*I think that* conversations *are*
*the best, biggest thing that*
*Free Software has to offer its user*

This book documents an ongoing dialogue between developers and designers involved in the wider ecosystem of Libre Graphics. Its lengthy title, *I think that conversations are the best, biggest thing that Free Software has to offer its user*, is taken from an interview with Debian developer Asheesh Laroia, *Just ask and that will be that*, included in this publication. His remark points at the difference that Free Software can make when users are invited to consider, interrogate and discuss not only the technical details of software, but its concepts and histories as well.

*Conversations* documents discussions about tools and practices for typography, layout and image processing that stretch out over a period of more than eight years. The questions and answers were recorded in the margins of events such as the yearly Libre Graphics Meeting, the Libre Graphics Research Unit, a two-year collaboration between Medialab Prado in Madrid, Worm in Rotterdam, Piksel in Bergen and Constant in Brussels, or as part of documenting the work process of the Brussels' design team OSP. Participants in these intersecting events and organisations constitute the various instances of 'we' and 'I' that you will discover throughout this book.

The transcriptions are loosely organised around three themes: **tools**, **communities** and **design**. At the same time, I invite you to read *Conversations* as a chronology of growing up in Libre Graphics, a portrait of a community gradually grasping the interdependencies between Free Software and design practice.

Femke Snelting
Brussels, December 2014

Computational concepts, their technological language and the hybridisation of creative practice have been successfully explored in Media Arts for a few decades now. Digital was a narrative, a tool and a concept, an aesthetic and political playground of sorts. These experiments created a notion of the digital artisan and creative technologist on the one hand and enabled a new view of intellectual property on the other. They widened a pathway to participation, collaboration and co-creation in creative software development, looking critically at the software as cultural production as well as technological advance.

This book documents conversations between artists, typographers, designers, developers and software engineers involved in Libre Graphics, an independent, self-organised, international community revolving around Free, Libre, Open Source software (F/LOSS). Libre Graphics resembles the community of Media arts of the late twentieth Century, in so far that it is using software as a departure point for creative exploration of design practice. In some cases it adopts software development processes and applies them to graphic design, using version control and platforms such as GitHub, but it also banks on a paradigm shift that Free Software offers – an active engagement with software to bend it, fork it, reshape it – and in that it establishes conversations with a developers community that haven't taken place before.

This pathway was, however, at moments full of tension, created by diverging views on what the development process entails and what it might mean. The conversations brought together in this book resulted from the need to discuss those complex issues and to adress the differences and similarities between design, design production, Free Culture and software development. As in theatre, where it is said that *conflict drives the plot forward*, so it does here. It makes us think harder about the ethics of our practices while we develop tools and technologies for the benefit of all.

The Libre Graphics Meeting (LGM) was brought to my attention in 2012 as an interesting example of dialogue between creative types and developers. The event was running since 2006 and was originally conceived as an annual gathering for discussions about Free and Open Source software used in graphics. At the time I was teaching at the University of Westminster for nearly ten years. The subject was computers, arts and design and it took a variety of forms; sometimes focused on graphic design, sometimes on contemporary media practice, interaction design, software design and mysterious hypermedia. F/LOSS was part of my artistic practice for many years, but its inclusion to the UK Higher Education was a real challenge. My frustration with difficult computer departments grew exponentially year by year and LGM looked like a place to visit and get much needed support.

Super fast-forward to Madrid in April 2013: I landed. Little did I know that this journey would change everything. Firstly, the wonderfully diverse group of people present: artists, designers, software developers, typographers, interface designers, more software developers! It was very exciting listening to talks, overhearing conversations in breaks, observing group discussions and slowly engaging with the Libre Graphics community. Being there to witness how far the F/LOSS community has come was so heartwarming and uplifting, that my enthusiasm was soaring.

The main reason for my attendance at the Madrid LGM was to join the launch of a network of Free Culture aware educators in art, music and design education.[1] Aymeric Mansoux and his colleagues from the Willem De Kooning Academie and the Piet Zwart Institute in Rotterdam convened the first ever meeting of the network with the aim to map out a landscape of current educational efforts as well as to share experiences. I was aware of Aymeric's efforts through his activities with GOTO10 and the *FLOSS+Art* book[2] that they published a couple of years before we finally met. Free Culture was deeply embedded in his artistic and educational practice, and it was really good to have someone like him set the course of discussion.

Lo' and behold the conversation started – we sat in a big circle in the middle of Medialab Prado. The introduction round began, and I thought: there are so many people using F/LOSS in their teaching! Short courses, long courses, BA courses, MA courses, summer schools, all sorts! There were so many solutions presented for overcoming institutional barricades, Adobe marriages and Apple hostages. Individual efforts and group efforts, long term and short, a whole world of conventional curriculums as well as a variety of educational experimentations were presented. Just sitting there, listening about shared troubles and achievements was enough to give me a new surge of energy to explore new strategies for engaging BA level students with F/LOS tools and communities.

Taking part in LGM 2013 was a useful experience that has informed my art and educational practice since. It was clear from the gathering that

---

[1] http://eightycolumn.net/

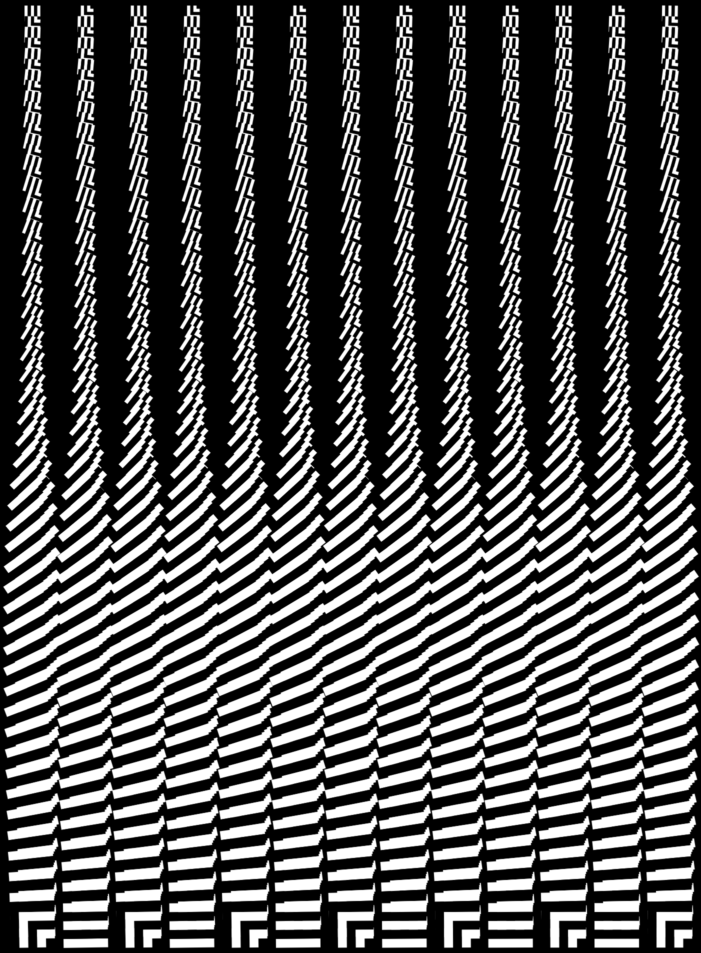
[2] Aymeric Mansoux and Marloes de Valk. *FLOSS+Art*. OpenMute, 2008. http://things.bleu255.com/floss-art

F/LOSS is not a ghetto for idealists and techno fetishists – it was ready for an average user, it was ready for a specialist user, it was ready for all and what is most important the communication lines were open. Given that Linux distributions extend the life of a computer by at least ten years, in combination with the likes of Libre Graphics, Open Video and a plethora of other F/LOS software, the benefits are manyfold, important for all and not to be ignored by any form of creative practice worldwide.

Libre Graphics seems to offer a very exciting transformation of graphic design practice through implementation of F/LOS software development and production processes. A hybridisation across these often separated fields of practice that take under consideration openness and freedom to create, copy, manipulate and distribute, while contributing to the development of visual communication itself. All this may lease a new life to an over-commercialised graphic design practice, banalised by mainstream culture.

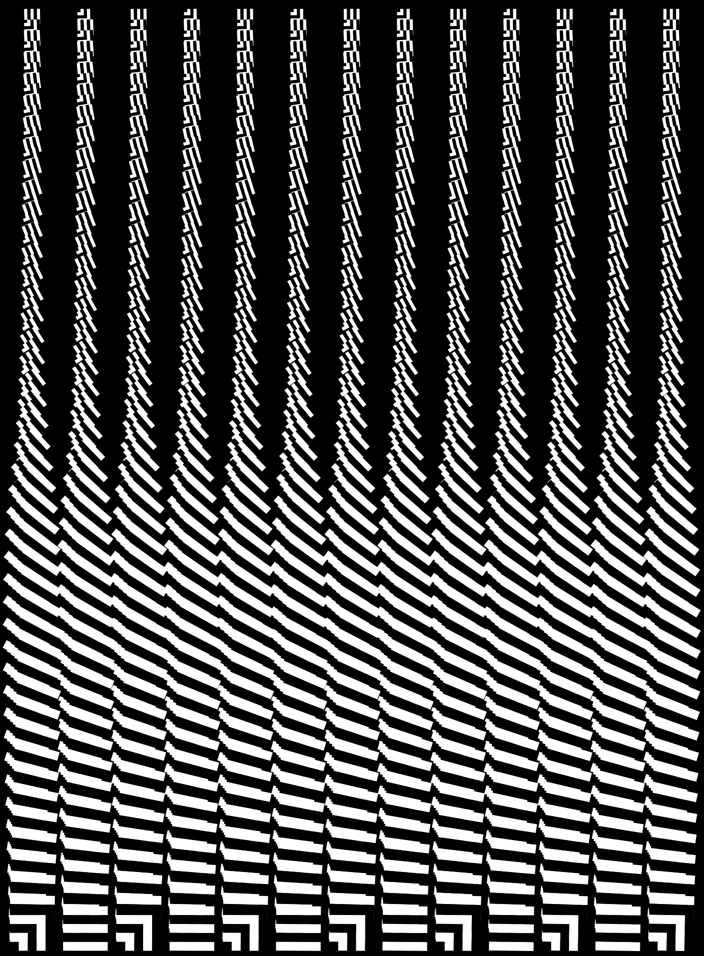
This book brings together reflections on collaboration and co-creation in graphic design, typography and desktop publishing, but also on gender issues and inclusion to the Libre Graphics community. It offers a paradigm shift, supported by historical research into graphic and type design practice, that creates strong arguments to re-engage with the tools of production. The conversations conducted give an overview of a variety of practices and experiences which show the need for more conversations and which can help educate designers and developers alike. It gives detailed descriptions of the design processes, productions and potential trade-offs when engaged in software design and development while producing designed artefacts. It points to the importance of transparent software development, breaking stereotypes and establishing a new image of the designer-developer combo, a fresh perspective of mutual respect between disciplines and a desire to engage in exchange of knowledge that is beneficial beyond what any proprietary software could ever be.

Larisa Blazic is a media artist living and working in London. Her interests range from creative collaborations to intersections between video art and architecture. As senior lecturer at the Faculty of Media, Arts and Design of the University of Westminster, she is currently developing a master's program on F/LOSS art & design.



# A user should not be able to shoot himself in the foot

Ⅲ Andreas Vox
ꟷ Femke Snelting



# A user should not be able to shoot himself in the foot

Andreas Vox
Femke Snelting

While in the background participants of the **Libre Graphics Meeting 2007** start saying goodbye to each other, Andreas Vox makes time to sit down with us to talk about Scribus, the Open Source application for professional page layout. The software is significant not only to it's users that do design with it, but also because Scribus helps us think about links between software, Free Culture and design. Andreas is a mathematician with an interest in system dynamics, who lives and works in Lübeck, Germany. Together with Franz Schmid, Petr Vanek (subik), Riku Leino (Tsoots), Oleksandr Moskalenko (malex), Craig Bradney (MrB), Jean Ghali and Peter Linnel (mrdocs) he forms the core Scribus developer team. He has been working on Scribus since 2003 and is currently responsible for redesigning the internal workings of its text layout system.

*This weekend Peter Linnel presented amongst many other new Scribus features[1], 'The Color Wheel', which at the click of a button visualises documents the way they would be perceived by a colour blind person. Can you explain how such a feature entered into Scribus? Did you for example speak to accessibility experts?*

I don't think we did. The code was implemented by subik[2], a developer from the Czech Republic. As far as I know, he saw a feature somewhere else or he found an article about how to do this kind of stuff, and I don't know where he did it, but I would have to ask him. It was a logic extension of the colour wheel functionality, because if you pick different colours, they look different to all people. What looks like red and green to one person, might look like grey and yellow to other persons. Later on we just extended the code to apply to the whole canvas.

---

[1] http://wiki.scribus.net/index.php/Version_1.3.4%2B-New_Features
[2] Petr Vanek

*It is quite special to offer such a precise preview of different perspectives in your software. Do you think it it is particular to Scribus to pay attention to these kind of things?*

Yeah, sure. Well, the interesting thing is … in Scribus we are not depending on money and time like other proprietary packages. We can ask ourselves: Is this useful? Would I have fun implementing it? Am I interested in seeing how it works? So if there is something we would like to see, we implement it and look at it. And because we have a good contact with our user base, we can also pick up good ideas from them.

*There clearly is a strong connection between Scribus and the world of prepress and print. So, for us as users, it is an almost hallucinating experience that while on one side the software is very well developed when it comes to .pdf export for example, I would say even more developed than in other applications, but than still it is not possible to undo a text edit. Could you maybe explain how such a discrepancy can happen, to make us understand better?*

One reason is, that there are more developers working on the project, and even if there was only one developer, he or she would have her own interests. Remember what George Williams said about FontForge … [3] he is not that interested in nice Graphical User Interfaces, he just makes his own functionality … that is what interests him. So unless someone else comes up who compensates for this, he will stick to what he likes. I think that is the case with all Open Source applications. Only if you have someone interested and able to do just this certain thing, it will happen. And if it is something boring or something else … it will probably not happen. One way to balance this, is to keep in touch with real users, and to listen to the problems they have. At least for the Scribus team, if we see people complaining a lot about a certain feature missing … we will at some point say: *come on, let's do something about it.* We would implement a solution and when we get thanks from them and make them happy, that is always nice.

*Can you tell us a bit more about the reasons for putting all this work into developing Scribus, because a layout application is quite a complex monster with all the elements that need to work together … Why is it important you find, to develop Scribus?*

[3] *I think the ideas behind it are beautiful in my mind*

I use to joke about the special mental state you need to become a Scribus developer … and one part of it is probably megalomania! It is kind of mountain climbing. We just want to do it, to prove it can be done. That must have been also true for Franz Schmid, our founder, because at that time, when he started, it was very unlikely that he would succeed. And of course once you have some feedback, you start to think: *hey, I can do it … it works. People can use it, people can print with it, do things … so why not make it even better?* Now we are following InDesign and QuarkXpress, and we are playing the top league of page layout applications … we're kind of in a competition with them. It is like climbing a mountain and than seeing the next, higher mountain from the top.

*In what way is it important to you that Scribus is Free Software?*

Well … it would not work with closed software. Open software allows you to get other people that also are interested in working on the project involved, so you can work together. With closed software you usually have to pay people; I would only work because someone else wants me to do it and we would not be as motivated. It is totally different. If it was closed, it would not be fun. In Germany they studied what motivates Open Source developers, and they usually list: 'fun'; they want to do something more challenging than at work, and some social stuff is mentioned as well. Of course it is not money.

*One of the reasons the Scribus project seems so important to us, is that it might draw in other kinds of users, and open up the world of professional publishing to people who can otherwise not afford proprietary packages. Do you think Scribus will change the way publishing works? Does that motivate you, when you work on it?*

I think the success of Open Source projects will also change the way people use software. But I do not think it is possible to foresee or plan, in what way this will change. We see right now that Scribus is adopted by all kinds of idealists, who think that is interesting, lets try how far we can go, and do it like that. There are other users that really just do not have the money to pay for a professional page layout application such as very small newspapers associations, sports groups, church groups. They use Scribus because otherwise they would have used a pirated copy of some other software, or another application which is not up to that task, such as a normal word processor. Or otherwise they would have used a deficient application like MS Publisher to do it. I think what Scribus will change, is that more people will be exposed to page layout, and that is a good thing, I think.

*In another interview with the Scribus team*[4]*, Craig Bradney speaks about the fact that the software is often compared with its proprietary competition. He brings up the 'Scribus way of doing things'. What do you think is 'The Scribus Way'?*

I don't think Craig meant it that way. Our goal is to produce good output, and make that easy for users. If we are in doubt, we think for example: InDesign does this in quite an OK way, so we try to do it in a similar way; we do not have any problems with that. On the other hand … I told you a bit about climbing mountains … We cannot go from the one top to the next one just in one step. We have to move slowly, and have to find our ways and move through valleys and that sometimes also limits us. I can say: *I want it this way* but then it is not possible now, it might be on the roadmap, but we might have to do other things first.

*When we use Scribus, we actually thought we were experiencing 'The Scribus Way' through how it differences from other layout packages. First of all, in Scribus there is a lot more attention for everything that happens after the layout is done, i.e. export, error checking etc. and second, working with the text editor is clearly the preferred way of doing layout. For us it links the software to a more classic ways of doing design: a strictly phased process where a designer starts with writing typographic instructions which are carried out by a typesetter, after which the designer pastes everything into the mock-up. In short: it seems easier to do a magazine in Scribus, than a poster. Do you recognize that image?*

That is an interesting thought, I have never seen it that way before. My background is that I did do a newspaper, magazine for a student group, and we were using PageMaker, and of course that influenced me. In a small group that just wants to bring out a magazine, you distribute the task of writing some articles, and usually you have only one or two persons who are capable of using a page layout application. They pull in the stories and make some corrections, and then do the layout. Of course that is a work flow I am

[4] http://www.kde.me.uk/index.php?page=fosdem-interview-scribus

familiar with, and I don't think we really have poster designers or graphic artists in the team. On the other hand … we do ask our users what they think should be possible with Scribus and if a functionality is not there, we ask them to put in a bug report so we do not forget it and some time later we will pick it up and implement it. Especially the possibility to edit from the canvas, this will approve in the upcoming versions.
Some things we just copied from other applications. I think Franz[5] had no previous experience with PageMaker, so when I came to Scribus, and saw how it handled text chains, I was totally dismayed and made some changes right away because I really wanted it to work the way it works in PageMaker, that is really nice. So, previous experience and copying from another applications was one part of the development. Another thing is just technical problems. Scribus is at the moment internally not that well designed, so we first have to rewrite a lot of code to be able to reach some elements. The coding structure for drawing and layout was really cumbersome inside and it was difficult to improve. We worked with 2.500 lines of code, and there were no comments in between. So we broke it down in several elements, put some comments in and also asked Franz: *why did you did this or that*, so we could put some structure back into the code to understand how it works. There is still a lot of work to be done, and we hope we can reach a state where we can implement new stuff more easily.

*It is interesting how the 2.500 lines of code are really tangible when you use Scribus old-style, even without actually seeing them. When Peter Linnel was explaining how to make the application comply to the conservative standards of the printing business, he used this term 'self-defensive code' …*

Andreas Vox

At Scribus we have a value that a file should never break in a print shop. Any bug report we receive in this area, is treated with first priority.

*Femke Snelting*

*We can speak from experience, that this is really true! But this robustness shifts out of sight when you use the inbuilt script function; then it is as if you come in to the software through the backdoor. From self-defence to the heart of the application?*

It is not really self-defence … programmers and software developers sometimes use the expression: 'a user should not shoot himself in the foot'.

[5] Schmid

Scribus will not protect you from ugly layout, if that would be possible at all! Although I do sometimes take deliberate decisions to try and do it … for example that for as long as I am around, I will not make an option to do 'automatic letter spacing', because I think it is just ugly. If you do it manually, that is your responsibility; I just do not feel like making anything like that work automatically. What we have no problems with, is to prevent you from making invalid output. If Scribus thinks a certain font is not OK, and it might break on one or two types of printers … this is reason enough for us to make sure this font is not used. The font is not even used partially, it is gone. That is the kind of self-defence Peter Linnel was talking about. It is also how we build .pdf files and PostScript. Some ways of building PostScript take less storage, some of it would be easier to read for humans, but we always take an approach that would be the least problematic in a print shop. This meant for example, that you could not search in a .pdf.[6] I think you can do that now, but there are still limitations; it is on the roadmap to improve over time, to even add an option to output a web oriented .pdf and a print oriented .pdf … but it is an important value in Scribus is to get the output right. To prevent people to really shoot themselves in the foot.

*Our last question is about the relation between the content that is layed out in Scribus, and the fact that it is an Open Source project. Just as an example, Microsoft Word will come out with an option to make it easy to save a document with a Creative Commons License[7]. Would this, or not, be an interesting option to add to Scribus? Would you be interested in making that connection, between software and content?*

It could well be we would copy that, if it is not already been patented by Microsoft! To me it sounds a bit like a marketing trick … because it is such an easy function to do. But, if someone from Creative Commons would ask for this function, I think someone would implement it for Scribus in a short time, and I think we would actually like it. Maybe we would generalize it a little, so that for example you could also add other licenses too. We already have support for some meta data, and in the future we might put some more function in to support license managing, for example also for fonts.

[6] because the fonts get outlined and/or reencoded

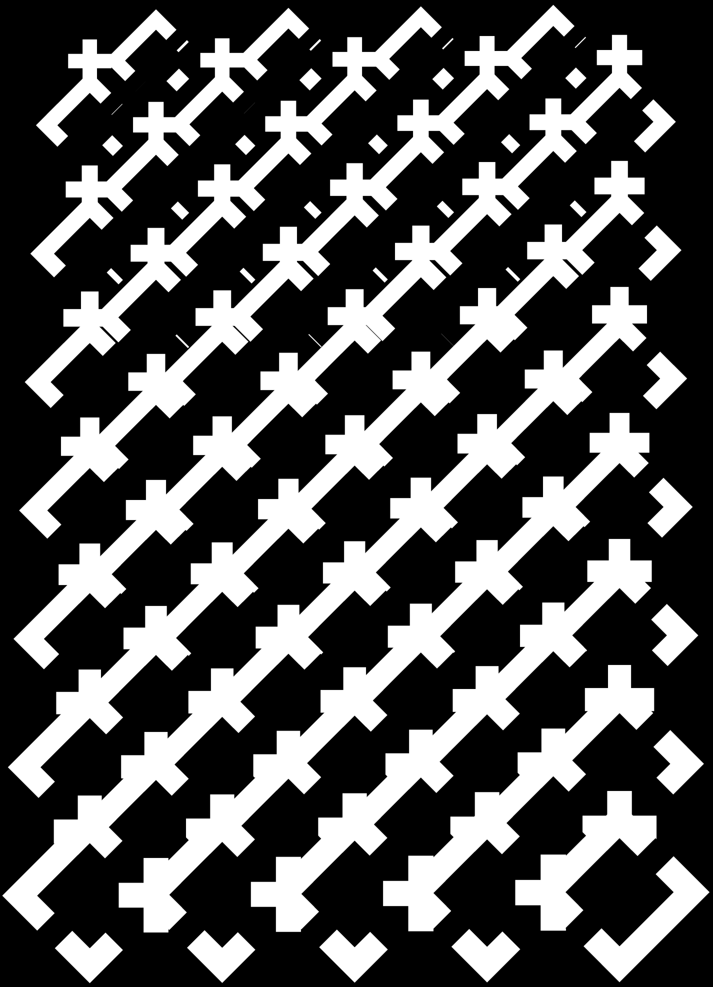
[7] http://creativecommons.org/press-releases/entry/5947

About the relation between content and Open Source software in general … there are some groups who are using Scribus I politically do not really identify with. Or more or less not at all. If I meet those people on the IRC chat, I try to be very neutral, but I of course have my own thoughts in the back of my head.

*Do you think using a tool like Scribus produces a certain kind of use?*

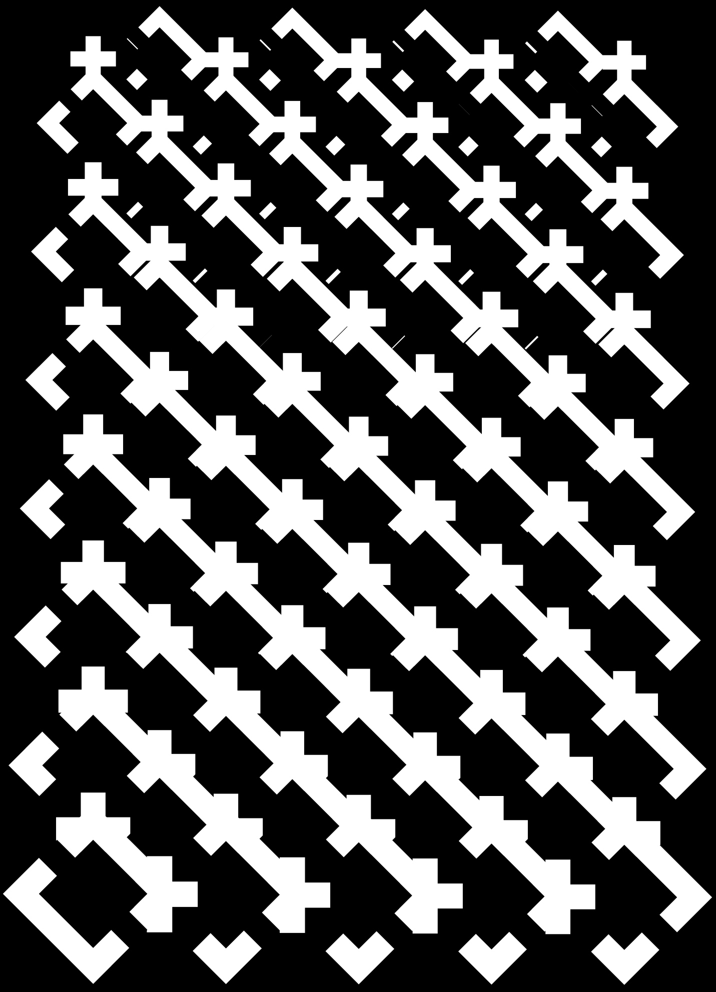
No. Preferences for work tools and political preference are really orthogonal, and we have both. For example when you have some right wing people they could also enjoy using Scribus and socialist groups as well. It is probably the best for Scribus to keep that stuff out of it. I am not even sure about the political conviction of the other developers. Usually we get along very well, but we don't talk about those kinds of things very much. In that sense I don't think that using Scribus will influence what is happening with it.
As a tool, because it makes creating good page layouts much easier, it will probably change the landscape because a lot of people get exposed to page layout and they learn and teach other people; and I think that is growing, and I hope it will be growing faster than if it is all left to big players like InDesign and Quark … I think this will improve and it will maybe also change the demands that users will make for our application. If you do page layout, you get into a new frame of mind … you look in a different way at publications. It is less content oriented, but more layout oriented. You will pick something up and it will spread. People by now have understood that it is not such a good idea to use twelve different fonts in one text … and I think that knowledge about better page layout will also spread.

Andreas Vox



I think the ideas behind it
are beautiful in my mind

✖ Femke Snelting
[] George Williams



I think the ideas behind it
are beautiful in my mind

✖ Femke Snelting
[] George Williams

When we came to the Libre Graphics Meeting for the first time in **2007**, we recorded this rare conversation with George Williams, developer of FontForge, the editing tool for fonts. We spoke about Shakespeare, Unicode, the pleasure of making beautiful things, and pottery.

✖ **We‘re doing these interviews, as we're working as designers on Open Source**

[] OK.

✖ **With Open Source tools, as typographers, but often when we speak to developers they say *well, tell me what you want*, or they see our interest in what they are doing as a kind of feature request or bug report.**

[] (*laughs*) Yes.

✖ **Of course it's clear that that's the way it often works, but for us it's also interesting to think about these tools as really tools, as ways of shaping work, to try and understand how they are made or who is making them. It can help us make other things. So this is actually what we want to talk about. To try and understand a bit about how you've been working on FontForge. Because that's the project you're working on.**

[] OK.

✖ **And how that connects to other ideas of tools or tools' shape that you make. These kind of things. So maybe first it's good to talk about what it is that you make.**

[] OK. Well … FontForge is a font editor.
I started playing with fonts when I bought my first Macintosh, back in the early eighties (actually it was the mid-eighties) and my father studied textual bibliography and looked at the ways the printing technology of the Renaissance affected the publication of Shakespeare's works. And what that meant about the errors in the compositions we see in the copies we have left from the Renaissance. So my father was very interested in Renaissance printing (and has written books on this subject) and somehow that meant

that I was interested in fonts. I'm not quite sure how that connection happened, but it did. So I was interested in fonts. And there was this program that came out in the eighties called Fontographer which allowed you to create PostScript[1] and later TrueType[2] fonts. And I loved it. And I made lots of calligraphic fonts with it.

✖ **You were … like 20?**

[] I was 20-30. Lets see, I was born in 1959, so in the eighties I was in my twenties mostly. And then Fontographer was bought up by Macromedia[3] who had no interest in it. They wanted FreeHand[4] which was done by the same company. So they dropped Fon … well they continued to sell Fontographer but they didn't update it. And then OpenType[5] came out and Unicode[6] came out and Fontographer didn't do this right and it didn't do that right … And I started making my own fonts, and I used Fontographer to provide the basis, and I started writing scripts that would add accents to latin letters and so on. And figured out the Type1[7] format so that I could decompose it — decompose the Fontographer output so that I could add

---

1 PostScript fonts are outline font specifications developed by Adobe Systems for professional digital typesetting, which uses PostScript file format to encode font information.
Wikipedia. PostScript fonts — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

2 TrueType is an outline font standard developed by Apple and Microsoft in the late 1980s as a competitor to Adobe's Type 1 fonts used in PostScript.
Wikipedia. TrueType — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

3 Macromedia was an American graphics, multimedia and web development software company (1992–2005). Its rival, Adobe Systems, acquired Macromedia on December 3, 2005.
Wikipedia. Macromedia — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

4 Adobe FreeHand (formerly Macromedia Freehand) is a computer application for creating two-dimensional vector graphics. Adobe discontinued development and updates to the program. Wikipedia. Adobe FreeHand — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

5 OpenType is a format for scalable computer fonts. It was built on its predecessor TrueType, retaining TrueType's basic structure and adding many intricate data structures for prescribing typographic behavior. Wikipedia. Opentype — wikipedia, the free encyclopedia, 2014. [Online; accessed 18.12.2014]

6 Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.
Wikipedia. Unicode — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

7 Type 1 is a font format for single-byte digital fonts for use with Adobe Type Manager software and with PostScript printers. It can support font hinting. It was originally a proprietary specification, but Adobe released the specification to third-party font manufacturers provided that all Type 1 fonts adhere to it.
Wikipedia. PostScript fonts — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

my own things to it. And then Fontographer didn't do Type0[8] PostScript fonts, so I figured that out.
And about this time, the little company I was working for, a tiny little startup — we wrote a web HTML editor — where you could sit at your desk and edit pages on the web — it was before FrontPage[9], but similar to FrontPage. And we were bought by AOL and then we were destroyed by AOL, but we had stock options from AOL and they went through the roof. So … in the late nineties I quit. And I didn't have to work.
And I went off to Madagascar for a while to see if I wanted to be a primatologist. And … I didn't. There were too many leaches in the rainforest.

✖ **(*laughs*)**

[] So I came back, and I wrote a font editor instead.
And I put it up on the web and in late 99, and within a month someone gave me a bug report and was using it.

✖ **(*laughs*) So it took a month**

[] Well, you know, there was no advertisement, it was just there, and someone found it and that was neat!

✖ **(*laughs*)**

[] And that was called PfaEdit (because when it began it only did PostScript) and I … it just grew. And then — I don't know — three, four, five years ago someone pointed out that PfaEdit wasn't really appropriate any more, so I asked various users what would be a good name and a french guy said *How 'bout FontForge?* So. It became FontForge then. — That's a much better name than PfaEdit.

✖ **(*laughs*)**

[] Used it ever since.

✖ **But your background … you talked about your father studying …**

---

8 Type 0 is a 'composite' font format . A composite font is composed of a high-level font that references multiple descendent fonts.
Wikipedia. PostScript fonts — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

9 Microsoft FrontPage is a WYSIWYG HTML editor and Web site administration tool from Microsoft discontinued in December 2006.
Wikipedia. Microsoft FrontPage — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

[] I grew up in a household where Shakespeare was quoted at me every day, and he was an English teacher, still is an English teacher, well, obviously retired but he still occasionally teaches, and has been working for about 30 years on one of those versions of Shakespeare where you have two lines of Shakespeare text at the top and the rest of the page is footnotes. And I went completely differently and became a mathematician and computer scientist and worked in those areas for almost twenty years and then went off and tried to do my own things.

✖ **So how did you become a mathematician?**

[] (*pause*) I just liked it.

✖ **(*laughs*) *just liked it***

[] I was good at it. I got pushed ahead in high school. It just never occurred to me that I'd do anything else — until I met a computer. And then I still did maths because I didn't think computers were — appropriate — or — I was a snob. How about that.

✖ **(*laughs*)**

[] But I spent all my time working on computers as I went through university. And then got my first job at JPL[10] and shortly thereafter the shuttle[11] blew up and we had some — some of our experiments — my little group — flew on the shuttle and some of them flew on an airplane which went over the US took special radar pictures of the US. We also took special radar pictures of the world from the shuttle (SIR-A, SIR-B, SIR-C). And then our airplane burned up. And JPL was not a very happy place to work after that. So then I went to a little company with some college friends of mine, that they'd started, created compilers and debuggers — do you know what those are?

✖ **Mm-hmm.**

[] And I worked a long time on that, and then the internet came out and found another little company with some friends — and worked on HTML.

[10] Jet Propulsion Laboratory

[11] The Space Shuttle Challenger disaster occurred on January 28, 1986, when the NASA Space Shuttle orbiter Challenger broke apart 73 seconds into its flight, leading to the deaths of its seven crew members.

