Spoken at Page Not Found , Den Haag November 2019.
Martijn van Boven

CODE RE-ENACTMENT OF THE DECAYING LANGUAGE INSIDE A MACHINE

Introduction

People and communities use languages, semiotic systems, and instructions that they can understand and operate within. But what if these languages disappear over time, deform, are misread, or need to be understood again, interpreted and applied. What does this say about the way we have information capturing our contemporary moment in seemingly endless data centers, compressed media or even independently operating musical instruments How can we recognise software as a cultural heritage, as art or artefact, and data as being endued with historical significance.

The alteration and interchangeability of knowledge in music, mathematics, fine arts and language—as described in the literature of *Godel, Escher Bach* by Douglas Hofstadter or *The Bead Game* by Herman Hesse—clearly shows that the abstract and immaterial world of culture and knowledge is transferable and able to change from one medium to another, without real significance or value. We do not live in the world of Marshall McLuhan's "the medium is the message", but that of Friedrich Kittler, "Nur was schaltbar ist, ist überhaupt" (Everything that is switchable, exists).

In the spirit of Kittler it can be thought that the resolution of digital recording and playback is higher than that of text, voice, images and thereby exchangeable at higher levels. The world is converted through sensors into digital information; in essence, it all depends on the degree of frequency (Hertz). From this base point of conversion, the world becomes information: programmed (code), encrypted (encryption), translated (conversion), sorted (database) and self-concealed (data cloaks).

Research question

Where language is often seen as a technology and carrier of valuable meaning, one often thinks code as only being procedural and meaningless by nature. It only executes orders. It seems as if we only have an aesthetic sense of understanding when we encounter software code, we might be able to read it but unable to understand its cultural meaning and significance.

Fundamentally we are now increasingly describing our world in code as opposed to the story line and its narrative structure. The story line is what we used to describe the world—to document it, to make it understandable, to interpret it—until the advent of computer technology. But we seem to be drifting away from understanding what is being written in code.

Is the world described in human language ultimately a matter of phenomenological perception and that of computer code a vision from inside of the machine out to the world? Written from the inside of its technological ecosystem and leaving an imprint on the physical world to whom it has no notion of. Is human language impressionistic and computer code expressionistic? Human language might be as finite as a pixel in its description of the world. Compare this to code and algorithms which can describe a vector field of objects, all in different relations to one another.
Arguably, by talking about the black box of software and trying to understand and describe it through human language a fundamental flaw is made.

For this research project I have chosen a piece of historic software from the 20th century. This piece of software was the result of two major interconnected global events of the 20th century: the space race and the cold war. Its development encompasses two decades of fundamental software engineering and resulted in the only perfect piece of software ever written.* Written by a anonymous subdivision of IBM who wrote 420.000 lines of code for the first space shuttle flight: STS-1.

This software produced a vast quantity of global events, images, actions, news and inventions which are not collected as objects in museums. It did not become a monument in public space or studied as a historic event. It exists only as information, code or data with hidden cultural and scientific significance.

There are some events from our recent history that were communicated to us via digital media. The first image of the surface of Mars was sent to earth as a series of ones and zeros. Within the art world, Mark Bain transposed seismological data from the collapse of the Twin Towers into two vibrating beds in his work 'A Seismological Recording of 9/11'. These are two concrete examples of historic data recording. A few more examples embedded in the cannon of digital history are Claude Shannon's concept of the information society, the first computer images on stamps, the Turing Machine, and the flash crash in the stock market. *We recognise these examples by their cultural importance but not in how we can understand them differently by reading them as software.*

Code Re-Enactment as an artistic expression

Code re-enactment deals with the notion that software and its performance, its purpose, its collective data, and its hardware are a historic event. All of these elements of software not only need to be documented, but can also be re-enacted, re-interpreted, studied, and used for artistic expression.

The process of code re-enactment can be understood through the analogy between music composition and software. Both are a meta language which can be performed by a wide variety of instruments and performers. They can both be freely interpreted or stringently followed. Both are ambiguous by nature and can easily be translated into different media. A score written for a chamber ensemble can be arranged for a large orchestra and vice versa. Data has no specific form and needs to be translated or printed to an output which can be a screen, paper, voltage controlled devices or made audible. Both a musical score and a software program are able to control machines and humans simultaneously, resulting in an expression and leaving an imprint on the world.

The Culture of Code

It's not difficult to appreciate the extraordinary audio visual qualities of a shuttle launch—how the software turns the space shuttle into one large instrument creating a visual imprint in the sky and an extremely powerful sound wave. The shuttle, in that sense, has always been an audio-visual instrument built and controlled by a large group of hard- and software engineers and astronauts.

As an example, instrument builders who constructed Northern-Europe's church organs, played and appreciated by composers such as Bach, dealt with the same conceptual, technical, and economical challenges. A deep understanding of physics was needed in order to withstand large pressure on the pipes that produce low and loud frequencies. Knowledge of woodworking and metallurgy was needed to build and connect a complex structure, and large sums of money were needed over time, collected from the church and the city counsel in order to finance the construction of the organ. Finally, the instrument builders needed to defy the extreme scrutiny of the composers who tested the church organs to their limits.

By placing the development of the shuttle in a cultural context, and not only within the purview of mechanical and software engineering or avionics history, new re-interpretations can be made. The Greek composer Iannis Xenakis famously spoke about the composer-pilot: the composer who writes music with the aid of a computer for electronic instruments and who oversees the mixing desk and all its peripheral instruments and filters. This is analogous to the astronaut-composer, who is in the Shuttle during the ascent and re-entry procedure, checking gauges and communicating with the onboard computer system.

Metaphorically, the complete Space Shuttle system, which includes the launchpad, payload, all the I/O sensors, environmental sensors, the astronauts, up-down link communication, and mission control are connected, controlled, and running in-sync by the software coming from the four IBM AP-101s. All the parts becoming one large ecosystem of highly technological systems interconnected with humans.

How to Approach Software? A working methodology on the STS-1 software

Since the actual software of the STS-1 is covered under US ITAR laws—International Traffic in Arms Regulations—working directly with the software is at this time not a possibility and may never be.

During the research period I used a combination of digital forensics and a reverse waterfall model (The Waterfall model, formalised in the 1970s, is a linear approach to software development. Each phase is finished before the next one begins, often by different people and or companies). So I will start will all the publicly accessible information, data and media and work my way up; constructing a software sketch of the STS-1 in the process.

I began with preliminary research mapping out all available online documentation.
Currently used resources during this time will be further expanded and are divided into the following categories:

- NASA documents such as space shuttle operation manuals, interviews with software engineers, ascent, orbital and re-entry telemetry points, flight simulation files, and trajectory data.
- HAL/S programming language manual and instruction files
- Aviation and navigation algorithms
- Integral audio and video flight recording
- Online documentaries about the space program, interviews with astronauts, etc.
- Articles from magazines such as Aviation Weekly and BYTE

* https://www.fastcompany.com/28121/they-write-right-stuff
** The Primary Avionics Software System, or PASS, is the software that runs in all the Shuttle's four primary computers. PASS is divided into two parts: system software and applications software. The system software is the Flight Computer Operating System (FCOS), the user interface programming, and the system control programs, whereas the applications software is divided into guidance, navigation and control, orbiter systems management, payload and checkout programs.
*** A program that converts instructions into a machine-code or lower-level form so that they can be read and executed by a computer.