Critical Mass, the software

by Barbara Lattanzi
11.6.04

[Run sine-wave algorithm software: satellite feed of Ralph Reed in silence staring at camera.]

As preface to my presentation, I will read a passage written by Hollis Frampton, one that he formulated in reference to time. This is a version of his statement – not the actual text - because, in rewriting it, I have taken the liberty of making a few substitutions...

If it is dragons we seek, or if it is angels, then we might reconsider our desperate searches through physical space, and hunt them, with code, where they seem to live: in the folds of systems, processes, protocols and networks. FOOTNOTE1

In this demonstration of my software, I will be focusing on the following 3 ideas:

1. One idea looks at an intersection between two cultures: that is, practices associated with computing culture applied to film culture.

I am referring specifically to cultural strategies of reverse engineering and to initiatives related to the “free software” and "open source" movements (of software development) that, in my work, become ironic strategies in relationship to films by Hollis Frampton and works by others including filmmaker Ernie Gehr and video artist Anne McGuire.

2. The second idea considers the simulation of film structures. What does it mean to encode structures abstracted from films by Frampton and others by means of algorithms that are intended to still remain referential to the prior films?

The idea that software can encode structures of films such as *Critical Mass* (that I will be showing you) resonates with a further idea about these films and about software in general: that structures *fundamentally perform something*, rather than being about something. And yet the performance of lineage, that is the simulation of a film structure in a software algorithm - where the software becomes **referential** to a specific film experience - *paradoxically registers a narrative in the algorithm*, a narrative with concrete reference within an abstraction.

3. The final idea proposes Frampton's work, generally, as a model for software construction because of the particular way that his work engages the intentionality and subjectivity of the viewer – the way his work stages viewers' acts of perception, or registration, as intentional achievements of the viewer.

Hollis Frampton's incorporation of deliberate errors in his structural systems (as he details in his writings about *Zorns Lemma*) serves as an example of how the viewer's attention can be challenged and thereby extended into a generative process. The extension of intentionality and subjectivity **into a generative process produced by code** reflects a critical agenda for software art as it engages pervasive technical systems of control that affect social and cultural domains. If "code is law," then resistance to its authoritarian expression - resistance to the way that abstract systems **instrumentalize** us - is an urgent project for software art. In this one area of critical mass, Frampton's work and ideas bear political fruit.

[Display wildernesspuppets website, Idiomorphic software webpage, and "HF Critical Mass" webpage]

[Display webpage for script of "HF Critical Mass" main algorithm]

1st. Practices associated with computing culture, applied to film culture.

For several years I have been writing code using high-level scripting languages such as Lingo and Perl for software of my own making. I refer to this work collectively as "idiomorphic software". The term "idiomorphic" is to register an individual and particular approach to computer-based functionality - one that hopefully diverges from the normalized model of the "user" that is constructed by proprietary software applications. The purpose of my software is to stage intensified forms of perception using sound and time-based images to which the software is interactively applied in real-time. My interests in code extend to possibilities for filmic improvisation -- using the software for what I call "affective projection." My approach to affective projection is to enable the software user to organize any given stream of images into specific time-intervals, in other words into "modal" structures that are mutable during real-time improvisation by the "projectionist" performer who interacts with the film's display.

The process of developing "idiomorphic" software sometimes has involved reverse-engineering specific works of film mainly from the 1970s and then "open sourcing"

the experience of viewing those films.

By 'reverse-engineer', I simply mean that I analyze a film object (for example, "Critical Mass") in order to re-construct it. In this case I translate the dominant structure of a given film into a programming script.

After that I make available the software for free on my website. I post the algorithm so the code is open sourced, but more significantly, the software itself, in its application and use, open sources the otherwise rare experience of viewing certain films. This tactic is applied to *Critical Mass* (1971) by Hollis Frampton, *Serene Velocity* (1970) by Ernie Gehr, and a 1992 work, *Strain Andromeda The*, by Anne McGuire. Frampton's *Zorns Lemma* (1970) also served as a point of reference for another software work that I will demonstrate titled "The Interrupting Annotator." FOOTNOTE2

[Demo "HF Critical Mass" software: satellite feed of Ralph Reed speaking with his wife off-frame.]

