ap - fm01

martin howse

ap

studio 22,38-40 upper clapton rd
london e5 8bq uk

m@1010.co.uk

jonathan kemp

ap

studio 22, 38-40 upper clapton rd
london e5 8bq uk

j@1010.co.uk

ABSTRACT

This paper examines a new discursive space which has been exposed by radical software art through the heuristic possibilities opened up by software abstraction.

Keywords

Machinic abstraction, environmental coding, endodata, software art, auto-destructive, distributed systems, reconfigurable computing

1. INTRODUCTION

fm01 is speculative software: that is it both explores the potentiality of possible programming thereby creating transversal connections between data/code, machines and networks, and software that reflexively addresses/investigates and finally reinvents itself as software by its own means (it is a bastard ontology): presenting a release of technology from the ontological idea of its usage, and permit its own ontological structuring as unknowable - placing it in the unknowable Kantian realm of practical reason (like infinity, god, morality, things in themselves) and outside of the operation of pure reason, (a sphere of necessity, of laws of cause and effect, syllogism, the law of identity/singularity, of the excluded middle); and evading the logic of the universal/particular coupling. so for us, the ontogenesis of fm01, isn't to be found in the retelling of a teleological story of image processing culminating in digitization or whatever (just as the immune system doesn't orientate itself to the survival of its host) and as fm01 doesn't take images as carriers of experiences and meanings, we are not interested or know where it sits in the context of expanded film.

instead fm01 attempts to confront between what man regards as being possible [function + value] and what machines present as feasible [in potentia]. technologies are permanently shifting this relation between the possible (potential) and the feasible (functional) and where the construct of the real constitutes such a "negotiation" between the potential and the functional, being constantly reformulated.

fm01 [also] attempts to ask the radical question of how data can be represented for the machinic without the supervenience of meta-data or the demand for the purely functional impacting on abstraction.

fm01 as investigation centres around the domains of:

1. new operational zones created by machinic ontologies [lying on the plane of fascism, outside humanism, because it demands a space outside the operation of pure reason in to the realm of practical reason (things in themselves)].
2. the specific exposition of a radical space for code art [denying the trope that if code defines how data gets handled on a technical level, then meta-code [like belief systems, ideologies and organizing principles] is the philosophically relevant level]

subthemes in relation to these two domains are: a) the decoding and recoding of data under non species dependent symbolic orders b) the form space of code and environment and its autopoeitic engineering. so the relationship we will present between image and generation is not a reenactment of the anthropological semantics of the human eye but is part of an integrative poeisis of processing between orders as extant between geology, plants, machines and humans. the making of images is not a return to imagination of iconic culture but the move to calculation + computation [vilem flusser], the morphospace of numerology. although we're used to being on the winning side of the image an image comes full circle when it is revealed by its own machinic underpinning.

2. ENVIRONMENT

in presenting the fm01 project, there is an underlying assumption that the audience is familiar with some current issues and practices in contemporary software art/cultural practice including digitally expanded cinema. a general synopsis of current paradigmatic thought around issues specifically of digital cinema may be found in Peter Weibel/Jeffrey Shaw's accompanying book for ZKM's 2002/3 Future Cinema exhibition[1], where they

write that, for example, “the biggest challenge for the digitally extended cinema is the conception and design of new narrative techniques that allow the interactive and emergent features of that medium to be fulfillingly embodied”[1], and outline three overlapping areas of recent praxis:

1. modular structures of narrative content which allow indeterminate numbers of permutations or parallelisms (eg Lev Manovich’s “soft cinema”; Jon Jost; Eija- Lisa Atthila; Raymond Tomin’s AVRA software; Jennifer + Kevin McCoy; Marc Lafia’s Max/Msp based “variable montage”).
2. algorithmic generation of narrative/mnemonic sequences/ markers that could be modulated by the user (eg. Martin Reinhart + Virgil Widrich’s “tx transform”; Marnix de Nijs “run, motherfucker, run!”; the work of George Legrady).
3. digitally extended cinema inhabited by audience who become immersed agents and protagonists in its narrative development (eg. Jeffrey Shaw; Michael Naimark; Margarete Jahrmann + Max Mosswitzer’s “rgani-engine-toolz”).

