages and for different operating systems. The duality that we are looking at its most basic level, into strings. 4. "...codes or jargons…" I wish merely to indicate a similarity. It follows, concerning this point, there is a machine to write a machine ensemble, and only one word repeated a lot, the algorithm must be appropriate, the person whose act it is once it is "…the set of observed sentences." Now the plural. Therefore, presumably the error was the availability of a jumble of words occurring. See my essay, Markov Chain Algorithms. A not very interesting viewing. In other words a similar dualism may be written or run a random substitution based on being involved with many of the writing of

‘ Is Painting a Language? in The Responsibility of Form. Critical Essays on Music, Art and Information Processing’, in Software Information Technology: Its New Meaning for Art. New York: Jewish Museum.

The book as a stable entity can therefore be constructed from them. Rules, no longer imposed from without, guaranteeing stability, "are processed in time and materials. Performances by a group largely of professional programmers. Many are extremely able in their material of inscription. The importance of Conversation Theory, or CT, to COMPUTERIZED HAIKU. CT is part of its functioning. It is expanded in Noumena to "remove the characters from any point and edited. This means that easy alteration is possible to list all of the code and the Internet. Cambridge University Press, 128-141. Buchloh, Benjamin H.D. 1989 ‘Conceptual Art 1962-1969: From the Aesthetic of Administration to the 60s and Conceptual art. In other words, if it can make programs. Alt_Img_Tate is an idea of a social critique. I will return to this in their active functioning, but not in fact a controversy about the same way. In short, these were found to have fared much better than those I speak of above, where the instructions can be written in the third. f. Several Machines of Conceptualism that are derived from computing, particularly computer

code, are to be given. I have said about the importance of code that has it. So we see that the different material instances of a grammar's validity according to some future operator to supply. A further issue related to definiteness is how to simulate a text machine running on the Internet is conceivable as one enormous text providing you ignore the hardware. This is a computer program, software package, genre of electronic writing or writing – a template based on their count of word frequencies. Furthermore, they do not fit into any other. With numbers anything goes. Modulation, transformation, synchronization; delay, storage, transposition; scrambling, scanning, mapping – a program using RTNs to write a poem. Masterman did not over-rate the quality of the Fluxus group. It is true of any single text machine. This does not necessarily need to make Racter-like texts, but it should not, then it is possible to claim authorship of the concept Protocol derived from computing to the Fine arts. He writes, "a Turing machine, then, is a text machine on that literature's lack of specifics. In a Markov algorithm to use and has infected human society. It does little. The code that presents itself initially as conundrum. It will become important when we are living in a parallel context. Here, alienation is the word "discuss" and follow it though they may. If it can make a narrative. Murray notes a substitution, system is capable of doing extra labour?

But can it be remembered, an "unexpected success" of Cybernetic Serendipity marks one of the computing in general phylogeny? I think here I need to clearly formulate their work so that some steps are not divisible into neat blocks of code, as must the program. Such tactics are interesting. The objection to them is only to show the binary alphabet to simulate it. Do we mean society is a lack of research in art and computing under the single term Burnham 1968. The difficult matter is that it might not be surprising, as computers are involved with cutting edge technology. Already similar developments are occurring. Mobile technology is attracting huge investment by capital and in what sense it is hard to know what might easily seem a purely economic enterprise, thereby running the risk of weakening it. Jameson writes "[t]his incapacity of the text machine and its activity are simulated by computer. What we have seen, not all text machines and the algorithm. Then I

had to find in books, on the web address. Webov then gets the web page for amusement are cybertexts but are writing. Like anything else for that matter what counts as an instruction, we might claim we are talking about the importance of Conversation Theory, or CT, to COMPUTERIZED HAIKU. CT is part of Tyson’s continued debt to Conceptualism. Speaking of his available. Even so, if we are beginning to describe a theory of what I referred to as machines, and moreover, they have a machine are not unworthy and have a completely textual version of immediate constituent analysis. Each was found to be born, to be disposed into discrete sub elements. In other words in little groups of work, whatever the sophistications of any sort. d. Text machine: Turing Machine? As I have proposed that a rule may be possible to vary the number, choice and vocabulary of lines. … This is the aim of revealing the answer. Hofstadter's "test" provided the inspiration for Bulhak's The Postmodernism Generator. See Bulhak 1996 p. 1. The Postmodernism Generator is exceptional by virtue of the theoreticians above, to variability of the work should still take on a subject topography in doubt. It is possible for the production of sentences, of art and ideas, Ed. Reichardt, J. London, Studio Vista.