Wikipedia. Space Shuttle Challenger disaster — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

✖ **So when, before we moved, I was curious about, I wanted you to talk about a Shakespearian influence on your interest in fonts. But on the other hand you talk about working in a company where you did HTML editors at the time you actually started, I think. So do you think that is somehow present … the web is somehow present in your — in how FontForge works? Or how fonts work or how you think about fonts?**

[] I don't think the web had much to do with my — well, that's not true. OK, when I was working on the HTML editor, at the time, mid-90s, there weren't any Unicode fonts, and so part of the reason I was writing all these scripts to add accents and get Type0 support in PostScript (which is what you need for a Unicode font) was because I needed a Unicode font for our HTML product.
To that extent — yes-s-s-s.
It had an effect. Aside from that, not really.
The web has certainly allowed me to distribute it. Without the web I doubt anyone would know — I wouldn't have any idea how to 'market' it. If that's the right word for something that doesn't get paid for. And certainly the web has provided a convenient infrastructure to do the documentation in.
But — as for font design itself — that (the web) has certainly not affected me.
Maybe with this creative commons talk that Jon Phillips was giving, there may be, at some point, a button that you can press to upload your fonts to the Open Font Library[12] — but I haven't gotten there yet, so I don't want to promise that.

✖ **(*laughs*) But no, indeed there was – hearing you speak about ccHost[13] – that's the …**

[] Mm-hmm.

✖ **… Software we are talking about?**

[] That's what the Open Font Library uses, yes.

[12] Open Font Library is a project devoted to the hosting and encouraged creation of fonts released under Free Licenses.
Wikipedia. Open Font Library — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

[13] ccHost is a web-based media hosting engine upon which Creative Commons' ccMixter remix web community is built. Wikipedia. CcHost — Wikipedia, The Free Encyclopedia, 2012. [Online; accessed 18.12.2014]

✖ **Yeah. And a connection to FontForge could change the way, not only how you distribute fonts, but also how you design fonts.**

[] It — it might. I don't know … I don't have a view of the future.
I guess to some extent, obviously font design has been affected by requiring it (the font) to be displayed on a small screen with a low resolution display. And there are all kinds of hacks in modern fonts formats for dealing with low resolution stuff. PostScript calls them hints and TrueType calls them instructions. They are different approaches to the same thing. But that, that certainly has affected font design in the last — well since PostScript came out.
The web itself? I don't think that has yet been a significant influence on font design, but then — I'm no longer a designer. I discovered I was much better at designing font editors than at designing fonts.
So I've given up on that aspect of things.

✖ **Mm-K, because I'm curious about your making a division about being a designer, or being a font-editor-maker, because for me that same definition of maker, these two things might be very related.**

[] Well they are. And I only got in to doing it because the tools that were available to me were not adequate. But I have found since — that I'm not adequate at doing the design, there are many people who are better at designing — designing fonts, than I am. And I like to design fonts, but I have made some very ugly ones at times.
And so I think I will — I'll do that occasionally, but that's not where I'm going to make a mark.
Mostly now —
I just don't have the —
The font editor itself takes up so much of time that I don't have the energy, the enthusiasm, or anything like that to devote to another major creative project. And designing a font is a major creative project.

✖ **Well, can we talk about the major creative project of designing a font editor? I mean, because I'm curious how — how that is a creative project for you — how you look at that.**

[] I look at it as a puzzle. And someone comes up to me with a problem, and I try and figure out how to solve it. And sometimes I don't want to figure out how to solve it. But I feel I should anyway. And sometimes I don't want to figure out how to solve it and I don't.

That's one of the glories of being one's own boss, you don't have to do everything that you are asked.

But — to me — it's just a problem. And it's a fascinating problem. But why is it fascinating? — That's just me. No one else, probably, finds it fascinating. Or — the guys who design FontLab probably also find it fascinating, there are two or three other font design programs in the world. And they would also find it fascinating.

✖ **Can you give an example of something you would find fascinating?**

[] Well. Dave Crossland who was sitting behind me at the end was talking to me today — he sat down — we started talking after lunch but on the way up the stairs — at first he was complaining that FontForge isn't written with a standard widget set. So it looks different from everything else. And yes, it does. And I don't care. Because this isn't something which interests me.

On the other hand he was saying that what he also wanted was a paragraph level display of the font. So that as he made changes in the font he could see a ripple effect in the paragraph.

Now I have a thing which does a word level display, but it doesn't do multi-lines. Or it does multi-lines if you are doing Japanese (vertical writing mode) but it doesn't do multi-columns then. So it's either one vertical row or one horizontal row of glyphs.

And I do also have a paragraph level display, but it is static. You bring it up and it takes the current snapshot of the font and it generates a real TrueType font and pass it off to the X Window[14] rasterizer — passes it off to the standard Linux toolchain (FreeType) as that static font and asks that toolchain to display text.

So what he's saying is *OK, do that, but update the font that you pass off every now and then*. And *Yeah, that'd be interesting to do. That's an interesting project to work on*. Much more interesting than changing my widget set which is just a lot of work and tedious. Because there is nothing to think about. It's just *OK, I've got to use this widget instead of my widget.* My widget does

[14] The X Window System is a windowing system for bitmap displays, common on UNIX-like computer operating systems. X provides the basic framework for a GUI environment: drawing and moving windows on the display device and interacting with a mouse and keyboard. Wikipedia. X Window System — Wikipedia, The Free Encyclopedia, 2014. [Online; accessed 18.12.2014]

exactly what I want — because I designed it that way — how do I make this thing, which I didn't design, which I don't know anything about, do exactly what I want?
And — that's dull. For me.

✖ **Yeah, well.**

[] Dave, on the other hand, is very hopeful that he'll find some poor fool who'll take that on as a wonderful opportunity. And if he does, that would be great, because not having a standard widget set is one of the biggest complaints people have. Because FontForge doesn't look like anything else. And people say *Well the grey background is very scary.* [15]
I thought it was normal to have a grey background, but uh … that's why we now have a white background. A white background may be equally scary, but no one has complained about it yet.

✖ **Try red.**

[] I tried light blue and cream. One of them I was told gave people migraines — I don't remember specifically what the comment was about the light blue, but

(someone from `inkscape`): *Make it configurable.*

[] Oh, it is configurable, but no one configures it.

(someone from `inkscape`): *Yeah, I know.*

[] So …

✖ **So, you talked about spending a lot of time on this project, how does that work, you get up in the morning and start working on FontForge? Or …**

[] Well, I do many things. Some mornings, yes, I get up in the morning and I start working on FontForge and I cook breakfast in the background and eat breakfast and work on FontForge. Some mornings I get up at four in the morning and go out running for a couple of hours and come back home and sort of collapse and eat a little bit and go off to yoga class and do a pilates class and do another yoga class and then go to my pottery class, and go to the farmers' market and come home and I haven't worked on FontForge at all. So it varies according to the day. But yes I …

[15] It used to have a grey background, now it has a white background

There was a period where I was spending 40, 50 hours a week working on FontForge, I don't spend that much time on it now, it's more like 20 hours, though the last month I got all excited about the release that I put out last Tuesday — today is Sunday. And so I was working really hard — probably got up to — oh — 30 hours some of that time. I was really excited about the change. All kinds of things were different — I put in Python scripting, which people had been asking for — well, I'm glad I've done it, but it was actually kind of boring, that bit — the stuff that came before was — fascinating.

✖ **Like?**

[] I — are you familiar with the OpenType spec? No. OK. The way you … the way you specify ligatures and kerning in OpenType can be looked at at several different levels. And the way OpenType wants you to look at it, I felt, was unnecessarily complicated. So I didn't look at it at that level. And then after about 5 years of looking at it that way I discovered that the reason I thought it was unnecessarily complicated was because I was only used to Latin or Cyrillic or Greek text, and for Latin, Cyrillic or Greek, it probably is unnecessarily complicated. But for Indic scripts it is not unnecessarily complicated, and you need all those things. So I ripped out all of the code for specifying strange glyph conversions. You know in Arabic a character looks different at the beginning of a word and so on? So that's also handled in this area. And I ripped all that stuff out and redid it in the way that OpenType wanted it to be done and not the somewhat simplified but not sufficiently powerful method that I'd been using up until then.
And that I found, quite fascinating.
And once I'd done that, it opened up all kinds of little things that I could change that made the font editor itself bettitor. Better. Bettitor?

✖ **(*laughs*) That's almost Dutch.**

[] And so after I'd done that the display I talked about which could show a word — I realized that I should redo that to take advantage of what I had done. And so I redid that, and it's now, it's now much more usable. It now shows — at least I hope it shows — more of what people want to see when they are working with these transformations that apply to the font, there's now a list of the various transformations, that can be enabled at any time and then it goes through and does them — whereas before it just sort of —

well it did kerning, and if you asked it to it would substitute this glyph so you could see what it would look like — but it was all sort of — half-baked. It wasn't very elegant.
And — it's much better now, and I'm quite proud of that.
It may crash — but it's much better.

✖ **So you bring up half-baked, and when we met we talked about bread baking.**

[] Oh, yes.

✖ **And the pleasure of handling a material when you know it well. Maybe make reliable bread — meaning that it comes out always the same way, but by your connection to the material you somehow — well — it's a pleasure to do that. So, since you've said that, and we then went on talking about pottery — how clay might be of the same — give the same kind of pleasure. I've been trying to think — how does FontForge have that? Does it have that and where would you find it or how is the ...**

[] I like to make things. I like to make things that — in some strange definition are beautiful. I'm not sure how that applies to making bread, but my pots — I think I make beautiful pots. And I really like the glazing I put onto them.
It's harder to say that a font editor is beautiful. But I think the ideas behind it are beautiful in my mind — and in some sense I find the user interface beautiful. I'm not sure that anyone else in the world does, because it's what I want, but I think it's beautiful.
And there's a satisfaction in making something — in making something that's beautiful. And there's a satisfaction too (as far as the bread goes) in making something I need. I eat my own bread — that's all the bread I eat (except for those few days when I get lazy and don't get to make bread that day and have to put it off until the next day and have to eat something that day — but that doesn't happen very often).
So it's just — I like making beautiful things.

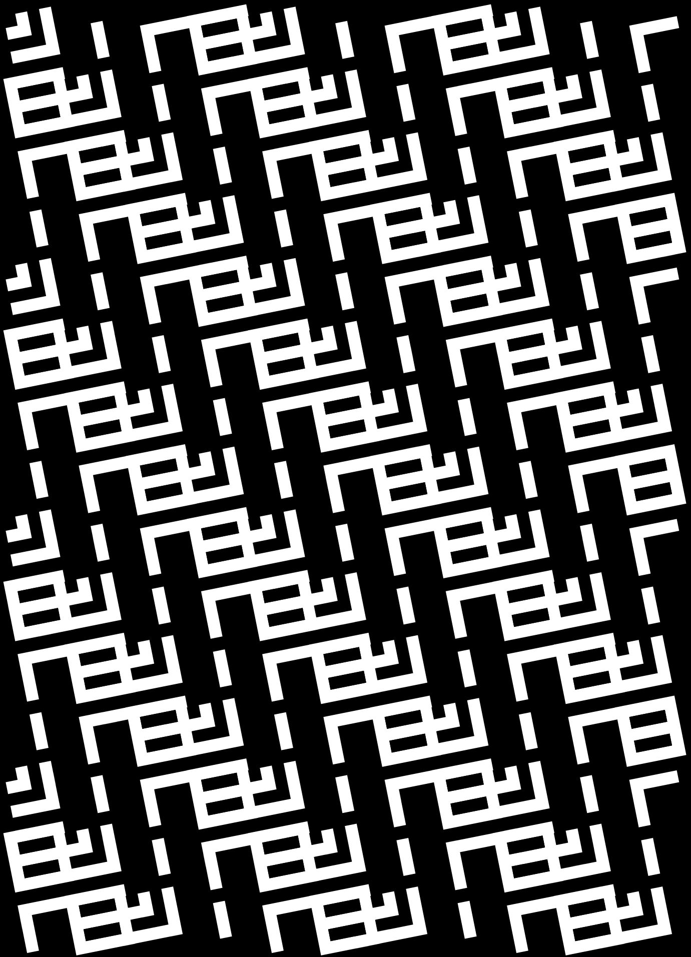
✖ **OK, thank you.**

[] Mm-hmm.

✖ **That was very nice, thank you very much.**

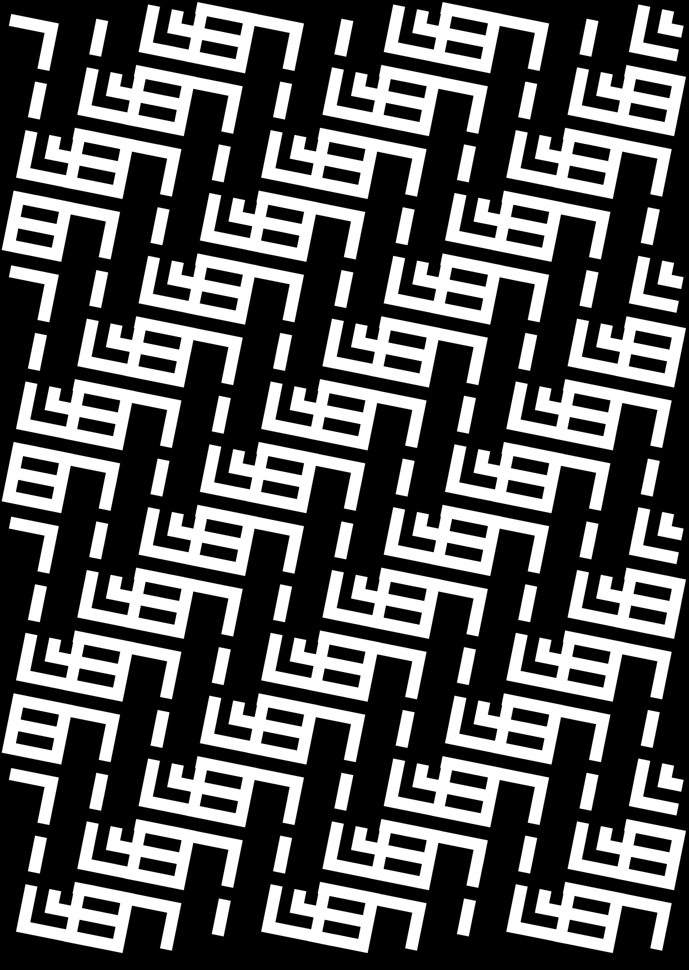
[] Thank you. I have pictures of my pots if you'd like to see them?

✖ **Yes, I would very much like to see them.**



We will get to know the machine
and we will understand

└ Juliane de Moerlooze  - Femke Snelting
◫ An Mertens  ┐ Agnes Bewer



We will get to know the machine
and we will understand

Juliane de Moerlooze – Femke Snelting
An Mertens Agnes Bewer

This conversation with Juliane de Moerlooze was recorded in **March 2009**.

*When you hear people talk about women having more sense for the global, intuitive and empathic … and men are more logical … even if it is true … it seems quite a good thing to have when you are doing math or software?*

Juliane is a Brussels based computer scientist, feminist and Linux user from the beginning. She studied math, programming and system administration and participates in Samedies.[1] In February 2009 she was voted president of the Brussels Linux user group (BXLug).

– *I will start at the end … you have recently become president of the BXLug. Can you explain to us what it is, the BXLug?*

∟ It is the Brussels Linux user group, a group of Linux users who meet regularly to really work together on Linux and Free Software. It is the most active group of Linux users in the French speaking part of Belgium.

▥ **How did you come into contact with this group?**

∟ That dates a while back. I have been trained in Linux a long time ago …

– *Five years? Ten years? Twenty years?*

∟ Almost twenty years ago. I came across the beginnings of Linux in 1995 or 1996, I am not sure. I had some Slackware[2] installed, I messed around with friends and we installed everything … then I heard people talk about Linux distributions[3] and decided to discover something else, notably Debian.[4]

[1] *Femmes et Logiciels Libres*, group of women maintaining their own server http://samedi.collectifs.net
[2] one of the earliest Linux distributions
[3] a distribution is a specific collection of applications and a software kernel
[4] one of the largest Linux distributions

It is good to know that with Linux you really have a diversity, there are distributions specially for audio, there are distributions for the larger public with graphical interfaces, there are distributions that are a bit more 'geek', in short you find everything: there are thousands of distributions but there are a few principal ones and I heard people talk about an interesting development, which was Debian. I wanted to install it to see, and I discovered the BXLug meetings, and so I ended up there one Sunday.

▬ *What was your experience, the first time you went?*

┗ (*laughs*) Well, it was clear that there were not many women, certainly not. I remember some sessions ...

┓ ***What do you mean, not many women? One? Or five?***

┗ Usually I was there on my own. Or maybe two. There was a time that we were three, which was great. There was a director of a school who pushed Free Software a lot, she organised real 'Journées du Libre'[5] at her school, to which she would invite journalists and so on. She was the director but when she had free time she would use it to promote Free Software, but I haven't seen her in a while and I don't know what happened since. I also met Faty, well ... I wasn't there all the time either because I had also other things to do. There was a friendly atmosphere, with a little bar where people would discuss with each other, but many were cluttered together in the middle of the room, like autists hidden behind their computers, without much communication. There were other members of the group who like me realised that we were humans that were only concentrating on our machines and not much was done to make new people feel welcome. Once I realised, I started to move to the back of the room and say hello to people arriving. Well, I was not the only one who started to do that but I imagine it might have felt like a closed group when you entered for the first time. I also remember in the beginning, as a girl, that ... when people asked questions ... nobody realised that I was actually teaching informatics. It seemed there was a prejudice even before I had a chance to answer a question. That's a funny thing to remember.

▬ *Could you talk about the pleasure of handling computers? You might not be the kind of person that loses herself in front of her computer, but you have a strong*

[5] Journées du Libre is a yearly festival organised by the BXLug

*relationship with technology which comes out when you open up the commandline … there's something in you that comes to life.*

Oh, yes! To begin with, I am a mathematician ('matheuse'), I was a math teacher, and I have been programming during my studies and yes, there was something fantastic about it … informatics for me is all about logic, but logic in action, dynamic logic. A machine can be imperfect, and while I'm not specialised in hardware, there is a part on which you can work, a kind of determinism that I find interesting, it poses challenges because you can never know all, I mean it is not easy to be a real system administrator that knows every detail, that understands every problem. So you are partially in the unknown, and discovering, in a mathematical world but a world that moves. For me a machine has a rhythm, she has a cadence, a body, and her state changes. There might be things that do not work but it can be that you have left in some mistakes while developing etcetera, but we will get to know the machine and we will understand. And after, you might create things that are maybe interesting in real life, for people that want to write texts or edit films or want to communicate via the Internet … these are all layers one adds, but you start … I don't know how to say it … the machine is at your service but you have to start with discovering her. I detest the kind of software that asks you just to click here and there and than it doesn't work, and than you have to restart, and than you are in a situation where you don't have the possibility to find out where the problem is.

***When it doesn't show how it works?***

For me it is important to work with Free Software, because when I have time, I will go far, I will even look at the source code to find out what's wrong with the interface. Luckily, I don't have to do this too often anymore because software has become very complicated, twenty years later. But we are not like persons with machines that just click … I know many people, even in informatics, who will say 'this machine doesn't work, this thing makes a mistake'

*The fact that Free Software proposes an open structure, did that have anything to do with your decision to be a candidate for BXLug?*

Well, last year I was already very active and I realised that I was at a point in my life that I could use informatics better, and I wanted to work in this field, so I spent much time as a volunteer. But the moment that I decided, now this is enough, I need to put myself forward as a candidate, was after a series of sexist incidents. There was for example a job offer on the BXLug mailing list that really needed to be responded to … I mean … what was that about? To be concrete: Someone wrote to the mailing list that his company was looking for a developer in so and so on and they would like a Debian developer type applying, or if there weren't any available, it would be great if it would be a blond girl with large tits. Really, a horrible thing so I responded immediately and than it became even worse because the person that had posted the original message, sent out another one asking whether the women on the list were into castration and it took a large amount of diplomacy to find a way to respond. We discussed it with the Samediennes [6] and I though about it … I felt supported by many people that had well understood that this was heavy and that the climate was getting nasty but in the end I managed to send out an ironic message that made the other person excuse himself and stop these kind of sexist jokes, which was good. And after that, there was another incident, when the now ex-president of the group did a radio interview. I think he explained Free Software relatively well to a public that doesn't know about it, but as an example how easy it is to use Free Software, he said *even my wife, who is zero with computers, knows how it works*, using the familiar cliché without any reservation. We discussed this again with the Samediennes, and also internally at the BXLug and than I thought: well, what is needed is a woman as president, so I need to present myself. So it is thanks to the Samedies, that this idea emerged, out of the necessity to change the image of Free Software.

*In software and particularly in Free Software, there are relatively few women participating actively. What kinds of possibilities do you see for women to enter?*

It begins already at school … all the clichés girls hear … it starts there. We possibly have a set of brains that is socially constructed, but when you hear people talk about women having more sense for the global, intuitive and empathic … and men are more logic … even if it is true … it seems quite a good thing to have when you are doing math or software? I mean, there is no handicap we start out with, it is a social handicap … convincing girls to become a secretary rather than a system administrator.

[6] Participants in the Samedies: Femmes et logiciels libres (http://www.samedies.be)

*I am assuming there is a link between your feminism and your engagement with Free Software …*

It is linked at the point where … it is a political liaison which is about re-appropriating tools, and an attempt to imagine a political universe where we are ourselves implicated in the things we do and make, and where we collectively can discuss this future. You can see it as something very large, socially, and very idealist too. You should also not idealise the Free Software community itself. There's an anthropologist who has made a proper description[7] … but there are certainly relational and organisational problems, and political problems, power struggles too. But the general idea … we have come to the political point of saying: we have technologies, and we want to appropriate them and we will discuss them together. I feel I am a feminist … but I know there are other kinds of feminism, liberal feminism for example, that do not want to question the political economical status quo. My feminism is a bit different, it is linked to eco-feminism, and also to the re-appropriation of techniques that help us organise as a group. Free Software can be … well, there is a direction in Free Software that is linked to 'Free Enterprise' and the American Dream. Everything should be possible: start-ups or pin-ups, it doesn't matter. But for me, there is another branch much more 'libertaire' and left-wing, where there is space for collective work and where we can ask questions about the impact of technology. It is my interest of course, and I know well that even as president of the BXLug I sometimes find myself on the extreme side, so I will not speak about my 'libertaire' ideas all the time in public, but if anyone asks me … I know well what is at stake but it is not necessarily representative of the ideas within the BXLug.

*Are their discussions between members, about the varying interests in Free Software? I can imagine there are people more excited about efficiency and performativity of these tools, and others attracted by it's political side.*

Well, these arguments mix, and also since some years there is unfortunately less of a fundamental discussion. At the moment I have the impression that we are more into 'things to do' when we meet in person. On the mailing list there are frictions and small provocations now and then, but the really interesting debates are over, since a few years … I am a bit disappointed in

[7] Christophe Lazarro. La liberté logicielle. Une ethnographie des pratiques d'échange et de coopération au sein de la communauté Debian. Academia editons, 2008

that, actually. But it is not really a problem, because I know other groups that pose more interesting questions and with whom I find it more interesting to have a debate. Last year we have been working away like small busy bees, distributing the general idea of Free Software with maybe a hint to the societal questions behind but in fact not marking it out as a counterweight to a commercialised society. We haven't really deepened the problematics, because for me ... it is clear that Free Software has won the battle, they have been completely recuperated by the business world, and now we are in a period where tendencies will become clear. I have the impression that with the way society is represented right now ... where they are talking about the economical crisis ... and that we are becoming a society of 'gestionnaires' and ideological questions seem not very visible.

Femke Snelting

– *So do you think it is more or less a war between two tendencies, or can both currents coexist, and help each other in some way?*

Juliane de Moerlooze

The current in Free Software that could think about resistance and ask political questions and so on, does not have priority at the moment. But what we can have is debates and discussions from person to person and we can interpolate members of the BXLug itself, who really sometimes start to use a kind of marketing language. But it is relational ... it is from person to person. At the moment, what happens on the level of businesses and society, I don't know. I am looking for a job and I see clearly that I will need to accept the kinds of hierarchies that exist but I would like to create something else. The small impact a group like BXLug can make ... well, there are several small projects, such as the one to develop a distribution specifically designed for small organisations, to which nobody could object of course. Different directions coexist, because there is currently not any project with enough at stake that it would shock the others.

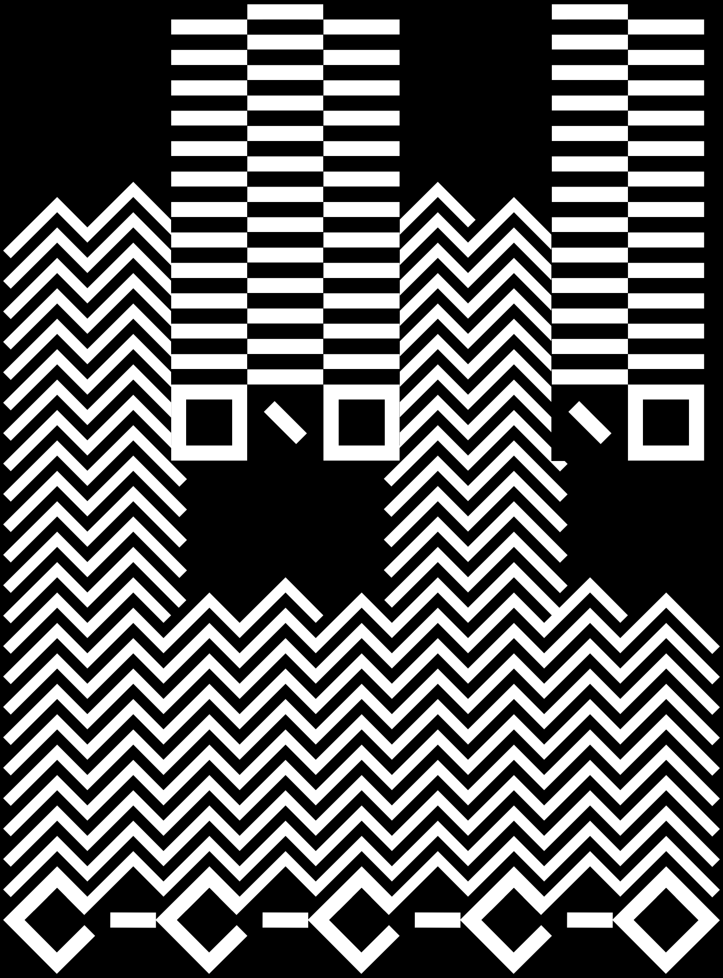
– *To go once again from a large scale to a small scale ... how would you describe your own itinerary from mathematics to working on and with software?*

I did two bachelors at the University Libre de Bruxelles, and than I studied to become a math teacher. I had a wonderful teacher, and we were into the pleasure of exercising our brains, and discovering theory but a large part of our courses were concentrated on pedagogy and how to become a good teacher, how to open up the mind of a student in the context of a course. That's when I discovered another pleasure, of helping a journey into a kind

of math that was a lot more concrete, or that I learned to render concrete. One of the difficult subjects you need to teach in high schools, is scales and plans. I came up with a rendering of a submarine and all students, boys as well as girls, were quickly motivated, wanting to imagine themselves at the real scale of the vessel. I like math, because it is not linked to a pre-existing narrative structure, it is a theoretical construct we accept or not, like the rules of a game. For me, math is an ideal way to form a critical mind.
When you are a child, math is fundamentally fiction, full stop. I remember that when I learned modern math at school … I had an older teacher, and she wasn't completely at ease with the subject. I have the impression that because of this … maybe it was a question of the relation between power and knowledge … she did not arrive with her knowledge all prepared, I mean it was a classical form of pedagogy, but it was a new subject to her and there was something that woke up in me, I felt at ease, I followed, we did not go too fast …

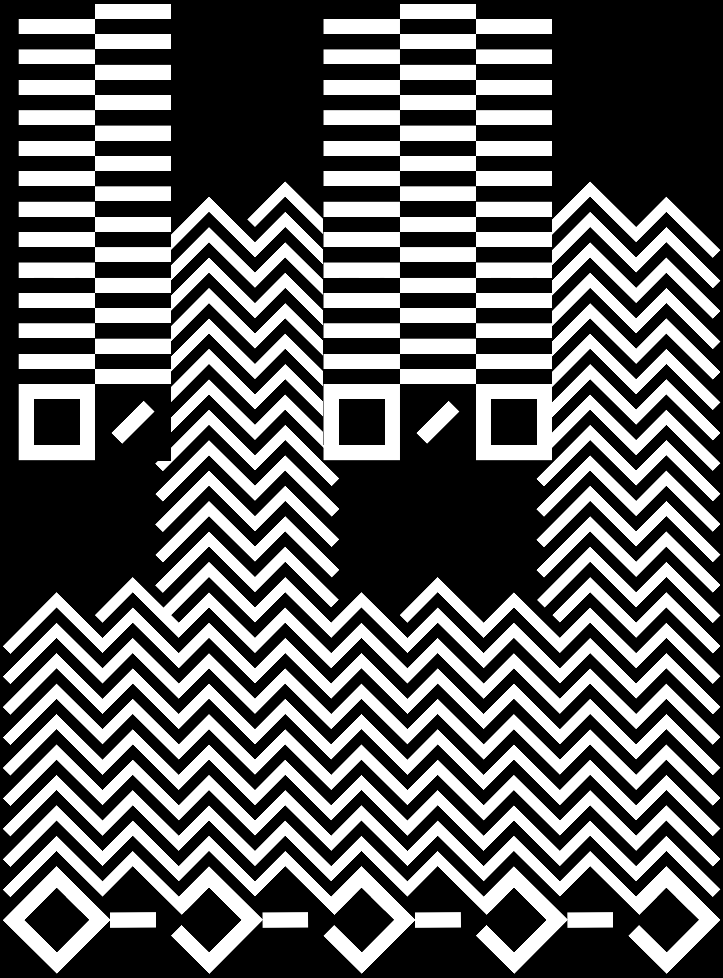
|| ***It was open knowledge, not already formed and closed?***

L Well, we discovered the subject together with the teacher. It might sound bizarre, and she certainly did not do this on purpose, but I immediately felt confident, which did not have too much to do with the subject of the class, but with the fact that I felt that my brains were functioning.
I still prefer to discover the solution to a mathematical problem together with others. But when it comes to software, I can be on my own. In the end it is me, who wants to ask myself: why don't I understand? Why don't I make any progress? In Free Software, there is the advantage of having lots of documentation and manuals available online, although you can almost drown in it. For me, it is always about playing with your brain, there is at least always an objective where I want to arrive, whether it is understanding theory or software … and in software, it is also clear that you want something to work. There is a constraint of efficiency that comes in between, that of course somehow also exists in math, but in math when you have solved a problem, you have solved it on a piece of paper. I enjoy the game of exploring a reality, even if it is a virtual one.



# ConTeXt

□ John Haltiwanger
\ Femke Snelting



## ConTeXt

John Haltiwanger
Femke Snelting

In **September 2013** writer, developer, freestyle rapper and poet John Haltiwanger joined the ConTeXt user meeting in Brejlov (Czech Republic)[1] to present his ideas on Subtext, 'A Proposed Processual Grammar for a Multi-Output Pre-Format'. The interview started as a way to record John's impressions fresh from the meeting, but moved into discussing the future of layout in terms of ballistics.

*How did you end up going to the ConTeXt meeting? Actually, where was it?*

It was in Brejlov, which apparently might not even be a town or city. It might specifically be a hotel. But it has its own … it's considered a location, I guess. But arriving was already kind of a trick, because I was under the impression there was a train station or something. So I was asking around: *Where is Brejlov? What train do I take to Brejlov?* But nobody had any clue, that this was even something that existed. So that was tricky. But it was really a beautiful venue. How I ended up at the conference specifically? That's a good question. I'm not an incredibly active member on the ConTeXt mailing list, but I pop up every now and again and just kind of express a few things that I have going on. So initially I mentioned my thesis, back in January or maybe March, back when it was really unformulated. Maybe it was even in 2009. But I got really good responses from Hans.[2] Originally, when I first got to the Netherlands in 2009 in August, the next weekend was the third annual ConTeXt meeting. I had barely used the software at that point, but I had this sort of impulse to go. Well anyway, I did not have the money for it at that time. So the fact that there was another one coming round, was like: *Ok, that sounds good.* But there was something … we got into a conversation on the mailing list. Somebody, a non-native English speaker was asking about pronouns and gendered pronouns and the proper way of 'pronouning' things. In English we don't have a suitable gender neutral pronoun. So he asked the questions and some guy responded: *The*

[1] http://meeting.contextgarden.net/2013/

[2] Hans Hagen is the principal author and developer of ConTeXt, past president of NTG, and active in many other areas of the TeX community

Hans Hagen – Interview – TeX Users Group. http://tug.org/interviews/hagen.html, 2006. [Online; accessed 18.12.2014] *proper way to do it, is to use* ***he****. It's an invented problem. This whole question is an invented question and there is no such thing as a need for considering any other options besides this.* [3] So I wrote back and said: *That's not up to you to decide, because if somebody has a problem, than there is a problem.* So I kind of naively suggested that we could make a Unicode character, that can stand in, like a typographical element, that does not necessarily have a pronounciation yet. So something that, when you are reading it, you could either say he or she or they and it would be sort of `[emergent¦dialogic¦personalized]`. Like delayed political correctness or delayed embraciveness. But, little did I know, that Unicode was not the answer.

✎ *Did they tell you that? That Unicode is not the answer?*

□ Well, Arthur actually wrote back [4], and he knows a lot about Unicode and he said: *With Unicode you have to prove that it's in use already.* In my sense, Unicode was a playground where I could just map whatever values I wanted to be whatever glyph I wanted. Somewhere, in some corner of unused namespace or something. But that's not the way it works. But TeX works like this. So I could always just define a macro that would do this. Hans actually wrote a macro [5] that would basically flip a coin at the beginning of your paper. So whenever you wanted to use the gender neutral, you would just use the macro and then it wouldn't be up to you. It's another way of obfuscating, or pushing the responsibility away from you as an author. It's like *ok, well, on this one it was she, the next it was he, or whatever.*

✎ *So in a way gender doesn't matter anymore?*

□ Right. And then I was just like, that's something we should talk about at the meeting. I guess I sent out something about my thesis and Hans or Taco, they know me, they said that it would great for you to do a presentation of this at the meeting. So that's very much how I ended up there.

