The Role of Programming in Computer Art

Jim Andrews
vispo.com

1. The Phenomenology of Computers

The single most important phenomenological observation about computers is that they're programmable[1]. It's the computer's programmability that sets it apart from other machines and provides its flexibility as a machine. Flexibility to the point where it seems that the human mind is, extrordinarily, a product of the most sophisticated 'computer' we have knowledge of: the human brain. Yet we do not even know the fundamental processing architecture of the human brain except in broad and sketchy outline. While this is true, it does seem likely that the fundamental architecture and operations, memory storage and information processing that takes place in the brain are understandable in terms of architectures and mechanisms we are becoming familiar with via the creation of computer architectures and software.

Not that the brain is OOP or encapsulates binary trees. But our understanding of data structures will be relevant to the way that information in the brain is stored and processed; it must store, retrieve and process memory information. And while there is no methodology in the brain's processing, because a methodology probably requires a designer, there will nonetheless be very interesting rules of the game to cerebral information processing; and our understanding of how that works in computers will be extremely relevant to understanding the rules of our cognitive biology concerning the 'protocols' of cerebral information processing.

The first question we ask about software is 'what does it do?'

The computer is potently and mysteriously protean not only in its function but in its processes and creative possibilities concerning language, image, sound, communications, social organization/structure/process, shaping the pursuits and activities of people, and even in simulating thinking and emotion. This flexibility is via programming. It's dangerous and, like any advanced technology, is capable of being severely detrimental to society, but computers are additionally dangerous because they are involved in control systems of so many kinds.

Programming must be viewed as a creative undertaking in the transformation not simply of art but of humanity and the world, not an activity reserved for engineer/accountants and regulatory or strictly commercial application. Artists must

1 See my other posts on computer art and the theory of computation starting here: netartery.vispo.com/?p=1174

inform the vision of where computing is going and offer works that shape the future. They are and will, but the cultures of art and Mathematics/Computer Science are still far from the sort of communication that creates numerous practitioners. The alternative is a world shaped without art, without joy, in the image of a machine, not humanity and our spiritual and generous, exuberant nature. It is more than a battle against the forces of dullness, but mayhem is dullness and numbing even on the news.

2. The Role of Programming in Computer Art

The role of programming in computer art is multiple. Computer art is art in which computers are crucial to the creation and display of the art. Computers are crucial as medium for computer art. Not for incidental reasons like it's digitized to be distributed digitally. In computer art, the art just couldn't be seen/heard the way it's supposed to be seen/heard without displaying it via computers.

Because programmability is the key feature of computers, programming plays a crucial role in computer art. Computer art that isn't in intense relationship with the programming of the piece is probably being digitized to be distributed digitally but isn't computer art, really, but yet another poseur attempting to execute the jig.

Programmability in computer art is like the peanuts in peanut butter or the cream in whipped cream.

What makes a medium unique is invariably important to strong art in that medium. If the art doesn't distinguish itself concerning its relation to the unique characteristics of the medium, the art is probably better experienced in some *other* medium where it uses the expressive properties of *that* medium to strong advantage and, consequently, strong focus.

The role of programming in computer art is like the writing in a novel, in that the programming is the language in which the work is expressed. But it is unlike the writing in a novel in that the programming is rarely read by the audience. The effects of the programming are 'read', are experienced, but the programming itself is usually unread. The behavior of the algorithms/code is experienced, but the code itself is usually unread.

If the programming is not where a lot of the magic of the work is, it probably isn't very good computer art. Computer art is a *new art form*; I agree with the philosopher Dominic Lopes about that[2]. What makes it a new art form is not interactivity. It's programmability. The level of programmability in computers surpasses the level of programmability typically present in machines such as looms prior to Turing's work in the theory of computation. Turing invented an abstract machine, a mathematization of the idea of the computer, and with that, a theory concerning the limits of computation.

2 See my review of Lopes's book A Philosophy of Computer Art (vispo.com/writings/essays/APhilosophyOfComputerArt-Reviewed-JimAndrews.pdf

Consequently, the programming in computer art can usually be as magical as the artist's vision can manage. The programming doesn't just reside in invisible code—it also resides in the *code ideas*, the *special behavior* of the program that makes it computer art.

Sometimes that behavior supports navigation within an interactive 3D world. Sometimes it isn't interactive, such as we see in considerable generative art that dynamically generates without viewer input. In any case, it's usually dynamic; that protean dynamism is at the heart of the power of computers as machines more flexible than those which preceded them.