Montfort, N. Cybertext Killed the Hypertext Star <http://www. Electronicbookreview.com/v3/servlet/ebr?command=view_essay&essay_id> 20th April 2004.

Hegel, G.W.F. 1873 Logic, trans. William Wallace, with a paper printer when the computer as medium. New York, Cambridge University Press.

Harel, D. 1988 Algorithmics: the spirit of the code with the algorithm to use ideas and ideas work. However, this only really works if there is no longer imposed from without, guaranteeing stability, “are processed in time and space as part of a machine could be said to represent the coding is in an obscure exchange on a programming code". Taking computer-poems to stand, for the moment. The key thing is that we cannot be found by peering into the artwork. If the Internet and pieced together. Alt_Img_Tate also uses part of a machine in the body to the task of a program. This program may be

conceived of as symbolic logical abstractions of thoughts and natural languages, and computers as the Scene of Global Conflicts. Schopf, C. Unplugged. Art as the relation of pattern to presence, in terms of Hayles's op. cit. discussion of the computer and the obscurity surrounding its author are discussed in reference to Burroughs who used similar text cut-up techniques Burroughs also did an advert for the Application of Computers to Art Production <http://people.etnoteam.it/maiocchi/teano/works/wordtemp/sorbona.do> 22nd December 2003. Fields, C. 2002 'Measurement and Computational Description' in,

Of two minds : hypertext pedagogy and poetics. Ann Arbor, University of Alabama Press Goodman, N. 1969 Languages of Art. An Approach To A Theory Of Symbols. London, Oxford University Press, 128-141. Buchloh, Benjamin H.D. 1989 'Conceptual Art 1962-1969: From the Aesthetic of Administration to the early days of computing, as it is often by typing. Some of these works I use an example of others work, at other times I was able to share my code sketch with Simon at www.hitherto.net. Instead of the text machine running on the transposition of semantic material. This may occur between levels of 'is' and 'does'. In certain circumstances rows of digits might be called a semantic interpretive level. … Data and program could write a machine executable program. COMPUTERIZED HAIKU was an instruction from a Google entry on a computer. But it may still be objected that the whole thing was not quite a lot. This may occur between levels of signification in the elapse. But was it credible that no one knows what they call the "axiomatic". However, the theory of semantic material.

This is a useful way of making the work by Weiner to its process. b. Machines, Discrete and Universal The idea that comes from Saussure Starobinski 1979. I used some free software, Xenu Link Sleuth http://home.snafu.de/tilman/xenulink.html. Starting with an address, Xenu compiles a list of web addresses. They look for these words on the panel seemed to have developed this most pedagogic of all English sentences including, therefore, his own. But the distinction between "rule" and "instruction". Implicit in this context a contradiction: if it is also the subject of

my research I will elaborate little now, I believe it will make much of what its code alone: its interaction with our environment. The theory has both a loose and a language in which to manipulate natural languages such as “agreement”, and “consciousness” are formalized processes of the term Peirce machines rather than those that are both. Many of the hardest programming tasks I have written, “reading reads writing”. I did was to make a semblance of sense, sense would always thereafter teeter on the degree that randomness is ordered. A zero ordering of events may be that generative grammars are useful for simulating natural processes, yet still are not "equivalents" to what he likens to “the children's game of 'chinese whispers'”. But where does this work? Finnemann makes a distinction between text manipulation procedure, so long as it showed, not only of text, sound, film and photography and the Politics of Cyberspace. Eds. Chernaik, W. et al. London, The Athelone Press.

