Gabriel Menotti (gb)
Media and Communications departments of PUC-SP and Goldsmiths University of London
Ph.D. Candidate

Computation as Dynamic Topography:

The Coordination of Algorithms, Apparatus and Architectures in the Production of Digital Images

The tradition that assimilates computer code to text is not completely appropriate for the analysis of audiovisual works based on digital media. It forces theory to take a detour through the field of linguistics, hiding the fact that – as physical phenomena – computation has more to do with architecture than with writing. This paper proposes the spatial *dispositif* as a concurrent paradigm for the evaluation of computational processes in the production of technical images. That, computational algorithms would share fundamental qualities with the cinematographic apparatus and the installation of art in a gallery.

The similarities between computation and the organization of objects can be traced back to the pre-historic origins of calculus. Men first kept track of quantities using proxies such as their fingers (*digits*) or small stones (*calculus*). This sort of calculation had no abstract dimension and it did not produce a directly 'greadable' outcome: the herdsman separated one pebble for each head of cattle that entered the cowshed; the ensuing pile of rocks did not represent a numerical value, but it could be used as a comparative mechanism to find if any animal was missing. Thus, both to count and to understand its results depended on the same activity of *processing discrete objects in closed territories.*

The physical computer gains abstract complexity and operability. These discrete entities are arranged according to rules of occupation, which preprocess the results of calculus. The chief example of this principle is the abacus, a matrix that organizes the pebbles in a system of relative, fixed positions. This striation of computational space produces a smooth visual pattern (Deleuze & Guattari 2004), where complex values can be immediately apprehended and go through sophisticated manipulations. Similarly, one could assume that the grid of coloured pixels in a modern computer screen is a particular surface of the machine's digital geometry, even before being any form of *representation*.

It is important to remark that the formalization of the aforementioned abstractions is visual being *conceptual*. In that sense, we'd argue that the parameters of computation are more like *rules of composition,* a strict textual grammar. One can easily perceive this operational logic in rudimentary programmable computers. For instance, Charles Babbage's mythical *analytical engine* (1837) borrowed its method of data storage and input (punch-hole cards) from the *Jacquard loom*, a device used to produce intricate visual patterns (Null & Lobur, 16). The processing method of the likewise theoretical *universal Turing machine* (1937) basically consisted of moving through the uni-dimensional space of an infinite tape and managing its binary occupation (Hayles, 176).

With electronics, the parameters of spatial arrangement become incorporated in the machine's circuit, as the once discrete entities are substituted by the constant flow of electricity. Modern-day CPUs function no differently from the first electronic computer, built 1939 using *relays* – switches that direct the electric current in one way or another (Kittler 1999, 18). Electronics take the spatial logic of computation to its last extents, as it reduces both data and its operation to the flow of energy (Kittler, 1995). In such mechanism, even the most static information depends on the system's overall motion (Kirschenbaum, 95). Hence, Bolter states that "all data in the computer world is a kind of controlled movement" (Ibid, 41) – the *trajectories* that stand as the necessary negative of the computer's dynamic topography. In other words, computation only exists while the system *runs*. It is not something the computer *does*; it is the manifested computer. Computation, all of the machine's logical and physical layers are reduced to the continuum of energy that enters through the power plug, goes through the processor and lights up the screen.

Likewise, software is never running in a computer; software is the computer running *in a particular way*, resulting in surface effects (Kittler 1999, 1). This also means that the algorithm is one of the computer's forms, impossible to tell apart from the machine's operation. Although the machine does *interpret* the code, it is not as a *reader* as much as a *performer*. Computation is not carried out in any language; even the machine's lowest ones and zeros are no more than "shorthand conveniences for voltage differentials" (Kirschenbaum, 116) – i.e. *another surface effect*. Programming languages mainly exist to bridge the gap between human use and computer operation. While low-level paradigms such as assembly still retain traces of spatial logic, high-level ones attain "superior flexibility and ease of design" by adopting natural semantics and syntax (Hayles, 58). Collaterally, they drive programming away from computation, as a "metaphor that hides the machine from its users" (Kittler, 1985).

We finish this essay by referring to modes of data input and programming through spatial arrangement, especially the *dataflow* paradigm employed in

systems such as *Pure Data* and *Max/MSP*. Based on the modular construction of electronic music synthesisers, these software call attention to the signal processing that happens on the core of the digital computer. As they become increasingly popular in the digital art scene, one hopes they inspire a critical framework more attentive to the qualities of algorithms as dispositives.

References

- Deleuze, Gilles & Guattari, Felix (2004) A Thousand Plateaus, London, Continuum International Publishing.
- Hayles, Katherine (2005) My Mother was a Computer, USA, University of Chicago.
- Kirschenbaum, Matthew (2009) Mechanisms, Massachusetts, MIT Press.
- Kittler, Friedrich (1999) Gramophone, Film, Typewriter, USA, Stanford.
- Kittler, Friedrich (1995) There is no Software [Online]. Retrieved from: http://www.ctheory.net/articles.aspx?id=74 [Accessed 25 May 2010].
- Null, Linda & Lobur, Julia (2006) The Essentials of Computer Organization and Architecture, Ontario, and Bartlett.