

nybble-engine & nybble-engine toolZ

by Margarete Jahrmann & Max Moswitzer

»To attack is to collaborate, to shoot is to communicate, to play is to edit code!«:

Inke Arns, Berlin

Published in: Joke Brower, Arjen Mulder, Anne Nigten, Laura Martz (eds.), *aRt&D. Research and Development in Art*, Rotterdam: V2_Publishing/NAi Publishers 2005, pp. 238-249

With software becoming ubiquitous in our everyday life, the attention of artists recently has shifted increasingly towards the performative potentials of program code. In the context of what has become known under the label of software art,¹ generative art or even game art², artists started fiddling around with what drives and controls (but what normally remains hidden behind) digital surfaces and graphical user interfaces, i.e. program code and its performativity, or, coded performativity.³ Program code is characterised by the fact that it is not a description or a representation of something, but, on the contrary, it directly affects, and literally sets in motion – or it even »kills« a process. This »coded performativity« has immediate, also political consequences on the actual and virtual spaces (or: programmed environments), in which we are increasingly moving and living: it means, ultimately, that this coded performativity has the potential to *mobilise or immobilizes* its users. Code thus becomes Law, or, as Lawrence Lessig has already put it in 1999, »Code is Law«⁴.

Interestingly, the more Code became Law, *the less it was visible*. This paradoxical relation is quickly explained: With the global success of personal computers and an exponentially growing usage of graphical user interfaces in the 1990s, program code

¹ Matthew Fuller distinguishes between social software, critical software, and speculative software. Cf. Matthew Fuller: *Behind the Blip. Essays on the Culture of Software*, New York 2003.

² Cf. Hartware MedienKunstVerein / Tilman Baumgärtel (eds.), *Games. Computerspiele von KünstlerInnen*, Frankfurt/Main 2003; Tilman Baumgärtel: Alle Nazis werden Dreiecke. Computerspiele verwandeln sich in hinter sinnige Kunstwerke, in: *Die Zeit*, 16/2002, http://www.zeit.de/archiv/2002/16/200216_computerspielkun.xml.

³ Cf. Inke Arns: *Read_me, run_me, execute_me: Software and its discontents, or: It's the performativity of code, stupid!* In: Olga Goriunova / Alexei Shulgin (eds.): *Read_me. Software Art and Cultures Conference*, Aarhus: University of Århus 2004, pp. 176-193; Inke Arns: *Read_me, run_me, execute_me. Code as Executable Text: Software Art and its Focus on Program Code as Performative Text*, in: Ihmels, Tjark (Hg./ed.), *Generative Tools*, Mainz 2004, <http://www.medienkunstnetz.de> (forthcoming)

⁴ Lawrence Lessig: *Code and other Laws of Cyberspace*. New York: Basic Books, 1999

(i.e. text, or command line-based operating systems like DOS) – increasingly disappeared behind graphical or metaphorical interfaces – like, e.g. the “desktop” which simulates a real desktop. As a direct result, the users’ awareness of the structures underlying programmed environments was decreasing. Ultimately, this development also made people forget that code is not something God-given, but that it’s written by programmers, i.e. humans, and can thus also be changed, re-written, shaped and conceived of in a radically different way.

Today we are witnessing a re-discovery of this knowledge – it is a re-discovery of a formerly common knowledge that today, with the ubiquitousness of software, gains almost an emancipatory meaning. One can think here of the free software movement (where this knowledge actually was never lost), the broader Linux “revolution” since 1999 or about recent software art and – computer games. Although computer games are not usually seen as contributing to the improvement of humanity, it can be said that here a very interesting change occurred in the first half of the 1990s concerning the relation between user (consumer) and developer (producer). When ID Software, the producer of one of the first so-called ego shooter games called *Doom*⁵ refused to make the code of the game engine publicly available, a world-wide community of users cracked and reverse-engineered the code in order to be able to add their own levels and modifications to the game.⁶ Since then it became a wide-spread practice.

nybble-engine and *nybble-engine toolZ*, initiated and developed by the visual artists Margarete Jahrmann and Max Moswitzer between 2001 and 2002, consists of – unplayable – modifications of an existing multiplayer first person shooter game (*Unreal Tournament*⁷). Jahrmann and Moswitzer, both born in 1968 and trained at the Academy of Fine Arts in Vienna, have specialized in manipulating program code of existing computer games. Game modifications (so-called “mods”), which are a usual practice within the various gaming communities and subcultures, can affect the game

⁵ Other notoriously brutal first person shooter games are *Doom*, *Quake*, *Counter-Strike*, *Wolfenstein*, *Half Life*, *Unreal Tournament*.