Each of the software programs (e.g., “HF Critical Mass” “EG Serene”, “AMG Strain”) enables individuals to view any video of their choosing [usually in quicktime movie format]. Each software program enables individuals to manipulate variables that affect the playback of the video in real time. Each software program is characterized by one narrowly-defined, specific algorithm.

According to Frampton, *"There is no evidence in the structural logic of the filmstrip that distinguishes 'footage' from a 'finished' work. Thus, any piece of film may be regarded as 'footage,' for use in any imaginable way to construct* ***or reconstruct*** *a new work. Therefore, it may be possible for the metahistorian [of film] to take old work as 'footage,' and construct from it identical new work necessary to a tradition. Wherever this is impossible, through loss or damage,* ***new footage must be made****. The result will be perfectly similar to the earlier work, but 'almost infinitely richer'."* [my emphases] FOOTNOTE3

The cinematic works that I have selected to be open-sourced are hard to find and to see because distribution of experimental films is so extremely limited. In fact, at the time of software construction, I had not looked at any of them in years, not *Critical Mass*, not *Zorns Lemma*, not Gehr's *Serene Velocity*, etc. The software-making has been a process of remembering a handful of viewing opportunities stretching over 30 years.

So, the simulation of film structures is not just aesthetic but participates in a kind of cultural politics. By invoking cultural politics, I mean that simulating these films

in regard to their structure enables the viewer to experience the films by re-making them. I have re-collected particular film works from memory (because that is all I have of them), translated the remembered structural aspects of the works into code and then, ironically consistent with hacker practice, I have released the code as free software for others to experience on their own.

[Demo "HF Critical Mass" software applied to video footage of Buffalo Clinic Defense Rehearsal: a tactical difference, 1992 documentation shot by Julie Zando]

2nd. What does it mean to encode structures abstracted from films by Hollis Frampton and others (including Ernie Gehr and Anne McGuire) by means of algorithms that are meant to remain referential to the films?

To reverse-engineer *Critical Mass*, I have to abstract out what I perceive or, rather, what I register and remember to be significant algorithmic structures. This involves a process of abstraction that does a kind of symbolic violence to the work on which it is based. You can assert fairly that much is lost in this abject attempt to simulate a total work such as "Critical Mass."

Nonetheless symbolic violence on the level of programming code, at least in this case, enables a kind of direct participation in the logic of the prior film seen as a generative system. In the next step I parametize the structures (a set of abstractions) as "behaviors" within a digital simulation. With code, these identified parameters are made applicable beyond the prior film from which they are derived.

Code is the material of software art. And with code, motion pictures are given malleable behaviors in addition to their visual display. Behaviors allow mutations of the original-derived structure. Behaviors become one of the technical methods for that which software artist Matthew Fuller has theorized as “a means of mutation.” FOOTNOTE4

When Frampton writes that "*A story is a stable pattern of energy through which an infinity of personages may pass, ourselves included*" FOOTNOTE5, he could have been talking about software and the network of code objects through which an infinity of instances may be birthed and processed. I can say this because Frampton was explicitly distinguishing a stable pattern of energy from the

physicality of the ribbon of motion picture film that passes through the gate of a projector.

"[F]ilm meets what may be, after all, the prime condition of music: it produces no object.[...] [A]t the instant the film is completed, the 'object'vanishes. The film strip is an elegant device for modulating standardized beams of energy. The phantom work itself transpires upon the screen as its notation is expended by a mechanical virtuoso performer, the projector." FOOTNOTE6

Frampton ironically describes the projection machine as a virtuoso performer. However, what happens when, without any irony, the machine ***is*** a performer - virtuoso or not? Or what happens when the notation guides a dynamically-evolving *conversation* between a performing projection system and a viewer?

Unlike the axiomatic narrative dimension of film, we don't usually experience the combined output of a network of code objects (the components of software) as narrative (for example, when these code objects coalesce as spreadsheet software, or word-processing software, or even as a digital video editing application like Final Cut Pro or Adobe AfterEffects) . Software is not narrativising in itself. Software is not about something. Software ***performs*** something.