Future Cinema was cast as a show where “the medium is the message” yet was situated in the familiar terrain of the hegemony of the species, the subjugation of machinic cinema embodiments to the plateau of pure reason, and the enslavement of technology to an ontology of relational functionalism. It is paradoxical that Future Cinema’s seeming activism is consistent with a series of static shots from a fixed point of view, between man and machine, as a homogenous relationship [the great Gutenberg program].

3. CODE

there’s a lot of data flying about that isn’t species dependent for its encoding, decoding or recoding, and in many ways we are no longer the sole traders in the realm of the symbolic as a species: for example, within the interaction of human text with machine coding, language is not the exclusive domain of human thought but also that of the internal logic of computers. Fm01 is an attempt within this context to offer a total software environment for the semi-automated production, scripting and editing of endless cinema outside the humanist realm of meta-data. Fm01 is not conceived as an editing engine for the manipulation of generic clips (in an expanding database of all possible scenes rganizatio according to a huge number of elements and relations) but rather offers an enmeshing within script, data streams and environment [WHERE ENVIRONMENT=CODE=ENVIRONMENT IN BEAUTIFUL TRAJECTORIES]. An environment of trajectorial machine nodes - from environment to language these machinic nodes are the environment - consisting of trajectories, macros-functions, relations, levels, scenes, scripts, machine-nodes.

Gdapp (generic data application) commenced coding in late 2003: the idea being that a pure lisp-coded prototype fm01 would meet with the arc described by the nodal, distributed gdapp experiment. An intensely nodal, modular, environmental, self-referencing system which builds on ap02 vm model but extending and multiplying this model/ dissecting it also into a nodal/ geologic strata model. Gdapp describes a globular/dynamic changing node structure of connections, of flow and instructions - a total environment, an operating system which dynamically re-codes itself, is recoded in operation. Layers of abstraction as environment: consisting of a base layer which mobile, almost viral code nodes sit and work upon and an open exchange as opposed to a protective, secure OS model. Gdapp codes promiscuity and open discovery rather than static upload/download or I/O structuring. Gdapp codes through a nodal model: nodes run/write code/data, nodes run other nodes, nodes link other nodes, nodes reflect macro linkages or embed code and linkages within neural model of linkage excitation.

The technologies of vision/representation have been driven by a indexical linking between reality and representation [image and its physical rganiza object] and we are compelled to locate [veracity] within the technologised image [the history of film and relation between celluloid/chemicals/light/scene]. If images have been conceived of only for human species then images that are machine-assisted or automated seeing render redundant imaging as an attempt to reconcile this contradiction, the indexical idea.

Digital images are not immaterial [as some think through the breakage of indexical link]

“hardware looks like programs: it is a configuration. Hence programs, hardware (configurations) and data have the same nature.” [2]

descriptions of cell matrix architecture (Macias 1999) well match the gdapp engineering philosophy. The cell matrix is internally configured by exchanging configuration data with its neighbours

“evolvable computing and some modern systems do not share computational scenario of a standard Turing machine and cannot be simulated on Turing machines.” [2] thus interactivity and infinity come into play with consequences for any theory of abstraction. That it is not a field of equivalence - here time enters the computational equation.

“at each time point evolvable devices have a finite description - however when one observes their computation in time they represent infinite sequences of reactive devices computing non-uniformly”[2]

“evolvable computing is beyond scope of an ordinary Turing machine. It does not violate the Church-Turing thesis because this thesis deals with a slightly different class of computations corresponding to the concept of algorithm. This class of computations is not typical for contemporary computational systems” [2]

the advent of the digital image is thus freed from its material support and mobilizes it through networks and recoded as mathematical information (bits, code) that allows it (the bit-real) to be recomposed infinitely and to flow indefinably as data.

Complex data cannot be comprehended by human reading (too slow). Data is rendered /abridged by images (like the celluloid film strip of Conrad Zuse holding both image/code and data). Digital calculation beyond the individual subject refers neither to the differential symbolic order represented on the screen nor to a world outside this screen (physical reality behind the screen is state and current only); the digital machinery retreats into total abstraction and is catachretic, not metaphoric, a baroque violence rendered to the potentia of the machinic phylum.