The main *code idea* in Aleph Null[3], for instance, is the graphic synthesizer and the instrument of color music via multiple brushes that sample from bitmaps or from complex gradients, creating a type of video and also a look of still images that requires a computer to make the images. It also provides an interface into the creation of a challenging type of art. Those who can make interesting art with this interface have imaginatively navigated not a 3D interactive world but a graphic synthesizer and instrument of color music. The *code ideas*, that is, the *special features* of the software, the *concept ideas* for the program, should be freighted with the magical possibilities of programming.

What's most important in the programming, concerning computer art, is the big picture, the code ideas, the main concepts, the main features, the unique characteristics of the feature set.

The vistas of computer art cannot be seen well without a poetic sense of the theory of computation. The theory of computation is all about the limits of computation. There is no horizon without a limiting line. Without limit, there is no vista; without limit, there is no perspective from which to view the vista. Limits provide shape. They may not show us how far we can go, but they let us see off into the distance. They inform our vision.

It's useful to computer artists to see computation as something that calls to imagination, that suggests we think of processes as interesting and as broad as *thought itself* or the very processes of the universe—or alternate universes—in our conception of the *code ideas* for digital art. This is a new form of art to change our world and change our minds.

Go boldly there. To make this a better world. Art is important in that vision.

3. Poetry and Programming

Whatever else poetry is, it involves an intense engagement with language. The fronts on which that intensity operate change over time as the media in which poetry is composed and disseminated change, and as language comes into intense relation with previously only peripherally-related fields and issues. T.S. Eliot observed many moons ago that art doesn't "progress," in the sense that, for instance, mathematics

3 Aleph Null 3.0 is at vispo.com/aleph3 .

progresses, but the materials change. Language changes; but also our understanding and application of language change.

The study of the formal properties of language, especially in the work of mathematicians/logicians from Leibniz to George Boole, Gottfried Frege, Georg Cantor, David Hilbert, Kurt Gödel, Alan Turing—and the linguists/mathematicians such as Chomsky—has been crucial to gaining a theoretical framework in which the possibilities and limitations of computation can be considered. The theory of computation, as it is commonly called, is a fascinating mathematization of language. Studied by mathematicians and computer scientists, we can think of the theory of computation as a kind of machine poetics of the media/um. Concomitantly, the resulting possibilities in computing have given rise to a 'communications revolution' that continues to change how people communicate with one another and change our image of ourselves and humanity.

At the root of the 'communications revolution' is our knowledge not of silicon and transistors, fibre optics, etc, but the formal properties of language that allow us to construct programmable languages and machines that can interpret and compile texts into instructions that these machines can carry out. I remember hearing about an MIT project in which a computer was constructed out of mechano or tinker toys, the point being that computers are made out of the theories of language and computation, and logic, not silicon.

Language has come into new relation with Mathematics and even Engineering over the last sixty years. One of the consequences of this, in art, is that the 'person vs machine' front is 'in the language' in different ways than it has been previously. We internalize the phenomenology of what a computer is and how to work one, ie, somewhat consciously, somewhat unconsciously, and the more we shape with it, the closer we come to the intersection of programming and, relatedly, mathematics.

There's the conflict between personal intent and programmed options (programmed intent): sometimes they don't line up. But also there's the conflict between technical language and artistic culture. And that is more to the 'person vs machine' conflict. In the case of programming, the troubles have a way of coming back to the conflict with the language of math, for many.

Programming must be viewed as a creative undertaking in the transformation not simply of art but of humanity and the world (like it will/is anyway), not an activity reserved for engineer/accountants and regulatory or strictly commercial application . Artists must inform the vision of where computing is going and offer works that shape the future. They are and will, but the cultures of art and Mathematics/Computer Science are still far from the sort of communication that creates numerous practitioners . The alternative is a world shaped without art, without joy, in the image of a machine, not humanity and our spiritual and generous, exuberant nature. It is more than a battle against the forces of dullness, but mayhem is dullness and numbing even on the news.

4. Computers are Language Machines

Computers are language machines. Some say they're math machines: they're computers, they compute. But they simply carry out instructions encoded in machine language even when they do math. They don't so much multiply or add, divide, etc numbers as they shift bits around according to instructions encoded in language. The gears of the machine are made of language. Language gears. Language widgets. Langwidgets. Their operation is entirely predicated on our understanding of the formal properties of language that support near flawlessly repeatable parsing, tokenization, interpretation, compilation, and execution.

A mathematical knowledge of, application of language.