⁶ Cf. Raphael Quinet, The subjective history of *Doom* editing, version 0.1, http://www.rome.ro/lee_killough/history/edhist.shtml

⁷ “*Unreal Tournament 2004* is a multiplayer first person shooter that combines the kill-or-be-killed experience of gladiatorial combat with cutting-edge technology. Ten game modes - both team-based and “every man for himself” -- provide even the most hardcore gamer with palm-sweating challenges through unbelievably detailed indoor arenas and vast outdoor environments. As the ultimate techno-gladiator of the future, players will take their fates into their hands, battling against up to 32 other players online in action-packed, frag-filled arenas.” (<http://www.unrealtournament.com/>)

surface, the level design (i.e. player representations, graphics, sounds) or go deeper and affect the so-called game engine, the „heart“ of the computer game. Out of such a “total conversion” (TC) completely new subversive game versions can be developed. *nybble-engine* (as a “mod”) and *nybble-engine-toolZ* (as a “TC”) cover the whole range of possible modifications of the game *Unreal Tournament*. By creatively modifying this game Jahrman and Moswitzer intervene directly in the digital technologies that increasingly shape our environment. They thus not only point to the possibilities and to the limits for action inscribed in the digital technologies but also subvert the notion of usability usually connected to technology.

The last five years have seen a growing artistic interest in computer games: Mongrel’s *Backlash* (1998), Jodi’s *SOD* (1999) which turned the ego-shooter *Wolfenstein 3D* into some kind of op art environment as well as Jodi’s project *Jet Set Willy c 1984* (2002)⁸, Cory Arcangel’s *Super Mario Clouds v2k3* (2002) and Joan Leandre’s *retroyou nostalg* (2003) which consists of a deconstruction of a car racing game are only a few examples. Artists have been using a broad range of games from early game classics which from today’s perspective look almost abstract, to very recent and hyperrealistic first person shooters.⁹

***nybble-engine* (2001-2002)**

However, Jahrman’s and Moswitzer’s motivation for dealing with computer games is less their interest in the content of these games, nor does it lie in critically addressing gaming culture as a whole. Rather, *nybble-engine*’s unique position in the context of “game art” is related to the artists’ interest in developing a working environment (if not operating system) that is structured like a game. *nybble-engine*’s game environment is entirely generated by real-time server processes (triggered by a user moving through the program or by the activities of re-programmed semi-autonomous agents) being fed as “raw material” or “raw data” into the *Unreal* game engine. The traffic on an ethernet card thus becomes the material basis for aesthetic transcoding.¹⁰ As if

⁸ [plugin], Tilman Baumgärtel, BüroFriedrich (eds.): *install.exe – Jodi*, Basel 2002.

⁹ C.f. on the history of computer games and their military origins Claus Pias: *Computer Spiel Welten*, Munich 2002.

¹⁰ Lev Manovich describes “transcoding” as one of the five main principles of digital media. The others are: numerical representation, modularity, automation, and variability. Cf. Lev Manovich: *The Language of New Media*. MIT Press: Cambridge, Massachusetts / London, England 2001.

short-circuiting the game with itself, the game engine is used to visualize the network traffic generated by the game, which in turn is fed back into the game engine itself.

Inspired by cybernetics pioneer Heinz von Foerster, and developed in cooperation with independent theorist F. E. Rakuschan, hypertext theorist Heiko Idensen, and musician Andi Kunzmann, *nybble-engine* produces so-called *nybble-engine*-movies (NEMs). NEMs are interactive movies – similar to machinima¹¹ – triggered and generated in real time by the interaction between players/users, bots, the 3-D surfaces and replicating program components. A NEM could be described as a kind of demo ride through a non-linear program architecture, or as a sort of “digital road movie” (Jahrmann/Moswitzer). NEMs are thus fully functional multi-player environments in which one can navigate and communicate. The bots from the Unreal shooter have been re-programmed into non-anthropomorphic non-linear tools and thus can trigger unpredictable events inside the program architecture. In addition to that, the hyperrealism of most computer games is being questioned by the fact that the structures generated by *nybble-engine* are covered by ASCII-code textures and are thus abstracted. This mapping of the code onto the very structures generated by it points to the total constructedness of virtual environments. *nybble-engine* thus shatters the naïve trust in any kind of reference to reality.