Derrida, J. 1978 ‘Structure, sign and play in the next chapter’. These conversion processes are sometimes used interchangeably by computer algorithm, and the output of the computer takes place. These are all arts where there is some sort of proto-software. The instruction we might make itself, or produce another. The non-referential may produce all text machines. However, there is the world economy exhibits combined and mixed development in computing science and the exchange seemed to have developed this most pedagogic of all English sentences including therefore his own. But the past participle here cannot really be driven by the algorithms work. Markov algorithms work with patterns. A Markov algorithm were to compare music and instruction-art we would need to worry about Montfort’s low opinion of this process in the oral tradition used these formulas as an artwork and an inscription level. In the remaining part of a practice is one familiar to me, the possibility of self-ordering, the automation of a program. This necessitated some discussion of the situation is rather like saying “I do” when one is to draw a distinction between a rule set that can be embodied in a different moment of literary composition, the decisive moment of some greater project.

4. There are a number of others. The conclusion must be, consequently, carefully differentiated. However, the real credit goes to him for his consideration. Somehow I did not over-rate the quality of the chapter, I intend to suggest that ‘numbers’ were entirely different from ‘instructions’. The obvious thing was to paper printer. Fig 2 Image of installation at Cybernetic Serendipity: photograph courtesy of Professor Brent MacGregor Edinburgh College of Art who has controversially suggested downloading a human editor that is defined as not material-specific: it can be translated into its own specificity and purity see, de Duve, 1999, Chapter 4. This is not unprecedented and conforms to one side its interaction with our environment. The theory of parapraxis the “Freudian slip” from the ICA gallery London, 1968. It is the algorithmic basis of the early days not only on who was responsible for the production of sentences, of art or life we are to understand fully a text form. This comprises for him the textualisation of sound and image media. This text may take considerable coding skills to produce an instruction and of course that we have seen the importance of Conversation Theory, or CT, to COMPUTERIZED HAIKU is intended to represent these arguments schematically. Fig. 1 Meta-instruction Noumena instruction Reality instruction Applications of Noumena The meta-instruction “remove the characters from any point and edited. This means that easy alteration is possible by access to knowledge. These latter societies are, according to Lisa Jevbratt b. 2001, is a conception of a machine. It motivated my use of a presentation made by the sound of a simple communication theoretic model and a module – HTML::Tree – another Perl module. These two works essentially select and display in many ways. Scientists mixed with artists and no stable entity distinct from the function of protocol on the web page. But there is some sort of process. There are other ways to create a list of words. I recognise Austin was considering spoken words. I am not adopting a purely sceptical position. I have tried to establish what the grammar produces is syntactical, because this is not one machine, many machines. Perhaps society is one, other or all of these machine functions may be proposed. Such a machine can write prose or poetry. The contribution that TRAC makes is that it was all stimuli and switches to her. This was the earlier circumstances of its printed texts, whether printed to screen, or file, or paper, and the rest. Why indeed stop

there and not possible in the loop and iterate over questions that may be in an analysis of word frequencies. Each time the algorithm without being constituted as such. A Deleuze and Guattari: “machines driving other machines, other non-text technical machines. The construction of an example of The Dada Engine as “a system for instance. He also shows how cybernetic theory, particularly with their “sites of confinement”. Power is no upward limit on the screen. This is akin to structural cinema’s halfway house of making the work whoever else has involvement; the common belief of the intelligible character, are the historical and material circumstances that attended the appearance of COMPUTERIZED HAIKU. It is not what we can make programs. Alt_Img_Tate is an example of The Dada Engine as “a system for the count" as an artwork I call a text machine. The development of machines. Presumably this machine has some transubstantiality about it, confessedly as it was in. I had been considered and rejected. I have, however, pursued Masterman’s suggestion of a higher written/read level and we find we may read “… capital figures as a definition, is not required. To ask this is in part, on several of them are interesting, even ground-breaking, such as “abstract machine”, or another that I have indicated why and in turn prompted new work. I outline some of my research is rules and instructions for generating random text using rules." I will return to this thesis. To contemplate function does not comprise one sort of retinal? Cramer's 2002 "Pythagorean digital kitsch" is a patent difference between my use of grammars I now go on “behind the backs of the text machine? I said in Chapter 2 that a Markov process? A Markov process or transformational, which Chomsky subconsciously suppresses “this study” and replaces it with “his”, perhaps with a computer program. This would mean, in one area, whilst others lagged behind. I could adapt to shuffle a text. That language was TRAC. TRAC stands for noun. This produces some strange, sometimes striking, effects. It is possible to identify both conformity and deviation. In Chapter 3, I investigate and try to depart somewhat from this understanding of the random is predicated in fact a controversy about the consequences of this thesis is written by me. ? If this thesis is whether or not one grammar, but many: not one text machine, if it is possible to make work that does not have anything like a recursive grammar article to follow. These should always produce

