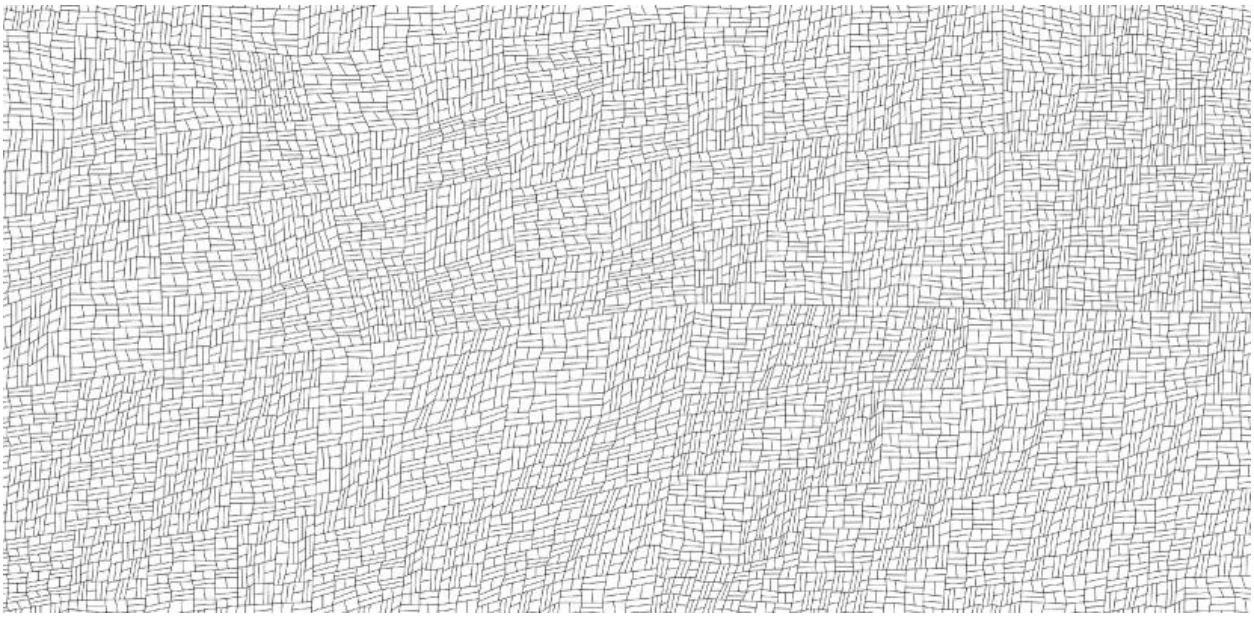
COMPUTATIONAL DRAWING: CODE AND INVISIBLE OPERATION

Brogan Bunt

Drawing upon my own experience in developing the algorithmic drawing project, *Loom*, this paper considers the relationship between conceptual and non-conceptual dimensions of drawing in computational art. It is concerned particularly to reflect upon the nature of this aesthetic labor, which involves not only programming but also the blind space of procedure.



Loom, 2011, Brogan Bunt, archival ink-jet print (author's image).

Subdivide an initial polygonal shape into a set of smaller polygonal shapes. Apply the same process to each of the polygons in the new set. Continue recursively.

This instruction could be regarded as the concept informing my recent exhibition of algorithmic drawing work, *Loom*. The work explores aspects of recursive geometric subdivision. Simple shapes are subdivided into further smaller shapes. Applied many times over, complex patterns and textures emerge. I have reservations, however, about expressing things in these terms, since my aim is to question the notion of a purely conceptual space that precedes and dominates the sphere of technical implementation and execution. The instruction above echoes the form of a Sol LeWitt wall drawing statement, yet it can hardly be said to be purely conceptual. It is expressed in linguistic terms. It is mediated through the impurity and materiality of language. More specifically, in conceiving the aesthetic possibility of polygonal subdivision, I am drawing upon particular programming constructs and dimensions of computational process. My creative ideas are shaped by the thinking of data structures, algorithmic pathways and iterative patterns.

However, my interest here is not so much in demonstrating the various ways in which my conceptual drawing statements are inevitably affected by the space of programmatic logic and implementation, but

in attempting to reconsider the relationship between the conceptual and non-conceptual aspects of computational making. Drawing upon the model of the Jacquard loom, my work positions computational processes as mechanical means of weaving virtual cloth from simple algorithmic patterns. My aim is to engage with the compelling power of computation, which is linked for me to the mystery of its dumb operation - its strange invisible labor. The computer is bound by regimes of instructional necessity, yet the opaqueness, scale and speed of its processes suggest an uncanny agency. This paradox is vital to my work. The abstract algorithmic schema – whether expressed as a conceptual statement or as a formal body of programming code – is never sufficient on its own. It must be played out on a surface. It must pass from the uncertain consciousness of code to the uncertain unconsciousness of iterative procedure. It is precisely in the tension between algorithmic conception and repetitive, non-reflective enactment that the process of drawing takes shape.