[Demo "The Interrupting Annotator" software]

3rd. Hollis Frampton's work, generally, presents a model for software construction because of the particular way that his work engages the intentionality and subjectivity of the viewer.

The insinuation of technical systems into every aspect of our lives, gives some urgency to the task of critical software production.

Media artist, Graham Harwood, has described this realm of technical systems, as an *"invisible shadow world of process"*. FOOTNOTE7

Hollis Frampton may have been in anticipatory agreement when he wrote:

"Images we make are part of our minds; they are living organisms that carry on our mental lives for us darkly, whether we pay them any mind or not." FOOTNOTE8

However, in an imaginary exchange, Graham Harwood points to a grey area in Hollis' formulation:

"...sometimes...reluctantly," he says, *"we have to depict the invisible in order to make it disappear."* FOOTNOTE9

Hollis, never at a loss for words, is just putting down his beer, about to reply, when filmmaker Harun Farocki takes advantage of the small gap in the conversation to smoothly deliver his own insight. *"Registers of the visual,"* he reminds them and us, *"are instances of the social."* FOOTNOTE10

He has barely gotten the words out of his mouth when a pack of software artists from the collective known as I.O.D. drown out whatever else he was about to say with their punk anthem,

"Software is mind control. Get some." FOOTNOTE11

In a recent essay on software art, Inke Arns cites the ground-breaking 1955 lectures by John Langshaw Austin, "How to Do Things With Words". Austin's speech act theory went beyond the conventional understandings of language as descriptive and referential. Speech act theory describes the performative dimension of language.

Applying speech act theory to software code, Arns writes: "Code as an effective speech act is not a description or a representation of something, but, on the contrary, it directly affects, and literally sets in motion... a process." FOOTNOTE12

Ultimately, she writes, *"...this coded performativity mobilises or immobilizes its users." And it is thus that code becomes Law. Code is Law."* FOOTNOTE13

[Run "HF Critical Mass" software, without sound, applied to NASA cinematography of Apollo moon mission]

Critical approachs to software art avoid focusing on code algorithms solely in the context of a technical or formal systems, but rather understand code, as Arns continues, *"in the context of social and political systems that are increasingly relying on these technical structures."* FOOTNOTE14

Frampton's films provide a resonant model for the production of critical and speculative software. This is chiefly reflected in the way in which Frampton's work integrates the agency of the viewer.

In her essay on Frampton's work, Chris Hill discusses the notion of the active

viewer. She points to Frampton's *"expectations of viewers as self-educators"*. She describes how certain of his films *"deny the apparent narrative its authority, and exercise the viewer's agency in experiencing variations on the theme of re-collecting, looking back and moving forward..."* FOOTNOTE15

This mobilizing of the agency and intentionality of the viewer resists what is simply generative. The algorithmic fetish of current generative art problematically valorizes the generation of visible surfaces through automated systems **that negate intentionality**.

In contrast to generative art's withdrawal of intentionality and its result-oriented non-interference with processes once launched, Frampton's work, instead, is resonant with software art that demonstrates an engaged “interest in the performativity of code” (quoting Inke Arns). FOOTNOTE16

Software art engages the balancing act between randomness and control. In this balancing act we find work that passionately projects and extends subjectivity and intentionality into the domain of systems.

[Run "HF Critical Mass" software, without sound, applied to blue-screen composite of a young woman wearing motion-capture hardware standing in front of Apollo astronauts on the moon.]

In his "Notes on Composing in Film," Frampton proposes a morphology for film study *"...that views film, not from the outside, as a product to be consumed, but from the inside, as a dynamically evolving organic code directly responsive and responsible, like every other code, to the supreme mediator: consciousness."* FOOTNOTE17

Frampton created algorithms for film without the error of thinking of algorithms as a static set of rules. For example, with *Zorns Lemma*, he reflexively acknowledges the contingency of any abstract system by deliberately incorporating errors into the structural rules. He wryly lists his categories of errors to include metric errors, lapses of taste, faking, and breaches of decorum, among others. FOOTNOTE18

Through such dynamic system destabilizations, carried out by means of controlled error, the attention of the viewer becomes **the main act** - a form of participation as an exacting perceptual achievement. In the Frampton model, the viewer does not voyage out to remote shores of virtual or artificial realities. Rather, the viewer reaches out to the world and, in the process, produces a dynamically-evolving sensorium, a stable pattern of energy out of noise and flux. FOOTNOTE19

The film works of Hollis Frampton have been inspirational for me as a software artist because of the emphasis on the viewing subject who is made aware of and responsible for her attentional acts. Frampton's work, for the software artist, interweaves constructivist and realist approaches towards making and understanding experience. What is registered in his work is never de-coupled from a way of inhabiting the world.

