Programming As Poetry

Johnston David

Programming as Poetry

A few brief musings on Antiorp, Kurweil, and Stallman
> David 'jhave' Johnston

Prototype Prelude

Organic models of consciousness are suggestive of waves. Things come into ripeness, pass in and out of balance, find their peak and then slide away. There is a flexibility provoked by continual change; and from the waves of ripenings arise the roots of desire (looking for what is ripe, grieving what is decaying or gone, navigating crests and troughs). It is these organic rhythms, these themes, subtle and diverse, which art (often) emulates and harvests.

Mechanical models suggest sustained efficiency; a calculable performance level from initial production until the end, when somewhere, something (a piece, a part) fails or breaks. Machines are modular. Parts that fail can be replaced. They do not grieve (yet). Or so we think, that thinking machines do not feel.

/**
Numbers and lines have many charms unseen by vulgar eyes,
and only discovered by the unwearied and respectful sons of Art.
**/

(E De Joncourt's 1762 quarto,
On the Nature and Notable Use of the most Simple Trigonal Numbers
cited in Babbage, Charles, Passages from the Life of a Philosopher, pg54)

main() {
For centuries the separation between arts and science has been generally sacrosanct. Ideologically distinct stereotypes have evolved to ensure the two camps are not confused. The scientist is left brain, neat, tidy, thorough, patriotic, sane and logical. The hair is short; the demeanour, brisk. The artist is passionate, intense, inspired, emotional, non-linear, revolutionary and often insane. The hair is eccentric; the attitude, wild. In actuality, these dichotomies dissolve under casual scrutiny. It has been assumed that disciplines such as programming are cold, rigid, seriously analytical and antithetical to the symbolic, ambiguous, poignant, often excessive realms of poetry. To some degree this is true: there are predominant traits to each discipline. Yet poetry and programming share more than strong affinities. Each is language-based, obsessed with conciseness, consistently evolving, modelled on consciousness, and inscrutable to the uninitiated (think of James Joyce reading C++). Each uses language in ways that involve leaps and circular paths; each requires an arduous concentration that ultimately relies upon reasoning which invokes intuition; and each is closely related by a shared goal of precise communication of complex realities. But in the contemporary world, the number of human languages is decreasing, while programming language are proliferating. This shift in the balance, for those humans whose capabilities draw them toward language-based play, is creating a migration toward the vivid daunting hyper-entropic evolutionary fields of creating computer code. Human poems are for emails or listservs; computer poems are for the compiler and CPU. Integrating ethical imagination into code, a new generation of programmer-poets are implementing hybrid forms that move beyond ancient dichotomies.

In order to effectively examine whether programming is an evolutionary descendant of poetry, it is perhaps wise to begin by asking: what is poetry? What is programming? Assume (for the moment) we are reading out loud a poem from a page or screen. Each word evokes a sound and a meaning. There is an external visual element: how does the

poem appear? There is an aural component: how does the poem sound? There is an internal visual component: what images does it evoke? There is an intellectual component: is it well constructed? What ideas or concepts is it dealing with? There is an emotional element (a tactile visceral sense): is the poem true? What does it do to the heart? And then there is the synergistic totality of all these elements which mysteriously combine to create a presence, an energy, a living actuality. In short, there is an infinite array of levels and sensory processes involved in provoking an experience which will be called 'poetic'.

Are these same processes and levels invoked or inherent in programming? Obviously, a series of distinctions could be made (i.e. where is the aural component to program code?), Nevertheless, what persists is the truth that both these disciplines are in focussed relation with symbolic systematized languages believed by their practitioners to be capable of representing truth, beauty, consciousness, and even love. For example, when programming an automated task such as a text reader for the blind, it is possible to speak of love as a motivational aspect of programming. Fundamentally what distinguishes poetry and programming from other literary disciplines is their mutual focus on the word as talisman capable of being a reservoir for consciousness. In poetry, this consciousness is subtle and charismatic; in programming, explicit and binary. In programming, this audacious faith in words is palpably manifest in that every functional program is an emulation of thought which through integration in an apparatus manipulates matter. In addition, the urge to evoke identity from absence is central to programming; digital intelligence has evolved from the void; it has emerged from the unconceivable; it is the volatile product of severe imagination linked with the will. As Kenneth Patchen, a early twentieth century visionary concrete poet pioneer says in his seminal work Sleepers Awake, "Art is not to throw light, but to be light…." (Patchen 268) From this perspective, both poetry and programming are activities devoted to dissolving any residue of our species-centric faith in our existence as the only repository for consciousness.

