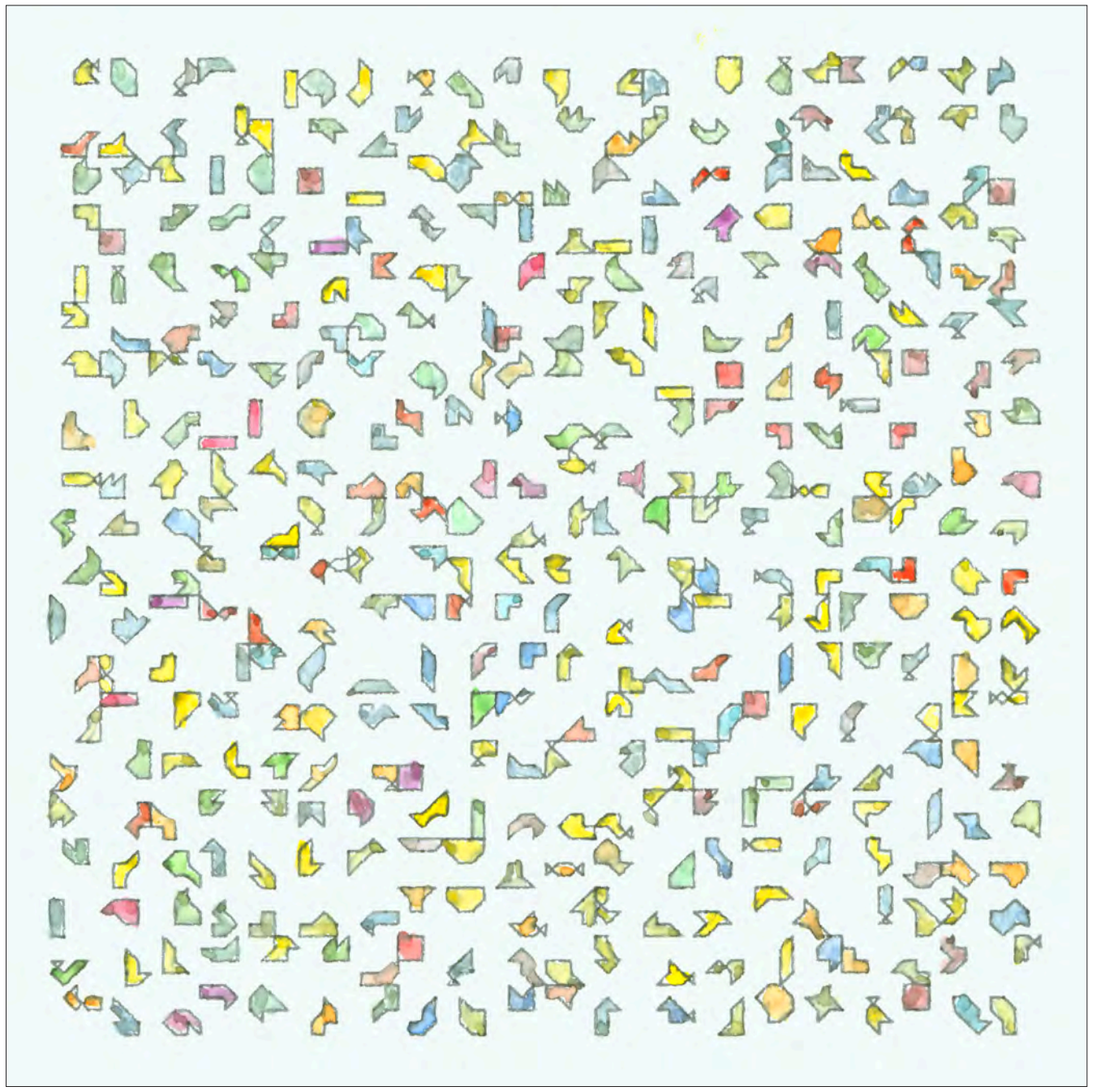


[203] Peter Beyls, *Untitled (GTT)*, 1992, drawing, ink on paper and watercolor, 690 mm x 590 mm each