COMPUTATIONAL LABOR

The images in my *Loom* exhibition depend upon a work of programming. However, another dimension of labor, the computational labor of machine execution (drawing), is also relevant. How can we make sense of this work? Can it even properly be considered a genuine form of labor?

Within the Hegelian-Marxist tradition, labor serves as a vital index of human rational and social activity. It represents the sublimation of immediate gratification towards the goal of producing useful things. [1] It is something that we undertake and endure with other ends in mind. In this sense, a key aspect of human labor relates to the awareness that we could be spending our time differently, that we are sacrificing the here and now for some other delayed space of superior satisfaction. This aspect of conscious, steadfast and resigned choice is clearly absent in mechanical forms of labor. The labor of the machine is unreflective. It simply proceeds. It is precisely this feature of machine labor that attracted Alan Turing when he set out to critique David Hilbert's axiom of decidability. [2] The distinctive characteristic of the finite state machine is that it proceeds step by step, without any contemplation of alternative possibilities. It is this incapacity to reflect that finally leads to its undoing. A recursive logic pushes it towards reflection, becoming trapped in an internal contradiction. Very importantly, however, Alan Turing's conception of computation does not represent an effort to distinguish the special character of human labor and thought. Instead, it serves to clarify the mechanical character of axiomatic mathematical procedure. In a critical and ironic manner, it demonstrates the relevance of the machine in conceiving the apparently pure workings of mathematical logic.

If machine labor appears especially alien, it is because it represents an aspect of ourselves that we are especially keen to avoid. It serves as the uncanny double of the repetitive, mechanical, materially determined and non-reflective dimensions of human action. In this manner – in its curious, unsettling agency – machine labor disturbs our self-image as free and rational agents. Within this context, it is worth recalling that Aristotle distinguishes between thinking and unthinking dimensions of *techne* (making). The habitual character of manual labor, which can proceed without a clear understanding of underlying causes, is contrasted to the conceptually informed practice of the master-artist.

Inanimate things bring about the effects of their actions by some nature, while manual workers do so through habit which results by practicing. Thus master-artists are considered wiser not in virtue of their ability to do something, but in virtue of having the theory and knowing the causes. [3]

Here the rift between human and mechanical labor takes clear social shape. Hegel also emphasises this social dimension, tracing its historical and dialectical development. He argues that the rise of industrial society transforms labor into a vehicle for alienation. In becoming social (and economic), in shifting from the sphere of individual and local production towards the general commodity market, labor grows increasingly distant from any space of immediate concrete realisation or exchange. Endlessly abstracted and endlessly deferring immediate gratification, concrete labor becomes decoupled from human scales of meaningful action. The rise of industrial manufacturing processes – of machine labor and of human labor rendered machinic – only exacerbates this sense of alienation: "[the worker] becomes through the work of the machine more and more machine-like, dull, spiritless. The spiritual element, the self-conscious plenitude of life, becomes as empty activity." Machine labor produces what Hegel terms a "life of death moving within itself." [4]

Despite this negative assessment of the implication of machine labor, from my point of view the interesting feature is that Hegel does not position mechanical labor as an entirely alien force (an external imposition). Instead, an intrinsic dialectic is acknowledged. The contours of modern alienation are immanent within human labor at the outset. They are evident in the initial split from immediate appetitive being. In its dimension of stoic self-abnegation, human labor takes shape as a paradox. It is both constitutive of rational human being and indicative of a turn away from the simplicity of integral organic being. In this sense the separation of the machine – the dull, dead, spiritually vacuous motion of its instrumental functioning – appears as an exacerbation or materialisation of a tendency that will have always been, in some sense, properly human.

PERFUNCTORY EXECUTION

In my experience, programming represents a liminal space. It projects an intimate and entangled relationship between human and machinic processes of coding and decoding, agency and determination. Nonetheless, software programming is typically conceived in terms of notions of conceptual priority and anteriority. Here, an interpretation of the legacy of conceptual art becomes relevant. The work of Sol LeWitt, for example, is often regarded as emblematic of a neat, hierarchical split between conceptual and material-practical aspects of making. In an article about his 2004 *{Software} Structures* exhibition, Casey Reas positions Sol LeWitt's wall drawings as a model for his own software art practice.