A very important element of the *nybble-engine* project is the live aspect. It is usually presented by Jahrmann and Moswitzer in a pop-coded form as a live modular performance-lecture. Here, the project emerges in three different equivalent formats: as a program, as a machinima, and as a 3-D object. This tactile object, called the “data-objectile”, is a three-dimensional “freeze frame” of a real time environment (or: a NEM) generated by processes in data space, navigations and movements, which is materialized via selective laser sintering (SLS). It is intended to allow for a better orientation in the virtual space. At the same time it is an object with out any reference which could be described as the ultimate simulacrum: a material copy without original.

¹¹ “machinimas” are realtime movies created with modified game engines. Within the gaming subculture the machinimas serve as “intro” movies for these mods.

***nybble-engine-toolZ* (2002/2003)**

while *nybble-engine* is used as a sound generating machine predominantly for Jahrmann's and Moswitzer's hilarious and self-ironic live performances, *nybble-engine-toolZ*, developed in close collaboration with the programmer Brigit Lichtenegger during a two-month residency at the V2_Lab in November-December 2002, is a networked multi user game environment and installation which was premiered at the exhibition „Metadata“ during the Dutch Electronic Arts Festival 2003 in Rotterdam. It consists of a peer-to-peer server network and – unlike *nybble-engine* which is only a “mod” – a “total conversion” of the C++ kernel of the Unreal game engine¹² that self-reflexively constructs the game itself out of network processes. This means that the log files produced by the network traffic itself and data on the hard disk (text, images and sound) become raw material for 3D movies (NEMs) out of which the game environment itself is constructed. Like in *nybble-engine*, the environment thus consists of real-time network activities.¹³ Players may log on to the game from various locations – including the installation, a 180 degree circular screen – to navigate the environment, meet other players and bots and communicate with them. By bumping into data objects players trigger network processes which in return are used to construct the game environment.

A very important further development of *nybble-engine-toolZ* is the possibility to launch Linux server operations from the running *toolZ* application. *nybble-engine toolZ* is thus much closer to the idea of an “operating environment for a server” (if not operating system) than *nybble-engine* ever was. One of the special technical features of *toolZ* is the display of live broadcast ASCII text messages that simultaneously generate visual objectiles, thus indicating and translating data-events. Originally, the idea was to implement each shooting action as an act of communication, i.e. sending an e-mail to an individual e-mail address or addressing individual servers with traceroutes. However, after the attack on Iraq in February 2003 this concept was changed quite radically by the artists. Shots fired in the game environment would now generate anti-war mails to the fixed mail address president@whitehouse.gov or calls for peace to a server of the government of the United States, while pressing

¹² Cf. Brigit Lichtenegger: *Nybble-Engine-ToolZ*, 2002, <http://retina.v2.nl:8080/evolutie/nybble.html>

¹³ Cf. Jury statement, Prix Ars 03, Interactive Arts, <http://www.aec.at/en/prix/updates/article2.asp?iNewsID=309&iFollowUpID=1051>

buttons on the gamepad would send ping commands and display traceroutes to targeted government servers.

Concerning usability, already when developing *nybble-engine* Jahrman and Moswitzer took into account experiences from gaming communities. Various test sessions of *nybble-engine* were performed, e.g., during a public deathmatch with gamers in museumsquartier in Vienna, during a performance at Public Netbase in Vienna, in installations at Casino Luxemburg, the OK Centrum in Linz and during the DEAF festival in Rotterdam. During these sessions, not unexpectedly it turned out that most gamers found *nybble-engine* and *nybble-engine-toolZ* quite unplayable. However, as a conceptual proposal Jahrman / Moswitzer's project opens up the engine and makes it reflective of the process of playing itself: »*nybble-engine-toolZ* points to the possibilities of game playing as editing of code and rewriting of the very tool that creates the game in an open collaborative system.«¹⁴ As a conceptual proposal, *nybble-engine* raises important issues about the constructedness of programmed environments. By creatively re-engineering existing programs Jahrman and Moswitzer thus suggest to think about alternatives to the code that is currently written.

2004

¹⁴ Jury statement, Prix Ars 03, Interactive Arts,
<http://www.aec.at/en/prix/updates/article2.asp?iNewsID=309&iFollowUpID=1051>