Digital architectures and languages, and the visions presented onscreen—which are one of the wonders of the current age—represent a synthesis of Mathematics and Linguistics, as well as other fields of human endeavor. The digital has a way of speeding introduction between ideas, ideas and people, people and people, markets, etc. The mathematics of Computer Science now drives Mathematics in the way that the mathematics of Physics used to. The study of the formal mathematical properties of language and algorithms expressed in various types of languages drives Computer Science at its foundations, which are mathematical and linguistic.

Language has come into new relation with Mathematics. And this is a strange edge of art in the contemporary world. It is this uneasy mix that is making other changes in what it means to be literate. For instance, we see how language, image, sound, video and other digitizable media are, in various ways, 'converging'. This is fundamentally because they are all capable of representation in language, ie, they are digitizable. In a sense, the 'convergence' happened forty or sixty years ago, when we realized that language, image, sound, etc could all be represented in language.

But, of course, we are used to certain divisions in media, even in electronic media, never mind the division between print and sound, for instance. Just because we have multimedia software does not mean that we have great multimedia work because the adaptation to technology lags behind the technology by fifty years sometimes. This is because some technologies require more than intellectual adaptation but almost a biological adaptation, ie, to make these technological extensions part of the mind and/or body or sensorium requires adjustment of what McLuhan called "the sense ratios" which are basically the relations between the senses during the act of 'reading'. When we 'read' multimedia, we are listening and interacting, reading, replaying/re'reading', thinking, etc, ie, we are 'reading' various media simultaneously or in turn, and driving the thing interactively, often. But we have some experience with this type of reading since that is more or less what we do to drive a computer in the first place, never mind the experience of net.art. That is the new literacy in front of our noses.

Poets who would explore this edge, including programming, should be prepared to ditch the time worn and musty divisions between the arts and sciences and use 'both sides of their brain', as it were, though even uttering that simplistic division probably does more to perpetuate it than break it down—-and it needs breaking down because so many of our abilities are not innate but shaped by our attitudes toward different subjects. I prefer Aristotle's approach: he said that the end of education is to unite our emotional and intellectual responses to life, and that such a synthesis is necessary to a fully realized response to life.

In Orality and Literacy, Walter Ong suggests that some well-known theories of personal or cultural development explain changes that are "more cogently" described as "shifts from orality to various stages of literacy":

"...shifts hitherto labelled as shifts from magic to science, or from the so-called 'prelogical' to the more and more 'rational' state of consciousness, or from Levi-Strauss's 'savage' mind to domesticated thought, can be more economically and cogently explained as shifts from orality to various stages of literacy."
Walter Ong from Orality and Literacy; see
http://vispo.com/writings/essays/mcluhana.htm

What IQ tests test for, according to this view, is not innate intelligence but literacy of certain types. Contemporary IQ tests stress knowledge of the formal properties of language—-that is the 'math' aspect to IQ tests, which are not presented in the formal language of mathematics but in a language of symbols wherein order and placement, pattern, and other formal aspects independent of content are present and knowledge of them is tested for.

Such types of literacy are learned, and their being present as acknowledged forms of literacy in the culture determines whether the people of the culture will understand them or use them in their lives. Not innate intelligence.

In our culture, education in mathematics can be sheer intellectual torture for some people. In part because people feel the importance of math as something against which their intelligence is unfairly evaluated, and curricula are constructed to make too little time for actual learning, as opposed to yet more material to cover.

The nature of that intensity and engagement has changed over time from pre-writing, oral culture through the possibilities posed by print while retaining, usually, some connection with the breath, with the spoken, even if that has been in visual poetry's relation with sound poetry. And there are poets of recorded sound, such as Gregory Whitehead, who use the studio in a kind of audio writing that cannot be translated to the page. And there are many other types of poetry apart from print poetry such as the holographic work done by Eduardo Kac, Joseph Kosuth's conceptual/visual investigations, Joseph Keppler's sculptural critiques and sculptural comments on the book and other media, Lettrismes' concentration on the alphabet, Oulipos' attention to process, Joe Keenan's visual language processes, Jaka Zeleznikar's animated, interactive, net-based programmed cogitations and visual feasts...

Additionally and conceptually, there's the sort of work David Ayre and Andrew Klobucar are doing in Computational Linguistics. They've been creating the Language Workbench for many years; it's a word processor for experimental writers.

My own stuff has yet a different type of engagement between programming and language. My engagement with language and programming has been more on the vispo side, in kinetic poetry, but also in bringing poetry and sound poetry/music into relation with synch and interactivity. And in publishing the source code with the projects and writing tools like Windows For Shockwave and also writing articles about programming like the article on audio programming in director but also on poetics of interactive audio and game/play/poetry and so on.