So what would be in this philosophy of images. Software routines and their non-traceable/path dependent images are a realm/ environment of data/code [in effect the break between the material support of digital imagery, the break between the indexical link]. And so in fm01 a relational, nodal language of connection will be formulated to descend through levels of scene, shot and frame. Instruction sets (which refer only to trajectories and grammars) are dependent on context/environment but offer the same functionality [non-functionality] across all flattened levels from pixels to major trajectories, from major conceptual conceits and rganizations of nodes to a single script particle.

Fm01 is running invisible code: its self-organising is ceaseless and in contradistinction is environmentally mobile: distributed and polyvalent. Code is an environment of subsystems in search of form or pattern, the orphaned bits without superstructure where nothing corresponds to anything inside nor outside the machineware that applies it. Nothing more than the autonomous movements of data within the clock of the machine - ways of doing things become situated within object orientated fields of influence (marked by a level of analysis, meta/o-nymed) (eg. The micro influenced by the macro, or a “case of” where a process is named) and we see in one sense a blind text/residue stratum ie. Electronic images are effects of a surface right from the start where the surface still appears as material of an object and as intensity of a pixel.

4. ENGINEERING PHILOSOPHY

textural lisp work with an extant written script (a description, a specification for fm01 almost) will describe a trajectory and meet with the arc of gdapp to constitute the first full version of fm01. this textual work (operating on text as stream rather than as file - text as particle physics) is towards creating a compiler/interpreter which actively runs that script. Gdapp is concerned with mobile code nodes which run and enact on data - both approaches lead to fm01 - runs the script creating new text or film - runs that script which acts on and suggests data sets - active across data sets and is inside-embedded within/as data sets.

Components outside an object model - trajectorised components include script, node, language implying an interpreter and particles with velocity and open destination. To begin with the script:

script as making evident of word-background/that-which-is-written within script as it stands - towards what-will-be filmed but without discarding the script. Background material includes data such as how it was written - how it is to be filmed (this is not meta-data but the being-script). Within such a landscape -relation the issue of vectorisation or trajectory is important.

Background/script (film script) stands in same relation to film (filmed) as program to execution - background as revelation of code and script rewriting film rewriting script.

Scene as node which can change/merge/enact upon other nodes/scenes. Functional data/code/scene nodes collaborate/ change each other - for example image analysis nodes/compression nodes

script as node, node as mobile, node as cross-level functional and able to negotiate, coerce and encompass trajectories of nodes, node as interpreter and interpreted ongoing, node as exchange with nodes and collaboration.

Back-foreground relation as question of interpretation (and context =environment) and the ability to alter-construct node trajectories. Nodes define a language which can be shared or argued over - misconceived. Fm01 is thus the description of a language. Language is described by the script. The script is defined in terms of mechanisms with the node as this mechanism which functions in/or constructing/constructs language.

Computer language (created on top of another and bootstrapping) describing an environment which is node, script, interpreter and trajectory (distributed) at the same time. Vectorisation of script and image - as image and in script across combinatorial senses of pages, scenes, shots and also within image and scene (eg. Concentric studio spaces and clues=trajectories) as well as being embedded in components script also defines trajectory (networked and across nodes) of/for software components

instructions or tokens (to use such a word?) are dependent on context - but offering the same functionality across all flattened levels of what could be seen as high and low levels - of pixels and major trajectories (some functions dependent on context = environment). trajectory of node-particle possesses speed and crosses levels as dictated. node is particle, relation and change of space of that particle.

to describe the meeting of both arcs: the co-evolution of cell data-code interpreter elements in lisp (eg. evolution of edge detection). co-evolving mobile modules explore and exchange different functionality and data/code whilst working on both script stream I/O and film stream I/O: simply to replace fixed database software with mobile nodes and changing code/data streams across parallel architectures. script-data-code IS these modules (how module is defined) thus software is really the interpreter or some sort of bootstrap of language and interpreter. nodes are the data, are the software - kein outside, kein addressing, purely trajectorial nodes impacting on an environment which is solely of nodes, of strata. fm01 has purely generic nodes which are just assemblages according to no function - function defined by relations and trajectory - interior (invisible) machinic function.

5. REFERENCES

1. Shaw, J, and Weibel, P. Future Cinema. MIT Press.
2. Sekanina, L. Evolvable computing by means of evolvable components. Natural Computing 00 1-31, 2004