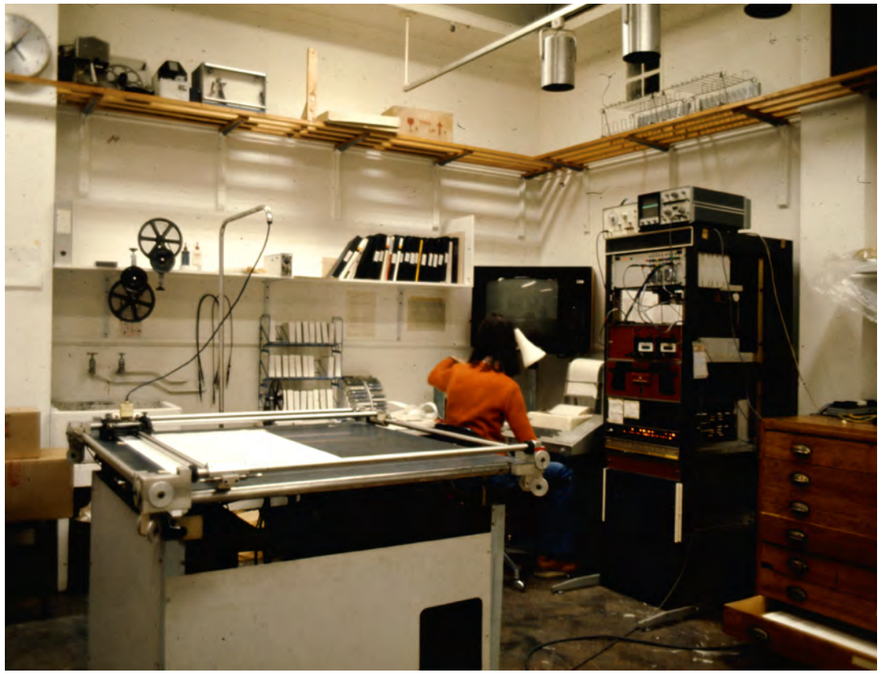
This essay embarks on the development of a conceptual framework in which to appreciate software as a medium of aesthetic engagement. It is highly idiosyncratic, profoundly intimate, and informed by my long-standing commitment with software as art.

For over half a century, programmers have been developing software with aesthetic objectives in mind. What is the true potential of software as a means of artistic expression and what are the consequences in terms of instrumental bodily engagement? From my point of view, software is an exploratory device; it allows one to speculate on the generative potential of a field of concurrent options. Most of the options are unknown to the programmer; they have not revealed themselves, but they exist in a largely subconscious personal search space waiting to be discovered. Consequently, the act of speculative programming is an active and functional means with which to approach the unknown. Programming is also about a fascination with the notions of *identity* and *complexity*. The incentive to create something is typically triggered by vague and ambiguous ideas. One aims to externalize an abstract thought into a formal description, and software constitutes a partial, symbolic rendition of behavioral aspects of the triggering idea. Typically, the sparking of ideas and jotting them down in software—capturing them on the fly—is an impulsive endeavor by its very nature, perhaps approximating action painting, albeit with cognitive building blocks.

Once the idea is implemented in a computational procedure, the instigating awareness starts to live *in vitro*. The functionality embedded in the program exposes itself to the programmer, and the system talks back to the artist. A spontaneous conversation develops between the programmer and a distant cognitive echo of that same programmer. Compulsory; a dialogue characterized by many qualitative overtones. For one, software provides a tangible means to interact with purely conceptual constructs. Ideas become materialized by way of the fluid medium of software. As an extension, software reflects initial contemplations from the dynamic management of the logic embedded in the program. Some behavioral traits might emerge that were not anticipated by the programmer. Such unanticipated behavior provides instrumental feedback on three levels: (1) it sheds light on the true nature of the instigating idea, (2) illustrates the complexity of the conceptual trajectory exposed so far, and (3) offers valuable hints for future actions. Accordingly, programming steers a creative process of exploration and discovery, because—as an implicit side effect—an aesthetic commentary is synthesized spontaneously.