For instance, I'm currently implementing dynamic creation and destruction of Windows, menus, and their elements. More detail, deeper concept: I'm implementing dynamic creation and destruction of multi-sprites. Multi-sprites are visual collections of sprites that may have visual and non-visual behavior and properties. Their behaviors are created, initialized, and attached to these multi-sprites either in authoring-mode, composing with the Score (or channeled timeline), or at run-time via Lingo methods (aka 'handlers'). Perspective: this is toward the creation of virtual entities that are numerous in 'moving parts' yet share membership in a window-like entity and may have parent-child relations between other windows and/or menus (aka forests of parent-child trees).

What has this to do with language? Several things. What is the body of language? It is changing through the digital and also the forms in which the digital is possible. And they change one another. Computer art changes the experience to engineer. And it deepens the conceptual significance. It also considers the politics, the 'position' of the work in the ethical and virtual realms of relations into which it enters in the verbo voco visuo, in the neath-textual behaviors and inscriptions, in the converse between people via the Net, in the somewhat glorious pit of language. Language and code get confused together.

Is there a language of DNA? Or is it just code? Certainly there's a whole lot of readin' n writin' of DNA goin' on. Readin' n writin' that makes, well, us. Isn't it obvious that language *is* operative there. Not a language we invented. But it's a language of DNA that is common to all creatures on Earth. Our notions of what language is (as opposed to merely code) is broadening.

Additionally, tools like Windows for Shockwave put otherwise difficult programming in the hands of people who may or may not have equal programming skills via providing drag and drop creation of multi-sprites into the Score. In a sense, poetry and art are always 'providing tools'. They are percepticons, sometimes dynamic, sometimes solely iconic. They are 'providing tools' to change your mind, among other things. Ease your broken heart or, better still, disquiet yet further your broken heart into understanding of the secret causes. But Windows for Shockwave merely provides the as-yet-not-fully-realized multi-sprite toward the body of language and virtually bodied verbo voco visuo.

The work done in Computational Linguistics and also in the related field of theoretical Computer Science has already had consequences for humanity that have changed the nature of intense engagement with language because that consequential work done by mathematicians on language has changed communications via computers and networks and what publication is, and dissemination, and also writing as 'information' that is encodable/decodable like sound and image and text in numbers and symbols...writing became 'information' and so did many other media such as sound, image, and video. All represented in a common set of languages and, from time to time, quite uncommon languages in a Computer Language field that was not with us fifty years ago but now seems like it's here to stay and both broaden and deepen the capabilities of computers. And also to understand the formal properties of language in greater depth and range. The two are strongly related and are related also to our understanding of ourselves and our own limits and possibilities since language draws a magic circle round the realm of the thinkable, beyond which lies not no language but the dawn silence of primal signs and the language of creatures without human language, which is by no means no language.

And beyond or prior to that? Some would say that the Tao or Logos is prior language, and language of necessity. In the beginning was the Word, the word of God, the will of God, the way of all things, the way that is not spoken, the unspoken, God's whispered surd, the language prior to language. Others would say that Logos or the Tao are humanly constructed notions that, rather than referring to the most natural and primal of languages, refer to the most sophisticated of human concepts, and involves inference taken to an abstract, ideal limiting case rather than being grounded in the primal. The influence of language even on the ground floors of being; language in the ground.

Language on the edge will hang around the confrontation between natural and machine language for some time, and this meeting with the primal sign. It is a 'confrontation' in some ways, and in others is a fully-meshed gift to natural language and a useful extension of our being. The confrontation between natural language and machine language arises out of the processual control exerted by the machine. Language is in process. Now you see it. What is the agenda here? Some is spam and in the can before it hits the screen. Now you don't. Some is chit chat. Some is publication. All is in the body of language. But whether it is chit chat or publication, etc, depends on the other properties of the body and what is being said, and to whom, and why, with what agenda and behavior, and with what awareness of and respect for the soft reading.

Poetry goes where language goes. And not only where it is a mass medium, but where language is being most intensely transformed and used to gain knowledge, where it is being expanded. Some poetry is eloquent with language in ways that weren't even considered poetry before. Some of my favorite poets aren't even considered to be poets at all.

⌛

I found much of this essay in 2018 in a 2003 file apparently written to present at trAce in England. I presented War Pigs there, in Nottingham in 2003, instead. And apparently forgot about this paper, which was unfinished. I finished it in 2018.