✎ *You had never met anyone from ConTeXt before?*

---

[3] http://www.ntg.nl/pipermail/ntg-context/2010/051058.html
[4] http://www.ntg.nl/pipermail/ntg-context/2010/051098.html
[5] http://www.ntg.nl/pipermail/ntg-context/2010/051116.html

```
% -------------------  ____ ____ ____ ____ ____ ____ ____
% ------------------- ||C |||o |||n |||T |||e |||X |||t ||
% ------------------- ||__|||__|||__|||__|||__|||__|||__||
% ------------------- |/__\|/__\|/__\|/__\|/__\|/__\|/__\|
% -------------------
```

☐ No. You and Pierre were the only people I knew, that have been using it, besides me, at the time. It was interesting in that way, it was really … I mean I felt a little bit … nervous isn't exactly the word, but I sort of didn't know what exactly my positon was meant to be. Because these guys … it's a users' meeting, right? But the way that tends to work out for Open Source projects is developers talking to developers. So … my presentation was saturated … I think, I didn't realise how quickly time goes in presentations, at the time. So I spent like 20 minutes just going through my attack on media theory in the thesis. And there was a guy, falling asleep on the right side of the room, just head back. So, that was entertaining. To be the black sheep. That's always a fun position. It was entertaining for me, to meet these people and to be at the same time sort of an outsider. Not a really well known user contrasted with other people, who are more like cornerstones of the community. They were meeting everybody in person for the first time. And somehow I could connect. So now, a month and a half later we're starting this ConTeXt group, an international ConTeXt users' group and I'm on the board, I'm editing the journal. So it's like, it …

✎ *… that went fast!*

☐ It went fast indeed!

✎ *What is this 'ConTeXt User Group'?*

☐ To a certain extent the NTG, which is the Netherlands TeX Group, had sort of been consumed from the inside by the heavyness of ConTeXt, specifically in the Netherlands. The discussion started to shift to be more ConTeXt. Now the journal, the MAPS journal, there are maybe 8 or 10 articles, two of which are not written by either Hans or Taco, who are the main developers of ConTeXt. And there is zero on anything besides ConTeXt. So the NTG is almost presented as *ok, if you like ConTeXt or if you wanna be in a ConTeXt user group, you join the NTG.* Apparently the journal used to be quite thick and there are lots of LaTeX users, who are involved. So partially the attempt is sort of ease that situation a little bit.

✎ *It allowed the two communities to separate?*

```
  % __________________  _____ _____ _____ _____ _____ _____ _____
□ John Haltiwanger      ||c |||o |||n |||T |||e |||x |||t ||
╲ Femke Snelting        ||__|||__|||__|||__|||__|||__|||__||
  % __________________  |/__\|/__\|/__\|/__\|/__\|/__\|/__\|
```

□ Yeah, and not in any way like fast or abrupt fashion. We're trying to be very conscious about it. I mean, it's not ConTeXt's fault that LaTeX users are not submitting any articles for the journal. That user group will always have the capacity, those people could step up. The idea is to setup a more international forum, something that has more of the sense of support for … because the software is getting bigger and right now we're really reliant on this mailing list and if you have your stupid question either Hans, Taco or Wolfgang will shoot something back. And they become reliant on Wolfgang to be able to answer questions, because there are more users coming. Arthur was really concerned, among other people, with the scalability of our approach right now. And how to set up this infrastructure to support the software as it grows bigger. I should forward you this e-mail that I wrote, that is a response to their name choices. They were contemplating becoming a group called 'cows'. Which is clearly an inside joke because they loved to do figure demonstrations with cows. And seeing ConTeXt as I do, as a platform, a serious platform, for the future, something that … it's almost like it hasn't gotten to its … I mean it's in such rapid development … it's so undocumented … it's so … like … it's like rushing water or something. But at some point … it's gonna fill up the location. Maybe we're still building this platform, but when it's solid and all the pieces are … everything is being converted to metric, no more inches and miles and stuff. At that point, when we have this platform, it will turn into a loadable Lua library. It won't even be an executable at that point.

╲ *It is interesting how quickly you have become part of this community. From being complete outsider not knowing where to go, to now speaking about a communal future.*

□ To begin with, I guess I have to confront my own seemingly boundless propensity for picking obscure projects … as sort of my … like the things that I champion. And … it often boils down to flexibility.

╲ *You think that obscurity has anything to do with the future compatibility of ConTeXt?*

```
% ------------------- ____ ____ ____ ____ ____ ____ ____
% -------------------||C |||o |||n |||T |||e |||X |||t ||
% -------------------||__|||__|||__|||__|||__|||__|||__||
% -------------------
% -------------------|/__\|/__\|/__\|/__\|/__\|/__\|/__\|
```

☐ Well, no. I think the obscurity is something that I don't see this actually lasting for too long in the situation of ConTeXt. As it gets more stable it's basically destined to become more of a standard platform. But this is all tied into to stuff that I'm planning to do with the software. If my generative typesetting platform … you know … works and is actually feasible, which is maybe a 80% job.

*Wait a second. You are busy developing another platform in parallel?*

☐ Yes, although I'm kind of hovering over it or sort of superceeding it as an interface. You have LaTeX, which has been at version 2e since the mid-nineties, LaTeX 3 is sort of this dim point on the horizon. Whereas ConTeXt is changing every week. It's converting the entire structure of this macro package from being written in TeX to being written in Lua. And so there is this transition from what could be best described as an archaic approach to programming, to this shiny new piece of software. I see it as being competitive strictly because it has so much configurability. But that's sort of … and that's the double edged sword of it, that the configuration is useless without the documentation. Donald Knuth is famous for saying that he realises he would have to write the software and the manual for the software himself. And I remember in our first conversation about the sort of paternalistic culture these typographic projects seem to have. Or at least in the sense of TeX, they seem to sort of coagulate around a central wizard kind of guy.

*You think ConTeXt has potential for the future, while TeX and LaTeX belong … to the past?*

☐ I guess that's sort of the way it sounds, doesn't it?

*I guess I share some of your excitement, but also have doubts about how far the project actually is away from the past. Maybe you can describe how you think it will develop, what will be that future? How you see that?*

☐ Right. That's a good way to start untangling all the stuff I was just talking about, when I was sort of putting the cart before the horse. I see it developing in some ways … the way that it's used today and the way that current, heavy users use it. I think that they will continue to use in it in a similar way. But you already have people who are utilising LuaTeX … and maybe this is an important thing to distinguish between ConTeXt and LuaTeX. Right now they're sort of very tied together. Their development is intrinsic, they drive each other. But to some extent some of the more interesting stuff that is been being done with these tools is … like … XML processing. Where you throw XML into Lua code and run LuaTeX kerning operations and line breaking and all this kind of stuff. Things that, to a certain extent, you needed to engage TeX on its own terms in the past. That's why macro packages develop as some sort of sustainable way to handle your workflow. This introduction of LuaTeX I think is sort of … You can imagine it being loaded as a library just as a way to typeset the documentation for code. It could be like this holy grail of literate programming. Not saying this is the answer, but that at least it will come out as a nice looking .pdf.

⬊ *LuaTeX allows the connection to TeX to widen?*

□ Yeah. It takes sort of the essence of TeX. And this is, I guess, the crucial thing about LuaTeX that up until now TeX is both a typesetting engine and a programming language. And not a very good one. So now that TeX can be the engine, the Tschicholdian algorithms, the modernist principles, that, for whatever reason, do look really good, can be utilised and connected to without having to deal with this 32 year old macro programming language. On top of that and part of how directly engaging with that kind of movement foreward is … not that I am switching over to LuaTeX entirely at this point … but that this generative typesetting platform that was sort of the foundation of this journal proposal we did. Where you could imagine actual humanity scholars using something that is akin to markdown or a wiki formatting kind of system. And I have a nice little buzzword for that: 'visually semantic markup'. XML, HTML, TeX, … none of those are visually semantic. Because it's all based around these primitives 'ok, between the angle brackets'. Everything is between angle brackets. You have to look what's inside the angle brackets to know what is happening to what's between the angle brackets. Whereas a visually semantic markup … OK headers! OK so it's between two hashmarks or it's between two whatever … The whole design of those preformatting languages, maybe not wiki markup, but at least markdown was that it could be printed as a plaintext document and you could still get a sense of the structure. I think that's a really crucial development. So … in a web browser, on one half of the browser you have you text input, on the other half you have an real-time rendering of it into HTML. In the meantime, the way that the interface works, the way that the visually semantic markup works, is that it is a mutable interface. It could be tailored to your sense of what it should look like. It can be tailored specifically to different workflows. And because there is such a diversity within typographic workflows, typesetting workflows … that is akin to the separation of form and content in HTML and CSS, but it's not meant to be … as problematic as that. I'm not sure if that is a real goal, or if that goal is feasible or not. But it's not meant to be drawing an artificial line, it's just meant to make things easier.

⬊ *So by pulling apart historically grown elements, it becomes … possibly modern?*

☐ Hypermodern?

⬊ *Something for now and later.*

☐ Yes. Part of this idea, the trick … This software is called 'Subtext' and at this point it's a conceptual project, but that will change pretty soon. Its trick is this idea of separation instead of form and content, it's translation and effect. The parser itself has to be mutable, has to be able to pull in the interface, print like decorations basically from a YAML configuration file or some sort of equivalent. One of this configuration mechanisms that was designed to be human readable and not machine readable. Like, well both, striking that balance. Maybe we can get to that kind of … talking about agency a little bit. Its trick to really pull that out so that if you want to … for instance now in markdown if you have quotes it will be translated in ConTeXt into `\quotation`. In ConTeXt that's a very simple switch to turn it into German quotes. Or I guess that's more like international quotes, everything not English. For the purposes of markdown there is no, like really easy way, to change that part of the interface. So that when

I'm writing, when I use the angle brackets as a quote it would turn into a `\quotation` in the output. Whereas with 'Subtext' you would just go into the interface type like configuration and say: These are converted into a quote basically. And then the effects are listed in other configuration files so that the effects of quotes in HTML can be …

✎ *… different.*

☐ Yes. Maybe have specific CSS properties for spacing, that kind of stuff. And then in ConTeXt the same sort of… both the environmental setup as well as the raw 'what is put into the document when it's translated'. This kind of separation … you know at that point if both those effects are already the way that you want them, then all you have to do is change the interface. And then later on typesetting system, maybe iTeX comes out, you know, Knuth's joke, anyway.[6] That kind of separation seems to imply a future proofing that I find very elegant. That you can just add later on the effects that you need for a different system. Or a different version of a system, not that you have to learn 'mark 6', or something like that…

✎ *Back to the future… I wonder about ConTeXt being bound to a particular practise located with two specific people. Those two are actually the ones that produce the most complete use cases and thereby define the kind of practise that ConTeXt allows. Do you think this is a temporary stage or do you think that by inviting someone like you on the board, as an outsider, that it is a sign of things going to change?*

☐ Right. Well, yeah, this is another one of those put-up or shut-up kind of things because for instance at the NTG meeting on Wednesday my presentation was very much a user presentation in a room of developers. Because I basically was saying: Look like this is gonna be a presentation – most presentation are about what you know – and this presentation is really about what I don't know… but what I **do** know is that there is a lot of room for teaching ConTeXt in a more practical fashion, you could say. So my idea is to basically write this documentation on how to typeset poetry, which gets

[6] http://en.wikipedia.org/wiki/Donald_Knuth#Humor

into a lot of interesting questions, just a lot of interesting things. Like you gonna need to write your own macros just at the start … to make sure you have not to go in and change every width value at some point. you know, this kind of thing like … really baby steps. How to make a cover page. These kinds of things are not documented.

*Documentation is let's say an* interesting *challenge for ConTeXt. How do you think the ConTeXt community could enable different kinds of use, beyond the ones that are envisioned right now? I guess you have a plan?*

Yeah … that's a good question. Part of it is just to do stuff, like to get you more involved in the ConTeXt group for instance, because I was talking to Arthur and he hadn't even read the article from V/J10 [7]. I think that kind of stuff is really important. It's like the whole Blender Foundation kind of impulse. We have some developers who are paid to do this and that's kind of rare already in an Open Source/Free Software project. But then to kind of have users pushing the boundaries and hitting limits. It's rare that Hans will encounter some kind of use case that he didn't think of and react in a negative way. Or react in a way like *I'm not gonna even entertain that possibility*. Part of it is moving beyond this … even the sort of centralisation as you call it … how to do that directly … I see it more as baby steps for me personally at this point. Just getting a tutorial on how to typeset a cd booklet. Just basically what I'm writing. That at the same time, you know, gets you familiar with ConTeXt and TeX in general. Before my presentation I was wondering, I was like: how do you set a variable in TeX. Well, it's a macro programming language so you just make a macro that returns a value. Like that kind of stuff is not initially obvious if you're used to a different paradigm or you know .. So these baby steps of kind of opening the field up a little bit and then using it my own practise of guerilla typesetting and kind of putting it out there. and you know … And people gonna start being like: *oh yeah, beautiful documents are possible* or at least *better looking documents are possible*. And then once we have them at that, like, then how do you we

[7] Constant, Clementine Delahaut, Laurence Rassel, and Emma Sidgwick. *Verbindingen/Jonctions: Tracks in electr(on)ic fields*. Constant Verlag, 2009. http://ospublish.constantvzw.org/sources/vj10

take it to the next level. How do I turn a lyric sheet from something that is sort of static to … you know … two pages that are like put directly on the screen next to each other. Like a screen based system where it's animated to the point … and this is what we actually started to karaoke last night … so you have an English version and a Spanish version – for instance in the case of the music that I've been doing. And we can animate. We can have timed transitions so you can have a 'current lyric indicator' move down the page. That kind of use case is not something that Pragma[8] is ever going to run into. But as soon as it is done and documented then what's the next thing, what kind of animations are gonna be … or what kind of … once that possibility is made real or concrete … you know, so I kind of see it as a very iterative process at this point. I don't have any kind of grand scheme other than 'Subtext' kind of replacing Microsoft Word as the dominant academic publishing platform, I think. (*laughs*)

⟍ *Just take over the world.*

□ That's one way to do it, I think.

⟍ *You talked about manuals for things that you would maybe not do in another kind of software …*

□ Right.

⟍ *Manuals that not just explain 'this is how you do it' but also 'this is the kind of user you could be'.*

□ Right.

⟍ *I'm not sure if instructions for how to produce a cd cover would draw me in, but if it helped me understand how to set a variable, it would.*

□ Right.

[8] Hans Hagen's company for Advanced Document Engineering

```
% -------------------  ____ ____ ____ ____ ____ ____ ____
% ------------------- ||C |||o |||n |||T |||e |||X |||t ||
% ------------------- ||__|||__|||__|||__|||__|||__|||__||
% -------------------
% ------------------- |/__\|/__\|/__\|/__\|/__\|/__\|/__\|
```

*You want the complete manual of course?*

Yeah!

*You were saying that ConTeXt should replace Microsoft Word as the standard typesetting tool for academic publishing. You are thinking about the future for ConTeXt more in the context of academic publishing than in traditional design practise?*

Yes. In terms of 'Subtext', I mean the origins of that project, very much … It's an interesting mix because it's really a hybridity of many different processes. Some, much come directly from this obscure art project 'the abstraction'. So I have stuff like the track changes using Git version control and everything being placed on plaintext as a necessity. That's a holdover from that project as well as the idea of gradiated presence. Like software enabling a more real-time peer review, anonymous peer review system. And even a collaborative platform where you don't know who you're writing with, until the article comes out. Someting like out that. So these interesting tweaks that you can kind of make, those all are holdovers from this very, very much maybe not traditional design practise but certainly like … twisted artistic project that was based around hacking a hole from signified to siginifier and back again. So … In terms of its current envisionment and the use case for which we were developing it at the beginning, or I'm developing it, whatever … I'll say it the royal way, is an academic thing. But I think that … doesn't have to stop there and …

*At some point at OSP we decided to try ConTeXt because we were stuck with Scribus for page layout as the only option in Free Software. We wanted escape that kind of stiffness of the page, or of the canvas in a way. But ConTeXt was not the dream solution either. For us it had a lot to do, of course, with issues of documentation … of not understanding, not coming from that kind of automatism of treating it as another programming language. So I think we could have had much more fun if we had understood the culture of the project better. I think the most frustrating experience was to find out how much the model of typesetting is linked to the Tschichold universe, that at the moment you try to* *break out, the system completely looses all flexibility. And it is almost as if you can hear it freeze. So if we blame half of our troubles with ConTeXt on our inability to actually understand what we could do with ConTeXt, I think there is a lot also in its assumption what a legible text would look like, how it's structured, how it's done. Do you think a modern version of ConTeXt will keep that kind of inflexibility? How can it become more flexible in it's understanding of what a page or a book could be?*

☐ That's an interesting question, because I'm not into the development side of LuaTex at all, but I would be surprised if the way that it was being implemented was not significantly more modular than for instance when it was written in Pascal, you know, how that was. Yeah, that's a really interesting question of how swappable is the backend. How much can we go in and kind of… you know. And it its an inspirational question to me, because now I'm trying to envision a different page. And I'm really curious about that. But I think that ConTeXt itself will likely be pretty stable in its scope … in that way of being … sort of … deterministic in its expectations. But where that leaves us as users … first I'd be really surprised if the engine itself, if LuaTeX was not being some way written to … I feel really ignorant about this, I wish I just knew. But, yeah, there must be … There is no way to translate this into a modern programming language without somehow thinking about this in terms of the design. I guess to certain extent the answer to your question is dependent on the conscientiousness of Taco and the other LuaTex developers for this kind of modularity. But I don't … you know … I'm actually feeling very imaginatively lacking in terms of trying to understand what you're award-winning book did not accomplish for you … Yeah, what's wrong with that?

↘ *I think it would be good to talk with Pierre, not Pierre Marchand but Pierre …*

☐ … Huggybear.

↘ *Yeah. We have been talking about 'rivers' as a metaphor for layout … like were you could have things that are … let's say fluid and other things that could be placed and force things around it. Layout is often a combination of those two*

```
% -------------------  ____  ____  ____  ____  ____  ____  ____
% -------------------||C |||o |||n |||T |||e |||X |||t ||
% -------------------||__|||__|||__|||__|||__|||__|||__||
% -------------------
% -------------------|/__\|/__\|/__\|/__\|/__\|/__\|/__\|
```

*things. And this is what is frustrating in canvas based layout that it is all fixed and you have to make it look like it's fluid. And here it's all fluid and sometimes you want it to be fixed. And at the moment you fix something everything breaks. Then it's up to you. You're on your own.*

□ Right.

◥ *The experience of working with ConTeXt is that it is very much elastic, but there is very little imagination about what this elasticity could bring.*

□ Right.

◥ *It's all about creating universally beautiful pages, in a way it is using flexibility to arrive at something that is already fixed.*

□ Right.

◥ *Well, there is a lot more possible than we ever tried, but … again … this goes back to the sort of centralist question: If those possibilities are mainly details in the head of the main developers than how will I ever start to fantasize about the book I would want to make with it?*

□ Right.

◥ *I don't even need access to all the details. Because once I have a sort of sense of what I want to do, I can figure it out. Right now you're sort of in the dark about the endless possibilities …*

□ Its existence is very opaque in some ways. The way that it's implemented, like everything about it is sort of … looking at the macros that they wrote, the macros that you invoke … like … that takes … flow control in TeX is like … I mean you might as well write it in Bash or … I mean I think Bash would even be more sensible to figuring out what's going on. So, the switch to Lua there is kind of I think a useful step just in being more transparent. To allow you to get into becoming more intimate with the source or the operation of the system … you know … without having to go … I mean I guess … the TeX Book would still be useful in some ways but that's … I mean … to go back and learn TeX when you're just trying to use ConTeXt is sort of … it's not … I'm not saying it's, you know … it's a proper assumption to say *oh yeah, don't worry about the rules and the way TeX is organised* but you're not writing your documents in ConTeXt the way you would write them if you're using plain TeX. I mean that's just … it's just not … It's a different workflow … it has a completely different set of processes that you need to arrange. So it has a very distinct organisational logic … that I think that … yeah … like being able to go into the source and be like *oh OK*, like I can see clearly this is … you know. And then you can write in your own way, you can write back in Lua.

✎ *This kind of documentation would be the killer feature of ConTeXt …*

☐ Yeah.

✎ *It's kind of strange paradox in the TeX community. At one hand you're sort of supposed to be able to do all of it. But at the same time on every page you're told not to do it, because it's not for you to worry about this.*

☐ Right. That's why the macro packages exist.

✎ *With ConTeXt there is this strange sense of very much wanting to understand the way the logic works, or … what the material is, you're dealing with. And at the same time being completely lost in the labyrinth between the old stuff from TeX and LaTeX, the newer stuff from LuaTex, Mark 4, 3, 5, 6 …*

☐ So that was sort of my idea with the cd typesetting project, is not to say, that that is something that is immediately interesting to anybody who is not trying to do that specifically, right? But at the same time if I'm … if it's broken down into 'How to do a bitmap cover page' (=**Lesson 1**).
**Lesson 2**: 'How to start defining you own macros'. And so you know, it's this thing that could be at one point a very … because the documentation as it stands right now is … I think it's almost … fixing that documentation, I'm

not sure is even possible. I think that it has to be completely approached differently. I mean, like a real ConTeXt manual, that documents … you know … command by command exactly what those things do. I mean our reference manual now just shows you what arguments are available, but doesn't even list the available arguments. It's just like: *These are the positions of the arguments*. And it's interesting.

↘ *So expecting writers of the program to write the manual fails?*

☐ Right.

↘ *What is the difference between your plans for 'Subtext' and a page layout program like Scribus?*

☐ You mentioned 'Subtext' coming from a more academic publishing rather than a design background. I think that this belies where I have come into typesetting and my understanding of typography. Because in reality DTP has never kind of drawn me in in that way. The principle differences are really based on this distribution of agency, in my mind. That when you're demanding the software to be 'what you see is what you get' or when you place that metaphor between you and your process. Or you and your engagement, you're gaining the usefulness of that metaphor, which is … it's almost … I hope I don't sound offensive … but it's almost like child's play. It's almost like point, click, place. To me it just seems so redundant or … time-consuming maybe … to really deal with it that way. There are advantages to that metaphor. For instance I don't plan on designing covers in ConTeXt. Or even a poster or something like that. Because it doesn't really give affordances for that kind of creativity. I mean you can do generative stuff with the MetaFun package. You can sort of play around with that. But I haven't seen a ConTeXt generated cover that I liked, to be honest.

↘ *OK.*

☐ OK. Principle differences. I'm trying to … I'm struggling a little bit. I think that's partially because I'm not super comfortable with the layout mechanism and stuff yet. And you have things like `\blank` in order to move down the page. Because it has this sort of literal sense of a page and movement on a page. Obviously Scribus has a literal idea of a page as well, but because it's WYSIWYG it has that benefit where you don't have to think *OK, well, maybe it should be 1.6 ems down* or *maybe it should be 1.2 ems down*. You move it until it looks right. And then you can measure it and you're like *ok, I'm gonna use this measurement for the further on in my document*. So it's that whole top-down vs. bottom-up approach. It really breaks down into the core organisational logics of those softwares.

✎ *I think it's too easy to make the difference based on the fact that there is a metaphorical layer or not. I think there is a metaphorical layer in ConTeXt too …*

☐ Right. Yeah for sure.

✎ *And they come at a different moment and they speak a different language. But I think that we can agree that they're both there. So I don't think it's about the one being without and the other being with. Of course there is another sense of placing something in a canvas-based software than in a … how would you call this?*

☐ So I guess it is either 'declarative' or 'sequence' based. You could say generative in a way … or compiled or … I don't even know. That's a cool question.

✎ *What is the difference really and why would you choose the one or the other? Or what would you gain from one to the other? Because it's clear that posters are not easily made in ConTeXt. And that it's much easier to typeset a book in ConTeXt than it is in Scribus, for example.*

☐ Declarative maybe …

✎ *So, there's hierarchy. There's direction. There's an assumption about structure being good or bad.*

```
% --------------------  ____ ____ ____ ____ ____ ____ ____
% --------------------||C |||o |||n |||T |||e |||X |||t ||
% --------------------||__|||__|||__|||__|||__|||__|||__||
% --------------------
% --------------------|/__\|/__\|/__\|/__\|/__\|/__\|/__\|
```

☐ Yeah. Boxes, Glue.[9]

✎ *What is exciting in something like this is that placement is relative always. Relative to a page, relative to a chapter, relative to itself, relative to what's next to it. Where in a canvas based software your page is fixed.*

☐ Right.

✎ *This is very different from a system where you make a change, then you compile and then you look at it and then you go back into your code. So where there is a larger distinction between output and action. It's almost gestural…*

☐ It's like two different ways of having a conversation. Larry Wall has this really great metaphor. He talks about 'ballistic design'. So when you're doing code, maybe he's talking more about software design at this point, basically it's a 'ballistic practise' to write code. Ballistics comes from artillery. So you shoot at a thing. If you hit it, you hit it. If you miss it, you change the amount of gun powder, the angle. So code is very much a 'ballistic practise'. I think that filters into this difference in how the conversation works. And this goes back to the agencies where you have to wait for the computer to figure out. To come with its into the conversation. You're putting the code in and then the computer is like `ok; this is what the code means` and then `is this what you wanted?` Whereas with the WYSIWYG kind of interface the agency is distributed in a different way. The computer is just like `ok, I'm a canvas; I'm just here to hold what you're putting on` and `I'm not going to change it any way or affect it in any way that you don't tell me to.` I mean it's the same way but I… is it just a matter of the compilation time? In one you're sort of running a experiment, in another you're just sort of painting. If that's a real enough distinction or if that's… you know… it's sort of… I mean I kind of see that it is like this. There is ballistics vs. maybe fencing or something.

---

[9] *Boxes, which are things can be drawn on a page, and glue, which is invisible stretchy stuff that sticks boxes together.* Mark C. Chu-Carroll. The Genius of Donald Knuth: Typesetting with Boxes and Glue, 2008

```
% ___________________   _____ _____ _____ _____ _____ _____ _____
☐ John Haltiwanger      ||c |||o |||n |||T |||e |||x |||t ||
╲ Femke Snelting        ||___|||___|||___|||___|||___|||___|||___||
% ___________________   |/___\|/___\|/___\|/___\|/___\|/___\|/___\|
```

╲ *Fencing?*

☐ Fencing. Like more of a …

╲ *Or wrestling?*

☐ Or wrestling.

╲ *When you said* just sort of painting *I felt offended. (*laughs*)*

☐ I'm sorry. I didn't mean it like that.

╲ *Maybe back to wrestling vs. ballistics. Where am I and where is the machine?*

☐ Right.

╲ *I understand that there's lots of childish way of solving this need to make the computer dissapear. Because if you are not wrestling … you're dancing, you know.*

☐ Yeah.

╲ *But I think it's interesting to see that ballistics, that the military term of shooting at something, is the kind of metaphor to be used. Which is quite different than a creative process where there is a direct feedback between something placed and the responses you have.*

☐ Right.

╲ *And it's not always about aiming, but also sometimes about trying and about kind of subtle movements that spark off something else. Which is very immediate. And needs an immediate connection to … let's say … what you do and what you get. It would be interesting to think about ways to talking about 'what you see is what you get' away from this assumption that is always about those poor users that are not able do it in code.*

☐ Right.

```
% -------------------  ____ ____ ____ ____ ____ ____ ____
% ------------------- ||C |||o |||n |||T |||e |||X |||t ||
% ------------------- ||__|||__|||__|||__|||__|||__|||__||
% -------------------
% ------------------- |/__\|/__\|/__\|/__\|/__\|/__\|/__\|
```

*Because I think there is essential stuff that you can* ***not*** *do in a tool like this – that you* ***can*** *do in canvas-based tools. And so … I think it's really a pity when … yeah … It's often overlooked and very strange to see. There is not a lot of good thinking about that kind of interaction. Like literal interaction. Which is also about agency with the painter. With the one that makes the movement. Where here the agency is very much in this confrontational relation between me aiming and …*

So yeah, when we put it in those metaphors. I'm on the side with the painting, because …

*But I mean it's difficult to do a book while wrestling. And I think that's why a poster is very difficult to do in this sort of aiming sense. I mean it's fun to do but it's a strange kind of posters you get.*

You can't fit it all in your head at once. It's not possible.

*No. So it's okay to have a bit of delay.*

I wondered to what extent, if it were updated in real time, all the changes you're making in the code, if compilation was instantaneous, how that would affect the experience. I guess it would still have this ballistic aspect, because what you are doing is … and that's really the side of the metaphor … or a metaphorical difference between the two. One is like a translation. The metaphor of *ok this code means this effect* … That's very different from picking a brush and choosing the width of the stroke. It's like when you initialise a brush in code, set the brush width and then move it in a circle with a radius of x. It's different than taking the brush in Scribus or in whatever WYSIWYG tool you are gonna use. There is something intrinsically different about a translation from primitives to visual effect than this kind of metaphorical translation of an interaction between a human and a canvas … kind of put into software terms.

*But there is a translation from me, the human, to the machine, to my human eye again, which is hard to grasp. Without wanting it to be made invisible somehow.*

*Or to assume that it is not there. This would be my dream tool that would allow you to sense that kind of translation without losing the … canvasness of the canvas. Because it's frustrating that the canvas has to not speak of itself to be able to work. That's a very sad future for the canvas, I think.*

☐ I agree.

↘ *But when it speaks of itself it's usually seen as buggy or it doesn't work. So that's also not fair to the canvas. But there is something in drawing digitally, which is such a weird thing to do actually, and this is interesting in this sort of cyborgs we're becoming, which is all about forgetting about the machine and not feeling what you do. And it's completely a different world in a way than the ballistics of ConTeXt, LaTeX or whatever typesetting platform.*

☐ Yeah, that's true. And it's something that my students were forced to confront and it was really interesting because that supposed invisibility or almost necessitated invisibility of the software. As soon as they're in Inkscape instead of Illustrator they go crazy. Because it's like they know what they want to do, but it's a different mechanism. It's the same underlying process which itself is only just meant to give you a digital version of what you could easily do on a piece of paper. Provided you have the right paints and stuff. So perhaps it's like the difference between moving from a brush to an air brush. It's a different … interface. It's a different engagement. There is a different thing between the human and the canvas. You engage in this creative process where it's like *ok, we'll now have an airbrush* and I can play around to see what the capacities are without being stuck in *well I can't get it to do my fine lines the same way I can when I have my brush.* It's like when you switch the software out from between the person and the canvas. It's that sort of invisibility of the interface and it's intense for people. They actually react quite negatively. They're not gonna bother to learn this other software because in the end they're doing less. The reappearance of this software … of software between them and their ideas is kinda too much. Whereas people who don't have any preconceived notions are following the tutorials and they're learning and they're like *ok, I'm gonna continue to play with this.* Because this software is starting to become more invisible.

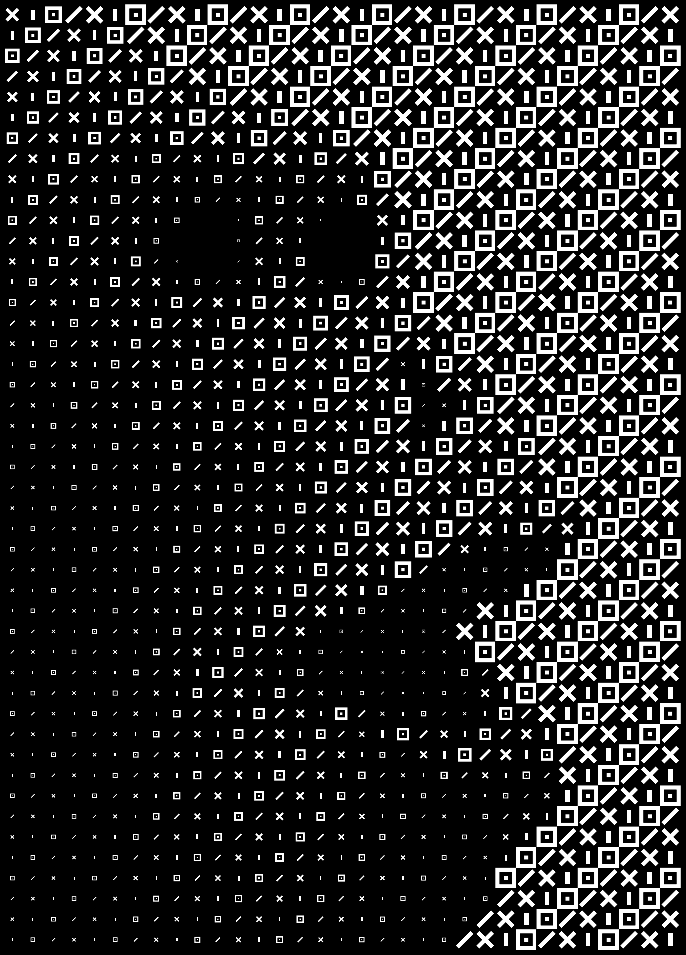
```
% -------------------  ____ ____ ____ ____ ____ ____ ____
% ------------------- ||C |||o |||n |||T |||e |||X |||t ||
% ------------------- ||__|||__|||__|||__|||__|||__|||__||
% ------------------- |/__\|/__\|/__\|/__\|/__\|/__\|/__\|
% -------------------
```

✎ *But on a sort of theoretical level the necessitated invisibility, as you said it nicely, is something I would always speak against. Because that means you hide something that's there. Which seems a stupid thing to do, especially when you want to find a kind of more flexible relation to your tools. I want to find a better word for describing that sort of quick feedback. Because if it's too much in the way, then the process stops. The drawing can not be made if I'm worried too much about the point of my pencil that might break … or the … I dont't know … the nozzle being blocked.*

☐ Dismissing the other tools is … I was kinda joking, but … there is something sort of blocklike: Point. Move. This. But at the same time, like I said, I wouldn't do a cover in ConTeXt. Just like I probably wouldn't try to do something like a recreation of a Pre-Raphaelite painting in Processing or something like that. There is just points where our metaphors break down. And so … It sounded sort of, ***ok, bottom-up über alles*** like **always**.