The relation between LeWitt and his draftsperson is often compared to the relation between a composer and performer, but I think it's also valid to look at the comparison between a programmer and the entity of execution. [5]

Software programming is likened to the conceptual field of LeWitt's written wall drawing instructions, while the field of program execution (of computational process) is likened to the manual labor of realising the instructions on any specific wall. At the same time, however, Reas acknowledges a key point of difference. LeWitt's instructions lack the precision of programming code. They are conveyed in natural language and directed towards human readers. Rather than entirely restricting the space of execution, they work to suggest a focused field of creative possibility. Reas is keen to regard software art in similar terms, aiming to identify a form of conceptual software practice that precedes actual software programming, providing a generative conceptual basis for all manner of actual algorithmic drawings.

The work develops in the vague domain of image and then matures in the more defined structures of natural language before any thought is given to a specific machine implementation. [6]

He employs the term “software structure” to designate this pre-computational, creative-conceptual field and associates it with a potential for intuition and expressive freedom.

I want programming to be as immediate and fluid as drawing and I work with software in a way that minimizes the technical aspects. I often spend a few days creating a core piece of technical code and then months working with it intuitively, modifying it without considering the core algorithms. I use the same code base to create myriad variations as I operate on the fundamental code structure as if it were a drawing – erasing, redrawing, reshaping lines, moulding the surface through instinctual actions. [7]

No doubt LeWitt’s wall drawing work is full of curious paradoxes in which the machinic and the intuitive intersect, but it seems odd to harness it in the interest of describing a notion of expressive and de-technologized computational drawing. LeWitt is associated much more with a critique of the modernist concern with subjective, materially-based expression. As Ana Lovatt suggests, “[a]gainst prevailing notions regarding the immediacy, directness and primacy of drawing, LeWitt devised a drawing practice that was always already mediated by technologies of reproduction and communication.” [8]

Now while Reas never positions software structures as literally material, he conceives them very much in terms of “the vague domain of image”. [9] In this manner, the notion of software structure recalls the mute and intuitive aesthetics of formalist modernism. It envisages an intimate, traditionally expressive realm of creative conceptualization that is grounded in the space of perceptual manifestation. In this respect, Reas reinforces the boundaries between the intuitive and the procedural. The domain of conceptual expression, of software programming, is positioned as a form of alienation from intuitive conceptualization. It manifests the underlying concept in an estranged language that is properly distinct from the inner sanctum of creative conceptual imagination. A conceptual space is delineated, but in terms that precisely correspond to the reassuring visibility of the material image.

I prefer another reading of LeWitt's wall drawing project. Rather than indicating a neatly hierarchical division between the conceptual and the operational, his work suggests a play of mutual imbrication, mirroring and exchange. Moreover, rather than the conceptual appearing as a subjectively grounded sphere of autonomy and dominance and the executable as an utterly derivative space of expressive material determination, their relation is articulated in profoundly curious and unsettling terms. Consider this classic statement from his 1967 *Paragraphs on Conceptual Art.*

In conceptual art the idea of concept is the most important aspect of the work. When an artist uses a conceptual form of art, it means that all the planning and decisions are made beforehand and the execution is a perfunctory affair. The idea becomes a machine that makes the art. [10]

This appears to belittle the sphere of actual making. The work of manual drawing is portrayed as trivial and secondary. However, there is an ambivalence. The term “perfunctory” suggests a task that is mechanically performed, without any sense of subjective investment. This strangely opens up an affinity to the nature of conceptual practice. LeWitt insists that “the idea is a machine that makes the art.” [11] The conceptual then is also interpreted in mechanical terms. Both the conceptual and the executable are stripped of subjectivity. They both preserve a procedural, non-reflective aspect. In his 1969 “Sentences on Conceptual Art,” LeWitt describes the ideational blindness of the conceptual: “The artist cannot imagine his art, and cannot perceive it until it is complete.” [12] Ultimately, the intuitive machinery of the conceptual enters into relation with the machinery of making.

28. Once the idea of the piece is established in the artist's mind and the final form is decided, the process is carried out blindly. There are many side effects that the artist cannot imagine. These may be used as ideas for new works.
29. The process is mechanical and should not be tampered with. It should run its course. [13]

The value of the "perfunctory" is clearly evident here. It is a productive dimension of mechanism that tests and inspires new concepts. Although apparently distant and distinct, the spaces of conception and execution find themselves allied and linked. They share a common antagonism to the thinking of subjective expression. Together, as paired coordinates, they suggest a notion of drawing that reaches beyond the human, struggling to find effective means to engage with dimensions of blind process.