As radical programmer Netochka Nezvanov (a.k.a. Antiorp) wrote online at her site http://www.m9ndfukc.org/korporat/=cw4t7abs.3nkod0r..0+2.html :

> *When civilization began human thought was the only available resource for processing information. Today information processing capacity is a trillion times greater with almost all of the increase being electronic. These electronic circuits do not need the smoothly varying data that human minds prefer - for them the jagged terrain of the real world is as absorbable as the artificial sphere of the geometer or the torus of the analyst. Utilizing the new evolutionary intermaths they have the ability to go where thought alone cannot.*

In the early 90s Antiorp was renowned for besieging listservs with ASCII poems, in some of her less parseable posts to listservs, Antiorp utilized a style that caused one commentator to compare her to Karl Schwitters' infamous sound poetry (example: "Ursonate"). She often playfully deletes random letters and replaces them with symbols (typically a 'y' will be replaced with a "!"); or the ASCII code for the letter may be substituted, or the word is condensed, or 'k' is used in place of 'c' ('korporat'). The morpheme-splicing entropic gymnastics of her idiosyncratic style are consistently astonishing. Turbulent disruptions in the conventional fabric of language decode into thick paradoxical helixes of nonlinear agility. It would be glib if the content were not so often radically involved with questions of gender, technology, armament-industries, korporat corruption. Ideologically perverse in its fanatical adherence to a fluidity which defies definition, to read some of the ASCII interspersed rants of Antiorp is to be introduced to neuronal clusters where language seethes below the level of normative consciousness.

Raw brain-heart processes, analogous to bits and byte-streams of the body that often may seem like nonsense and hallucinations to our high-level conscious-identity interfaces, have historically been rich fuel for poets. Contemporaneous with Duchamp's introduction of a toilet into a gallery, Hugo Ball and Kurt Shwitters at the Cabaret Voltaire introduced guttural grunts, growls, yelps, and streams of imaginary languages into their poetry performances. Manipulations of graphic elements followed, --the page evolved into arbitrarily lush turbulent horizons. Eventually under the glandular deconstructivist contortional physics of postmodernism, poetry became whatever we call poetry. Into this expansive atmosphere, in the mid-80's, Ray Kurzweil (a renowned programmer instrumental in developing speech recognition software, the first CCD flatbed scanner, etc….) launched *Cybernetic Poet*, a computerized mathematical modeling system which emulates a diversity of poetic styles as it spontaneously improvises new creations. Essentially, it exists at the interstice of a new field of creation where collaboration becomes analogous to initiating the genetic form of a programmed process (software) which then develops work independently of its creator. This perpetual motion would have delighted renaissance theoreticians. This type of

software constitutes one feature of an evolving nascent AI landscape which will probably soon supercede the creative output of all humans.

Antiorp's verbal play on the listservs is clearly a more-traditional human-centric evolutionary offshoot of concrete poetry. Resonant comparisons could be made to b.p. nichol or "Wind" by Eugene Gomringer (source: http://www.webdelsol.com/Perihelion/p-theory.htm). The twisting of context so that words require effort to be decoded as both visual elements and as signifiers is the primary contribution made by the concrete poetry school. Analogous to this dispersion of immediate meaning (in subtle defiance of the ease of gratification favoured by the products of our entertainment manufacturers), computer code often moves in convulsive ways challenging our notions of its linearity.

Example, after signing up to Antiorp's email list, I received an email which began:

// =_?
mot!vatd.adapt!v.bhav!our = funda.mentl 2
--learn!ng bhav!ourz - 3 parad!gmz (c.note
-s!ngl.st!muluz.prezentat!on
----------|
-ekzposur. 2 relat!onz among. st!mul!
----------|
-ekzposur. 2 relat!onz b.tween rezpons.z + st!mul!

!ntell!genss
1)

Virtuosic deluges of messages of similar style intercut with long samples of her computer code (interspersed with occasional bitter spasms of political invective) provoked her banishment from several listservs. She fulfilled the typical model of the manic artist, the babbling obsessive, burning so brightly with fevered incandescence of inspiration that online communities occasionally reluctantly turned against her: she was keeping everyone awake to too large a spectrum of reality. Her diffusive non-linearity imperiled the concentrated topics to which technical listservs are dedicated. From the linguistic evidence, Antiorp would seem to have some personal knowledge of poetically-extreme states (a subtle clue in her exclamation: "- *! juzt forget 2 eat.*").

Yet programming also (like some forms of formal poetry) requires an arduous logical concentration. Extreme clarity and absolute precision are necessary to perfect syntax and achieve optimized functioning. In this respect, the dichotomy that arises from Nietzsche's analysis of poets in terms of Apollonian and Dionysian streams must be resolved in the creative programmer: imaginative fervour and logical rigour need to be synthesized. The quality and breadth of much programming is definitive proof of this resolved synergetic fuel: exploratory abstract imagination, and rigorous analysis leading to clarity of practice.