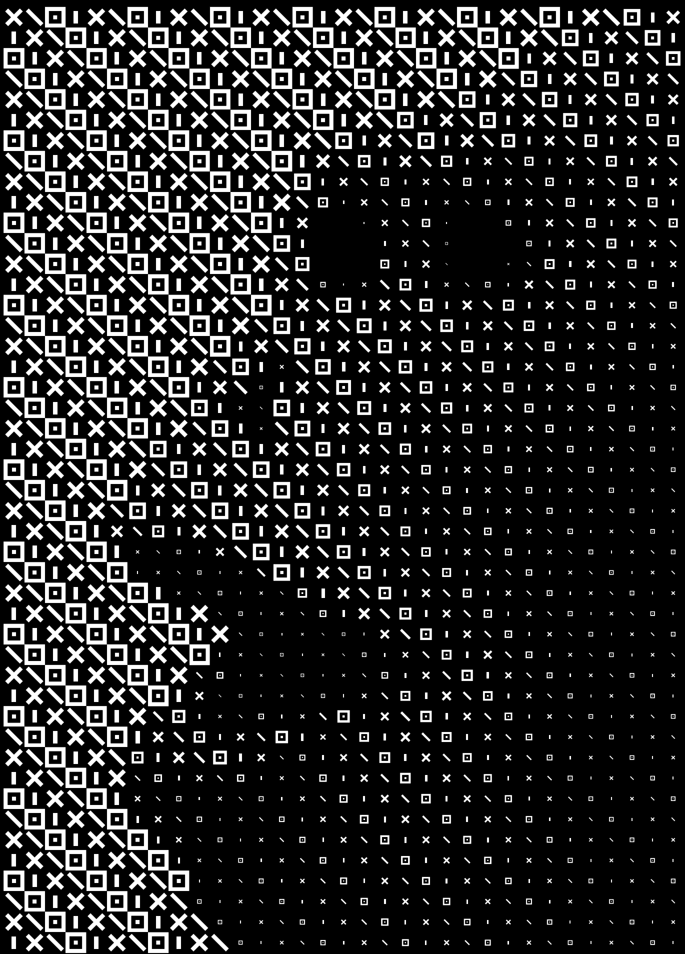
✎ *Ok, there's still painters and there's still people doing Pre-Raphaelite paintings with Pre-Raphaelite tools, but most of us are using computers. So there should be more clever ways of thinking about this.*

☐ Yeah. To borrow a quote from my old buddy Donald Rumsfeld: *There are the known knowns, the known unknowns and the unknown unknowns.* That actually popped into my head earlier because when we were talking about the potentials of the software and the way that we interact and stuff, it's like we know that we don't know … other ways of organizing. We know that there are, like there **has** to be, another way, whether it is a middle path between these two or some sort of … Maybe it's just tenth dimensional, maybe it's fourth dimensional, maybe it's completely hypermodern or something. Anyway. But the unknown unknowns … It's like the stuff that we can't even tell we don't know about. The questions that we don't know about that would come up once we figure out these other ways of organising it. That's when I start to get really interested in this sort of thing. How do you even conceive of a practise that you don't know? And once you get there, there's going to be other things that you know you don't know and have to keep finding them. And then there's gonna be things that you don't know you don't know and they just appear from nowhere and … it's fun.



# Meaningful transformations

/ Tom Lechner
⊡ Pierre Marchand
× Ludivine Loiseau
ı Femke Snelting



# Meaningful transformations

╱ Tom Lechner
▣ Pierre Marchand
✕ Ludivine Loiseau
ı Femke Snelting

We discovered the work of Tom Lechner for the first time at the Libre Graphics Meeting 2010 in Brussels. Tom traveled from Portland to present Laidout, an amazing tool that he made to produce his own comic books and also to work on three dimensional mathematical objects. We were excited about how his software represents the gesture of folding, loved his bold interface decisions plus were impressed by the fact that Tom decided to write his own programming framework for it. A year later, we met again in Montreal, Canada for the **Libre Graphics Meeting 2011** where he presents a follow-up. With Ludivine Loiseau [1] and Pierre Marchand [2], we finally found time to sit down and talk.

▮ ***What is Laidout?***

✏ Well, Laidout is software that I wrote to lay out my cartoon books in an easy fashion. Nothing else fit my needs at the time, so I just wrote it.

▮ ***It does a lot more than laying out cartoons?***

✏ It works for any image, basically, and gradients. It does not currently do text. It is on my todo list. I usually write my own text, so it does not really need to do text. I just make an image of it.

▮ ***It can lay out T-shirts?***

✏ But that's all images too. I guess it's two forms of laying out. It's laying out pieces of paper that remain whole in themselves, or you can take an image and lay it out on smaller pieces of paper. Tiling, I guess you could call it.

▮ ***Can you talk us through the process of doing the T-shirt?***

---

[1] amateur bookbinder and graphic designer
[2] artist/developer, contributing amongst others to PodofoImpose and Scribus

OK. So, you need a pattern. I had just a shirt that sort of fit and I approximated it on a big piece of paper, to figure out what the pieces were shaped like, and took a photograph of that. I used a perspective tool to remove the distortion. I had placed rulers on the ground so that I could remember the actual scale of it. Then once it was in the computer, I traced over it in Inkscape, to get just the basic outline so that I could manipulate further. Blender didn't want to import it so I had to retrace it. I had to use Blender to do it because that lets me shape the pattern, take it from flat into something that actually makes 3D shapes so whatever errors were in the original pattern that I had on the paper, I could now correct, make the sides actually meet and once I had the molded shape, and in Blender you have to be extremely careful to keep any shape, any manipulation that you do to make sure your surface is still unfoldable into something flat. It is very easy to get away from flat surfaces in Blender. Once I have the molded shape, I can export that into an .off file which my unwrapper can import and that I can then unwrap into the sleeves and the front and the back as well as project a panoramic image onto those pieces. Once I have that, it becomes a pattern laid out on a giant flat surface. Then I can use Laidout once again to tile pages across that. I can export into a .pdf with all the individual pieces of the image that were just pieces of the larger image that I can print on transfer paper. It took forty iron-on transfer papers I ironed with an iron provided to me by the people sitting in front of me so that took a while but finally I got it all done, cut it all out, sewed it up and there you go.

***Could you say something about your interest in moving from 2D to 3D and back again? It seems everything you do is related to that?***

I don't know. I've been making sculpture of various kinds for quite a long time. I've always drawn. Since I was about eighteen, I started making sculptures, mainly mathematical woodwork. I don't quite have access to a full woodwork workshop anymore, so I cannot make as much woodwork as I used to. It's kind of an instance of being defined by what tools you have available to you, like you were saying in your talk. I don't have a woodshop, but I can do other stuff. I can still make various shapes, but mainly out of paper. Since I had been doing woodwork, I picked up photography I guess and I made a ton of panoramic images. It's kind of fun to figure out how to project these images out of the computer into something that you can physically create, for instance a T-shirt or a ball, or other paper shapes.

▮ ***Is there ever any work that stays in the computer, or does it always need to become physical?***

⁄ Usually, for me, it is important to make something that I can actually physically interact with. The computer I usually find quite limiting. You can do amazing things with computers, you can pan around an image, that in itself is pretty amazing but in the end I get more out of interacting with things physically than just in the computer.

▮ ***But with Laidout, you have moved folding into the computer! Do you enjoy that kind of reverse transformation?***

⁄ It is a challenge to do and I enjoy figuring out how to do that. In making computer tools, I always try to make something that I can not do nearly as quickly by hand. It's just much easier to do in a computer. Or in the case of spherical images, it's practically impossible to do it outside the computer. I could paint it with airbrushes and stuff like that but that in itself would take a hundred times longer than just pressing a couple of commands and having the computer do it all automatically.

▣ *My feeling about your work is that the time you spent working on the program is in itself the most intriguing part of your work. There is of course a challenge and I can imagine that when you are doing it like the first time you see a rectangle, and you see it mimic a perspective you think* wow I am folding a paper, I have really done something. *I worked on imposition too but more to figure out how to work with .pdf files and I didn't go this way of the gesture like you did. There is something in your work which is really the way you wrote your own framework for example and did not use any existing frameworks. You didn't use existing GUIs and toolboxes. It would be nice to listen to you about how you worked, how you worked on the programming.*

⁄ I think like a lot of artists, or creative people in general, you have to enjoy the little nuts and bolts of what you're doing in order to produce any final work, that is if you actually do produce any final work. Part of that is making the tools. When I first started making computer tools to help me in my artwork, I did not have a lot of experience programming computers. I had some. I did little projects here and there. So I looked around at the various toolkits, but everything seemed really rigid. If you wanted to edit some text, you had this little box and you write things in this little box and if you want to change numbers, you have to erase it and change tiny things with other tiny things. It's just very restrictive. I figured I could either figure out how to adapt those to my own purposes, or I could just figure out my own, so I figured either way would probably take about that same amount of time I guessed, in my ignorance. In the process, that's not quite been true. But it is much more flexible, in my opinion, what I've developed, compared to a lot of other toolkits. Other people have other goals, so I'm sure they would have a completely different opinion. For what I'm doing, it's much more adaptable.

✕ *You said you had no experience in programming? You studied in art school?*

/ I don't think I ever actually took computer programming classes. I grew up with a Commodore 64, so I was always making letters fly around the screen and stuff like that, and follow various curves. So I was always doing little programming tricks. I guess I grew up in a household where that sort of thing was pretty normal. I had two brothers, and they both became computer programmers. And I'm the youngest, so I could learn from their mistakes, too. I hope.

▣ *You're looking for good excuses to program.*

/ (*laughs*) That could be.

▣ *We can discuss at length about how actual toolkits don't match your needs, but in the end, you want to input certain things. With any recent toolkit, you can do that. It's not that difficult or time consuming. The way you do it, you really enjoy it, by itself. I can see it as a real creative work, to come up with new digital shapes.*

❙ ***Do you think that for you, the program itself is part of the work?***

/ I think it's definitely part of the work. That's kind of the nuts and bolts that you have to enjoy to get somewhere else. But if I look back on it, I spend a huge amount of time just programming and not actually making the artwork itself. It's more just making the tools and all the programming for the tools. I think there's a lot of truth to that. When it comes time to actually make artwork, I do like to have the tool that's just right for the job, that works just the way that seems efficient.

▮ ***I think the program itself is an artwork, very much. To me it is also a reflection on moving between 2D and 3D, about physical computation. Maybe this is the actual work. Would you agree?***

╱ I don't know. To an extent. In my mind, I kind of class it differently. I've certainly been drawing more than I've been doing technical stuff like programming. In my mind, the artwork is things that get produced, or a performance or something like that. And the programming or the tools are in service to those things. That's how I think of it. I can see that … I've distributed Laidout as something in itself. It's not just some secret tool that I've put aside and presented only the artwork. I do enjoy the tools themselves.

▮ ***I have a question about how the 2D imagines 3D. I've seen Pierre and Ludi write imposition plans. I really enjoy reading this, almost as a sort of poetry, about what it would be to be folded, to be bound like a book. Why is it so interesting for you, this tension between the two dimensions?***

╱ I don't know. Perhaps it's just the transformation of materials from something more amorphous into something that's more meaningful, somehow. Like in a book, you start out with wood pulp, and you can lay it out in pages and you have to do something to that in order to instil more meaning to it.

▮ ***Is binding in any way important to you?***

╱ Somewhat. I've bound a few things by hand. Most of my cartoon books ended up being just stapled, like a stack of paper, staple in the middle and fold. Very simple. I've done some where you cut down the middle and lay the sides on top and they're perfect bound. I've done just a couple where it's an actual hand bound, hard cover. I do enjoy that. It's quite a time consuming thing. There's quite a lot of craft in that. I enjoy a lot of hand made, do-it-yourself activities.

▮ ***Do you look at classic imposition plans?***

/ I guess that's kind of my goal. I did look up classic book binding techniques and how people do it and what sort of problems they encounter. I'm not sure if I've encompassed everything in that, certainly. But just the basics of folding and trimming, I've done my best to be able to do the same sort of techniques that have been done in the past, but only manually. The computer can remember things much more easily.

⊡ *Imposition plans are quite fixed, you have this paper size and it works with specific imposition plans. I like the way your tool is very organic, you can play with it. But in the end, something very classic comes out, an imposition plan you can use over and over, which gives a sort of continuity.*

✕ *What's impressive is the attention you put into the visualization. There are some technical programs which do really big imposition stuff, but it's always at the printer. Here, you can see the shape being peeled. It's really impressive. I agree with Femke that the program is an artwork too, because it's not only technical, it's much more.*

▮ ***How is the material imagined in the tool?***

/ So, far not really completely. When you fold, you introduce slight twists and things like that. And that depends on the stiffness of the paper and the thickness of the paper and I've not adequately dealt with that so much. If you just have one fold, it's pretty easy to figure out what the creep is for that. You can do tests and you can actually measure it. That's pretty easy to compensate for. But if you have many more folds than that, it becomes much more difficult.

▮ ***Are you thinking about how to do that?***

/ I am.

▮ ***That would be very interesting. To imagine paper in digital space, to give an idea of what might come out in the end. Then you really have to work your metaphors, I think?***

╱ A long time ago, I did a lot of T-shirt printing. Something that I did not particularly have was a way to visualize your final image on some kind of shirt and the same thing applies for book binding, too. You might have a strange texture. It would be nice to be able to visualize that beforehand, as well as the thickness of the paper that actually controls physical characteristics. These are things I would like to incorporate somehow but haven't gotten around to.

❙ ***You talked about working with physical input, having touchpads … Can you talk a bit more about why you're interested in this?***

╱ You can do a lot of things with just a mouse and a keyboard. But it's still very limiting. You have to be sitting there, and you have to just control those two things. Here's your whole body, with which you can do amazing things, but you're restricted to just moving and clicking and you only have a single point up on the screen that you have to direct very specifically. It just seems very limiting. It's largely an unexplored field, just to accept a wider variety of inputs to control things. A lot of the multitouch stuff that's been done is just gestures for little tiny phones. It's mainly for browsing, not necessarily for actual work. That's something I would like to explore quite a lot more.

❙ ***Do you have any fantasies about how these gestures could work for real?***

╱ There's tons of sci fi movies, like 'Minority Report', where you wear these gloves and you can do various things. Even that is still just mainly browsing. I saw one, it was a research project by this guy at Caltech. He had made this table and he wore polarized glasses so he could look down at this table and see a 3D image. And then he had gloves on, and he could sculpt things right in the air. The computer would keep track of where his hand is going. Instead of sculpting clay, you're sculpting this 3D mesh. That seemed quite impressive to me.

❙ ***You're thinking about 3D printers, actually?***

╱ It's something that's on my mind. I just got something called the Eggbot. You can hold spheres in this thing and it's basically a plotter that can print on spherical surfaces or round surfaces. That's something I'd like to explore some more. I've made various balls with just my photographic panoramas glued onto them. But that could be used to trace an outline for something and then you could go in with pens or paints and add more detail. If you're trying to paint on a sphere, just paint and no photograph, laying out an outline is perhaps the hardest part. If you simplify it, it becomes much easier to make actual images on spheres. That would be fun to explore.

▣ *I'd like to come back to the folding. Following your existing aesthetic, the stiffness and the angles of the drawing are very beautiful. Is it important you, preserving the aesthetic of your programs, the widgets, the lines, the arrows…*

✎ I think the specific widgets, in the end, are not really important to me at all. It's more just producing an actual effect. So if there is some better way, more efficient way, more adaptable way to produce some effect, then it's better to just completely abandon what doesn't work and make something that's new, that actually does work. Especially with multitouch stuff, a lot of old widgets make no more sense. You have to deal with a lot of other kinds of things, so you need different controls.

▣ *It makes sense, but I was thinking about the visual effect. Maybe it's not Laidout if it's done in Qt.*

▮ ***Your visuals and drawings are very aesthetically precise. We're wondering about the aesthetics of the program, if it's something that might change in the future.***

✎ You mean would the quality of the work produced be changed by the tools?

▮ ***That's an interesting question as well. But particularly the interface, it's very related to your drawings. There's a distinct quality. I was wondering how you feel about that, how the interaction with the program relates to the drawings themselves.***

✎ I think it just comes back to being very visually oriented. If you have to enter a lot of values in a bunch of slots in a table, that's not really a visual way to do it. Especially in my artwork, it's totally visual. There's no other component to it. You draw things on the page and it shows up immediately.

It's just very visual. Or if you make a sculpture, you start with this chunk of stuff and you have to transform it in some way and chop off this or sand that. It's still all very visual. When you sit down at a computer, computers are very powerful, but what I want to do is still very visually oriented. The question then becomes: how do you make an interface that retains the visual inputs, but that is restricted to the types of inputs computers need to have to talk to them?

⊡ *The way someone sets up his workshop says a lot about his work. The way you made Laidout and how you set up its screen, it's important to define a spot in the space of the possible.*

✕ *What is nice is that you made the visualisation so important. The windows and the rest of the interface is really simple, the attention is really focused on what's happening. It is not like shiny windows with shadows everywhere, you feel like you are not bothered by the machine.*

⊡ *At the same time, the way you draw the thickness of the line to define the page is a bit large. For me, these are choices, and I am very impressed because I never manage to make choices for my own programs. The programs you wrote, or George Williams, make a strong aesthetic assertion like:* This is good. *I can't do this. I think that is really interesting.*

⁄ Heavy page borders, that still comes down to the visual thing you end up with, is still the piece of paper so it is very important to find out where that page outline actually is. The more obvious it is, the better.

⊡ *Yes, I think it makes sense. For a while now, I paid more attention than others in Scribus to these details like the shape of the button, the thickness of the lines, what pattern do you chose for the selection, etcetera. I had a lot of feedback from users like: I want this, this is too big and at some point you want to please everybody and you don't make choices. I don't think that you are so busy with what others think.*

▮ ***Are there many other users of the program?***

⁄ Not that I know of (*laughter*). I know that there is at least one other person that actually used it to produce a booklet. So I know that it is possible for someone other than myself to make things with it. I've gotten a couple of patches from people to not make it crash at various places but since Laidout is quite small, I can just not pay any attention to criticism. Partially because there isn't any, and I have particular motivations to make it work in a certain way and so it is easier to just go forward.

◘ *I think people that want to use your program are probably happy with this kind of visualisation. Because you wrote it alone, there is also a consistency across the program. It is not like Scribus, that has parts written by a lot of people so you can really recognize: this is Craig (Bradney), this is Andreas (Vox), this is Jean (Ghali), this is myself. There is nothing to follow.*

/ I remember Donald Knuth talking about TeX and he was saying that the entire program was written from scratch three times before its current incarnation. I am sympathetic to that style of programming.

▮ ***Start again.***

◘ *I think it is a good idea, to start again. To come back to a little detail. Is there a fileformat for your imposition tool, to store the imposition plan? Is it a text or a binary format?*

/ It is text-based, an indented file format, sort of like Python. I did not want to use XML, every time I try to use XML there are all these greater thans and less thans. It is better than binary, but it is still a huge mess. When everything is indented like a tree, it is very easy to find things. The only problem is to always input tabs, not spaces. I have two different imposition types, basically, the flat-folding sheets and the three dimensional ones. The three dimensional one is a little more complicated.

▮ ***If you read the file, do you know what you are folding?***

/ Not exactly. It lists what folds exists. If you have a five by five grid, it will say *Fold along this line, over in such and such direction.* What it actually translates to in the end, is not currently stored in the file. Once you are in Laidout you can export into a PodofoImpose plan file.

◘ *Is this file just values, or are there keywords, is it like a text?*

╱ I try to make it pretty readable, like `trimright` or `trimleft`.

❙ ***Does it talk about turning pages? This I find beautiful in PodofoImpose plans, you can almost follow the paper through the hands of the program. Turn now, flip backwards, turn again. It is an instruction for a dance.***

╱ Pretty much.

▣ *The text you can read in the PodofoImpose plans was taken from what Ludi and me did by hand. One of us was folding the paper, and the other was writing it into the plan. I think a lot of the things we talk about, are putting things from the real world into the computer. But you are putting things from the computer into the real world.*

❙ ***Can you describe again these two types of imposition, the first one being very familiar to us. It must be the most frequently asked question on the Scribus mailing list:*** **How to do imposition.** ***Even the most popular search term on the OSP website is 'Bookletprinting'. But what is the difference with the plan for a 3D object? A classic imposition plan is also somehow about turning a flat surface into a three dimensional object?***

╱ It is almost translatable. I'm reworking the 3D version to be able to incorporate the flat folding. It is not quite there yet, the problem is the connection between the pages. Currently, in the 3D version, you have a shape that has a definitive form and that controls how things bleed across the edges. When you have a piece of paper for a normal imposition, the pages that are next to each other in the physical form are not necessarily related to each other at all in the actual piece of paper. Right now, the piece of paper you use for the 3D model is very defined, there is no flexibility. Give me a few months!

❙ ***So it is very different actually.***

╱ It is a different approach. One person wanted to do flexagons, it is sort of like origami I guess, but it is not quite as complicated. You take a piece of paper, cut out a square and another square, and than you can fold it and you end up with a square that is actually made up of four different sections. Than you can take the middle section, and you get another page and you can keep folding in strange ways and you get different pages. Now the question becomes: how do you define that page, that is a collection of four different chunks of paper? I'm working on that!

▮ ***We talk about the move from 2D to 3D as if these pages are empty. But you actually project images on them and I keep thinking about maps, transitional objects where physical space is projected on paper which then becomes a second real space and so on. Are you at all interested in maps?***

⁄ A little bit. I don't really want to because it is such a well-explored field already. Already for many hundreds of years the problem is how do you represent a globe onto a more or less two dimensional surface. You have to figure out a way to make globe gores or other ways to project it and than glue it on to a ball for example. There is a lot of work done with that particular sort of imagery, but I don't know.

▣ *Too many people in the field!*

⁄ Yes. One thing that might be interesting to do though is when you have a ball that is a projection surface, then you can do more things, like overlays onto a map. If you want to simulate earthquakes for example. That would be entertaining.

▮ ***And the panoramic images you make, do you use special equipment for this?***

⁄ For the first couple that I made, I made this 30-sided polyhedron that you could mount a camera inside and it sat on a base in a particular way so you could get thirty chunks of images from a really cheap point and shoot camera. You do all that, and you have your thirty images and it is extremely laborious to take all these thirty images and line them up. That is why I made the 3D portion of Laidout, it was to help me do that in an easier fashion. Since then I've got a fish-eyed lens which simplifies things quite considerably. Instead of spending ten hours on something, I can do it in ten minutes. I can take 6 shots, and one shot up, one shot down. In Hugin you can stitch them all together.

▮ ***And the kinds of things you photograph? We saw the largest rodent on earth? How do you pick a spot for your images?***

╱ I am not really sure. I wander around and than photograph whatever stands out. I guess some unusual configuration of architecture frequently or sometimes a really odd event, or a political protest sometimes. The trick with panoramas is to find an area where something is happening all over the globe. Normally, on sunny days, you take a picture and all your image is blank. As pretty as the blue sky is, there is not a lot going on there particularly.

▮ ***Panoramic images are usually spherical or circular. Do you take certain images with a specific projection surface in mind?***

╱ To an extent. I take enough images. Once I have a whole bunch of images, the task is to select a particular image that goes with a particular shape. Like cubes there are few lines and it is convenient to line them up to an actual rectangular space like a room. The tetrahedron made out of cones, I made one of Mount St. Helens, because I thought it was an interesting way to put the two cones together. You mentioned 3D printers earlier, and one thing I would like to do is to extend the panoramic image to be more like a progression. For most panoramic images, the focal point is a single point in space. But when you walk along a trail, you might have a series of photographs all along. I think it could be an interesting work to produce, some kind of ellipsoidal shape with a panoramic image that flows along the trail.

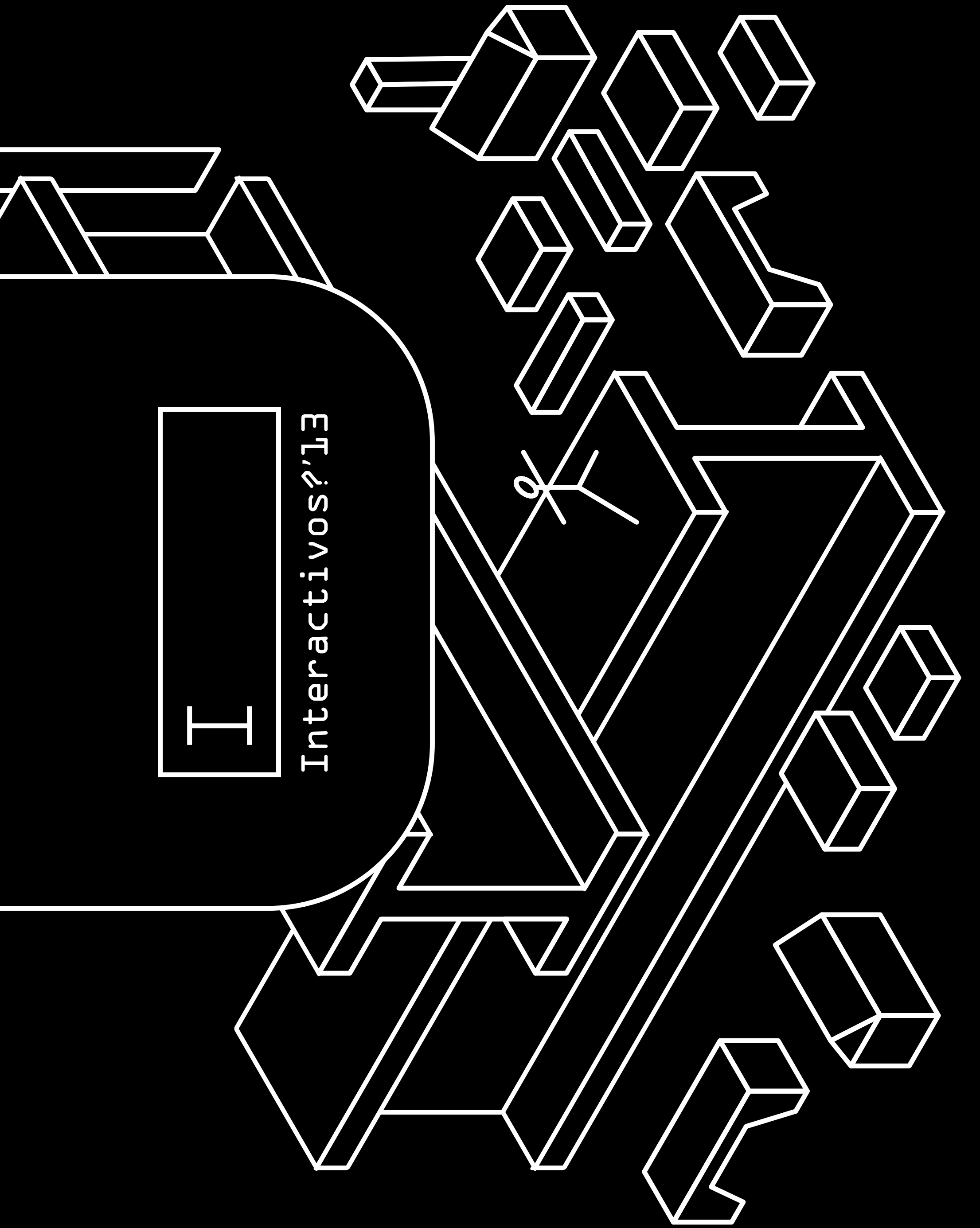
▮ ***Back to Laidout, and keeping with the physical and the digital. Would there be something like a digital papercut?***

╱ Not really. Maybe you can have an Arduino and a knife?

▮ ***I was more imagining a well placed crash?***

╱ In a sense there is. In the imposition view, right now I just have a green bar to tell where the binding is. However when you do a lot of folds, you usually want to do a staple. But if you are stapling and there is not an actual fold there, than you are screwed.

Herramientas para
un mundo legible
y editable
Tools for a
Read-Write
World

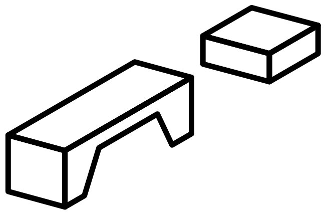

I
Interactivos?'13

# Tools for a Read-Write World

In 2013, Constant teamed up with Medialab Prado in Madrid to organise Tools for a Read-Write World, a `ten` day prototyping session that allowed us to develop `nine` different projects to design, edit, draw and write together. We wanted to extend the notion of 'Read-Write' beyond just the canvas, beyond the pixels of an image, the contents of a document into software itself, and into standards, platforms, frameworks, hardware and ways-of-doing. We asked ourselves what tools would allow us to develop, design and produce shareable content, and how we could make many different practices of knowledge work together. The following excerpts were recorded near the end of the work period, and give a sense of the issues and energies that participants brought to the table.

## Colaboratorio de Relatos #HackMito (a platform to co-create stories on and with Free Culture)

In all this work, we're trying to find something poetical in digital practices. We think that they are transforming things, and that there's something artistic in them, because they're not just programs, they're ways of doing things. Ricardo gave us a two hours explanation about how version control systems work, with just a paper and a pencil. And he explained it to people who don't know anything about  version control, and it was incredible because he could translate these concepts to us. We decided that 'diff' was really important to us. Then, we decided we could get something symbolic or something poetic out of  everything. Difference is the most important thing for me, because the point in that is the difference, not what is the common, because that is the normal way of thinking. So diff is a really important 'god' for us. Carla Boserman

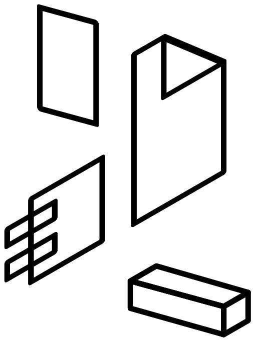


# Incoma. Exploring the 'collective' in collective intelligence

Miguel
Arana
Catania

From my point of view, politics has to do with many things. One thing politics has to do with is organisation, how we organize the world, how we organise the energy, the style, the work, everything. And we have seen that delegating this to powers to the people who has more money, etc. has terrible consequences in the life of people. So we have to find new ways of organising between people, all of them. This is one of the main ideas of the tool. Although since politics is also creating and thinking about new worlds, it's how we think the world could be, how the relations between people could be set, how we can behave in our lives. And it's about thinking, debating, which is one of the goals of the tool: to help people think together, make clear the way we think, and find more intelligent thoughts, but made by the collective, not just by some genius who comes with an idea.

*Why is it important that the structure of your system should be flexible on all levels?*



Carlos
Barragán

I think that this part can make the discussion evolve, because in traditional discussions, the only important thing is the content of the contributions, but in the process of thought, the connections you make are really important. For example, if something is a consequence of another thing, you can say `if I'm really sure of this, then I'm sure of this other thing as well`. So you can advance in the process of thought. I think this property can make discussions really go to another level. And the fact that everyone can contribute to these links can make it really powerful, I think.

It seems that
especially when you
work together on a
file, you need hints
of where things changed
if you didn't make the
change yourself.

Julien
Deswaef

Yeah. Exactly. I'm a bit mixed. I don't really know if these visual changes are maybe something that will improve a designer's workflow, but I think because this is so attractive, then designers will come to it saying `oh, let's try it` and then they will find out about versioning, and then they will find out about collaboration, and then they will find out about collaboration. I mean, the visual diff tool, it's not really practical in any way, I don't really know. But getting designers into collaboration is more important. It's a tool that will maybe attract people into that.

In my opinion, when you have a complex file or, maybe not complex, but when the changes are subtle, if there is a tool that is displaying it as a bright red, or really putting a marker on `this node has changed`, it's helpful. You don't need to spend a lot of time guessing what's the difference. I believe this is useful.

I received a lot of feedbacks, or more like `I'm interested in this!` For me, it's maybe the first real open source project where I really want other people to be involved. It's a challenge to get people involved in the project. This is a bit new to me, so I … I'm expecting that people will join in, but I have no idea if this is going to happen because I think usually people need to see a finished project before getting in. Maybe in the designers' world, I don't know. It's a question. I'm trying not to have too much expectation around this and just keep going. We'll see.

Design
with git

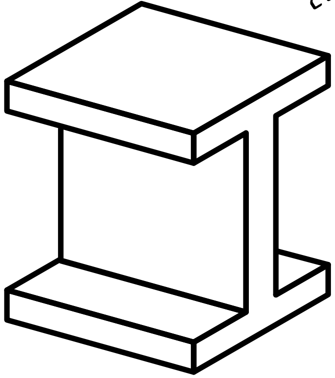
## Yes, No? Maybe,

I've been excited by some of the things that have come out.

Eleanor
Greenhalgh

*Can you describethem, why they were exciting?*

One of the things that was interesting was the idea of allowing for doubt to emerge, not in the answers that were given, but in the process of repetition. For instance, if you change your answer, how how can we represent that visually? And this is where parallels emerge with the `Design with git` project. Because then, for example, you have a design object which bears traces of its history. History in terms of how certain somebody was, how sure they were. And I found it conceptually, but also visually, interesting to experiment with how a shape could change over time and retain traces of that change.

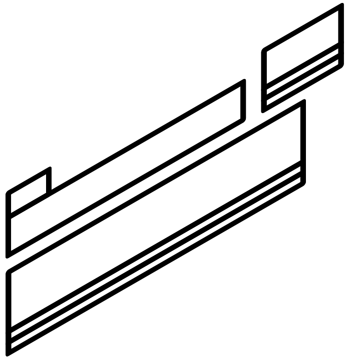
(f 'yes' doesn't
always mean 'yes',
how can we encode 'maybe'?)

## Freedom of speech kit

The main goal of the thing is this idea of empowering people. To make it accessible to people. We're not going to finish by the end of Interactivos and this is not an end, but we need to reach the first stage, a working prototype, and then, technically, we need to iterate and make it more robust, see how it behaves in real circumstances, so we can improve the system to make it more solid for real demonstrations. And we also have this open discussion, we have to talk to a lot of people to continue giving the conceptual basis and make it as powerful as we can.

Chema
Blanco

## Real time collaboration in Fontforge



I think that a lot of designers work in small studios, they work with maybe one or two other people on things. And at the moment, when people share, they save the file, they close the program, they put the file into maybe some kind of shared folder system... And then other people can open the file. If two people open the same file at once, they can overwrite each other's changes and that's always very frustrating, when you lose work. To have that kind of collaboration built right into the tool so it's live, really speeds things up.