SHIMMERING

I will conclude by briefly considering an alternative model for thinking the relation between conceptual and non-conceptual dimensions of computational practice. It is drawn from a specific mode of painting within Australian Indigenous art. Howard Morphy describes the technique of Eastern and Central Arnhem Land painting: "painting is seen as a process of transforming a surface from a state of dullness to that of shimmering brilliance (*bir'yunhamirri*)." [14] He describes a clearly defined set of steps:

1. The painting surface is covered in an overall wash (typically red-ochre).
2. The key forms are outlined in yellow and black and basic figurative elements are coloured in.
3. Large portions of the surface are covered in "cross-hatched" infill with a special long brush.
4. The final work involves "outlining the figures and cross-hatched areas in white to create a clear edge which defines their form." [15]

Stage one is a straightforward process. Stage two depends upon high-order artistic skill and a close understanding of relevant representational traditions and protocols. Morphy notes that the second stage is performed relatively quickly by "a senior person." [16] Stage three is the most time-consuming, demanding technical skill but less demonstrable cultural knowledge. The final stage draws the painting together and is closely directed by the senior artist.

My specific interest is in the sophisticated mediation that this artistic process enables between elements of conceptualization and mechanical technique. The term 'mechanical' has to be used carefully here. It is less, in this instance, to liken Aboriginal painting to the characteristic forms of industrial production than to pinpoint a dimension of iterative, non-conceptually grounded process within Aboriginal art-making. It is not as though the work of producing cross-hatched infill does not have conceptual, aesthetic resonance, it is that it gains this resonance and this potential to shape a shimmering aesthetic surface by casting itself in terms of a repetitive articulation of time and space. The work has a ritual, performative aspect. In relation to the cross-hatching, Morphy argues that "Yolngu are not merely producing an aesthetic effect but moving the image towards the ancestral domain. The cross-hatched surface of the painting reflects the power of the ancestral being it represents, the quality of the shininess is the power of the ancestral beings incarnate in the object." [17] In this sense, the work becomes a means of summoning and invocation. Slow and mechanical, it shapes a real and affective alignment with dimensions of ancestral being and opens up the possibility of manifestation. From this perspective then, processes of conceptualization and mechanical technique are mutually imbricated. The distinction between concept and technique does not take a binary shape, but is instead structured as a play of mediation within the overall creative process. Concepts emerge as much from the labor of mechanical repetition, which

serves as a field of intimate communication and connection, as from the processes of mechanical repetition are inevitably inflected by the rich context of cultural meaning.

This example indicates other ways of making sense of the relationship between conceptualization and practical making within art; suggesting the need to re-evaluate the non-reflective character of making and to acknowledge the dynamic exchange between concept and mechanism within art. The relation between the two is no longer cast in binary and hierarchical terms – rather they appear congruent and enmeshed. I would argue that something like this is also what the creative programmer experiences. The close relation between writing, compilation and running that programming entails fosters a new, uncertain relation between the regimes of conceptual logic and mechanical operation. The programmer seeks not only to choreograph and determine computational processes, but also, at the same time, to explore an uncanny space in which the already alien algorithmic concept passes into the executable, non-reflective event and phenomenon.

References and Notes:

1. *Shlomo Avineri, Hegel's Theory of the Modern State (London, Cambridge Uni. Press, 1972), chapter 5, http://www.marxists.org/reference/subject/philosophy/works/ot/avineri5.htm (accessed June 23, 2011).*
2. *Alan Turing, "On Computable Numbers [...]," in From Gutenberg to the Internet: A Sourcebook on the History of Information Technology, ed. M. Norman (Novato, California: Historyofscience.com, 1995).*
3. *Aristotle, Metaphysics (Bloomington Indiana: Indiana Uni. Press, 1966), 13.*
4. *Shlomo Avineri, chapter 5.*
5. *Casey Reas, "{Software} Structures," ARTPORT: Whitney Museum of American Art, 2004, http://artport.whitney.org/commissions/softwarestructures/text.html (accessed June 12, 2011).*
6. *Ibid.*
7. *Ibid.*
8. *Ana Lovatt, "On Drawing, Ideas in Transmission: LeWitt's Wall Drawings and the Question of Medium," Tate Papers, Autumn 2010, http://www.tate.org.uk/research/tateresearch/tatepapers/10autumn/lovatt.shtm (accessed June 12, 2011).*
9. *Casey Reas, "{Software} Structures," ARTPORT: Whitney Museum of American Art.*
10. *Sol LeWitt, "Paragraphs on Conceptual Art," in Conceptual Art: A Critical Anthology, ed. A. Alberro and B. Stimson (Cambridge Mass.: MIT Press, 1999), 12-16.*
11. *Sol LeWitt, "Paragraphs on Conceptual Art," 12-16.*
12. *Sol LeWitt, "Sentences on Conceptual Art," in Conceptual Art: A Critical Anthology, ed. A. Alberro and B. Stimson, 106-108 (Cambridge Mass., MIT Press, 1999).*
13. *Ibid.*
14. *Howard Morphy, Aboriginal Art (London: Phaidon, 1998), 188-189.*
15. *Ibid.*
16. *Ibid.*
17. *Ibid.*