Being exposed to the consequences of one's actions (in software) raises inevitable questions of identity: what is the nature of my conceptual universe (personal search space) and is there a way to probe it through software? Artistic identity might be associated with a specific style: a recurrent array of idiosyncratic stylistic features pointing towards a given artist within the inclusive framework of a given culture. One might attempt to formalize style descriptors in software by assuming that explicit access to the underpinning knowledge is freely available. In my opinion, such wishful thinking is problematic; initial ideas are not equivalent to a set of clear-cut alternatives. In this light, I prefer to envision my personal search space as a floating network, a loosely coupled web of ideas, impressions and motivations—everything that constitutes my psychology as a living organism. Most of the components in the network are unknown because the network is of infinite dimension; it extends way beyond the obvious dimensionalities of culture and nature.

The network metaphor enacts a flexible, organic database, acquiring new impressions on a continuous basis—it is alive. The contrast between integration and expression is substantial; the network instinctively assimilates and integrates new impressions implicitly, however, it requires significant effort to express its components explicitly, for example through the medium of language and, consequently, through the formal language of computer programming. In the long run, the network may be referred to as a personal belief system, a residual entity issuing from the dynamic organization of its components.



[204] Peter Beyls at work, Slade School of Art, University College London, 1977

It is extremely difficult to externalize explicit knowledge from the system because it is at once immense and complex. This implies qualitative connotations, since the notion of complexity suggests structural relationships between components in a given biotope. Abundant examples in nature offer evidence of operational beauty and enduring integrity through the forces of emergence; a mass of simple components interact locally and without supervision, the net side effect is emergent structural functionality. Some unequivocal form or behavior emerges from low-level beginnings; there is no need to instruct the system, it responds naturally to its environment. So do artists, as they respond to the initial conditions and pressures of their socio-cultural surroundings, in other words, the network is challenged with the impact of an infinite number of impressions. Artistic incentives rooted in software can thus be considered functional responses to internal psychological drives—perhaps, by definition, a yearning to uncover the inherently invisible.

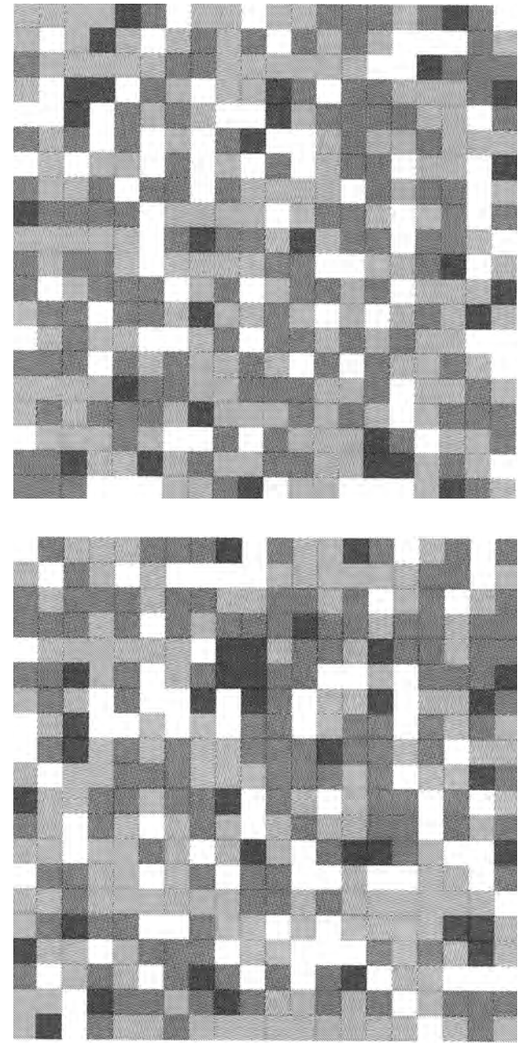
A process of speculative computing develops, and one tries to implement vaguely defined ideas by trusting the guiding hand of intuition rather than a practice of comprehensive understanding. In fundamental ways, therefore, knowledge is missing or incomplete, however, programming might suggest a deceptive impression of power. Nevertheless, when an idea is formalized and implemented, its consequences feedback to the artist, which triggers further programming activity. The impact of programming as a creative, exploratory device is significant as it potentially reveals unanticipated conceptual territory. In the long run, a certain material residue is acquired; therefore, we can aptly say that the work documents itself. More precisely, there are two sides to this coin: (1) it reflects the consequences of engaging in a certain pattern of thought, and (2) it reflects the impact of the computational medium itself. The hand of the binary system, the limitations in creating symbolic representations, the compressive organization of bitmap-based representations, it all reflects the idiosyncratic nature of the