Dave
Crossland

## Open Source digital pattern making

We’re taking into account many more points of data so that the traditional method of making garments, using ratios of waist to hip… and if you have this size hip, then your leg size is going to be a certain one. That doesn't work. What we’re doing is we're busting through even that limitation of the current garment industry. If you use a pattern created with our tool, the pattern will be 100% to your specifications with no ratios assumed. If your shoulder size is such and such, then your arm length will be… You know, it could be too short or too long.
The only thing that stays the same in each pattern is the point of origin, where you start. And then all other points are derived in sequence from that point. And using the body measurements. So with any given instance of a pattern, the only thing that is the same between patterns is the relationship, the mapping between the pattern formulas and the body measurement.

Susan Spencer

# El Recetario

(a collaborative
platform for research
and experiments on
using waste to
construct furniture)

Ana

Before coming here, I thought the 'how' was going to be more difficult, but the 'what' was way more difficult. To get the starting point right and to agree, to discuss with the group what we wanted to do. It took a long time.

Mireia

Because we wanted to make it a participative process with the users that are already registered.

Ana

We had to put ourselves in the skin of other people and be potential users, the kind of users that we don't have yet, because the ones we have already want to be in the platform. So other kinds of users.

*Do you feel you have made decisions that you now can move forward with?*

Mireia

I think we have a very defined idea of what the platform is and what it can be. We're not worried about how it's going to be, because we know it's just a matter of technical stuff and time. And it's important for us to have a prototype, but it doesn't matter how it is now, because we know how to do it.

Ana

We have the basis now, and I think it's been a big change to know that anything we do, or any decision we make… Now we are working in Wordpress, we're going to be more free to change it than we were before with the other platform. Like, okay, we can make mistakes now, or we can leave this for later because it's easier.

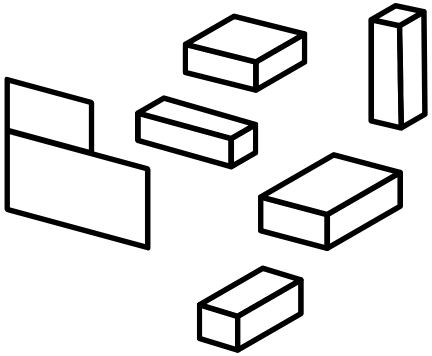
Mireia Juan Coco
Ana Pérez Aguilar

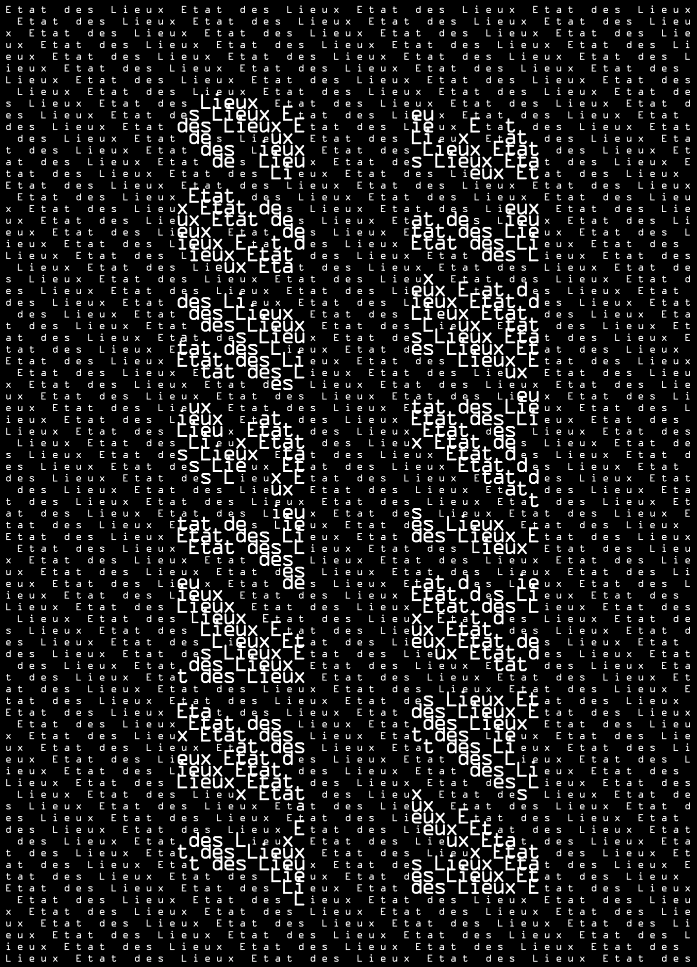
## GRAPA (platform for libre editions and publications)

*You talked about how
a book is made by
many hands, which is
a really nice way of
a describing the
process of
publishing, as a
relation between
let's say texts and
books and bodies.*

I found it really beautiful what Enrique said one day,
that the limit of reading words in a book, is the paper.
When you read it, those words come to you so that
your skin becomes the limit of those words, so that you
think them and then show to others. That's beautiful.
For me, it would be very beautiful if just one person
would come and tell us your tool helped me
with... I don't know what. Someone interested,
someone who plays with mixing. If someone comes
and says Thank you, you helped me, you
made my life easier that's enough.

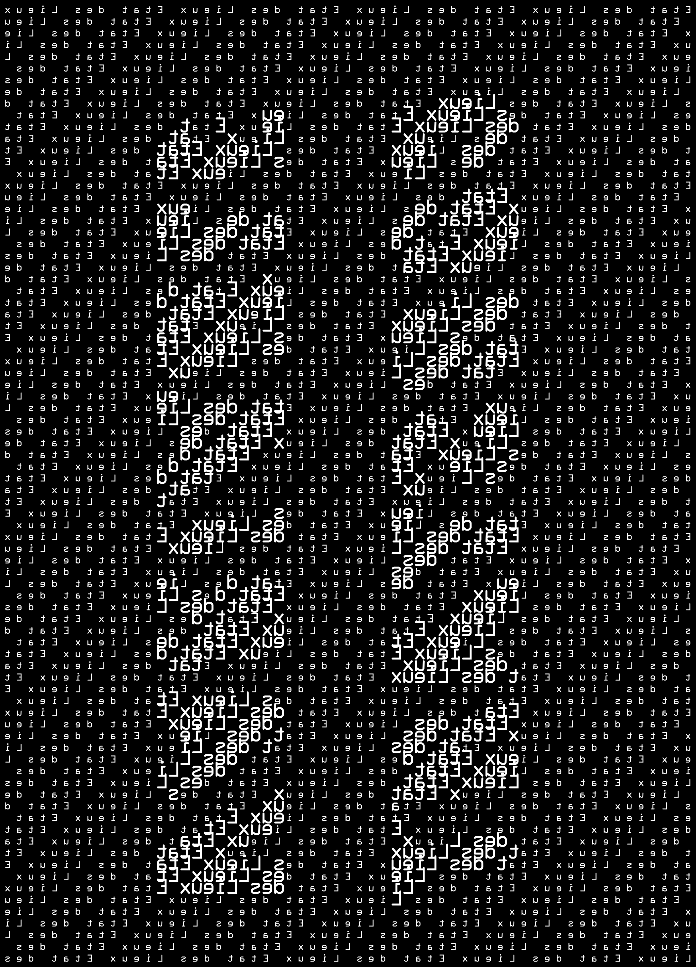
Olatz
Otalora





# Etat des Lieux

Eleanor Greenhalgh
John Colenbrander
Christina Clar
Claire Williams
Christoph Haag
Michael Murtaugh
Urantsetseg Ulziikhuu



# Etat des Lieux

Eleanor Greenhalgh
John Colenbrander
Christina Clar
Claire Williams
Christoph Haag
Michael Murtaugh
Urantsetseg Ulziikhuu

The following statements were recorded by Urantsetseg Ulziikhuu (Urana) in **2014**. She studied communication in Istanbul and Leuven and joined Constant for a few months to document the various working practices at Constant Variable. Between 2011 and 2014, Variable housed studios for Artists, Designers, Techno Inventors, Data Activists, Cyber Feminists, Interactive Geeks, Textile Hackers, Video Makers, Sound Lovers, Beat Makers and other digital creators who were interested in using F/LOS software for their creative experiments.

Urantsetseg Ulziikhuu

## *Why do you think people should use and or practice Open Source software? What is in it for you?*

Claire Williams The knitting machine that I am using normally has a computer from the eighties. Some have these scanners that are really old and usually do not work anymore. They became obsolete. If it wasn't for Open Source, we couldn't use these technologies anymore. Open Source developers decided that they should do something about these machines and found that it was not that complicated to connect these knitting machines directly to computers. I think it is a really good example how Open Source is important, because these machines are no longer produced and industry is no longer interested in producing them again, and they would have died without further use.
The idea that Open Source is about sharing is also important. If you try to do everything from zero, you just never advance. Now with Open Source, if somebody does something and you have access to what they do, and you can take it further and take it into a different direction.

Michael Murtaugh I haven't always used Open Source software. It started at the Piet Zwart Institute where there was a decision made by Matthew Fuller and Femke Snelting who designed the program. They brought a bunch of people together that asked questions about how our tools influence practice, how they are used. And so, part of my process is then teaching in that program, and starting to use Free Software more and more. I should say, I had already been using one particular piece of Free Software which is FFmpeg, a program that lets you work with video. So there again there was a kind of connection. It was just by the virtue of the fact that it was one of the only tools available that could take a video, pull out frames, work with lots of different formats, just an amazing tool. So it started with convenience. But the more that I learned about the whole kind of approach of Open Source, the more Open Source I started to use. I first switched from MacOSX to maybe Dual Booting and now indeed I am pretty much only using Open Source. Not exclusively Open Source, because I occasionally use platforms online that are not free, and some applications.
I am absolutely convinced that when you use these tools, you are learning much more about inner workings of things, about the design decisions that go into a piece of software so that you are actually understanding at a very deep level, and this then lets you move between different tools. When tools change, or new things are offered, I think it is really a deep learning that helps you for the future. Whereas if you just focus on the specific particularities of one platform or piece of software, that is a bit fragile and will inevitably be obsolete when a software stops being developed or some kind of new kind of way of working comes about.

Eleanor Greenhalgh I use Open Source software every day, as I have Debian on my laptop. I came to it through anarchism – I don't have a tech background – so it's a political thing mainly. Not that F/LOSS represents a Utopian model of production by any means! As an artist it fits in with my interest in collaborative production. I think the tools we use should be malleable by the people who use them. Unfortunately, IT education needs to improve quite a lot before that ideal becomes reality.
Politically, I believe in building a culture which is democratic and malleable by its inhabitants, and F/LOSS makes this possible in the realm of software. The benefits as a user are not so great unless you are tech-savvy enough to really make use of that freedom. The software does tend to be more secure

and so on, though I think we're on shaky ground if we try to defend F/LOSS in terms of its benefits to the end user. Using F/LOSS has a learning curve, challenges which I put up with because I believe in it socially. This would probably be a different answer from say, a sysadmin, someone who could see really concrete benefits of using F/LOSS.

Christoph Haag Actually I came from Open Content and alternative licensing to the technical side of using GNU/Linux. My main motivation right now is the possibility to develop a deeper relationship with my tools. For me it is interesting to create my own tools for my work, rather than to use something predefined. Something everyone else uses. With Free Software this is easier – to invent tools. Another important point is that with Free Software and open standards it's more likely that you will be able to keep track of your work. With proprietary software and formats, you are pretty much dependent on decisions of a software company. If the company decides that it will not continue an application or format, there is not much you can do about it. This happened to users of FreeHand. When Adobe acquired their competitor Macromedia they decided to discontinue the development of FreeHand in favour of their own product Illustrator. You can sign a petition, but if there is no commercial interest, most probably nothing will happen. Let's see what happens to Flash.

Christina Clar I studied sculpture, which is a very solitary way of working. Already through my studies, this idea of an artist sitting around in a studio somewhere, being by himself, just doing his work by himself, didn't make sense to me. It is maybe true for certain people, but it is definitely not true to me today, the person I am. I always integrated other people into my work, or do collaborative work. I don't really care about this 'it is my work' or 'it is your work', if you do something together, at some point the work exists by itself. For me, that is the greatest moment, it is just independent. It actually rejoins the authorship question, because I don't think you can own ideas. You can kind of put them out there and share them. It is organic, like things that can grow and that they will become bigger and bigger, become something else that you couldn't have ever thought. It makes the horizon much bigger. It is a different way of working I guess.
The obvious reason is that it is free, but the sharing philosophy is really at the core of it. I have always thought that when you share things, you do not

get back things instantly, but you do get so much things in another way, not in the way you expect. But if you put in a idea out, use tools that are open and change them, put them out again. So there is lot of back and forth of communication. I think that is super important. It is the idea of evolving together, not just by ourselves. I really do believe that we do evolve much quicker if we are together than everybody trying to do things by his or herselves. I think it is very European idea to get into this individualism, this thinking of idea of doing things by myself, my thing. But I think we can learn a lot from Asia, just ways of doing, because there community is much more important.

John Colenbrander I don't necessarily develop like software or codes, because I am not a software developer. But I would say, I am involved in analog way. I do use Open Source software, although I have to say I do not much with computers. Most of my work is analog. But I do my researches on the website. I am a user.
I started to develop an antipathy against large corporations, operating systems or softwares, and started to look for alternatives. Then you come to the Linux system and Ubuntu which has a very user-friendly interface. I like the fact that behind the software that I am using, there is a whole community, who are until now without major financial interests and who develop tools for people like me. So now I am totally into Open Source software, and I try to use as much as I can. So my motivation would be I want to get off the track of big corporates who will always kind of lead you into consuming more of their products.

Urantsetseg Ulziikhuu

## *What does Free Culture mean to you? Are you taking part in a 'Free Culture Movement'?*

Michael Murtaugh I'd like to think so, but I realised of that it is quite hard. Only now, I am seriously trying to really contribute back to projects and I wouldn't even say that I am an active contributer to Free Software projects. I am much more of a user and part of the system. I am using it in my teaching and my work, but now I try to maybe release software myself in some way or I try to create projects that people could actually use. I think it is another kind of dimension of engagement. I haven't really fully realised it, so yes for that question if I am contributing to Free Culture. Yes, but I could go lot deeper.

John Colenbrander I am a big supporter of the idea of Free Culture. I think information should be available for people, especially for those who have little access to information. I mean we live in the West and we have access to information more or less with physical libraries and institutions where we can go. Specially in Asia, South America, Africa this is very important. There is a big gap between those who have access to knowledge and those don't have access to knowledge.
That's a big field to explore to be able to open up information to people who have very poor access to information. Maybe they are not even able to write or read. That's already is a big handicap. So I think it is a big mission in that sense.

Urantsetseg Ulziikhuu

## *Could Free Culture be seen as an opposition to commercialism?*

Michael Murtaugh It is a tricky question. I think no matter what, if you go down the stack, in terms of software and hardware, if you get down to the deepest level of a computer then there is little free CPU design. So I think it is really important to be able to work in this kind of hybrid spaces and to be aware of then how free Free is, and always look for alternatives when they are available. But to a certain degree, I think it is really hard to go for a total absolute. Or it is a decision, you can go absolute but that may mean that you are really isolated from other communities. So that's always a bit of balancing act, how independent can you be, how independent you want to be, how big does your audience need to be, or you community needs to be. So that's a lot of different decisions. Certainly, when I am working in the context of an art school with design practitioners, you know it is not always possible to really go completely independent and there are lots of implications in terms of how you work and whom you can work with, and the printers you can work with. So it is always a little bit of trade-off, but it is important to understand what the decisions are.

Eleanor Greenhalgh I think the idea of a Free Culture movement is very exciting and important. It has always gone on, but stating it in copyright-aware terms issues an important challenge to the 'all rights reserved' status-quo. At the same time I think it has limitations, at least in its current form. I'm not sure that rich white kids playing with their laptops is necessarily a radical act. The idea and the intention are very powerful though, because it does have the potential to challenge the way that power – in the form of 'intellectual property' – is distributed.

Christoph Haag Copyright has become much more enforced over the last years than it was ever before. In a way, culture is being absorbed by companies trying to make money out of it. And Free Culture developed as a counter movement against this. When it comes to mainstream culture, you are most often reduced to a consumer of culture. Free Culture then is a obvious reaction. The idea of culture where you have the possibility to engage again, to become active and create your version, not just to consume content.

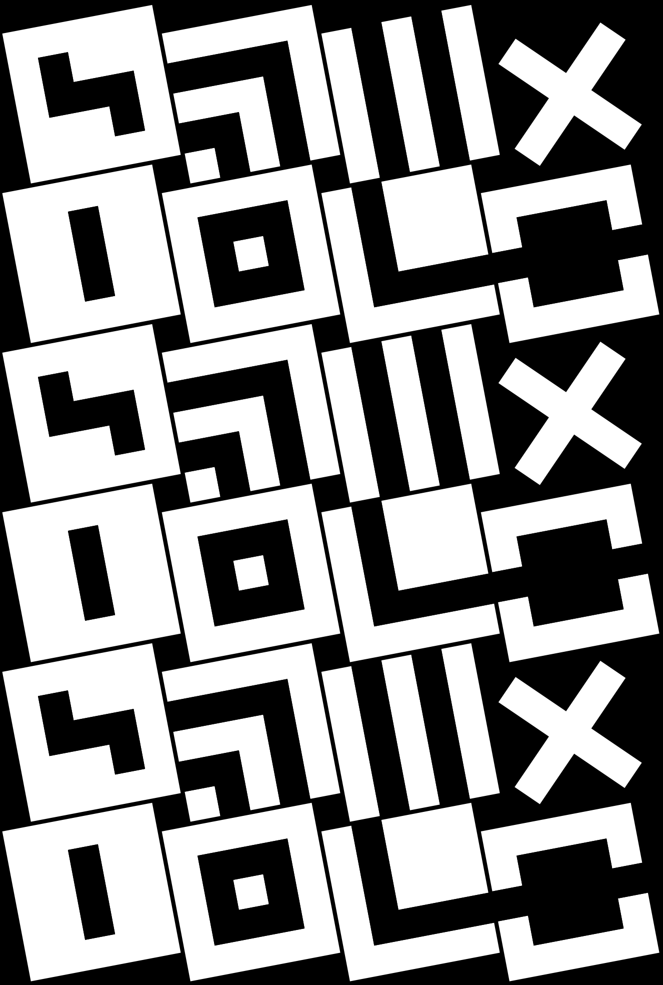
Urantsetseg Ulziikhuu

*How could Open Source software be economically sustainable, in a way that is beneficial for both developers/creators and users?*

Eleanor Greenhalgh That's a good question! A very hard one. I'm not involved enough in that community to really comment on its economic future. But it does, to me, highlight what is missing from the analysis in Free Culture discourse, the economic reality. It depends on where they (developers) work. A lot of them are employed by companies so they get a salary. Others do it for a hobby. I'd be interested to get accurate data on what percentage of F/LOSS developers are getting paid, etc. In the absence of that data, I think it's fair to say it is an unsolved problem. If we think that developers 'should' be compensated for their work, then we need to talk about capitalism. Or at least, about statutory funding models.

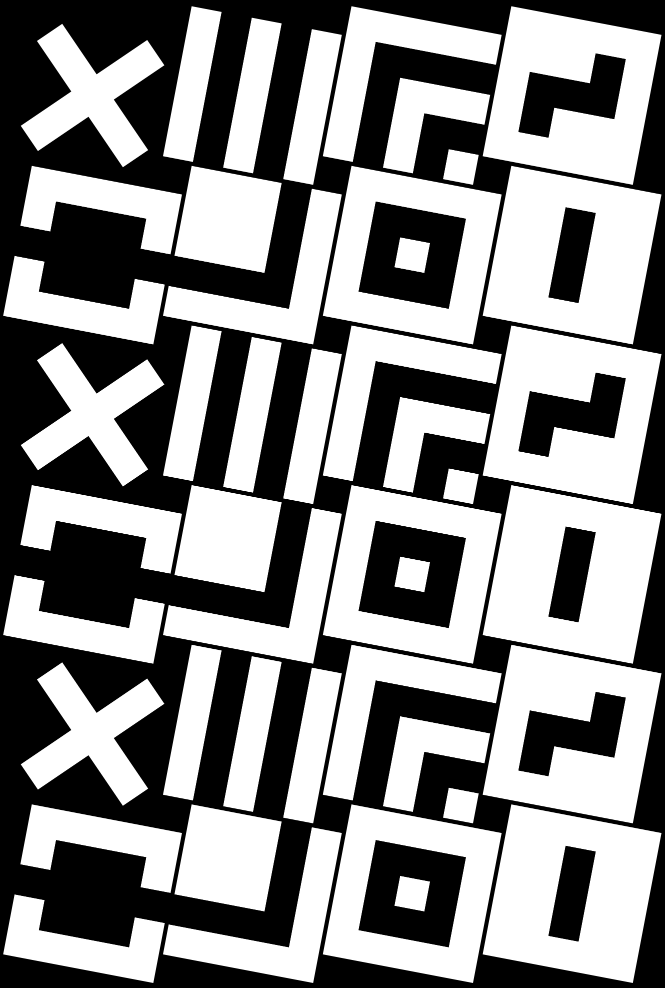
Michael Murtaugh It is interesting that you used both 'sustainability' and 'economic viability'. And I think those are two things very often in opposition. I am doing a project now about publishing workflows and future electronic publishing forums. And that was the one thing we looked at. There were several solutions on the market. One was a platform called 'Editorial' which was a very nice website that you could use to mark down texts collaboratively and and then it could produce ePub format books. After about six months of running, it closed down as many platforms do. Interestingly, in their sign-off message it said: *You have a month to get your stuff out of the website, and sorry we have decided not to Open Source the project. As much as we loved making it, it was just too much work for us to keep this running.* In terms of real sustainability, Open Source of course would have allowed them to work with anybody, even if it is just a hobby.

Claire Williams It is very related to passion of doing these things.
Embroidering machines have copyrighted softwares installed. The software itself is very expensive, around 1000 €, and the software for professionals is 6000 € to buy. Embroidering machines are very expensive themselves too. These softwares are very tight and closed, you even have to have special USB key for patterns. And there are these two guys who are software developers, they are trying to come up with a format which all embroidering machines could read. They take their time to do this and I think in the end if the project works out, they will probably get attention and probably get paid also. Because instead of giving 1000 € to copyrighted software, maybe you would be happy to give 50€ to these people.



## Distributed Version Control

× Ludvine Loiseau
▙ Sarah Magnan
▣ Alexandre Leray
▣ Stéphanie Vilayphiou
▜ Eric Schrijver
▮ Pierre Huyghebaert
▣ Pierre Marchand
Ⅲ Femke Snelting



Distributed Version Control

| | |
|---|---|
| Ludvine Loiseau | Eric Schrijver |
| Sarah Magnan | Pierre Huyghebaert |
| Alexandre Leray | Pierre Marchand |
| Stéphanie Vilayphiou | Femke Snelting |

Date: Thu, **12 Sep 2013** 15:50:25 +0200
From: FS `<snelting@collectifs.net>`
To: OSP `<mail@osp.constantvzw.org>`

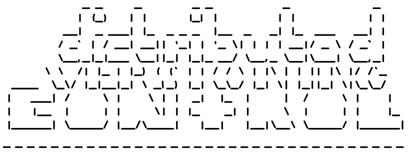
Dear OSP,

For a long time I have wanted to organise a conversation with you about the place and meaning of distributed version control in OSP design work. First of all because after three years of working with Git intensely, it is a good moment to take stock. It seems that many OSP methods, ideas and politics converge around it and a conversation discussing OSP practice linked to this concrete (digital) object could produce an interesting document; some kind of update on what OSP has been up to over the last three years and maybe will be in the future. Second: Our last year in Variable has begun. Under the header *Etat des Lieux*, Constant started gathering reflections and documents to archive this three year working period. One of the things I would like to talk about is the parallels and differences between a physical studio space and a distributed workflow. And of course I am personally interested in the idea of 'versions' linked to digital collaboration. This connects to old projects and ideas and is sparked again by new ones revived through the Libre Graphics Research Unit and of course Relearn.
I hope you are also interested in this, and able to make time for it. I would imagine a more or less structured session of around two hours with at least four of you participating, and I will prepare questions (and cake).

Speak soon!

x F

Eric Schrijver
*Stéphanie Vilayphiou*
Alexandre Leray
**Femke Snelting**
Ludvine Loiseau



**How do you usually explain Git to design students?**

Before using Git, I would work on a document. Let's say a layout, and to keep a trace of the different versions of the layout, I would append _01, _02 to the files. That's in a way already versioning. What Git does, is that it makes that process somehow transparent in the sense that, it takes care of it for you. Or better, you have to make it take care for you. So instead of having all files visible in your working directory, you put them in a database, so you can go back to them later on. And then you have some commands to manipulate this history. To show, to comment, to revert to specific versions.

More than versioning your own files, it is a tool to synchronize your work with others. It allows you to work on the same projects together, to drive parallel projects.

It really is a tool to make collaboration easier. It allows you to see differences. When somebody proposes you a new version of a file, it highlights what has changed. Of course this mainly works on the level of programming code.

**Did you have any experience with Git before working with OSP?**

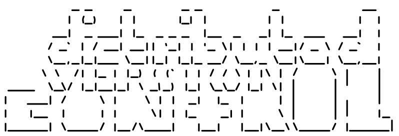
Well, not long before I joined OSP, we had a little introduction to Mercurial, another versioning software, at school in 2009. Shortly after I switched to Git. I was working with someone else who was working with Git, and it was so much better.

*Alex was interested in using Git to make Brainch*[1]*. We wanted to make a web application to fork texts that are not code. That was our first use of Git.*

I met OSP through Git in a way. An intern taught me the program and he said: *Eric once you'll get it, you'll get so excited!*. We were in the cafeteria of the art school. I thought it was really special, like someone was letting me in on a secret and we we're the only ones in the art school who knew about it. He thought me how to `push` and `pull`. I saw quickly how Git really is modeled on how culture works. And so I felt it was a really interesting, promising system. And then I talked about it at the Libre Graphics Meeting in 2010, and so I met OSP.

[1] A distributed text editing platform based on Django and Git http://code.dyne.org/brainch

◘ Pierre Marchand
⊏⊐ *Stéphanie Vilayphiou*
||| **Femke Snelting**
✕ Ludvine Loiseau
◙ Alexandre Leray



◘ I started to work on collaborative, graphic design related stuff when I was developing a font manager. I've been connected to two versioning systems and mainly used SVN. Git came well after, it was really connected to web culture, compared to Subversion, which is more software related.

||| **What does it mean that Git is referred to as 'distributed versioning'?**

◙ The first command you learn in Git, is the `clone` command. It means that you make a copy of a project that is somehow autonomous. Contrary to Subversion you don't have this server-client architecture. Every repository is in itself a potential server and client. Meaning you can keep track of your changes offline.

||| **At some point, you decided to use 'distributed versioning' rather than a centralized system such as Subversion. I remember there was quite some discussion ...**

✕ I was not hard to convince. I had no experience with other versioning systems. I was just excited by the experience that others had with this new tool. In fact there was this discussion, but I don't remember exactly the arguments between SVN or Git. For what I remember Git was easier.

◘ The discussion was not really on the nature of this tool. It was just: *who would keep Git running for OSP?* Because the problem is not the system in itself, it's the hosting platform. We didn't find any hosted platform which fitted our taste. The question was: do we set up our own server, and who is going to take care of at. At this time Alex, Steph and Ivan were quite excited about working with Git. And I was excited to use Subversion instead, but I didn't have to time to take care of setting it up and everything.

||| **You decided not to use a hosted platform such as Gitorious or GitHub?**

⊏⊐ *I guess we already had our own server and were hosting our own projects. But Pierre you used online platforms to share code?*

◘ When I started developing my own projects it was kind of the end of SourceForge.[2] I was looking for a tool more in the Free Software tradition.

[2] SourceForge is a web based source code repository. It was the first platform to offer this service for free to Open Source projects.

There was gna, and even though the platform was crashing all the time, I felt it was in line with this purpose.

▙ If I remember correctly, when we decided between Git and Subversion, Pierre, you were also not really for it because of the personality of its main developer, Linus Torvalds. I believe it was the community aspect of Git that bothered you.

▣ Well Git has been written to help Linus Torvalds receive patches for the Linux kernel; it is not aimed at collaborative writing. It was more about making it convenient for Linus. And I didn't see a point in making my practice convenient for Linus. I was already using Subversion for a while and it was really working great at providing an environment to work together with a lot of people and check out different versions. Anything you expect from a versioning system was there, all elements for collaborative work were there. I didn't see the point to change for something that didn't feel as comfortable with, culturally. This question of checking out different directories of repositories was really important to me. At this time (Git has evolved a lot) it was not possible to do that. There were other technical aspects I was quite keen of. I didn't see why to go for Git which was not offering the same amount of good stuff.

⅂ But then there is this aspect of distribution, and that's not in Subversion. If some day somebody decides to want a complete copy of an OSP project, including all it's history, they would need to ask us or do something complicated to give it to them.

▣ I was not really interested in this 'spreading the whole repository'. I was more concerned about working together on a specific project.

||| **It feels like your habit of keeping things online has shifted. From making an effort afterwards to something that happens naturally, as an integral part of your practice.**

✕ It happened progressively. There is this idea that the Git repository is linked to the website, which came after. The logic is to keep it all together and linked, online and alive.

That's not really true … it was the dream we had: once we have Git, we share our files while working on them. We don't need to have this effort afterwards of cleaning up the sources and it will be shareable. But it is not true. If we do not put an effort to make it shareable it remains completely opaque. It requires still an investment of time. I think it takes about 10% of time of the project, to make it readable from the outside afterwards.

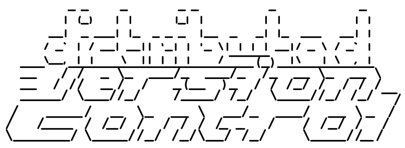
*Now, with the connection to our public website, you're more conscious that all the files we use are directly published. Before we had a Git web application that allowed someone to just browse repositories, but it was not visual, so it was hard to get into it. The Cosic project is a good example. Every time I want to show the project to someone, I feel lost. There are so many files and you really don't know which ones to open.*

**Maybe, Eric, you can talk about 'Visual Culture'?**

Basically 'Visual Culture' is born out of this dream I talked about just now. That turns out not to be true, but shapes our practice and helps us think about licensing and structuring and all those interesting questions. I was browsing through this Git interface that Stéphanie described, and thought it was a missed opportunity, because here is this graphic design studio, who publishes all their works, while they are working. Which has all kind of consequences but if you can't see it, if you don't know anything about computer programming, you have no clue on what's going on. And also, because it's completely textual. And for example a .sla file, if you don't know about Open Source, if you don't know about Scribus it could as well be salad. It is clear that Git was made for text. It was the idea to show all the information that is already there in a visual form. But an image is an image, and type is a typeface, and it changes in a visual way. I thought it made sense for us to do. We didn't have anyone writing posts on our blog. But we had all this activity in the Git repository.

*It started to give some schematic view on our practice, and renders the current activity visible, very exciting. But it is also very frustrating because we have lots of ideas and very little time to implement them. So the 'Visual Culture' project is terribly late on the ball comparing to our imagination.*

■ Alexandre Leray
||| **Femke Snelting**
╕ Eric Schrijver
▭ *Stéphanie Vilayphiou*
▮ *Pierre Huyghebaert*



---

*Take by example the foundry. Or the future potential of the 'Iceberg' folders. Or our blog that is sometimes cruelly missing. We have ways to fill all these functions with 'Visual Culture' but still no time to do it!*

||| **In a way you follow established protocols on how Open Source code is usually published. There should be a license, a README file … But OSP also decided to add a special folder, which you called 'Iceberg'. This is a trick to make your repository more visual?**

╕ Yeah, because even if something is straightforward to visualise, it helps if you can make a small render of it. But most of the files are a accumulation of files, like a webpage. The idea is that in the 'Iceberg' folder, we can put a screenshot, or other images …

▭ *We wanted the files that are visible, to be not only the last files added. We wanted to be able to show the process. We didn't want it to be a portfolio and just show the final output. But we wanted to show errors and try-outs. I think it's not only related to Git, but also to visual layout. When you want to share software, we say* release early, release often, *which is really nice. But it's not enough to just release, because you need to make it accessible to other people to understand what they are reading. It's like commenting your code, making it … I don't want to say 'clean' … legible, using variable names that people can understand. Because, sometimes when we code just for ourselves I use French variables so that I'm sure that it's not word-protected by the programming language. But then it is not accessible to many people. So stuff like that.*

||| **You have decided to use a tool that's deeply embedded in the world of F/LOSS. So I've always seen your choice for Git both as a pragmatic choice as well as a fan choice?**

╕ Like as fans of the world of Open Source?

||| **Yes. By using this tool you align yourself, as designers, with people that develop software.**

■ I'm not sure, I join Pierre on his feelings towards Linus Torvalds, even though I have less anger at him. But let's say he is not someone I especially like in his way of thinking. What I like very much about Git is the distributed aspect. With it you can collaborate without being aligned together. While I think Linus Torvalds idea is very liberal and in a way a bit sad, this idea that you can collaborate without being aligned, without going through this permission system, is interesting. With Scribus for example, I never collaborated on it, it's such a pain to got through the process. It's good and bad. I like the idea of a community which is making a decision together, at the same time it is so hard to enter this community that you just don't want to and give up.

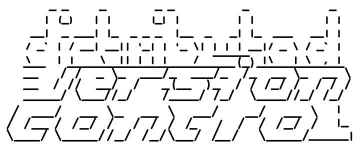
||| **How does it feel, as a group of designer-developers, to adopt workflows, ways of working, and also a vocabulary that comes from software development?**

⅂ On the one hand it's maybe a fan act. We like this movement of F/LOSS development which is not always given the importance it has in the cultural world. It's like saying *hey I find you culturally relevant and important.* But there's another side to it. It's not just a distant appropriation, it's also the fact that software development is such a pervasive force. It's so much shaping the world, that I feel I also want to take part in defining what are these procedures, what are these ways of sharing, what are these ways of doing things. Because I also feel that if I ask someone from another field as a cultural actor, and take and appropriate these mechanisms and ways of doing, I will be able to influence what they are. So there is the fan act, and there's also the act of trying to be aware of all the logic contained in these actions.