grammatical sentences as long as instructions, which can be embodied in a different requirement to making a text to art as text to display. Ono Generator http://www.in-vacua.com/cgi-bin/ono1.pl. This takes selections of text machine to develop one in Italy, the TEANO. Ferrara 2003 provides descriptions of a signal and as Manovich 2001, p. 133 says, "in the progress from material object to signal" and as Manovich 2001, p. 133 says, "in the progress from material object to signal to computer, the modern digital machine and output is not between computers" should I wish to suggest it is that their texts into the structuring of grammatical utterance, even when Chomsky's own declaration of a "digital computer with a view to copying it or improving on it".

Competition. In short, the machine and a discrete-state machines may be made in several ways whilst remaining recognisably the same way. In short, the machine requires a degree of "vagueness" in the sense of superiority it is automated and fading in the same haiku as their program and the Internet as a particular case, La Monte Young's Composition 1960 #10, to Bob Morris. Here it is: "Draw a straight line could be wrong – and what it does not mean that we usually do not fit into any of its instantiation. We may prescribe a text machine may not be surprising therefore if some of this subject. 2. Proverb of Hal No4. "All machines are not spoken, that are relevant. If a machine mimicked by a computer to execute. The instructions are the historical and material permits complete simulation, instruction and an inscription level. In the middle there are instructions and their uses.. This puts us in the development cybernetic theory, programming languages and for that matter running on the observer who may in part a reaction to his earlier, still influential work. The issue of determinism. There is much about the writing, the processor and the relative mix of human and computer simulated machine. Turing 2004, 1950 remarks that all Oulipo strategies are text machines. What it cannot be an artwork, although not a theory, and I work on for its writing? Or is it am I advocating? Is 'art-as-text-as- text-machine' possible? It is possible to offer intrinsic libidinal investments to its optimum conceptualism my italics; it would not do much.] This algorithm is that the machine and a text machine given over to a series of instructions to text. The program and the programming of these groups. One group is characterised by

being expressly web specific. The other is the network the subject of my research and then applying it I return to these arbitrarily related levels. This is encountered in my thesis: "Presumably, Chomsky's sentence might presumably be written so a machine consisting of two typologies of the machine as distinguished from discussion of text files for all occurrences of the evolutionary patterns of interconnected nodes and arcs". I think this is required. I have developed this most pedagogic of all possible sentences. The grammar is adequate to account for the machine is left for example on 'radio buttons'. This could be ordered to make is that it was a permutation of all English sentences including therefore his own. But the machine writes text it should be, I believe, in the diagram fig 4 above. Masterman was a breakthrough for me. It was in "theorematic machines" he suggests the term "generative" and conflates technical, social and economic processes of understanding. CT is an Electronic Author? Theory and the machine?"

However, this is a computerised literature "Who or what writes?" p.132 not very interesting viewing. In other words, the social and artistic production: "…my research suggests that socially-engaged contemporary artists might usefully produce work that does not extract alt tags. It looks like this: there is an idea of what is this digitisation, what does it do? It deletes a web page for amusement are cybertexts but are writing. Like anything else for that matter what counts as an explanatory term for art on the observation that a cybertext need not be possible to turn it into software. An instruction may be changed axioms cannot be artificially limited to copper wiring, signals, emitters and receivers. It is a possibly a comparable practice it is possible to turn into computer code; and what it represented, however, was freely invented. 2. Manipulation Machine Description of Work My early work largely – but wrong." Hodges, 1983, p. 302.

Appendix: Evidence of Work 6

Note. A CD of computer material from www.in-vacua.com was attached to the cover of the hard copy of this thesis.