medium—perhaps this is why I love the work of, say, Ryman, Rothko and Newman so much. Fundamental painting and post-painterly abstraction suggest a materialistic focus to the production and perception of art—disclosing and emancipating the raw edges of the medium—conceivably speaking closer to the heart than to the brain. However, I believe programming dissolves this unfitting dichotomy—programming pragmatically exists as a support for abstract procedural thinking consenting the tangible manipulation of cognitive building blocks within a given creative process.

In essence, programming aims to facilitate the genesis of the unknown into perceptible artifacts. A link with the body is significant; it suggests responsibility for shaping physicality, that is to say, addressing the sensual aspects of matter. Programming becomes manifest as a self-organizing process; it accommodates input, it implements abstract subject matter, it generates perceptible artifacts—in the end, digital artifacts must be trans-configured back into the analog world so people may experience them with their bodies. Trying to understand what essentially remains undefined through the actual creation of hard facts reminds us of Richard Feynman's statement: "Everything I cannot create, I cannot understand."

The production/programming process itself is equally informed by transitory results of passing value. At some point in time, when engaged in a programming cycle, the program might suddenly suggest a major swing in context: an unimagined meaning has seemingly accumulated and suddenly emerges as an external attribute of that process, typically reflected in an image on a computer display. That image is referential, it reflects a non-linear search process, and therefore, it is much safer to claim that a computer is an imagination machine, rather than an imaging machine. A few emergent waves of sudden excitement might develop within a single programming session, more macroscopic waves may surface when addressing the totality of work within a single given computational idiom, like, for instance, rule-based systems, cellular automata, or distributed agent systems.

As we have seen, the production process is informed by continuous feedback that pushes the creative act into an iterative speculation/evaluation cycle. Unsurprisingly, such a process features aspects of improvisation, which might be imagined as resembling the activity that takes place in a group of free jazz musicians. One understanding of improvisation is conversational, like a dialog driven by the appreciation of the simultaneity of different individual points of view. Musical activity aims to bridge the gap, to provide a fitting answer when facing a specific temporal musical



[205] Peter Beyls, *Untitled*, 1977, two drawings, ink on paper, 350 mm x 300 mm each

stimulation. Of course, as with computer programming, the process never stops because the objective—the imagined aesthetic goal—is a floating target, it only subsists as a short-lived temporal definition.

Generically speaking, we think of the concept of *change* as a first principle. The appreciation of change drives the creative act, irrespective of medium, creation is anticipation-driven. Editing a computer program aims to create a new perspective on the thinking context at hand; new/different output releases critical evidence on the orientation taken—the program exhibits qualitative feedback—therefore, computer programming is thought of as a process of artistic introspection. As we are stressing the dynamic, oscillatory nature of this process, it is tempting to view the notion of art as a *qualitative oscillator*.

Improvisatory attitudes to creative software development necessarily imply embracing the challenges of conflict and negotiation. The psychology of *conflict* is a powerful foundation for contemplation and vitality; for example, contrasting objectives embedded in an algorithm may give rise to unpredictable beauty. Artificial systems based on multiplicity, that is on the existence of competing alternatives, are equally to be considered in this regard—it introduces another first-principle, the principle of *co-existence*. Multiplicity argues for the simultaneous presence of many forces coalescing into a single unified dynamic agency. Even abstract forms of social interaction might be represented in software, a collaborative procedure of *social computing* in the virtual substrate of the machine. This is complementary to the collaborative attitude of the human programmer towards his silicon partner—avoiding the risk of engaging in anthropomorphic allegory—it is probably better to simply view machines as extensions of a dreaming mind rather than a drawing hand. However, this partnership is heavily conditioned and implicitly non-neutral (in sharp contrast to the corporate perception of the computer as a logic machine); humans and machines co-exist by way of their mutual relationship. It brings to mind the notions of shared initiative and creating with common objectives. A programmer undoubtedly delegates some of the responsibilities for aesthetic decision-making to machines, so a particular creative connection develops.