Finally, I would like to breach decorum during my few remaining minutes by asking you to join me in a song...

[Demo "C-SPAN Karoake" software]

Footnotes

1. Transcoded from essay by Hollis Frampton, "Incisions in History/Segments of Eternity", pg.103, *Circles of Confusion* (Rochester, NY: Visual Studies Workshop Press, 1983).

Original quote:

"If it is dragons we seek, or if it is angels, then we might reconsider our desperate searches through space, and hunt them, with our cameras, where they seem to live: in the reaches of temporality."

2. "Strain Andromeda, The" is a video by Anne McGuire from 1992 that itself is a detournement of the narrative structure of the 1972 Hollywood movie "The Andromeda Strain" directed by Robert Wise and based on the 1968 Michael Crichton novel.

An earlier software program I developed, titled "Surface Tension: Applied Memory Mutation Software", attempted this same reverse-engineering of Hollis Frampton's films "Poetic Justice" and "Gloria!". I was dissatified with the results. However, it led eventually to my most recent software efforts that include "The Interrupting Annotator" and the related software "C-Span Karaoke".

Information and downloads for the software mentioned can be found at:

http://www.wildernesspuppets.net

3. Hollis Frampton, "For a Metahistory of Film: Commonplace Notes and Hypotheses", in *Circles of Confusion,* pg.113-114

4. "A Means of Mutation," essay from Matthew Fuller, *Behind the Blip: Essays on the Culture of Software* (Brooklyn, NY: Autonomedia, 2003).

Fuller describes a *"poetics of potentional"* in which everything *"- every bit, every on or off fact – is understood in terms of its radical coefficiency, against the range of mutation from which it emerged and amongst the potential syntheses with which it remains fecund. It is the production of sensoria that are productive not just of 'worlds' but of the world."* pg.65

5. Hollis Frampton, "Pentagram for Conjuring the Narrative",*Circles of Confusion* , pg.67

6. Hollis Frampton, "For a Metahistory of Film", *Circles of Confusion,* pg115

7. quoted in Inke Arns essay, "Read_me, Run_me, Execute_me: Software and its discontents, or: It's the performativity of code, stupid!", pg.10, PDF file, 2004, downloaded from the writer's website at:

http://www.v2.nl/~arns/Lecture/ArnsNoviSad2004.pdf

8. Hollis Frampton. Unknown source.

9. Graham Harwood. Unknown source.

10. Harun Farocki. Unknown source.

11. IOD, British art collective, from their website circa 2000

http://www.backspace.org/iod/

12. Inke Arns, pg.9, PDF file.

13. Ibid, page 9

14. Ibid, page 9

15. Chris Hill, "(Re)performing the Archive: Barbara Lattanzi & Hollis Frampton in Dialogue" (*Millenium Film Journal,* Nos.39/40, pg. 78 and 79)

16. Inke Arns, pg.6, PDF file.

17. Hollis Frampton, "Notes on Composing in Film", *Circles of Confusion,* p.124

18. Hollis Frampton, "Notes on Zorns Lemma", *Screen Writings*, p.60, Scott MacDonald, editor.

19. Using a term that I have borrowed from Brian Cantwell Smith's elaboration of an ontological foundation for computational systems, we might refer to this intentional stance of the viewer as *registration*. As I understand this concept, registration describes an achievement of perception, and of a process of objectification wrested out of flux. See, Brian Cantwell Smith, *On the Origin of Objects* (Cambridge, Mass: MIT Press, 1998)

Acknowledgements

Thanks to Keith Sanborn, Su Friedrich, and P. Adams Sitney for the opportunity to participate in the conference, "Gloria! The Legacy of Hollis Frampton" - The Program in Visual Arts, Princeton University, 2004.