Mag.net reader: Experiences in Electronic Cultural Publishing (forthcoming)
Images: http://ospublish.constantvzw.org/documents/awkward

Awkward gestures: designing with Free Software

Open Source Publishing (OSP) is a Brussels based design team that uses Free Libre and Open Source Software (FLOSS), open fonts and copyleft licences for its productions. We aim to make our designs available as source material whenever possible and try to convince our clients to do the same.

[1]

Print Party avant la lettre: production line for an 'exquisite corpse' publication as part of The Tomorrow Book Project (Jan van Eyck Academie, Maastricht, 2006)

We launched OSP because our portfolio started to fill up with designs for alternative music, copyleft activities and Linux Install Parties and the gap between the language of our work and the jargon of the commercial software we used became more obvious with every new job. We were also interested in the role that software plays in the creative process and trying to find out how our digital tools could become a creative and substantial element in design itself. But since the software packages of Adobe Inc. have become quite the standard in art academies, creative studios and printshops, it is difficult to detect their influence, let alone analyse their effect.
The past two years, OSP has made a number of publications, posters, brochures and websites with Free Software and this experience has changed our practice. Although this was clearly our objective, it also led to surprising discoveries about the way we work and what we actually expect from software.

[2]

Print Party 0.2: How To Print A Booklet In 19 Easy Steps; the projected command line interface shows the second last step (Interface 3, Brussels, 2006)

Almost every poster, website or publication that is made nowadays, is the result of a partial or a complete digital process, but worldwide there is just one single company that supplies designers with tools to make them. Adobe's out-of-the-box packages are certainly powerful but as they can be customised only superficially, the wish to make a difference starts to become an argument to choose for a more active engagement with software. It has even lead to the acknowledgement of Open Source as an option, most notably by the Adobe company itself. Design critic David Womack compares it to the production of the T-Ford[1]. Although a streamlined process might be faster, it runs the risk of everything looking the same in the end. Thus, in order to make your mark, a diversification of tools is necessary.

Like with the production of the T-Ford, that of course had much more to it than the fact that from then on cars looked more or less identical, software does not merely determine the boundaries of visual expression. Because it is constantly present, it conditions our practice in terms of division of labour, vocabulary and the physical relationship with the digital medium. Our choice for a different toolset is therefore as much related to ethical as it is to aesthetic considerations; OSP is first of all an attempt to facilitate a design practice that starts from a critical use of technology and explicitly functions in an ecology of knowledge based on distribution and circulation rather than competition and exclusion.

Mastering your tools

At the end of the 19th century, machines increasingly took over the work of typographers, printers and typesetters. Designer and socialist William Morris was convinced that workers should not only have collective ownership of their own means of production, he also believed in another form of mastery, i.e. the skilful employment of techniques and materials. For Morris, there was more to it than just being handy; his Arts and Crafts movement brought together artists and designers who thoroughly reflected upon the influence of the production process on the nature and meaning of everyday objects. For them, getting the job right implied not only the economic ownership of machines and resources, but also the technical mastery of the work instead of being the machine s slave.

[3]

Print Party 0.2: Live publication of Irina Aristarkhova's Maternal Politics during Digitales (Interface 3, Brussels, 2006)

Designer David Reinfurt observes, that the overdetermined functionality and staggering complexity of professional design software makes users restrict themselves to standard techniques and tools[3]. How could Free Software be more empowering? The fundamental difference it makes, is that it allows users to use, analyse, change and distribute source code and in a sense users literally get hold of their means of production. But while a computer programmer, by having the right to adjust software, can feel in control, every other power user with the same rights, is practically blown away by the explosion of procedures, formats and processes she is confronted with. Let alone the fact that the 'means of production' for designers include more than their software[4], our experience of designing with Free Software has shown us over and over again that 'owning' our tools is not the same as 'mastering' them.

[4]

Print Party 2.0: Sophia Loren, All you see I owe to spaghetti *(Quarantaine, Brussels, 2006)*

In Design by numbers[5], the book that led to the development of Processing, a visual programming language that has become popular among designers, John Maeda warns that a clever use of software is often wrongfully considered as craftsmanship. His point is clear; unless we learn to use code as a material, we will never become the master of our software. A comparable argument can be found in the enthusiasm for the commandline interface, as this facilitates a communication with the numerical operations of the machine itself. Without detracting from the thrilling experience of effortlessly commanding the shell or self-confidently manipulating squares and circles in Processing, we need to avoid a tunnel vision of technology where practices, conditions and perspectives can and must be pushed aside to enable a sense of control.