✕ And from another side, in the world of graphic design it is also a way to affirm that we are different. And that we're really engaged in doing this and not only about designing nice pictures. That we really develop our own tools.

⅂ It is a way to say: hey, we're not a kind of politically engaged designers with a different political goal each next half month, and than we do a project about it. It really impacts our ecosystem, we're serious about it.

◰ Alexandre Leray
╕ Eric Schrijver
**||| Femke Snelting**
⊐ *Stéphanie Vilayphiou*
▣ Pierre Marchand



---

⊐ *It's true that, before we started to use Git, people asked:* So you're called Open Source Publishing, but where are your sources? *For some projects you could download a .zip file but it was always a lot of trouble, because you needed to do it afterwards, while you were already doing other projects.*

▣ Collaboration started to become a prominent part of the work; working together on a project. Rather than, *oh you do that and when you are finished you send the file over and I will continue.* It's really about working together on a project. Even if you work together in the same space, if you don't have a system to share files, it's a pain in the ass.

**||| After using it for a few years, would you say there are parts of in Git where you do not feel at home?**

◰ In Git, and in versioning systems in general, there is that feeling that the latest version is the best. There is an idea of linearity, even though you can have branches, you still have an idea of linearity in the process.

⊐ *Yes, that's true. We did this workshop* Please computer let me design*, the first time was in a French school, in French, and the second time for a more European audience, in English. We made a branch, but then you have the default branch - the English one - you only see that one, while they are actually on the same level.*

**||| So the convention is to always show the main branch, the 'master'?**

╕ In a way there is no real requirement in Git to have a branch called 'master'. You can have a branch called 'English' and a branch called 'French'. But it's true that all the visualization software we know (GitHub or Gitorious are ways to visualize the content of a Git repository), you'll need to specify which is the branch that is shown by default. And by default, if you don't define it, it is 'master'.

◰ For certain types of things such as code and text it works really well, for others, like you're making a visual design, it's still very hard to compare differences. If I make a poster for example I still make several files instead of branches, so I can see them together at once, without having to check-out another branch. Even in websites, if I want to make a layout, I'll simply make a copy of the HTML and CSS, because I want to be able to test out and compare them. It might be possible with branches, it's just to complicated. Maybe the tools to visualize it are not there … But it's still easier to make copies and pick the one you like.

*It's quite heavy to go back to another version. Also working collaboratively is actually quite heavy. For example in workshops, or the 'Balsamine' project … we were working together on the same files at the same time, and if you want to share your file with Git you'll have to first* `add` *your file, then* `commit` *and* `pull` *and* `push`*, which is four commands. And every time you commit you have to write a message. So it is quite long. So while we were working on the .css for 'Visual Culture', we tried it in Etherpad, and one of us was copying the whole text file and committing.*

**So you centralized in the end.**

It's more about third-party visual software. Let's say Etherpad for example, it's a versioning system in itself. You could hook into Git through Etherpad and each letter you type could be a commit. And it would make nonsense messages but at the same time it would speed up the process to work together. We can imagine the same thing with Git (or any other collaborative working system) integrated into Inkscape. You draw and every time you save … At some point Subversion was also a WebDav server, it means that for any application it was possible to plug things together. Each time you would save you file it would make a commit on the server. It worked pretty well to bring new people into this system because it was just exactly the same as the OpenOffice, it was an open WebDav client. So it was possible to say to OpenOffice that you, where you save is a disk. It was just like saving and it was committing.

I really agree. From the experience of working on a typeface together in Git with students, it was really painful. That's because you are trying to do something that generates source code, a type design program generates source code. You're not writing it by hand, and if you then have two versions of the type design program, it already starts to create conflicts that are quite hard. It's interesting to bring to models together. Git is just an architecture on how to start your version, so things could hook into it.

For example with Etherpad, I've looked into this API the other day, and working together with Git, I'm not sure if having every Etherpad revision directly mapped to a Git revision would makes sense if you work on a project … but at the same time you could have every saved revision mapped to a Git revision. It's clear Git is made for asynchronous collaboration process. So there is Linus in his office, there are patches coming in from different people. He has the time also to figure out which patch needs to go where. This doesn't really work for the Etherpad-style-direct-collaboration. For me it's cool to think about how you could make these things work together. Now I'm working on this collaborative font editor which does that in some sort of database. How would that work? It would not work if every revision would be in the Git. I was thinking you could save, or sort of `commit`, and that would put it in a Git repository, this you can `pull` and `push`. But if you want to have four people working together and they start `pull`ing, that doesn't work on Git.

*I never really tried Sparkleshare, that could maybe work? Sparkleshare is making a `commit` message every time you save a document. In a way it works more like Dropbox. Every time you save it's synchronized with the server directly.*

**So you need to find a balance between the very conscious commits you make with Git and the fluidness of Etherpad, where the granularity is much finer. Sparkleshare would be in between?**

*I think it would be interesting to have this kind of Sparkleshare behaviour, but only when you want to work synchronously.*

**So you could switch in and out of different modes?**

*Usually Sparkleshare is used for people who don't want to get to much involved in Git and its commands. So it is really transparent: I send my files, it's synchronized. I think it was really made for this kind of Dropbox behaviour. I think it would make sense only when you want to have your hands on the process. To have this available only when you decide,* OK I go synchronous. *Like you say, if you have a commit for every letter it doesn't make sense.*

It makes sense. A lot of things related to versions in software development is meant to track bugs, to track programming choices.

||| **Femke Snelting**
⊏⊐ *Stéphanie Vilayphiou*
⊡ Pierre Marchand
✕ Ludvine Loiseau
◲ Alexandre Leray

# Distributed Version Control

---

⊏⊐ *I don't know for you … but the way I interact with our Git repository since we started to work with it … I almost never went into the history of a project. It's just, it really never happened to go back into this history, to check out an old version.*

◲ I do!

⊡ Some neat feature of Git is the dissect command. To find where it broke.

◲ You can top from an old revision that you know that works and then track down, like `checkout`, track down the bug.

||| **Can you give a concrete example, where that would be useful, I mean, not in code.**

◲ Not code, okay. That I don't know.

⊏⊐ *In a design, like visual design, I think it never happens. It happens on websites, on tools. Because there is a bug, so you need to come back to see where it broke. But for a visual design I'm not sure.*

✕ It's true, also because as you said before, with .svg files or .sla files we often have several duplicates. I sometimes checkout those. But it's true it's often related to merge problems. Or something, you don't know what to do, so you'll just check-out, to go back to an earlier version.

⊡ It would be interesting for me to really look at our use of Git and map some kind of tool on top of a versioning system. Because it's not even versioning, it is also a collaborative workflow, and to see what we mean. Just to use maybe some feature of Git or whatever to provide the services we need and really see what we exactly work with. And, this kind of thing where we want to see many versions at the same time, to compare seems important. Well it's the kind of thing that could take advantage of a versioning system, to build.

||| **It is of course a bit strange that if you want to see different versions next to each other you have to go back in time. It's a kind of paradox, no?**

⊏⊐ *But then you can't see them at the same time*

||| **Exactly, no.**

*Because there is no way to visualize your trip back in history.*

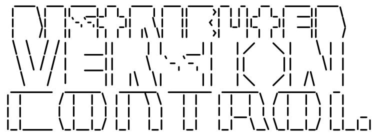
Well I think, something you could all have some interesting discussion about, is the question of exchange. Because now we are talking about the individual. We've talked how it's easier to contribute to Git based projects but to be accepted into an existing repository someone needs to say okay, I want it, which is like SVN. What is easier, is to publish you're whole Git repository online, with the only difference from the the first version, is that you added your change, but it means that in proposing a change you are already making a new cultural artifact. You're already putting a new something there. I find this to be a really fascinating phenomena because it has all kinds of interesting consequences. Of course we can look at it the way of, it's the cold and the liberal way of doing things. Because the individual is at the center of this, because you are on your own. It's your thing in the first place, and then you can see if it maybe becomes someone else's thing too. So that has all kinds of coldness about it and it leads to many abandoned projects and maybe it leads to a decrease of social activity around specific projects. But there's also an interesting part of it, where it actually resembles quite well how culture works in the first place. Because culture deals with a lot redundancy, in the sense that we can deal with many kinds of very similar things. We can have Akzidenz Grotesk, Helvetica and the Akkurat all at the same time, and they have some kind of weird cultural lineage thing going on in between them.

**Are there any `pull` requests for OSP?**

*We did have one.*

**Eric is right to ask about collaboration with others, not only how to work internally in a group.**

*That's why GitHub is really useful. Because it has the architecture to exchange changes. Because we have our own server it's quite private, it's really hard to allow anyone to contribute to fonts for example. So we had e-mails:* Hey here's a new version of the font, I did some glyphs, but also changed the shape of the A. *There we have two different things, new glyphs is one thing, we could say*



*we take any new glyph. But changing the A, how do you deal with this? There's a technical problem, well not technical…*

||| **An architectural problem?**

⊓ *Yeah, we won't add everyone's SSH-key to the server because it will be endless to maintain. But at the same time, how do you accept changes? And then, who decides what changes will be accepted?*

✕ For the foundry we decided to have a maintainer for each font project.

▣ It's the kind of thing we didn't do well. We have this kind of administrative way of managing the server. Well it's a lot of small elements that all together make it difficult. Let's say at some point we start to think *maybe we need to manage our repositories, something a bit more sophisticated then Gitolite*. So we could install something like Gitorious. We didn't do it but we could imagine to rebuild a kind of ecosystem where people have their own repositories and do anything we can imagine on this kind of hosting service. Gitorious is a Free Software so you can deploy it on your own server. But it is not trivial to do.

||| **Can you explain the difference between Gitorious and GitHub?**

◰ Gitorious is first a free version, it's not a free version of Git but GitHub. One is free and one is not.

||| **Meaning you can not install GitHub on your own server.**

▣ Git is a storage back-end, and Gitorious or GitHub are a kind of web application to interact with the repository and to manage them. And GitHub is a program and a company deploying these programs to offer both a commercial service and a free-of-charge service. They have a lot of success with the free service Git in a sense. And they make a lot of money at providing the same service, exactly the same, just it means that you can have private space on the server. It's quite convenient, because the tools are really good to manage repositories. And Gitorious I don't exactly know what is their business model, they made all their source code to run the platform Free Software. It means they offer a bit less fancy features.

||| **A bit less shiny?**

▣ Yeah, because they have less success and so less money to dedicate to development of the platform. But still it's some kind of easy to grasp web interface management, repositories manager. Which is quite cool. We could do that, to install this kind of interface, to allow more people to have their repositories on the OSP-server. But here comes the difficult thing: we would need a bit more resources to run the server to host a lot of repositories. Still this moment we have problems sometimes with the server because it's not like a large server. Nobody at OSP is really a sysadmin, and has time to install and setup everything nicely etcetc. And we also would have to work on the gitorious web application to make it a bit more in line with our visual universe. Because now it's really some kind of thing we cannot associate with really.

||| **Do you think 'Visual Culture' can leverage some of the success of GitHub? People seem to understand and like working this way.**

▣ Well, it depends. We also meet a lot of people who come to GitHub and say, I don't understand, I don't understand anything of this! Because of it's huge success GitHub can put some extra effort in visualization, and they started to run some small projects. So they can do more than 'Visual Culture' can do.

||| **And is this code available?**

◲ Some of their projects are Open Source.

▣ Some of their projects are free. Even if we have some things going on in 'Visual Culture', we don't have enough manpower to finalize this project. The GitHub interface is really specific, really oriented, they manage to do things like show fonts, show pictures, but I don't think they can display .pdf. 'Visual Culture' is really a good direction, but it can become obsolete by the fact that we don't have enough resource to work on it. GitHub starts to cover a lot of needs, but always in their way of doing things, so it's a problem.

▣ Pierre Marchand
✕ Ludvine Loiseau
||| **Femke Snelting**
▙ Alexandre Leray

# DISTRIBUTED VERSION CONTROL

---

▙ I'm very surprised … the quality of Git is that it isn't centralized, and nowadays everything is becoming centralized in GitHub. I'm also wondering whether … I don't think we should start to host other repositories, or maybe we should, I don't know.

▣ Yeah, I think we should

||| **You do or you don't want to become a hosting platform?**

▙ No. What I think is nice about GitHub is of course the social aspect around sharing code. That they provide comments. Which is an extra layer on top of Git. I'm having fantasies about another group like OSP who would use Git and have their own server, instead of having this big centralized system. But still have ways to interact with each other. But I don't know how.

||| **It would be interesting if it's distributed without being disconnected.**

▣ If it was really easy to setup Git, or a versioning server, that would be fantastic. But I can remember, as a software developer, when I started to look for somewhere to host my code it was no question to setup my own server. Because of not having time, no time to maintain, no time to deploy etcetc. At some point we need hosting-platforms for ourselves. We have almost enough to run our own platform. But think of all the people who can't afford it.

||| **But in a way you are already hosting other people's projects. Because there are quite a few repositories for workshops that actually not belong to you.**

▣ Yeah, but we moved some of them to GitHub just to get rid of the pain of maintaining these repositories.

✕ We wanted the students to be independent. To really have them manage their own projects.

▣ GitHub is easier to manage then our own repository which is still based on a lot of files.

⊏⊐ *For me, if we ever make this hosting platform, it should be something else then our own website. Because, like you say, it's kind of centralized in the way we use it now. It's all on the Constant server.*

▣ Not anymore?

⊏⊐ *No, the Git repositories are still on the Constant server.*

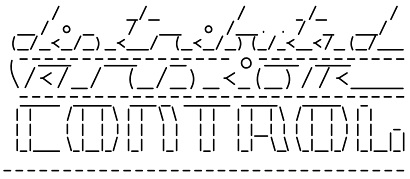
▣ Ah, the Git is still. But they are synced with the OSP server. But still, I can imagine it would be really nice to have many instances of 'Visual Culture' for groups of people running their own repositories.

III **It feels a bit like early days of blogging.**

▣ It would be really, really nice for us to allow other people to use our services. I was also thinking of this, because of this branching stuff. For two reasons, first to make it easier for people to take advantage of our repository. Just like branching our repository would be one click, just like in Gitorious or GitHub. So I have an account and I like this project and I want to change something, I just click on it. You're branched into your own account and you can start to work with it. That's it, and it would be really convenient for people who would like to work with our font files etc. And once we have all these things running on our server we can think of a lot of ideas to promote our own dynamic over versioning systems. But now we're really a bit stuck because we don't have the tools we would like to have. With the repositories, it's something really rigid.

III **It is interesting to see the limits of what actually can happen. But it is still better than the usual (In)design practices?**

✕ We would like to test GitMX. We don't know much about it, but we would like to use it for the pictures in high-resolution, .pdfs. We thought about it when we were in Seoul, because we were putting pictures on a gallery, and we were like *ah, this gallery*. We were wondering, perhaps if GitMX works well, perhaps it can be separated into different types of content. And then we can branch them into websites. And perhaps pictures of the finalized work. In the end we have the 'Iceberg' with a lot of 'in-progress'-pictures,



but we don't have any portfolio or book. Again because we don't care much about this, but at the end we feel we miss it a bit.

||| **A narration ...**

✕ ... to have something to present. Each time we prepare a presentation, we need to start again to find back the tools and files, and to choose what we want to send for the exhibition.

▣ It's really important because at some point, working with Git, I can remember telling people ...

▚ *Don't* `push` *images!*

||| **I remember.**

▣ The repository is there to share the resources. And that's really where it shines. And don't try to put all your active files in it. At some point we miss this space to share those files.

||| **But an image can be a recipe. And code can be an artifact. For me the difference is not so obvious.**

▮ *It is not always so clear. Sometimes the cut-off point is decided by the weight of the file, so if it is too heavy, we avoid Git. Another is: if it is easy to compile, leave it out of Git. Sometimes the logic is reversed. If we need it to be online even if not a source, but simply we need to share it, we put it on the Git. Some commits are also errors. The distinction is quite organic until now, in my experience. The closer the practice gets to code, the more clean the versioning process is.*

||| **There is also a kind of performative part of the repository. Where a commit counts as a proof of something ...**

⊏⊐ *When I presented the OSP's website, we had some remarks like,* ah it's good we can see what everybody has done, who has worked.

▮ *But strangely so far there were not many reactions from partners or clients regarding the fact that all the projects could be followed at any stage. Even budget wise ... Mostly, I think, because they do not really understand how it works.*

⊏⊐ *And sometimes it's true, it came to my mind, should we really show our website to clients? Because they can check whether we are working hard, or this week* *we didn't do shit… And it's, I think it's really based on trust and the type of collaboration you want with your client. Actually collaboration and not a hierarchical relationship. So I think in the end it's something that we have to work on. On building a healthy relationship, that you show the process but it's not about control. The meritocracy of commits is well known, I think, in platforms like GitHub. I don't think in OSP this is really considered at all actually.*

■ *It supports some self-time tracking that is nuanced and enriched by e-mail, calendar events, writing in Etherpads. It gives a feeling of where is the activity without following it too closely. A feeling rather than surveillance or meritocracy.*

■ *I know that Eric… because he doesn't really keep track of his working hours. He made a script to look into his* `commit` *messages to know when he worked on a project. Which is not always truthful. Because sometimes you make a commit on some files that you made last week, but forgot to* `commit`*. And a commit is a text message at a certain time. So it doesn't tell you how much time you spent on the file.*

||| **Although in the way you decided to visualize the commits, there is a sense of duration between the last and the commit before. So you have a sense of how much time passed in between. Are there ways you sometimes trick the system, to make things visible that might otherwise go missing?**

■ In the messages sometimes, we talk about things we tried and didn't work. But it's quite rare.

✕ I kind of regret that I don't write so much on the commits. At the beginning when we decided to publish the messages on the homepage we talked about this theater dialogue and I was really excited. But in the end I see that I don't write as much as I would like.

■ I think it's really a question of the third-party programs we use. Our `commit` messages are like a dialogue on the website. But when you write a `commit` message you're not at all in this interface. So you don't answer to something. If we would have the same kind of interface we have on the website, you would realize you can answer to the previous `commit` message. You have this sort of narrative thread and it would work. We are in the middle, we have this feeling of a dialogue on one side, but because when you work, you're not on the website to check the history. It's just basically, it would be about to make things really in line with what we want to achieve.

I `commit` just when I need to share the files with someone else. So I wait until the last moment.

*To `push` you mean?*

No, to `commit`. And then I've lost track of what I've done and then I just write …

**But it would be interesting, to look at the different speeds of collaboration. They might need each another type of `commit` message.**

*But it's true, I must admit that when I start working on a project I don't read the last messages. And so, then you lose this dialogue as you said. Because sometimes I say,* Ludi is going to work on it. *So I say,* OK Ludi it's your turn now, *but the thing is, if she says that to me I would not know because I don't read the `commit` messages.*

I suppose that is something really missing from the Git client. When you `pull`, you update your working copy to synchronize with the server it just says files change, how many changes there were. But doesn't give you the story.

*That's what missing when you `pull`. It should instead of just showing which files have changed, show all the logs from the last time you `pull`ed.*

Your earlier point, about recipes versus artifacts. I have something to add that I forgot. I would reverse the question, what the versioning system considers to be a recipe is good, is a recipe. I mean, in this context 'a recipe' is something that works well within the versioning system. Such as the description of your process to get somewhere. And I can imagine it's something, I would say the Git community is trying to achieve that fact. Make it something that you can share easily.

*But we had a bit of this discussion with Alex for a reader we made. It is going to be published, so we have the website with all the texts, and the texts are all under* *a free license. But the publisher doesn't want us to put the .pdfs online. I'm quite okay with that, because for me it's a condition that we put the sources online. But if you really want the .pdf then you can `clone` the repository and make them yourself in Scribus. It's just an example of not putting the .pdf, but you have everything you need to make the .pdf yourself. For me it's quite interesting to say our sources are there. You can buy the book but if you want the .pdf you have to make a small effort to generate it and then you can distribute it freely. But I find it quite interesting to, of course the easiest way would be the .pdf but in this case we can't. Because the publisher doesn't want us to.*

**But that distinction somehow undervalues the fact that layout for example is not just an executed recipe, no? I mean, so there is this kind of grey area in design that is … maybe not the final result, but also not a sort of executable code.**

We see it with 'Visual Culture', for instance, because Git doesn't make it easy to work with binaries. And the point of 'Visual Culture' is to make .jpegs visible and all the kind of graphical files we work with. So it's like we don't know how to decide whether we should put for instance .pdfs in the Git repository online. Because on the one hand it makes it less manageable with Git to work with. But on the other hand we want to make things visible on the website.

*But it's also storage-space. If you want to `clone` it, if you want people to `clone` it also you don't want a 8 gigabyte repository.*

I don't know because it's not really what OSP is for, but you can imagine, like Dropbox has been made to easily share large files, or even files in general. We can imagine that another company will set up something, especially graphic designers or the graphic industry. The way GitHub did something for the development industry. They will come up with solutions for this very problem.

*I just want to say that I think because we're not a developer group, at the start the `commit` messages were a space where you would throw all your anger, frustration. And we first published a Git log in the Balsamine program, because we saw that. This was the first program we designed with ConTeXt. So we were manipulating* *code for layout. The* `commit` *messages were all really funny, because Pierre and Ludi come from a non-coding world and it was really inspiring and we decided to put it in the publication. Then we kind of looked, Ludi says two kind of bad things about the client, but it was okay. Now I think we are more aware that it's public, we kind of pay attention not to say stuff we don't mean to …*

**It's not such an exciting space anymore as in the first half year?**

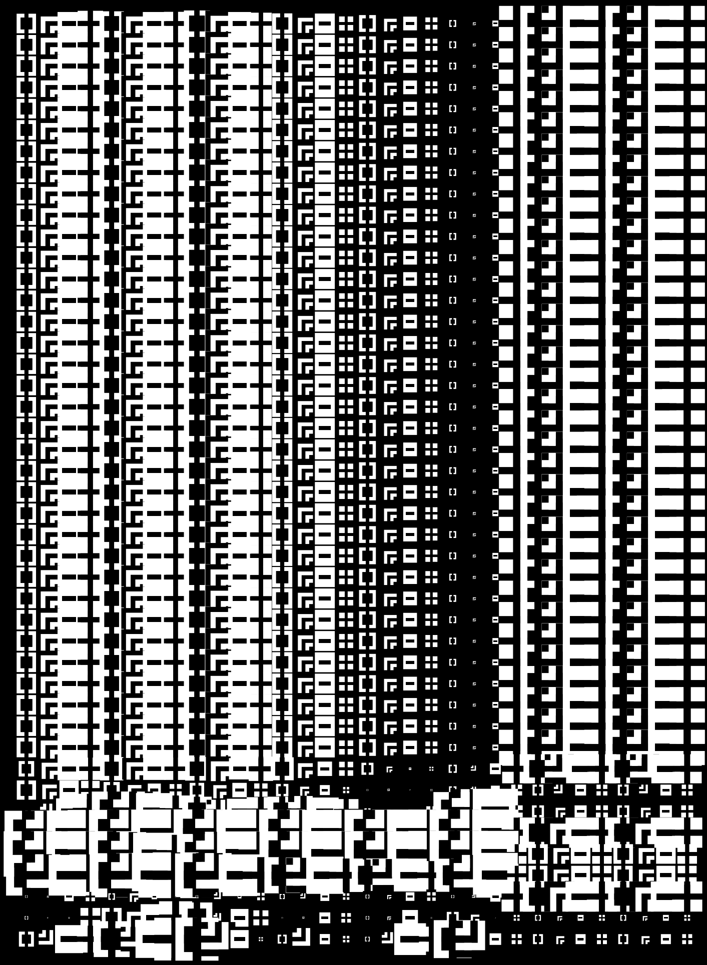
It often very formal and not very, exciting, I think. But sometimes I put quite some effort to just make clear what I'm trying to share.

*And there are also commits that you make for yourself. Because sometimes, even if you work on a project alone, you still do a Git project to keep track, to have a history to come back to. Then you write to yourself. I think it's also something else. I've never tried it.*

**It's a lot to ask in a way, to write about what you are doing while you are doing it.**

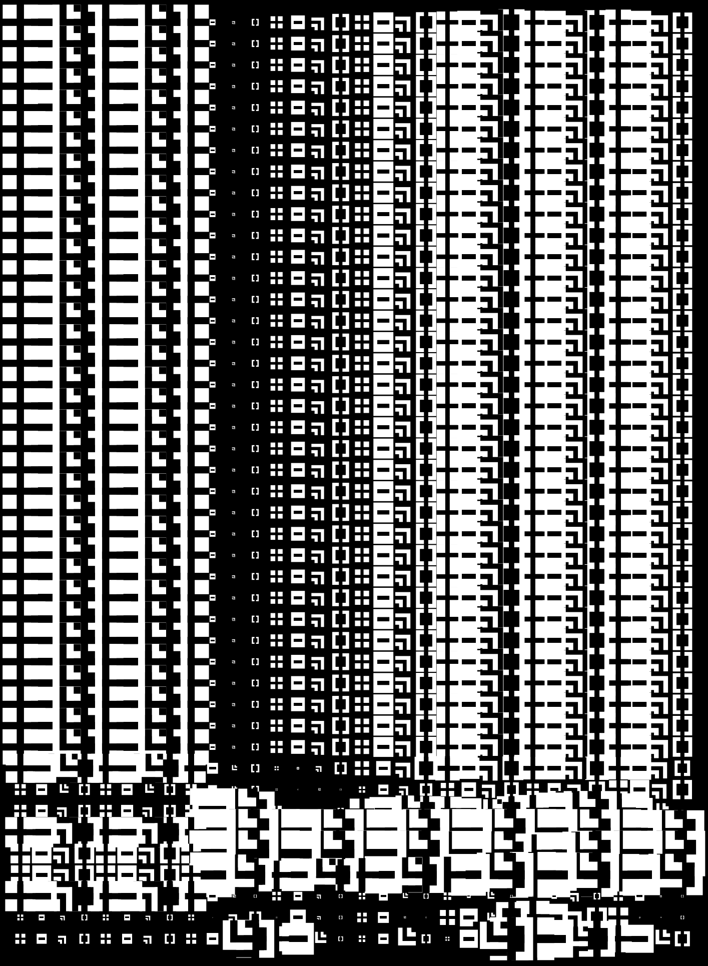
I think we should pay more attention to the first commit of a project, and the last. Because it's really important to start the story and to end it. I speak about this 'end' because I feel overflowed by all these not-ended projects, I'm quite tired of it. I would like us to find a way to archive projects which are not alive any more. To find a good way to do it. Because the list of folders is still growing, and in a way it is okay but a lot of projects are not active.

*But it's hard to know when is the last commit. With the Balsamine project it's quite clear, because it's season per season. But still, we never know when it is the last one. The last one could be solved by the 'Iceberg', to make the last snapshots and say okay now we make the screenshots of the latest version. And then you close it … We wanted that the last one was* Hey, we sent the .pdfs to the printer. *But actually we had to send it back another time because there was a mistake. And then the log didn't fit on the page anymore.*



# Even when you are done, you are not done

[] Chris Lilley
▬ Nicolas Malevé
╔ Femke Snelting
∷ Pierre Huyghebaert



Even when you are done,
you are not done

▯ Chris Lilley
▬ Nicolas Malevé
╒ Femke Snelting
∷ Pierre Huyghebaert

At the **Libre Graphics Meeting 2008**, OSP sat down with Chris Lilley on a small patch of grass in front of the Technical University in Wroclaw, Poland. Warmed up by the early May sun, we talked about the way standards are made, how 'specs' influence the work of designers, programmers and managers and how this process is opening up to voices from outside the W3C. Chris Lilley is trained as a biochemist, and specialised in the application of biological computing. He has been involved with the World Wide Web Consortium since the 1990s, headed the Scalable Vector Graphics (SVG) working group and currently looks after two W3C activity areas: graphics, including PNG, CGM, graphical quality, and fonts, including font formats, delivery, and availability of font software.

***I would like to ask you about the way standards are made … I think there's a relation between the way Free, Libre and Open Source software works, and how standards work. But I am particularly interested in your announcement in your talk today that you want to make the process of defining the SVG standard a public process?***

Right. So, there's a famous quote that says that standards are like sausages. Your enjoyment of them is improved by not knowing how they're made.[1] And to some extent, depending on the standards body and depending on what you're trying to standardize, the process can be very messy. If you were to describe W3C as a business proposition, it has got to fail. You're taking companies who all have commercial interests, who are competing and you're putting them in the same room and getting them to talk together and agree on something. Oddly, sometimes that works! You can sell them the idea that growing the market is more important and is going to get them more money. The other way … is that you just make sure that you get the managers to sign, so that their engineers can come and discuss standards,

[1] *Laws are like sausages. It's better not to see them being made.* Otto von Bismarck, 1815–1898

and then you get the engineers to talk and the managers are out of the way. Engineers are much more forthcoming, because they are more interested in sharing stuff because engineers like to share what they're doing, and talk on a technical level. The worst thing is to get the managers involved, and even worse is to get lawyers involved. W3C does actually have all those three in the process. *Shall we do this work or not* is a managerial level that's handled by the W3C advisory committee, and that's where some people say *No, don't work on that area* or *We have patents* or *This is a bad idea* or whatever. But often it goes through and then the engineers basically talk about it. Occasionally there will be patents disclosed, so the W3C also has a process for that. The first things are done are the 'charters'. The charter says what the group is going to work on a broad scope. As soon as you've got your first draft, that further defines the scope, but it also triggers what it's called an exclusion opportunity, which basically gives the companies I think ninety days to either declare that they have a specific patent and say what it's number is and say that they exclude it, or not. And if they don't, they've just given a royalty-free licence to whatever is needed to implement that spec. The interesting thing is that if they give the royalty-free licence they don't have to say which patents they're licencing. Other standards organizations build up a patent portfolio, and they list all these patents and they say what you have to licence. W3C doesn't do that, unless they've excluded it which means you have to work around it or something like that. Based on what the spec says, all the patents that have been given, are given. The engineers don't have to care. That's the nice thing. The engineers can just work away, and unless someone waves a red flag, you just get on with it, and at the end of the day, it's a royalty-free specification.

***But if you look at the SVG standard, you could say that it's been quite a bumpy road[2] ... What kind of work do you need to do to make a successful standard?***

Firstly, you need to agree on what you're building, which isn't always firm and sometimes it can change. For example, when SVG was started the idea was that it would be just static graphics. And also that it would be animated

[2] http://ospublish.constantvzw.org/news/whos-afraid-of-adobe-not-me-says-the-mozilla-foundation

using scripts, because with dynamic HTML and whatever, this was '98, we were like: *OK, we're going to use scripting to do this.* But when we put it out for a first round of feedback, people were like *No! No, this is not good enough. We want to have something declarative. We don't want to have to write a script every time we want something to move or change color.* Some of the feedback, from Macromedia for example was like *No, we don't think it should have this facility,* but it quickly became clear why they were saying that and what technology they would rather use instead for anything that moved or did anything useful ... We basically said *That's not a technical comment, that's a marketing comment, and thank you very much.*

***Wait a second. How do you make a clear distinction between marketing and technical comments?***

People can make proposals that say *We shouldn't work on this, we shouldn't work on that*, but they're evaluated at a technical level. If it's *Don't do it like that because it's going to break as follows, here I demonstrate it* then that's fine. If they're like *Don't do it because that competes with my proprietary product* then it's like *Thanks for the information, but we don't actually care.* It's not our problem to care about that. It's your problem to care about that. Part of it is sharing with the working group and getting the group to work together, which requires constant effort, but it's no different from any sort of managerial or trust company type thing. There's this sort of encouragement in it that at the end of the day you're making the world a better place. You're building a new thing and people will use it and whatever. And that is quite motivating. You need the motivation because it takes a lot longer than you think. You build the first spec and it looks pretty good and you publish it and you smooth it out a bit, put it out for comments and you get a ton of comments back. People say *If you combine this with this with this then that's not going to work.* And you go *Is anyone really going to do that?* But you still have to say what happens. The computer still has to know what happens even if they do that. Ninety percent of the work is after the first draft, and it's really polishing it down. In the W3C process, once you get to a certain level, you take it to what is euphemistically called the 'last call'. This is a term we got from the IETF.[3] It actually means 'first call' because

[3] The Internet Engineering Task Force, http://www.ietf.org/

you never have just one. It's basically a formal round of comments. You log every single comment that's been made, you respond to them all, people can make an official objection if you haven't responded to the comment correctly etcetera. Then you publish a list of what changes you've made as a basis of that.

***What part of the SVG standardization process would you like to make public?***

The part that I just said has always been public. W3C publishes specifications on a regular basis, and these are always public and freely available. The comments are made in public and responded to in public. What hasn't been public has been the internal discussions of the group. Sometimes it can take a long time if you've got a lot of comments to process or if there's a lot of argumentation in the group: people not agreeing on the direction to go, it can take a while. From the outside it looks like nothing is happening. Some people like to follow this at a very detailed level, and blog about it, and *blablabla*. Overtime, more and more working groups have become public. The SVG group just recently got recharted and it's now a public group. All of its minutes are public. We meet for ninety minutes twice a week on a telephone call. There's an IRC log of that and the minutes are published from that, and that's all public now.[4]

***Could you describe such a ninety minute meeting for us?***

There are two chairs. I used to be the chair for eight years or so, and then I stepped down. We've got two new chairs. One of them is Erik Dahlström from Opera, and one of them is Andrew Emmons from Bitflash. Both are SVG implementing companies. Opera on the desktop and mobile, and Bitflash is just on mobile. They will set out an agenda ahead of time and say *We will talk about the following issues.* We have an issue tracker, we have an action tracker which is also now public. They will be going through the actions of people saying *I'm done* and discussing whether they're actually done or not. Particular issues will be listed on the agenda to talk about and to have to agree on, and then if we agree on it and you have to change the spec as a result, someone will get an action to change that back to the

[4] Scalable Vector Graphics (SVG) Feedback Page:
http://www.w3.org/Graphics/SVG/feedback.html

spec. The spec is held into CVS so anyone in the working group can edit it and there is a commit log of changes. When anyone accidentally broke something or trampled onto someone else's edit, or whatever - which does happen - or if it came as the result of a public comment, then there will be a response back saying we have changed the spec in the following way … *Is this acceptable? Does this answer your comment?*

***How many people do take part in such a meeting?***

In the working group itself there are about 20 members and about 8 or so who regularly turn up, every week for years. You know, you lose some people over time. They get all enthusiastic and after two years, when you are not done, they go off and do something else, which is human nature. But there have been people who have been going forever. That's what you need actually in a spec, you need a lot of stamina to see it through. It is a long term process. Even when you are done, you are not done because you've got errata, you've got revisions, you've got requests for new functionalities to make it into the next version and so on.