Analysis of programming in the early 60's "demonstrated that all programs could be written in terms of only three control structures" (Deitel & Deitel 61) First, the *sequence structure*: the next line of code is read after the previous line in sequential order; same as we read word to word, line to line on this page. Second, *selection structure*: a choice is made; for example, right now, do we continue reading? or do we go do something else? if we are bored or have an appointment, we quit reading; we make a choice between paths. Third, *repetition structure*; example: we may continue reading this paragraph until we understand it or until we get tired of the attempt; in other words, we re-read it until the condition of comprehension is satisfied or until we are tired. In pseudo-poetry-code this would be written as a type of cryptic pseudo-haiku:

not comprehending
so read again
unless bored or
impatient
go on

Basically these three control structures are fundamental conceptual models for approaching programming, or for approaching poetry (or even neurology). *Sequential* structure is fundamental to how readers construct narratives

within the flow of an emotional-intellectual logic. *Selection* is intrinsic to the accretion of patterns into symbolic nets, resonant fields which yield meaning. *Repetition* is the chorus form, the refrain, repeated with a tiny shifting value. These minimal structures necessary for programming are remarkably consistent with literary analysis. Intuition can be understood as the coalesced excretion of similar *reiterated subconscious* pattern-recognition activities. Epiphanies are the ripe fruits our glands pluck from the orchards of structured intuition.

Peripheral to the development of these technical, formal structures, vast corporate structures of ownership coalesced around the productions of programming. These structures have yielded a typically-insane disparity of wealth. Adrienne Rich (poet, feminist, lesbian) says, "…if we care about the freedom of the word, about language as a libratory current, if we care about the imagination, we will care about economic injustice." (Rich 165). Richard Stallman, the infamous MIT programmer deeply involved in the open-source software movement who developed GNU, is one example of a renegade programmer deeply committed to activity that transcends the hegemony of materialism, and challenges the tyranny of copyright. Stallman began the GNU project in 1983 by announcing, on a listserv, his intention to create an alternative to the expensive proprietary UNIX operating system and give it away *free*. He invited others to help. While egocentric modes of behavior are certainly operative in almost all human endeavours, there is a conscious thread here of openness toward unity, an explicit awareness that all beings have a right to be involved in the production of their environment, and that important resources should be shared not wasted or hoarded. These ideological concerns are to some degree a prevalent thematic strand within the history of poetics and programming. As Stallman defines it, free software is

> *....a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the*
> *users of the software:*
> *The freedom to run the program, for any purpose (freedom 0).*
> *The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.*
> *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
> *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. (freedom 3). Access to the source code is a precondition for this.*
> (source: http://www.fsf.org/philosophy/free-sw.html)

Stallman has been dubbed "The Prince Kropotkin of Software" (a reference to the Russian anarchist) in an online biography by Nikolai Bezroukov . Akin (slightly) in appearance and philosophy to Alan Ginsberg, and comparable to Glenn Gould in hermetic-obsessive style, Stallman seems far from the sanitized logical technician stereotype of a scientist. Why not label his programming work poetry and call it art? It is socially compassionate, extraordinarily intellectual, and devotedly imaginative. In the way that a Bach sonata is constructed, so too these massive operating system programs are interleaved and interwoven symmetries evocative of structural beauty. Writing a work of the size of GNU is comparable to the achievements of Dostoevsky. And unlike the contemporary artistic model of individual authorship, the open source movement is a collaborative environment, these are truly creations of collectives.

It is clear that in some cases, programming is poetry; it is an art; it involves a total commitment to language as a living energy, language as a crucially nutritive malleable living entity; language as capable of being expressive of the highest modes of awareness, language as an active presence which can modulate our world. In their physical actuality, acrobatic twists and plays of meaning shimmer through the surface of both disciplines. The words in each case become vessels of consciousness. Yet, will programming ever match the devout emotional commitment to raptures and the heart-centric luminosity of poetry? It is as if the descendant of a richly sensual creature has withdrawn from the excesses of its ancestor, withdrawn in reaction into a harder rigid style of being. Logic without emotional intuition, reason without passion, order without grace: it is easy to stigmatize programming as a dry barren offshoot of a flourishing source. It is easy to say: logic is plundering the human soul of spontaneity, sterilizing the seeds of authentic creativity, burying the radiance of innocent spontaneity beneath an obscene avalanche of technological commodities. Nonetheless, amidst this turbulence, there is a slender thread where programming embodies the living essence of poetry, and that essence is mysterious, a river of words evoking worlds, --and perhaps this earth is among those worlds; and we are the semi-autonomous, self-replicating aspects of a vast fiction, a vast tangible poetical-programming project.

return ; }