[5]

Print Party 2.0: Kate Rich presents the Cube Cola reverse engineering project, serving Cuba Libre (Quarantaine, Brussels, 2006)

Through cutting a comfortably coherent slice out of the unruly entity that software is, you might miss the opportunity to engage with it in other ways than as a means to an end. Software is source code, but also an interface which, whether graphic or not, represents a particular interaction with the underlying processes. Groups of users gather around certain applications and thereby create patterns of use that make sense of this interaction. Mailing lists and documentation on software are characterized by a specific language and tone, as is the way software developers converse with each

other and their users. When we consider software as culture, it is perhaps possible to drop the rhetoric of master and slave, and we can begin to think about how 'competence' can mean more than 'control'.

Making an account of itself

In The Confessions of Zeno[6], Italo Svevo describes how one evening Zeno strikes up a conversation with a doctor who explains to him at length how 54 muscles come in motion when you walk rapidly. Zeno becomes fascinated by this extraordinary account of the monstrous machinery of his own body, but his curiosity proves to be fatal: *Of course I could not distinguish all its fifty-four parts, but I discovered something terrifically complicated which seemed to get out of order directly I began thinking about it. I limped, leaving that café; and I went on limping for several days.* From that moment on he is unable to think about this memorable evening, the doctor or even about his own legs without starting to stagger.

[6]

Print Party 2.0 (Quarantaine, Brussels, 2006)

Is a similar principle at work in software? Apple promotes its operating system with '*software that just works*' (apparently you don't need to worry about it at all) and also Adobe makes every effort to push the simulations and algorithms, the monstrous machinery, that define the software, into the background. Recognizable patterns are inventively arranged in well-organized and reliable interfaces, minimizing their own presence and creating a feel of naturalness. Free Software on the contrary categorically refuses to disappear out of sight, if only because it's not mainstream. Simply by offering an alternative, it already makes a statement about itself and without even making a spectacular difference, certain automatic actions become visible that otherwise would have remained unnoticed.

[7]

Print Party 2.0: Each of the 19 steps is carefully followed from the paper recipe (Quarantaine, Brussels, 2006)

It could also be a side effect of the Linux/Unix philosophy itself, where the emphasis is on small specific tools that are good at executing relatively simple and well-defined tasks with the intention of giving users as much freedom as possible in order to let them compose their own more complex configurations later. The software remains tangible, because the same recognizable elements can be connected to each other again and again in many different ways. With this modular structure of clearly defined 'clutches' in the form of pipes and standard streams (stdin and stdout), the shift from one action to another is easy to experience. And once you get to know this versatile set of tools a little better, you will detect their traces everywhere, even in more complex graphic applications.

[8]

Print Party 2.0: The 19 commands that we typed one by one into the terminal caused a funny yet fascinating spectacle that ended only when 16 pages were correctly printed, folded and stapled together. (Quarantaine, Brussels, 2006)

The generative principle that characterizes FLOSS has lead to an incredible variety of programs; in graphic interfaces alone there are numerous differences. A volunteering developers community is less motivated to hide their efforts from users (the identity of the project actually matters) so the

convergence of tools that we are accustomed to from Adobe and Apple, is less likely to happen. This can be clearly experienced when working through the differences between Scribus (desktop publishing), Gimp (image editing) and Inkscape (vector graphics editor), three programmes that OSP often employs side-by-side. Whether it's the result of a lack of attention or the outcome of deliberate choices, moving between these programmes reveals the culture of it's developers, it's technical construction and development history. At times this can be destabilizing but more often it is inspiring, as it constantly reminds us of the cultural aspect of software production.
Matthew Fuller introduced the term interrogibility[7] to describe the quality of software to make an account of itself and to share the premises on which it is based with it's users. It is important how well something can be put to use for a specific purpose, but also to what extent it clarifies the processes that it generates and it is here where FLOSS can make a difference. By considering interrogability beyond the obvious level of source code, software opens up to be used in different ways than intended, even as tool to think with.