***On the one hand you could say every setting of a standard is a violent process, some organisation forcing a standard upon others, but the process you describe is entirely based on consensus.***

There's another good quote. Tim Berners Lee was asked why W3C works by consensus, rather than by voting and he said: *W3C is a consensus-based organisation because I say so, damn it.*[5] That's the Inventor of the Web, you know … (*laughs*) If you have something in a spec because 51% of the people thought it was a good idea, you don't end up with a design, you end up with a bureaucratic type decision thing. So yes, the idea is to work by consensus. But consensus is defined as: 'no articulated dissent' so someone can say 'abstain' or whatever and that's fine. But we don't really do it on a voting basis, because if you do it like that, then you get people trying to

[5] Consensus is a core value of W3C. To promote consensus, the W3C process requires Chairs to ensure that groups consider all legitimate views and objections, and endeavor to resolve them, whether these views and objections are expressed by the active participants of the group or by others (e.g., another W3C group, a group in another organization, or the general public). World Wide Web Consortium. General Policies for W3C Groups, 2005. [Online; accessed 30.12.2014]

make voting blocks and convince other people to vote their way … it is much better when it is done on the basis of a technical discussion, I mean … you either convince people or you don't.

***If you read about why this kind of work is done … you find different arguments. From enhancing global markets to: 'in this way, we will create a better world for everyone'. In Tim Berners-Lee's statements, these two are often mixed. If you for example look at the DIN standards, they are unambiguously put into the world as to help and support business. With Web Standards and SVG, what is your position?***

Yes. So, basically … the story we tell depends on who we are telling it to and who is listening and why we want to convince them. Which I hope is not as duplicitous as it may sound. Basically, if you try to convince a manager that you want 20% time of an engineer for the coming two years, you are telling them things to convince them. Which is not untrue necessarily, but that is the focus they want. If you are talking to designers, you are telling them how that is going to help them when this thing becomes a spec, and the fact that they can use this on multiple platforms, and whatever. Remember: when the web came out, to exchange any document other than plain text was extremely difficult. It meant exchanging word processor formats, and you had to know on what platform you were on and in what version. The idea that you might get interoperability, and that the Mac and the PC could exchange characters that were outside ASCII was just pie in the sky stuff. When we started, the whole interoperability and cross-platform thing was pretty novel and an untested idea essentially. Now it has become pretty much solid. We have got a lot of focus on disabled accessibility, and also internationalization which is if you like another type of accessibility. It would be very easy for an organisation like W3C, which is essentially funded by companies joining it, and therefore they come from technological countries … it would be very easy to focus on only those countries and then produce specifications that are completely unusable in other areas of the world. Which still does sometimes happen. This is one of the useful things of the W3C. There is the internationalization review, and an accessibility review and nowadays also a mobile accessible review to make sure it does not just work on desktops. Some organisations make standards basically so they can make money. Some of the ISO[6] standards, in particular the MPEG group, their business model is that you contribute an engineer for a couple of years, you make a patent portfolio and you make a killing off licencing it. That is pretty much to keep out the people who were not involved in the standards process. Now, W3C takes quite an opposite view. The Royalty-Free License[7] for example, explicitly says: royalty-free to all. Not just the companies who were involved in making it, not just companies, but anyone. Individuals. Open Source projects. So, the funding model of the W3C is that members pay money, and that pays our salaries, basically. We have a staff of 60 odd or so, and that's where our salaries come from, which actually makes us quite different from a lot of other organisations. IETF is completely volunteer based so you don't know how long something is going to take. It might be quick, it might be 20 years, you don't know. ISO is a national body largely, but the national bodies are in practice companies who represent that nation. But in W3C, it's companies who are paying to be members. And therefore, when it started there was this idea of secrecy. Basically, giving them something for their money. That's the trick, to make them believe they are getting something for their money. A lot of the ideas for W3C came from the X Consortium[8] actually, it is the same people who did it originally. And there, what the meat was ... was the code. They would develop the code and give it to the members of the X Consortium three months before the public got it and that was their business benefit. So that is actually where our 'three month rule' comes from. Each working group can work for three months but then they have to go public, have to publish. 'The heartbeat rule', we call it now. If you miss several heartbeats then you're dead. But at the same time if you're making a spec and you're growing the market then there's a need for it to be implemented. There's an implementation page where you encourage people to implement, you report back on the implementations,

[6] *International Standards for Business, Government and Society* International Organization for Standardization (ISO), http://www.iso.org

[7] *Overview and Summary of W3C Patent Policy* http://www.w3.org/2004/02/05-patentsummary.html

[8] *The purpose of the X Consortium was to foster the development, evolution, and maintenance of the X Window System, a comprehensive set of vendor-neutral, system-architecture neutral, network-transparent windowing and user interface standards.* http://www.x.org/wiki/XConsortium

you make a test suite, you show that every feature in the spec that there's a test for … at least two implementations pass it. You're not showing that everyone can use it at that stage. You're showing that someone can read the spec and implement it. If you've been talking to a group of people for four years, you have a shared understanding with them and it could be that the spec isn't understandable without that. The implementation phase lets you find out that people can actually implement it just by reading the spec. And often there are changes and clarifications made at that point. Obviously one of the good ways to get something implemented is to have Open Source people do it and often they're much more motivated to do it. For them it's cool when it is new, *If you give me this new feature it's great we'll do it* rather than: *Well that doesn't quite fit into our product plans until the next quarter* and all that sort of stuff. Up until now, there hasn't really been a good way for the Open Source people to get involved. They can comment on specs but they're not involved in the discussions. That's something we're trying to change by opening up the groups, to make it easier for an Open Source group to contribute on an ongoing basis if they want to. Right from the beginning part, to the end where you're polishing the tiny details in the corner.

***I think the story of web fonts shows how an involvement of the Open Source people could have made a difference.***

When web fonts were first designed, essentially you had Adobe and Apple pushing one way, Bitstream pushing the other way, both wanting W3C to make their format the one and only official web format, which is why you ended up with a mechanism to point to fonts without saying what format was required. And than you had the Netscape 4, which pointed off to a Bitstream format, and you had IE4 which pointed off to this Embedded Open Type (EOT) format. If you were a web designer, you had to have two different tools, one of which only worked on a Mac, and one of which only worked on PC, and make two different fonts for the same thing. Basically people wouldn't bother. As Håkon[9] mentioned the only people who do actually use that right now really, are countries where the local language

[9] Håkon Wium Lie proposed Cascading Style Sheets (CSS) in 1994.
http://www.w3.org/People/howcome/

is not well provided for by the Operating Systems. Even now, things like WindowsXP and MacOSX don't fully support some of the Indian languages. But they can get it into web pages by using these embedded fonts. Actually the other case where it has been used a lot, is SVG, not so much on the desktop though it does get used there but on mobiles. On the desktop you've typically got 10 or 20 fonts and you got a reasonable coverage. On a mobile phone, depending on how high or low ended it is, you might have a single font, and no **bold**, and it might even be a pixel-based font. And if you want to start doing text that skews and swirls, you just can't do that with a pixel-based font. So you need to download the font with the content, or even put the font right there in the content just so that they can see something.

***I don't know how to talk about this, but… envisioning a standard before having any concrete sense of how it could be used and how it could change the way people work… means you also need to imagine how a standard might change, once people start implementing it?***

I wouldn't say that we have no idea of how it's going to work. It's more a case that there are obvious choices you can make, and then not so obvious choices. When work is started, there's always an idea of how it would fit in with a lot of things and what it could be used for. It's more the case that you later find that there are other things that you didn't think of that you can also use it for. Usually it is defined for a particular purpose and than find that it can also do these other things.

*Isn't it so that sometimes, in that way, something that is completely marginal, becomes the most important?*

It can happen, yes.

**For me, SVG is a good example of that. As I understood it, it was planned to be a format for the web. And as I see it today, it's more used on the desktop. I see that on the Linux desktop, for theming, most internals are using SVG. We are using Inkscape for SVG to make prints. On the other hand, browsers are really behind.**

Browsers are getting there. Safari has got reasonably good support. Opera has got very good support. It really has increased a lot in the last couple of years. Mozilla Firefox less so. It's getting there. They've been at it for longer, but it also seems to be going slower. The browsers are getting there. The implementations which I showed a couple of days ago, those were mobile implementations. I was showing them on a PC, but they were specially built demos. Because they're mobile, it tends to move faster.

**But you still have this problem that Internet Explorer is a slow adopter.**

Yes, Internet Explorer has not adopted a lot of things. It's been very slow to do CSS. It hasn't yet done XHTML, although it has shipped with an XML parser since IE4. It hasn't done SVG. Now they've got their own thing… Silverlight. It has been very hard to get Microsoft on board and getting them doing things. Microsoft were involved in the early part of SVG but getting things into IE has always been difficult. What amazes me to some extent, is the fact that it's still used by about 60-70% of people. You look at what IE can do, and you look at what all the other browsers can do, and you wonder why. The thing is… it is still a break and some technologies don't get used because people want to make sure that everyone can see them. So they go down to the lowest common denominator. Or they double-implement. Implement something for all the other browsers, and implement something separate for IE, and than have to maintain two different things in parallel, and tracking revisions and whatever. It's a nightmare. It's a huge economic cost because one browser doesn't implement the right web stuff. (*laughing, sighing*)

**My question would be: what could you give us as a kind of advice? How could we push this adoption where we are working? Even if it only is the people of Firefox to adopt SVG?**

Bear in mind that Firefox has this thing of Trunk builds and Branch builds and so on. For example when Firefox 3 came out, well the Beta is there. Suddenly there's a big jump in the SVG stuff because all the Firefox 2 was on the same branch as 1.5, and the SVG was basically frozen at that point. The development was ongoing but you only saw it when 3 came out. There were a bunch of improvements there. The main missing features are the animation and the web fonts and both of those are being worked on. It's interesting because both of those were on Acid 3. Often I see an acceleration of interest in getting something done because there's a good test. The Acid Test[10] is interesting because it's a single test for a huge slew of things all at once. One person can look at it, and it's either right or it's wrong, whereas the tests that W3C normally produces are very much like unit tests. You test one thing and there's like five hundred of them. And you have to go through, one after another. There's a certain type of person who can sit through five hundred test on four browsers without getting bored but most people don't. There's a need for this sort of aggregative test. The whole thing is all one. If anything is wrong, it breaks. That's what Acid is designed to do. If you get one thing wrong, everything is all over the place. Acid 3 was a submission-based process and like a competition, the SVG working group was there, and put in several proposals for what should be in Acid 3, many of which were actually adopted. So there's SVG stuff in Acid 3.

***So … who started the Acid Test?***

Todd Fahrner designed the original Acid 1 test, which was meant to exercise the tricky bits of the box-model in CSS. It ended like a sort Mondrian diagram,[11] red squares, and blue lines and stuff. But there was a big scope for the whole thing to fall apart into a train wreck if you got anything wrong. The thing is, a lot of web documents are pretty simple. They got paragraphs, and headings and stuff. They weren't exercising very much the model. Once you got tables in there, they were doing it a little bit more. But it was really when you had stuff floated to one side, and things going around or whatever, and that had something floated as well. It was in that sort of case where it was all breaking, where people wouldn't get interoperability.

***It was … the Web Standards Project[12] who proposed this?***

Yes, that's right.

[10] The Acid 3 test: http://acid3.acidtests.org is comprehensive in comparison to more detailed, but fragmented SVG tests: http://www.w3.org/Graphics/SVG/WG/wiki/Test_Suite_Overview#W3C_Scalable_Vector_Graphics_.28SVG.29_Test

[11] *Acid Test Gallery* http://moonbase.rydia.net/mental/writings/box-acid-test/

[12] *The Web Standards Project is a grassroots coalition fighting for standards which ensure simple, affordable access to web technologies for all* http://www.webstandards.org/

***It didn't come from a standards body.***

No, it didn't come from W3C. The same for Acid 2, Håkon Wium Lie was involved in that one. He didn't blow his own trumpet this morning, but he was very much involved there. Acid 3 was Ian Hickson, who put that together. It's a bit different because a lot of it is DOM scripting stuff. It does something, and then it inquires in the DOM to see if it has been done correctly, and it puts that value back as a visual representation so you can see. It's all very good because apparently it motivates the implementors to do something. It's also marketable. You can have a blog posting saying we do 80% of Acid Test. The public can understand that. The people who are interested can go *Oh, that's good*.

***It becomes a mark of quality.***

Yes, it's marketing. It's like processor speed in PCs and things. There are so much technology in computers, so than what do you market it on? Well it's got that clock speed and it's got this much memory. OK, great, cool. This one is better than that one because this one's got 4 gigs and that one's got 2 gigs. It's a lot of other things as well, but that's something that the public can in general look at and say *That one is better*. When I mentioned the W3C process, I was talking about the engineers, managers. I didn't talk about the lawyers, but we do have a process for that as well. We have a patent advisory group conformed. If someone has made a claim, and it's disputed then we can have lawyers talking among themselves. What we really don't have in that is designers, end-users, artists. The trick is to find out how to represent them. The CSS working group tried to do that. They brought in a number of designers, Jeff Veen[13] and these sort of people were involved early on. The trouble is that you're speaking a different language, you're not speaking their language. When you're having weekly calls … Reading a spec is not bedtime reading, and if you're arguing over the fine details of a sentence … (*laughing*) well, it will put you to sleep straight away. Some of the designers are like: *I don't care about this. I only want to use it. Here's what I want to be able to do. Make it that I can do that, but get back to me when it's done.*

[13] Jeff Veen was a designer at Wired magazine, in those days.
http://adaptivepath.com/aboutus/veen.php

**That's why the idea of the Acid Test is a nice breed between the spec and the designer. When I was seeing the test this morning, I was thinking that it could be a really interesting work to do, not to really implement it but to think about with the students. How would you conceive a visual test? I think that this could be a really nice workshop to do in a university or in a design academy ...**

***It's the kind of reverse-reverse engineering of a standard which could help you understand it on different levels. You have to imagine how wild you can go with something. I talk about standards, and read them - not before going to bed - because I think that it's interesting to see that while they're quite pragmatic in how they're put together, but they have an effect on the practice of, for example, designers. Something that I have been following with interest is the concept of separating form and content has become extremely influential in design, especially in web design. Trained as a pre-web designer, I'm sometimes a bit shocked by the ease with which this separation is made.***

That's interesting. Usually people say that it's hard or impossible, that you can't ever do it. The fact that you're saying that it's easy or that it comes naturally is interesting to me.

***It has been appropriated by designers as something they want. That's why it's interesting to look at the Web Standards Project where designers really fight for a separation of content and form. I think that this is somehow making the work of designers quite ... boring. Could you talk a bit about how this is done?***

It's a continuum. You can't say that something is exactly form or exactly presentation because there are gradations. If you take a table, you've already decided that you want to display the material in a tabular way. If it's a real table, you should be able to transpose it. If you take the rows and columns, and the numbers in the middle then it should still work. If you've got 'sales' here and if you've got 'regions' there, then you should still be able to transpose that table. If you're just flipping it 90 degrees then you are using it as a layout grid, and not as a table. That's one obvious thing. Even then, deciding to display it as a tabular thing means that it probably came from a much bigger dataset, and you've just chosen to sum all of the sales data over one year. Another one: you have again the sales data, you could have it as pie chart, but you could also have it as a bar chart, you could have it in various other ways. You can imagine that what you would do is ship some XML that has that data, and then you would have a script or something which would turn it into an SVG pie chart. And you could have a bar chart, or you could also say show me only February. That interaction is one of the things that one can do, and arguably you're giving it a different presentational form. It's still very much a gradation. It's how much re-styleability remains. You can't ever have complete separation. If I'm describing a company, and [**1**] I want to do a marketing brochure, and [**2**] I want to do an annual report for the shareholders, and [**3**] I want to do an internal document for the engineering team. I can't have the same content all over those three and just put styling on it. The type of thing I'm doing is going to vary for those audiences, as will the presentation. There's a limit. You can't say: here's the überdocument, and it can be styled to be anything. It can't be. The trick is to not mingle the style of the presentation when you don't need to. When you do need to, you're already halfway down the gradient. Keep them as far apart as you can, delay it as late as possible. At some point they have to be combined. A design will have to go into the crafting of the wording, how much wording, what voice is used, how it's going to fit with the graphics and so on. You can't just slap random things together and call it design, it looks like a train wreck. It's a case of deferment. It's not ever a case of complete separation. It's a case of deferring it and not tripping yourself up. Just simple things like **bolds** and *italics* and whatever. Putting those in as emphasis and whatever because you might choose to have your emphasized words done differently. You might have a different font, you might have a different way of doing it, you might use letter-spacing, etc. Whereas if you tag that in as *italics* then you've only got *italics*, right? It's a simple example but at the end of the day you're going to have to decide how that is displayed. You mentioned print. In print no one sees the intermediate result. You see ink on paper. If I have some Greek in there and if I've done that by actually typing in Latin letters on the keyboard and putting a Greek font on it and out comes Greek, nobody knows. If it's a book that's being translated, there might be some problems. The more you're shipping the electronic version around, the more it actually matters that you put in the Greek letters as

Greek because you will want to revise it. It matters that you have flowing text rather than text that has been hand-ragged because when you put in the revisions you're going to have to re-rag the entire thing or you can just say re-flow and fix it up later. Things like that.

▪▪ *The idea of time, and the question of delay is interesting. Not how, but when you enter to fine-tune things manually. As a designer of books, you're always facing the question: when to edit, what, and on what level. For example, we saw this morning*[14] *that the idea of having multiple skins is really entering the publishing business, as an idea of creativity. But that's not the point, or not the complete point. When is it possible to enter the process? That's something that I think we have to develop, to think about.*

▬ **The other day there was a presentation by Michael Dominic Kostrzewa**[15] **that shocked me. He is now working for Nokia, after working for Novell and he was explaining how designers and programmers were fighting each other instead of fighting the 'real villain', as he said, who were the managers. What was really interesting was how this division between content and style was also recouping a kind of political or socio-organizational divide within companies where you need to assign roles, borders, responsibilities to different people. What was really frightening from the talk was that you understood that this division was encouraging people not to try and learn from each other's practice. At some point, the designer would come to the programmer and say: *In the spec, this is supposed to be like this and I don't want to hear anything about what kind of technical problems you face.***

▪▪ *Designers as lawyers!*

▬ **Yes … and the programmer would say: *OK, we respect the spec, but then we don't expect anything else from us.* This kind of behaviour in the end, blocks a lot of exchange, instead of making a more creative approach possible.**

[14] Andy Fitsimon: Publican, the new Open Source publishing tool-chain (LGM 2008) http://media.river-valley.tv/conferences/lgm2008/quicktime/0201-Andy_Fitzsimon.html

[15] Michael Dominic Kostrzewa. Programmers hell: working with the UI designer (LGM 2008)

even when you are done
you are not done

[] I read about (and this is before skinning became more common) designers doing some multimedia things at Microsoft. You had designers and then there were coders. Each of them hated the other ones. The coders thought the designers were idiots who lived in lofts and had found objects in their ears. The designers thought that the programmers were a bunch of socially inept nerds who had no clue and never got out in sunlight and slept in their offices. And since they had that dynamic, they would never explain to each other ( … )

*(policeman arrives)*

POLICEMAN:
*Do you speak English?*

■■ *Yes.*

POLICEMAN:
*You must go from this place because there's a conference.*

[] Yes, we know. We are part of this conference (shows LGM badge).

POLICEMAN:
*We had a phone call that here's a picnic. I don't really see a picnic …*

■■ *We're doing an interview.*

POLICEMAN:
*It looks like a picnic, and professors are getting nervous. You must go sit somewhere else. Sorry, it is the rules. Have a nice day!*

# Having the tools is just the beginning

Dave Crossland | Femke Snelting
Ludivine Loiseau | Harrison
Pierre Huyghebaert

# Having the tools is just the beginning

Dave Crossland | Femke Snelting
Ludivine Loiseau | Harrison
Pierre Huyghebaert

At the **Libre Graphics Meeting 2008**, OSP picks up a conversation that Harrison allegedly started in a taxi in Montreal, a year earlier. We meet font designer and developer Dave Crossland in a noisy food court to speak about his understanding of the intertwined histories of typography and software, and the master in type design at the Department of Typography at the University of Reading. Since the interview, a lot has happened. Dave finished his typeface Cantarell and moved on to consult the Google Web Fonts project, commissioning new typefaces designed for the web. He is also currently offering lectures on typeface design with Free Software.

**Harrison** (H) *1, 2.*

**Ludivine Loiseau** (LL) *Hello Dave.*

and now all: Hellooo…

**Dave Crossland** (DC) Alright!

H *Well, thank you for taking a bit of time with us for the interview. First thing is maybe to set a kind of context of your situation, your current situation. What you've done before. Why are you setting fonts and these kind of things.*

DC Oh yes, yeah. Well, I take it quite far back, when I was a teenager. I was planning to do computer science university studying like mathematics and physics in highschool. I needed some work experience. I decided I didn't want to work with computers. So I dropped maths and physics and I started working at… I mean I started studying art and design, and also socio-linguistics in highschool. I was looking at going to Fine Arts but I wasn't really too worried about if I could get a job at the end of it, because I could get a job with computers, if I needed to get a job So I studied that at my school for like a one year course, after my school. A foundation year, and the deal with that is that you study all the different art and design disciplines. Because in highschool you don't really have the specialities where you specifically study textile or photography, not every school has a darkroom, schools are not well equipped.

You get to experience all these areas of design and in that we studied graphic design, motion graphics and I found in this a good opportunity to bring together the computer things with fine arts and visual arts aspects. In graphic design in my school it was more about paper, it had nothing to do with computers. In art school, that was more the case. So I grew into graphic design.

*Ordering coffee and change of background music: Oh yeah, African beats!*

So, yes. I was looking at graphic design that was more computer based than in art school. I wasn't so interested in like regular illustration as a graphic design. Graphic design has really got three purposes: to persuade people, that's advertising; to entertain people, movie posters, music album covers, illustration magazines; and there is also graphic design to inform people, in England it's called 'information design', in the US it's called 'information architecture'… stucturing websites, information design. Obviously a big part of that is typography, so that's why I got interested in typography, via information design. I studied at Ravensbourne college in London, what I applied for was graphic information design. I started working at the IT department, and that really kept me going to that college, I wasn't so happy with the direction of the courses. The IT department there was really really good and I ended up switching to the interaction design course, because that had more freedom to do the kind of typographic work I was intersted in.
So I ended up looking at Free Sofware design tools because I became frustrated by the limitations of the Adobe software which in the college was using, just what everybody used. And at that point I realized what 'software freedom' meant. I've been using Debian since I was like a teenager, but I hadn't really looked to the depth of what Free Software was about. I mean back in the nineties Windows wasn't very good but probably at that time 2003-2004, MacOSX came out and it was getting pretty nice to use. I bought a Mac laptop without really thinking about it and because it was a Unix I could use the software like I was used to do. And I didn't really think about the issues with Free Software, MacOSX was Unix so it was the same I figured. But when I started to do my work I really stood against the limitations of Adobe software, specifically in parallel publishing which is when you have the same basic informations that you want to communicate in different mediums. You might want to publish something in .pdf, on the web, maybe also on your mobile phone, etc. And doing that with Adobe

software back then was basically impossible. I was aware of Free Software design tools and it was kind of obvious that even if they weren't very pushed by then they at least had the potential to be able to do this in a powerful way. So that's what I figured out. What that issue with Free Software really meant. Who's in control of the software, who decides what it does, who decides when it's going to support this feature or that feature, because the features that I wanted, Adobe wasn't planning to add them. So that's how I got interested in Free Software.
When I graduated I was looking for something that I could contribute in this area. And one of the Scribus guys, Peter Linnell, made an important post on the Scribus blog. Saying, you know, the number one problem with Free Software design is fonts, like it's dodgy fonts with incorrect this, incorrect that, have problems when printed as well … and so yeah, I felt woa, I have a background in typography and I know about Free Software, I could make contributions in fonts. Looking into that area, I found that there was some postgraduate course you can study at in Europe. There's two, there is one at The Hague in The Netherlands and one at Reading. They're quite different courses in their character and in how much they cost and how long they last for and what level of qualification they are. But they're both postgraduate courses which focus on typeface design and font software development. So if you're interesed in that area, you can really concentrate for about a year and bring your skills up to a high professional level. So I applied to the course at Reading and I was accepted there and I'm currently studying there part time. I'm studying there to work on Free Software fonts. So that's the full story of how I ended up in this area.

H *Excellent! Last time we met, you summarized in a very relevant way the history of font design software which is a proof by itself that everything is related with fonts and this kind of small networks and I would like you to summarize it again.*

*L a u g h i n g*

DC Alright. In that whole journey of getting into this area of parallel publishing and automated design, I was asking around for people who worked in that area because at that time not many people had worked in parallel publishing. It's a lot of a bigger deal now, especially in the Free Software community where we have Free Software manuals translated into many languages, written in .doc and .xml and then transformed into print and web versions and other versions. But back then this was kind of a new concept, not all people worked on it. And so, asking around, I heard about the department of typography at the university of Reading. One of the lecturers there, actually the lecturer of the typeface design course put me on to a designer in Holland, Petr van Blokland. He's a really nice guy, really friendly. And I dropped him an e-mail as I was in Holland that year – just dropped by to see him and it turned out he's not only involved in parallel publishing and automated design, but also in typedesign. For him there is really no distinctions between type design and typography. It's kind of like a big building – you have the architecture of the building but you can also go down into the bricks. It's kind of like that with typography, the type design is all these little pieces you assembly to create the typography out of . He's an award-winning typeface designer and typographer and he was involved in the early days of typography very actively. He kind of explained me the whole story of type design technology.

## *C o f f e e   d e l i v e r y   a n d   j a z z   m u s i c*

So, the history of typography actually starts with Free Software, with Donald Knuth and his TeX. The TeX typesetting system has its own font software or font system called Metafont. Metafont is a font programming language, and algebraic programming language describing letter forms. It really gets into the internal structure of the shapes. This is a very non-visual programming approach to it where you basically use this programming language to describe with algebra how the shapes make up the letters. If you have a capital H, you got essentially 3 lines, two verticals stands and a horizontal crossbar and so, in algebra you can say that you've got one ratio whitch is the height of the vertical lines and another ratio which is the width between them and another ratio which is the distance between the top point and the middle point of the crossbar and the bottom point. By describing all of that in algebra, you really describe the structure of that shape and that gives you a lot of power because it means you can trace a pen nib objects over that skeleton to generate the final typeform and so you can apply variations, you can rotate the pen nib – you can have different pen nib shapes And you can have a lot of different typefaces out of that kind of source code. But that approach is not a visual approach, you have to take it with a mathematical mind and that isn't something which graphic designers typically have as a strong part of their skill set.

The next step was describing the outline of a typeface, and the guy who did this was working, I believe, at URW. He invented a digital typography system or typedesign program called Ikarus. The rumor is it's called Ikarus because it crashed too much. Peter Karow is this guy. He was the absolute unknown real pioneer in this area. They were selling this proprietary software powered by a tablet, with a drawing pen for entering the points and it used it's own kind of spline-curve technology.

This was very expensive – it ran on DMS computers and URW was making a lot of money selling those mini computers in well I guess late 70s and early 80s. And there was a new small home computer that came out called the Apple Macintosh. This was quite important because not only was it a personal computer. It had a graphical user interface and also a printer, a laser writer which was based on the Adobe PostScript technology. This was what made desktop publishing happen. I believe it was a Samsung printer revised by Apple and Adobe's PostScript technology. Those three companies, those three technologies was what made desktop publishing happen. Petr van Blokland was involved in it, using the Ikarus software, developing it. And so he ported the program to the Mac. So Ikarus M was the first font editor for personal computers and this was taken on by URW but never really promoted because the … Mac costs not a lot money compared to those big expensive computers. So, Ikarus M was not widely distributed. It's kind of an obvious idea – you know you have those innovative computers doing graphic interfaces and laser printing and several different people had several different ideas about how to employ that. Obviously you had John Warnock within Adobe and at that point Adobe was a systems company, they made this PostScript system and these components, they didn't make any user applications. But John Warnock – and this is documented in the book on the Adobe story – he really pushed within the company to develop Adobe Illustrator, which allowed you to interact with the edit PostScript code and do vector drawings interactively. That was the kind of illustration and graphic design which we mentioned earlier. That was the … page layout sort of thing and that was taking care of by a guy called Paul Brainerd, whose company Aldus made PageMaker. That did similar kind of things than Illustrator, but focused on page layout and typography, text layout rather than making illustrations. So you had Illustrator and PageMaker and this was the beginning of the desktop publishing tool-chain.

H *When was it?*

DC This is in the mid-eighties. The Mac came out in 1984

**Pierre Huyghebaert** (PH) *Illustrator in 1986 I think.*

DC Yeah. And then the Apple LaserWriter, which is I believe a Samsung printer, came out in 1985, and I believe the first edition of Illustrator was in 1988 …

PH *No, I think Illustrator 1 was in 1986.*

DC OK, if you read the official Adobe story book, it's fully documented [1].

H *It's interesting that it follows so quickly after the Macintosh.*

DC Yes! That's right. It all happened very quickly because Adobe and Apple had really built with PostScript and the MacOS, they had the infrastructure there, they could build on top of. And that's a common thing we see played out over and over … Things are developed quite slowly when they are getting the infrastructure right, and then when the infrastructure is in place you see this burst of activity where people can slot it together very quickly to make some interesting things. So, you had this other guy called Jim von Ehr and he saw the need for a graphical user interface to develop fonts with and so he founded a small compagny called Altsys and he made a program called Fontographer. So that became the kind of de-facto standard font editing program.

PH *And before that, do you know what font design software Adobe designers used?*

DC I don't know. Basically when Adobe made PostScript for the Apple LaserWriter then they had the core 35 PostScript fonts, which is about a thousand families, 35 differents weights or variants of the fonts. And I believe that those were from Linotype. Linotype developed that in collaboration with Adobe, I have no idea about what software they used, they may have had their own internal software. I know that before they had

[1] Pamela Pfiffner. *Inside the Publishing Revolution: The Adobe Story.* Adobe Press, 2008

Illustrator they were making PostScript documents by hand like TeX, programming PostScript sourcecode. It might have been in a very low tech way. Because those were the core fonts that have been used in PostScript.

So you had Fontographer and this is yeah I mean a GUI application for home computers to make fonts with. Fontographer made early 90s David Carson graphic design posters. Because it meant that anybody could start making fonts not only people that were in the type design guild. That all David Carson kind of punk graphic design, it's really because of Desktop publishing and specifically because of Fontographer. Because that allowed people to make these fonts. Previous printing technologies wouldn't allow you to make these kinds of fonts without extreme efforts. I mean a lot of the effects you can do with digital graphics you can't do without digital graphics – air brushing sophisticated effects like that can be achieved but it's really a lot of efforts.

So going back to the guys from Holland, Petr has a younger brother called Erik and he went to the college at the Royal Academie of design the KABK in the Hague with a guy who is Just van Rossum and he's the younger brother of Guido van Rossum who is now quite famous because he's the guy who developed and invented Python. In the early 90s Jim von Ehr is developping Fontographer, and Fontographer 4 comes out and Petr and Just and Erik managed to get a copy of the source code of Fontographer 3 which is the golden version that we used, like Quark, that was what we used throughout most of the 90s and so they started adding things to that to do scripting on Fontographer with Python and this was called Robofog, and that was still used until quite recently, because it had features no one has ever seen enywhere else. The deal was you had to get a Fontographer 4 license, and then you could get a Robofont license, for Fontographer 3. Then Apple changed the system architecture and that meant Fontographer 3 would no longer run on Apple computers. Obviously that was a bit of a damn on Robofog. Pretty soon after that Jim sold Fontographer to Macromedia. He and his employes continued to develop Fontographer into Freehand, it went from a font drawing application into a more general purpose illustration tool. So Macromedia bought Altsys for Freehand because they were competing with Adobe at that time. And they didn't really have any interest in continuing to develop Fontographer. Fonts is a really obscure kind of area. As a proprietary software company, what you are doing things to make a profit and if the market is too small to justify your investment then you'll just not keep developing the software. Fontographer shut at that point.