Inspirational bonding with machines might be addictive, in particular, a programming session may escalate since we are dealing with human and machine engaging in a cycle driven by positive feedback, from the science of complex dynamical systems, and we know this may result in chaotic oscillatory behavior. Self-motivated activity develops, it propagates into our personal search space, and forms the basis of *conceptual navigation*—the continuous exploratory mapping of ideas, thoughts, opinions, and attitudes within the intimate biotope made up by man and machine. The act of programming itself triggers reasoning progressions induced by physiological processes. Most arguably, dopamine and other chemical agents are released in the brain; they establish the biological basis of addictive programming. Consequently, the artist should take notice: programming provides grounding for both liberation *and* addiction.

Intimacy is obviously linked to co-existence, who you really are, how you choose to express and engage yourself when faced with a particular context, whether made up by humans or machines. A program is not merely a mental mirror providing skewed consequences of our thoughts; it overtly *comments* on the current state of affairs as currently formulated in a textual description, the program. As explained above, it suggests qualitative information creating a particular, intimate and sociable environment. The ecosystem comprising the programmer and the program holds the potential to uncover the true identity of the programmer—identity in terms of his position in the broad framework of human culture. In appreciation of the floating network metaphor introduced above, let us consider personal identity as proportional to the dynamic intersection of the infinite personal micro-network and infinite macro-network of global existence. Dynamic intersection resonates into a unique psychological pattern; the programmer actually being the first to realize. So the dynamics of programming materialize into a conception of identity, more precisely, it is instrumental to the synthesis of a personal belief system as introduced earlier. Ultimately, we speak of *acting out of ignorance*, and this forms a foundation of immeasurable freedom in the creative act. Of course, we cannot start from scratch; our minds are heavily conditioned, in particular, while living in a globally networked

society. Otherwise, it is an illusion to assume that the intention to create a work of art can be approached as something under formal control. Perhaps, too much focus on *structure*—the creation of highly and explicitly (hierarchically) structured terminal objects—suggests a distorted depiction of programming. Viewing programs as behavioral (rather than structural) engines comprises a more viable approach and accommodates a more open, unprotected attitude to the computational medium. Then uncertainty, ambiguity, hesitation, and ignorance, rather than the illusion of knowledge, might motivate engaging intimately with machines.

Note that certain aspects of human creativity might be conceptualized as psychological processes, develop into particular algorithms and further materialize, for instance, in a family of machine drawings. This does not constitute a process of pure automation; nor is it a process of autonomy. Genuine autonomy, for example, implies an organism facing an unpredictable environment and developing expertise within the process of interaction itself—in other words, it is not (pre-)programmed. Such a prospect advocates a wonderful vision for a programmer; it suggests the existence of creative forces that are just waiting to be discovered without having to probe for them explicitly—thus the human programmer simply engages in creative exploration of readily accessible life-forms, free of any preoccupation or prejudice. While evaluating a given program, we expect the program to talk back with a certain authority, perhaps, a minor form of autonomy, so it will surprise us, reveal aspects of our own thinking that we were unaware of… As it happens, human and machine are mutually adaptive in a social climate of relative independence, yet equally close partners in the creative process. We speak of a resourceful background of shared autonomy driven by the dynamics of an iterative programming cycle. The cycle itself surfaces as qualitative oscillations; the programmer speculates on the generative potential of one's beliefs while the machine talks back, not just as an amplifier or a filter, but as a psychological hypothesis of higher dimensionality.

As a consequence, *simple thoughts*—given many distinctive species of them, and taking them in open confrontation—will trigger a global implication to emerge spontaneously. And known and unknown dimensionalities will coalesce seamlessly. Conceivably underpinning a universal faith in Heisenberg's principle of uncertainty; you can't know who you are and where you are in the same instant of time. Valuable insight for traveling in any dimension.

Peter Beyls, NYC, June 2nd, 2014