[9]

Canadian Printing Breakfast: travel report of a visit to the Libre Graphics Meeting in Montreal (Nepomuk, Brussels, 2007)

A sane person, says Zeno, *doesn't analyse himself, doesn t look in the mirror*[8], just like software is paid attention to most of all when it doesn't work. When a hammer is broken, you realize how heavy and how big it actually is, how its weight is relative to your own strength and how its size relates to what you actually wanted to do with it.[9] Also proprietary programs have their bugs and glitches, but it is the automatic reflex of FLOSS developers not to avoid or hide them. On the contrary, it is important that imperfections remain visible so that users feel inspired to report them or do something about them.

[10]

Canadian Printing Breakfast: Pancakes with maple syrup (Nepomuk, Brussels, 2007)

The obligatory use of open standards is the last but not least reason for processes being more explicit in FLOSS. Far from being normalised, they often cause obstructions in the publishing workflow where documents are sent back and forth between authors, designers and printers. The risk of a possible incompatibility compels us to warn, to explain and to be alert during each moment of the process. Conversions are never flawless.

Awkward gestures

Not unlike Zeno's experience, it is difficult to stay in motion when the machinery comes to the foreground. Anyone who has seen a designer at work, knows that the self-assured agility with which a layout is done or how the tension of a digital curve is determined, leaves little or no room for questions about the nature of the underlying processes. Taking doubt into account implies breaking with the natural 'flow' of things and accepting the hitches that aren't always that easy to deal with. It is in this way we have started to understand the importance of performing our practice publicly because it brings out unusual gestures that break with the appeasing elegance of the typical self-assured designer who has everything sorted.

[11]

Canadian Printing Breakfast: Turning a frog into a prince and back. Scribus meets Python (Nepomuk, Brussels, 2007)

While a familiar gesture is one that fits perfectly well in a generally accepted model, an awkward gesture is a movement that is not completely synchronic. It's not a countermovement, nor a break from the norm; it doesn t exist outside of the pattern, nor completely in it. Like a moiré effect reveals the presence of a grid, awkward behaviour can lead to a state of increased awareness; a form of productive insecurity that presents us with openings that help understand the complex interaction between skills, tools and medium.

The Print parties that we organize now and then in a vacant café, a bookstore or a classroom are irregular public appearances whenever we feel the need to report on what we discovered and where we've been; as anti-heroes of our own adventures we keep contact with our fellow designers who are interested in our journey into the exotic territory of BoF, Version Control and GPL3[10]. We make a point of presenting each time a new experiment, of producing something printed and also something edible on site; it is the tension between those parallel processes that defines those infectious events.

Throughout our practice we are looking for forms of reflection that can do without comfortable distance. We use our awkwardness as a strategy to cause interference, to create pivotal moments between falling and moving, an awkward in-between that makes space for thinking without stopping us to act.

[12]

Free Operations: design students produce, cook and eat pasta while we talk to them about Free, Libre and Open Source Software (Werkplaats Typografie, Arnhem, 2007)

1 Steven Heller en David Womack. Becoming a Digital Designer, A Guide to Careers in Web, Video, Broadcast, Game and Animation Design. John Wiley & Sons, 2007.

2 "*It is not this or that... machine which we want to get rid of, but the great intangible machine of commercial tyranny which oppresses the lives of all of us.*" William Morris. Art and Its Producers, and The Arts and Crafts of To-day: Two Addresses Delivered Before the National Association for the Advancement of Art. Longmans & Co., London, 1901.

3 David Reinfurt. Making do and getting by. Software and design. Adobe Design Center Think Tank. http://www.adobe.com/designcenter/thinktank/makingdo (March 2008).

4 See also: Why you should own the beer company you design for (interview with Dmytri Kleiner). http://ospublish.constantvzw.org/?p=380, 2007

5 John Maeda. Design By Numbers. The MIT Press , 2001.

6 Italo Svevo. De bekentenissen van Zeno. Athenaeum-Polak & Van Gennep, 2000.

7 Matthew Fuller. Softness, Interrogability, general intellect; art methodologies in software. Media Research Centre, Huddersfield, 2006.

8 Svevo. 2000.

9 Sarah Ahmed. Queer Phenomenology: Orientations, Objects, Others. Durham and London: Duke University Press, 2006.

10 BoF: Birds of a Feather, informal meetings based on shared interest. Version Control: system to track changes in software development. GPL3: fiercely debated update of the General Public License, now explicitly excluding Digtial Rights Management.