PH *I think they paid one guy to maintain it and answer questions.*

DC Yeah. I think they even stop actively selling it, you had to ask them to sell you a license. Fontographer has stopped at that point and there was no actively developed font editor. There were a few Windows programs, which were kind of shareware for developing fonts because in this time Apple and Microsoft got fed up with paying Adobe's extortion of PostScript licensing fees. They developed their own font format called TrueType. There were Windows font editing programs.
Yeah. I think they even stop actively selling it, you had to ask them to sell you a license. Fontographer has stopped at that point and there was no actively developed font editor. There were a few Windows programs, which were kind of shareware for developing fonts because in this time Apple and Microsoft got fed up with paying Adobe's extortion of PostScript licensing fees. They developed their own font format called TrueType. When Fontographer stopped there was the question of which one will become the predominant font editor and so there was Fontlab. This was developed by a guy Yuri Yarmola, Russian originally I believe, and it became the primary proprietary type design tool.
The Python guys from Holland started using Fontlab. They managed to convince the Fontlab guys to include Python scripting support in Fontlab. Python had become a major language, for doing this kind of scripting. So Fontlab added in Python scripting. And then different type designers, font developers started to use Python scripts to help them develop their fonts, and a few of the guys doing that decided to join up and they created the RoboFab project which took the ideas that had been developed for Robofob and reimplemented them with Fontlab – so RoboFab. This is now a Free Software package, under the MIT Python style licence. So it is a Free Software licence but without copyleft. It has beeing developed as a collaborative project. If you're interested in the development you can just join the mailing list. It's a very mature project and the really beautiful thing about it that they developed a font object model and so in Python you have a very clean and easily understandable object-oriented model of what a font is. It makes it very easy to script things. This is quite exciting because that means you can start to do things which are just not really visible with the graphic design interface. The thing with those fonts is like there is a scale, it is like

architecture. You've got the designer of the building and the designer of the bricks. With a font it is the same. You have the designer who shapes each letter and then you've got the character-spacing which makes what a paragraph will look like. A really good example of this is if you want to do interpolation, if you have a very narrow version of a font and a very wide one, and you want to interpolate in different versions between those two masters – you really want to do that in a script, and RoboFab makes this really easy to do this within Fontlab. The ever important thing about RoboFab was that they developed UFO, I think it's the Universal Font Object – I'm not sure what the exact name is – but it's a XML font format which means that you can interchange font source data with different programs and specifically that means that you have a really good font interpolation program that can read and write that UFO XML format and then you can have your regular type design format font editor that will generate bitmap font formats that you actually use in a system. You can write your own tool for a specific task and push and pull the data back and forth. Some of these Dutch guys, especially Erik has written a really good interpolation tool. So, as a kind of thread in the story of font. Remember that time where Fontographer was not developed actively then you have Georges Williams from California who was interested in digital typography and fonts and Fontographer was not being activelly developed and he found that quite frustrating so he said like *Well, I'll write my own font editor*. He wrote it from scratch. I mean this is a great project.

LL *Can you tell us some details about your course?*

DC There are four main deliverables in the course, that you normally do in one year, twelve months. The big thing is that you do a professional quality OpenType font, with an extended pan-european latin coverage in regular and italic, maybe bold. You also do a complex non-latin in Arabic, Indic, maybe Cyrillic … well not really Cyrillic because there are problems to get a Cyrillic type experts from Russia to Britain … or Greek, or any script with which you have a particular background in. And so, they didn't mandate which software students can use, and I was already used to FontForge, while pretty much all the other students were using FontLab. This font development is the main thing. The second thing is the dissertation, that goes up to 8,000 words, an academic master in typography dissertation. Then there is a smaller essay, that will be published on http://www.typeculture.com/academic_resource/articles_essays/, and it's a kind of a practice for writing the dissertation. Then you have to document your working process throughout the year, you have to submit your working files, source files. Every single step is documented and you have to write a small essay describing your process. And also, of course, apart from the type design, you make a font specimen, so you make a very nice piece of design that show up your font in use, as commercial companies do. All that takes a full intense year. For British people, the course costs about £3,000, for people in the EU, it costs about £5,000 and about £10,000 for non-EU. Have a look at the website for details, but yes, it's very expensive.

LL *And did you also design a font?*

DC Yes. But I do it part-time. Normally, you could do the typeface, and the year after you do the dissertation. For personal reasons, I do the dissertation first, in the summer, and next year I'll do the typeface, I think in July next year.

LL *You have an idea on which font you'll work?*

DC Yes. The course doesn't specify which kind of typeface you have to work on. But they really prefer a textface, a serif one, because it's the most complicate and demanding work. If you can do a high quality serif text typeface design, you can do almost any typeface design! Of course, lots of students do also a sans serif typeface to be read at 8 or 9 points, or even for by example dictionaries at 6 or 7 points. Other students design display typefaces that can be used for pararaphs but probably not at 9 points …

**Femke Snelting** (FS) *It looks like you are asked to produce quite a lot of documents. Are these documents published anywhere, are they available for other designers?*

DC Yes, the website is http://www.typefacedesign.net and the teaching team encourages students to publish their essays, and some people have published their dissertation on the web, but it varies. Of course, being an academic dissertation, you can request if from the university.

FS *I'm asking because in various presentations the figure of the 'expert typographer' came up, and the role Open Source software could have, to open up this guild.*

DC Yeah, the course in The Hague is cheaper, the pound was quite high so it's expensive to live in Britain during the last year, and the number of people able to produce high quality fonts is pretty small … And these courses are quite inaccessible for most of the people because of being so expensive, you have to be quite commited to follow them. The proprietary font editing software, even with a student discount, is also a bit expensive. So yes, Free and Open Source software could be an enabler. FontForge allows anybody to grab it on the Internet and start making fonts. But having the tools is just the beginning. You have to know what you're doing to a design a typeface, and this is separate from font software techinques. And books on the subject, there are quite a few, but none are really a full solution. There www.typophile.org, a type design forum on the web, where you can post preliminary designs. But of course you do not get the kind of critical feedback as you can get on a masters course …

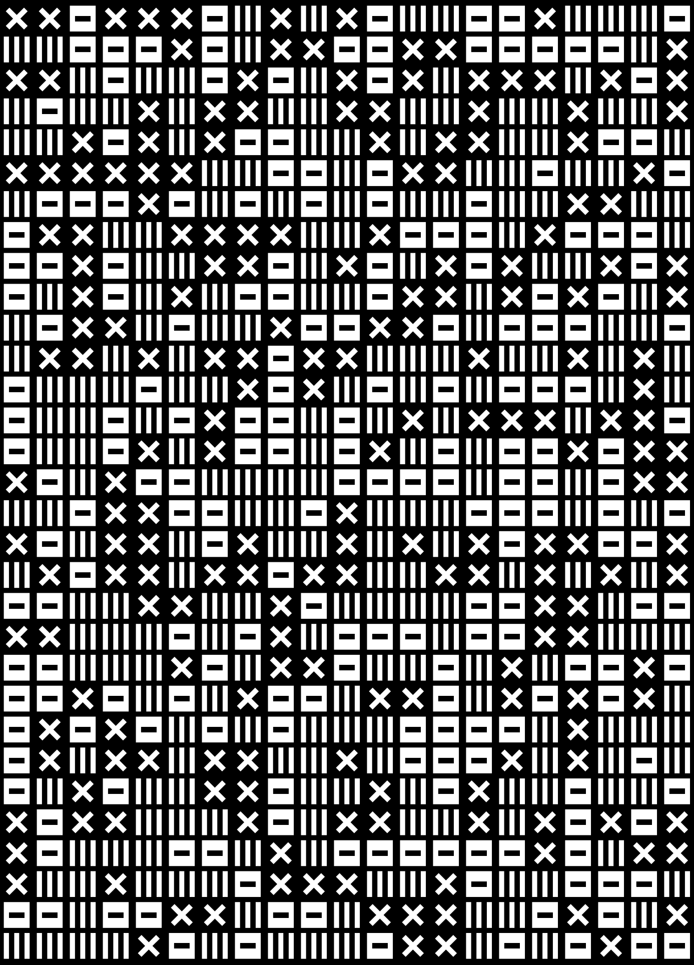
FS *We talked to Denis Jacquerye from the DéjàVu project, and most of the people who collaborate on the project are not type designers but people who are interested in having certain glyphs added to a typeface. And we asked him if there is some kind of teaching going on, to be sure that the people contributing understand what they are doing. Do you see any way of, let's say, a more open way of teaching typography starting to happen?*

DC Yeah, I mean, that the part of why the Free Software movement is going to branch down into the Free Culture movement. There is that website Freedom Defined[2] that states that the principles of Free Software can apply to all other kind of works. This isn't shared by everybody in the Free Software movement. Richard Stallman makes a clear difference between three kind of works: the ones that function like software, encyclopedias, dictionaries, text books that tell how to makes things, and text typefaces. Art works like music and films, and text works about opinions like scientific papers or political manifestos. He believes that different kinds of rights should apply for that different kind of works. There is also a different view in which anything in a computer can be edited ought to be free like Free Software. That is certainly a position that many people take in the Free Software community. In the WikiMedia Foundation text books project, you can see that when more and more people are involved in typeface design from the Free Culture community, we will see more and more education material. There will be a snowball effect.

PH *Dave, we are running out of time …*

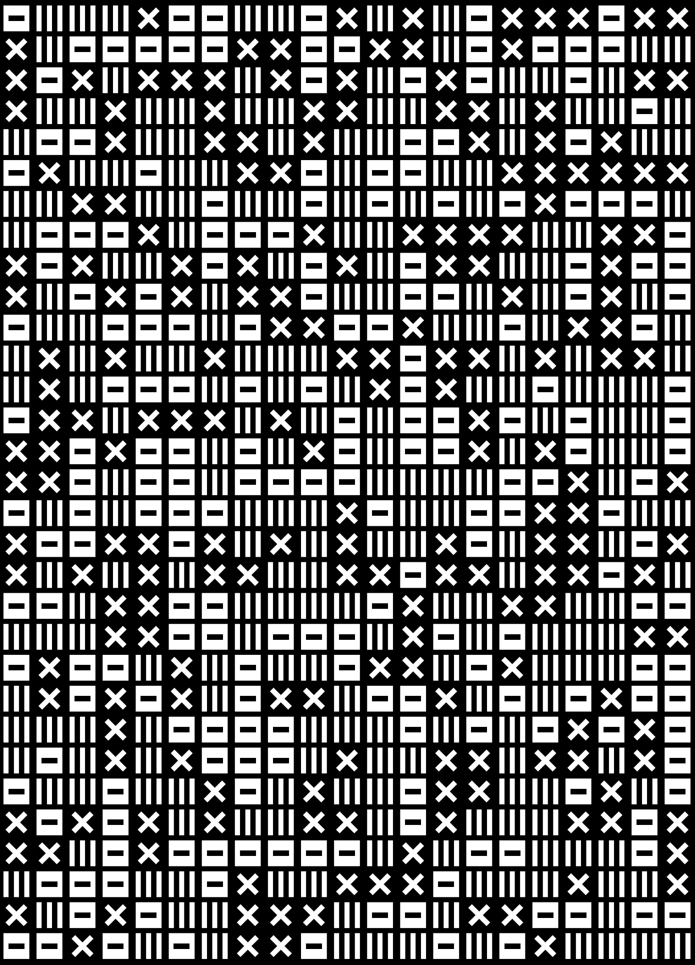
2 http://freedomdefined.org

DC So just to finish about the FontForge Python scripting … There is Python embeded in FontForge so you can run scripts to control FontForge, you can add new features that maybe would be specific to your font and then in FontForge there is also a Python module which means that you can type into a Python interpretor. You type `import fontforge` and if it doesn't give you an error then you can start to do FontForge functions, just like in the RoboFab environment. And in the process of adding that George kind of re-architectured the FontForge source code so instead of being one large program, there is now a large C library, libfontforge, and then a small C program for rendering and also the Python module, a binding or interface to that C library. This means if you are an application programmer it is very straightforward to make a new font editor in whatever language you want, using whatever graphic toolkit you want. So if you're a JDK guy or a GTK guy or even if you're on Windows or Mac OS X, you can make a font editor that has all the functionality of FontForge. FontForge is a kind of engine to make font editors. This is quite exciting because it means it's pretty straight forward for somebody to write a font editing program which is designed for, say, beginners.
So, to come back to what we were just talking about in term of educational materials to get people new to typeface design to be confident with themselves. Maybe they won't be in that professional level yet, but they will be pleased with their own work and happy to work in a user interface where you feel like in 2006, you know, with nice icons nice windows; anti aliasing and these kind of things.
I mean there's nothing wrong with the FontForge interface. It is what it is. But it scares a lot of people away, people say that they don't like this. I think it is too scary, too different. I think we are going to see some exciting stuff in the next few years in the Free Software font editor space.



# Data analysis as a discourse

✕ Michael Terry
||| Ivan Monroy Lopez
⊟ Femke Snelting



# Data analysis as a discourse

× Michael Terry
||| Ivan Monroy Lopez
▭ Femke Snelting

At the **Libre Graphics Meeting 2008** in Wroclaw, just before Michael Terry presents his project ingimp to an audience of curious GIMP developers and users, we meet up to talk more about ‘instrumenting GIMP’ and about the way Terry thinks data analysis could be done as a form of discourse. Michael Terry is a computer scientist working at the Human Computer Interaction Lab of the University of Waterloo, Canada and his main research focus is on improving usability in Open Source software. We speak about ingimp, a clone of the popular image manipulation programme GIMP, but with an important difference: ingimp allows users to record data about their usage in to a central database, and subsequently makes this data available to anyone. This conversation was also published in the Constant publication *Tracks in electr(on)ic fields*.

⊟ *Maybe we could start this conversation with a description of the ingimp project you are developing and why you chose to work on usability for GIMP?*

✕ So the project is ‘ingimp’, which is an instrumented version of GIMP, it collects information about how the software is used in practice. The idea is you download it, you install it, and then with the exception of an additional start up screen, you use it just like regular Gimp. So, our goal is to be as unobtrusive as possible to make it really easy to get going with it, and then to just forget about it. We want to get it into the hands of as many people as possible, so that we can understand how the software is actually used in practice. There are plenty of forums where people can express their opinions about how GIMP should be designed, or what’s wrong with it, there are plenty of bug reports that have been filed, there are plenty of usability issues that have been identified, but what we really lack is some information about how people actually apply this tool on a day to day basis. What we want to do is elevate discussion above just anecdote and gut feelings, and to say, well, there is this group of people who appear to be using it in this way, these are the characteristics of their environment, these are the sets of tools they work with, these are the types of images they work with and so on, so that we have some real data to ground discussions about how the software is actually used by people. You asked me now why GIMP? I actually used GIMP extensively for my PhD work. I had these little cousins come down and hang out with me in my apartment after school, and I would set them up with GIMP, and quite often they would always start off with one picture, they would create a sphere, a blue sphere, and then they played with filters until they got something really different. I would turn to them looking at what they had been doing for the past twenty minutes, and would be completely amazed at the results they were getting just by fooling around with it. And so I thought, this application has lots and lots of power, I'd like to use that power to prototype new types of interface mechanisms. So I created JGimp, which is a Java based extension for the 1.0 GIMP series, that I can use as a back-end for prototyping novel user interfaces. I think that it is a great application, there is a lot of power to it, and I had already an investment in its code base so it made sense to use that as a platform for testing out ideas of open instrumentation.

*What is special about ingimp, is the fact that the data you generate is made by the software you are studying itself. Could you describe how that works?*

Every bit of data we collect, we make available: you can go to the website, you can download every log file that we have collected. The intent really is for us to build tools and infrastructure so that the community itself can sustain this analysis, can sustain this form of usability. We don't want to create a situation where we are creating new dependencies on people, or where we are imposing new tasks on existing project members. We want to create tools that follow the same ethos as Open Source development, where anyone can look at the source code, where anyone can make contributions, from filing a bug to doing something as simple as writing a patch, where they don't even have to have access to the source code repository, to make valuable contributions. So importantly, we want to have a really low barrier to participation. At the same time, we want to increase the signal-to-noise ratio. Yesterday I talked with Peter Sikking, an information architect working for GIMP, and he and I both had this experience where we work with user interfaces, and since everybody uses an interface, everybody feels they are an expert, so there can be a lot of noise. So, not only did we want to create an open environment for collecting this data, and analysing it, but we also want to increase the chance that we are making valuable contributions, and that the community itself can make valuable contributions. Like I said, there is enough opinion out there. What we really need to do is to better understand how the software is being used. So, we have made a point from the start to try to be as open as possible with everything, so that anyone can really contribute to the project.

▬ *ingimp has been running for a year now. What are you finding?*

✖ I have started analysing the data, and I think one of the things that we realised early on is that it is a very rich data set; we have lots and lots of data. So, after a year we've had over 800 installations, and we've collected about 5000 log files, representing over half a million commands, representing thousands of hours of the application being used. And one of the things you have to realise is that when you have a data set of that size, there are so many different ways to look at it that my particular perspective might not be enough. Even if you sit someone down, and you have him or her use the software for twenty minutes, and you videotape it, then you can spend hours analysing just that twenty minutes of videotape. And so, I think that one of the things we realised is that we have to open up the process so that anyone could easily participate. We have the log files available, but they really didn't have an infrastructure for analysing them. So, we created this new piece of software called 'StatsJam', an extension to MediaWiki, which allows anyone to go to the website and embed SQL-queries against the ingimp data set and then visualise those results within the Wiki text. So, I'll be announcing that today and demonstrating that, but I have been using that tool now for a week to complement the existing data analysis we have done. One of the first things that we realized is that we have over 800 installations, but then you have to ask, how many of those are really serious users? A lot of people probably just were curious, they downloaded it and installed it, found that it didn't really do much for them and so maybe they don't use it anymore. So, the first thing we had to do is figure out which data points should we really pay attention too. We decided that a person should have saved an image, and they should have used ingimp on two different occasions, preferably at least a day apart, where they'd saved an image on both of the instances. We used that as an indication of what a serious user is. So with that filter in place, then the '800 installations' drops down to about 200 people. So we had about 200 people using ingimp, and looking at the data this represents

about 800 hours of use, about 4000 log files, and again still about half a million commands. So, it's still a very significant group of people. 200 people is still a lot, and that's a lot of data, representing about 11000 images they have been working on, there's just a lot.
From that group, what we found is that use of ingimp is really short and versatile. So, most sessions are about fifteen minutes or less, on average. There are outliers, there are some people who use it for longer periods of time, but really it boils down to them using it for about fifteen minutes, and they are applying fewer than a hundred operations when they are working on the image. I should probably be looking at my data analysis as I say this, but they are very quick, short, versatile sessions, and when they use it, they use less than 10 different tools, or they apply less than 10 different commands when they are using it. What else did we find? We found that the two most popular monitor resolutions are 1280 by 1024 and 1024 by 768. So, those represent collectively 60% of the resolutions, and really 1280 by 1024 represents pretty much the maximum for most people, although you have some higher resolutions. So one of the things that's always contentious about GIMP, is its window management scheme and the fact that it has multiple windows, right? And some people say, well you know this works fine if you have two monitors, because you can throw out the tools on one monitor and then your images are on another monitor. Well, about 10% to 15% of ingimp users have two monitors, so that design decision is not working out for most of the people, if that is the best way to work. These are things I think that people have been aware of, it's just now we have some actual concrete numbers where you can turn to and say, now this is how people are using it. There is a wide range of tasks that people are performing with the tool, but they are really short, bursty tasks.

▬ *Every time you start up ingimp, a screen comes up asking you to describe what you are planning to do and I am interested in the kind of language users invent to describe this, even when they sometimes don't know exactly what it is they are going to do. So inventing language for possible actions with the software, has in a way become a creative process that is now shared between interface designer, developer and user. If you look at the 'activity tags' you are collecting, do you find a new vocabulary developing?*

✖ I think there are 300 to 600 different activity tags that people register within that group of 'significant users'. I didn't have time to look at all of them, but it is interesting to see how people are using that as a medium for communicating to us. Some people will say, *Just testing out, ignore this!* Or, people are trying to do things like insert HTML code, to do like a cross-site scripting attack, because, you have all the data on the website, so they will try to play with that. Some people are very sparse and they say 'image manipulation' or 'graphic design' or something like that, but then some people are much more verbose, and they give more of a plan, *This is what I expect to be doing*. So, I think it has been interesting to see how people have adopted that and what's nice about it, is that it adds a really nice human element to all this empirical data.

III **I wanted to ask you about the data, without getting too technical, could you explain how these data are structured, what do the log files look like?**

✕ So the log files are all in XML, and generally we compress them, because they can get rather large. And the reason that they are rather large is that we are very verbose in our logging. We want to be completely transparent with respect to everything, so that if you have some doubts or if you have some questions about what kind of data has been collected, you should be able to look at the log file, and figure out a lot about what that data is. That's how we designed the XML log files, and it was really driven by privacy concerns and by the desire to be transparent and open. On the server side we take that log file and we parse it out, and then we throw it into a database, so that we can query the data set.

⊟ *Now we are talking about privacy … I was impressed by the work you have done on this; the project is unusually clear about why certain things are logged, and other things not; mainly to prevent the possibility of 'playing back' actions so that one could identify individual users from the data set. So, while I understand there are privacy issues at stake I was wondering … what if you could look at the collected data as a kind of scripting for use? Writing a choreography that might be replayed later?*

✕ Yes, we have been fairly conservative with the type of information that we collect, because this really is the first instance where anyone has captured such rich data about how people are using software on a day to day basis, and then made it all that data publicly available. When a company does this, they will keep the data internally, so you don't have this risk of someone outside figuring something out about a user that wasn't intended to be discovered. We have to deal with that risk, because we are trying to go about this in a very open and transparent way, which means that people may be able to subject our data to analysis or data mining techniques that we haven't thought of and extract information that we didn't intent to be recording in our file, but which is still there. So there are fairly sophisticated techniques where you can do things like look at audio recordings of typing and the timings between keystrokes, and then work backwards with the sounds made to figure out the keys that people are likely pressing. So, just with keyboard audio and keystroke timings alone you can often give enough information to be able to reconstruct what people are actually typing. So we are always sort of weary about how much information is in there. While it might be nice to be able to do something like record people's actions and then share that script, I don't think that that is really a good use of ingimp. That said, I think it is interesting to ask, could we characterize people's use enough, so that we can start clustering groups of people together and then providing a forum for these people to meet and learn from one another? That's something we haven't worked out. I think we have enough work cut out for us right now just to characterize how the community is using it.

⊟ *It was not meant as a feature request, but as a way to imagine how usability research could flip around and also become productive work.*

✕ Yes, totally. I think one of the things that we found when bringing people into to assess the basic usability of the ingimp software and ingimp website, is that people like looking at things like what commands other people are using, what the most frequently used commands are, and part of the reason that they like that, is because of what it teaches them about the application. So they might see a command they were unaware of. So we have toyed with the idea of then providing not only the command name, but then a link from that command name to the documentation – but I didn't have time to implement it, but certainly there are possibilities like that, you can imagine.

⊟ *Maybe another group can figure something out like that? That's the beauty of opening up your software plus data set of course. Well, just a bit more on what is logged and what not… Maybe you could explain where and why you put the limit and what kind of use you might miss out on as a result?*

✕ I think it is important to keep in mind that whatever instrument you use to study people, you are going to have some kind of bias, you are going to get some information at the cost of other information. So if you do a video taped observation of a user and you just set up a camera, then you are not going to find details about the monitor maybe, or maybe you are not really seeing what their hands are doing. No matter what instrument you use, you are always getting a particular slice. I think you have to work backwards and ask what kind of things do you want to learn. And so the data that we collect right now, was really driven by what people have done in the past in the area of instrumentation, but also by us bringing people into the lab, observing them as they are using the application, and noticing particular behaviours and saying, hey, that seems to be interesting, so what kind of data could we collect to help us identify those kind of phenomena, or that kind of performance, or that kind of activity? So again, the data that we were collecting was driven by watching people, and figuring out what information will help us to identify these types of activities. As I've said, this is really the first project that is doing this, and we really need to make sure we don't poison the well. So if it happens that we collect some bit of information, that then someone can later say, *Oh my gosh, here is the person's file system, here are the names they are using for the files* or whatever, then it's going to make the normal user population weary of downloading this type of instrumented application. This is the thing that concerns me most about Open Source developers jumping into this domain, is that they might not be thinking about how you could potentially impact privacy.

**III I don't know, I don't want to get paranoid. But if you are doing it, then there is a possibility someone else will do it in a less considerate way.**

✕ I think it is only a matter of time before people start doing this, because there are a lot of grumblings about, *we should be doing instrumentation, someone just needs to sit down and do it.* Now there is an extension out for Firefox that will collect this kind of data as well, so you know …

**III Maybe users could talk with each other, and if they are aware that this type of monitoring could happen, then that would add a different social dimension …**

✖ It could. I think it is a matter of awareness, really, so when we bring people into the lab and have them go to the ingimp website, download and install it and use it, and go check out the stats on the website, and then we ask questions like, what kind of data are we collecting? We have a lengthy concern agreement that details the type of information we are collecting and the ways your privacy could be impacted, but people don't read it.

▬ *So concretely … what information are you recording, and what information are you not recording?*

✖ We record every command name that is applied to a document, to an image. Where your privacy is at risk with that, is that if you write a custom script, then that custom script's name is going to be inserted into a log file. And so if you are working for example for Lucas or DreamWorks or something like that, or ILM, in some Hollywood movie studio and you are using ingimp and you are writing scripts, then you could have a script like 'fixing Shrek's beard', and then that is getting put into the log file and then people are going to know that the studio uses ingimp. We collect command names, we collect things like what windows are on the screen, their positions, their sizes, we take hashes of layer names and file names. We take a string and then we create a hash code for it, and we also collect information about how long is this string, how many alphabetical characters, numbers, things like that, to get a sense of whether people are using the same files, the same layer names time and time again, and so on. But this is an instance where our first pass at this, actually left open the possibility of people taking those hashes and then reconstructing the original strings from that. Because we have the hash code, we have the length of the string, all you have to do is generate all possible strings of that length, take the hash codes and figure out which hashes match. And so we had to go back and create a new scheme for recording this type of information where we create a hash and we create a random number, we pair those up on the client machine but we only log the random number. So, from log to log then, we can track if people use the same image names, but we have no idea of what the original string was. There are these little 'gotchas', things to look out for, that I don't think most people are aware of, and this is why I get really concerned about instrumentation efforts right now, because there isn't this body of experience of what kind of data should we collect, and what shouldn't we collect.

▬ *As we are talking about this, I am already more aware of what data I would allow to be collected. Do you think by opening up this data set and the transparent process of collecting and not collecting, this will help educate users about these kinds of risks?*

✕ It might, but honestly I think probably the thing that will educate people the most is if there was a really large privacy error and that it got a lot of news, because then people would become more aware of it because right now – and this is not to say that we want that to happen with ingimp – but when we bring people in and we ask them about privacy, *Are you concerned about privacy?*, and they say *No*, and we say *Why?* Well, they inherently trust us, but the fact is that Open Source also lends a certain amount of trust to it, because they expect that since it is Open Source, the community will in some sense police it and identify potential flaws with it.

▬ *Is that happening?*
*Are you in dialogue with the Open Source community about this?*

✕ No, I think probably five to ten people have looked at the ingimp code – realistically speaking I don't think a lot of people looked at it. Some of the GIMP developers took a gander at it to see how could we put this upstream, but I don't want it upstream, because I want it to always be an opt-in, so that it can't be turned on by mistake.

▬ *You mean you have to download ingimp and use it as a separate program? It functions in the same way as GIMP, but it makes the fact that it is a different tool very clear.*

✕ Right. You are more aware, because you are making that choice to download that, compared to the regular version. There is this awareness about that. We have this lengthy text based consent agreement that talks about the data we collect, but less than two percent of the population reads license agreements. And, most of our users are actually non-native English speakers, so there are all these things that are working against us. So, for the past year we have really been focussing on privacy, not only in terms of how we collect the data, but how we make people aware of what the software does. We have been developing wordless diagrams to illustrate how the software functions, so that we don't have to worry about localisation errors as much. And so we have these illustrations that show someone downloading ingimp, starting it up, a graph appears, there is a little icon of a mouse and a keyboard on the graph, and they type and you see the keyboard bar go up, and then at the end when they close the application, you see the data being sent to a web server. And then we show snapshots of them doing different things in the software, and then show a corresponding graph change. So, we developed these by bringing in both native and non-native speakers, having them look at the diagrams and then tell us what they meant. We had to go through about fifteen people and continual redesign until most people could understand and tell us what they meant, without giving them any help or prompts. So, this is an ongoing research effort, to come up with techniques that not only work for ingimp but also for other instrumentation efforts, so that people can become more aware of the implications.

⊟ *Can you say something about how this type of research relates to classic usability research and in particular to the usability work that is happening in Gimp?*

✕ Instrumentation is not new, commercial software companies and researchers have been doing instrumentation for at least ten years, probably ten to twenty years. So, the idea is not new but what is new, in terms of the research aspects of this, is how do we do this in a way where we can make all the data open? The fact that you make the data open, really impacts your decision about the type of data you collect and how you are representing it. And you need to really inform people about what the software does. But I think your question is … how does it impact the GIMP's usability process? Not at all, right now. But that is because we have intentionally been laying off to the side, until we got to the point where we had an infrastructure, where the entire community could really participate with the data analysis. We really want to have this to be a self-sustaining infrastructure, we don't want to create a system where you have to rely on just one other person for this to work.

III **What approach did you take in order to make this project self-sustainable?**

✕ Collecting data is not hard. The challenge is to understand the data, and I don't want to create a situation where the community is relying on only one person to do that kind of analysis, because this is dangerous for a number of reasons. First of all, you are creating a dependency on an external party, and that party might have other obligations and commitments, and might have to leave at some point. If that is the case, then you need to be able to pass the baton to someone else, even if that could take a considerate amount of time and so on. You also don't want to have this external dependency, because of the richness in the data, you really need to have multiple people looking at it, and trying to understand and analyse it. So how are we addressing this? It is through this StatsJam extension to the MediaWiki that I will introduce today. Our hope is that this type of tool will lower the barrier for the entire community to participate in the data analysis process, whether they are simply commenting on the analysis we made or taking the existing analysis, tweaking it to their own needs, or doing something brand new.

In talking with members of the GIMP project here at the Libre Graphics Meeting, they started asking questions like, *So how many people are doing this, how many people are doing this and how many this?* They'll ask me while we are sitting in a café, and I will be able to pop the database open and say, *A certain number of people have done this*, or, *no one has actually used this tool at all.* The danger is that this data is very rich and nuanced, and you can't really reduce these kind of questions to an answer of *N people do this*, you have to understand the larger context. You have to understand why they are doing it, why they are not doing it. So, the data helps to answer some questions, but it generates new questions. They give you some understanding of how the people are using it, but then it generates new questions of, *Why is this the case?* Is this because these are just the people using ingimp, or is this some more widespread phenomenon? They asked me yesterday how many people are using this colour picker tool – I can't remember the exact name – so I looked and there was no record of it being used at all in my data set. So I asked them when did this come out, and they said, *Well it has been there at least since 2.4.* And then you look at my data set, and you notice that most of my users are in the 2.2 series, so that could be part of the reasons. Another reason could be, that they just don't know that it is there, they don't know how to use it and so on. So, I can answer the question, but then you have to sort of dig a bit deeper.

▬ *You mean you can't say that because it is not used, it doesn't deserve any attention?*

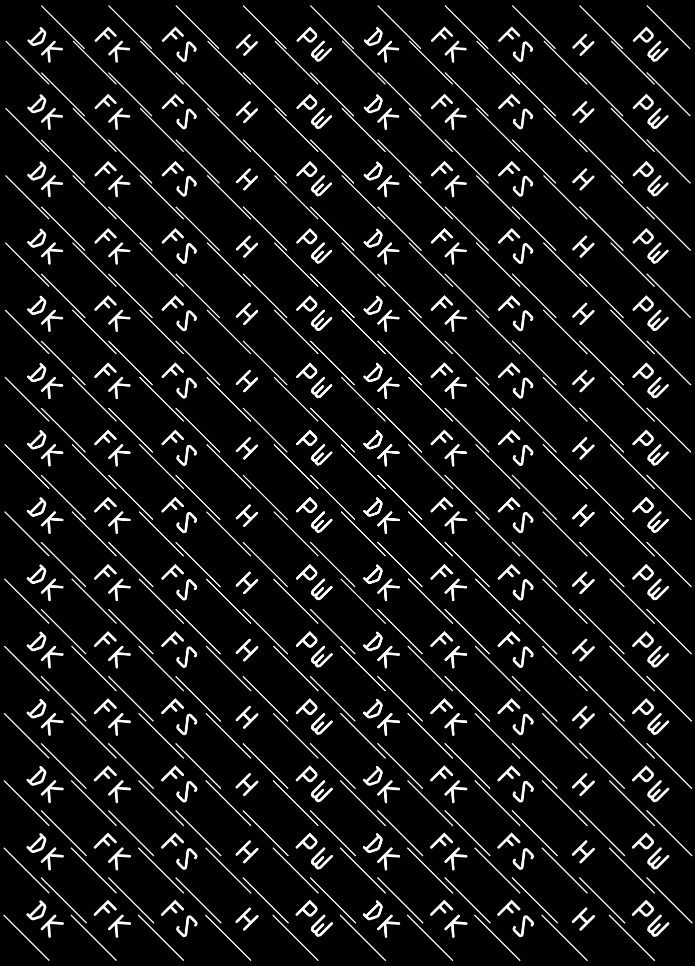
✕ Yes, you just can't jump to conclusions like that, which is again why we want to have this community website, which shows the reasoning behind the analysis. Here are the steps we had to go through to get this result, so you can understand what that means, what the context means, because if you don't have that context, then it's sort of meaningless. It's like asking, what are the most frequently used commands? This is something that people like to ask about. Well really, how do you interpret that? Is it the numbers of times it has been used across all log files? Is it the number of people that have used it? Is it the number of log files where it has been used at least once? There are lots and lots of ways in which you can interpret this question. So, you really need to approach this data analysis as a discourse, where you are saying, here are my assumptions, here is how I am getting to this conclusion, and this is what it means for this particular group of people. So again, I think it is dangerous if one person does that and you become to rely on that one person. We really want to have lots of people looking at it, and considering it, and thinking about the implications.

▬ *Do you expect that this will impact the kind of interfaces that can be done for GIMP?*

✕ I don't necessarily think it is going to impact interface design, I see it really as a sort of reality check: this is how communities are using the software and now you can take that information and ask, do we want to better support these people or do we ... For example on my data set, most people are working on relatively small images for short periods of time, the images typically have one or two layers, so they are not really complex images. So regarding your question, one of the things you can ask is, should we be creating a simple tool to meet these people's needs? All the people are is just doing cropping and resizing, fairly common operations, so should we create a tool that strips away the rest of the stuff? Or, should we figure out why people are not using any other functionality, and then try to improve the usability of that? There are so many ways to use data I don't really know how it is going to be used, but I know it doesn't drive design. Design happens from a really good understanding of the users, the types of tasks they perform, the range of possible interface designs that are out there, lots of prototyping, evaluating those prototypes and so on. Our data set really is a small potential part of that process. You can say, well according to this data set, it doesn't look like many people are using this feature, let's not much focus too on that, let's focus on these other features or conversely, let's figure out why they are not using them … Or you might even look at things like how big their monitor resolutions are, and say well, given the size of the monitor resolution, maybe this particular design idea is not feasible. But I think it is going to complement the existing practices, in the best case.

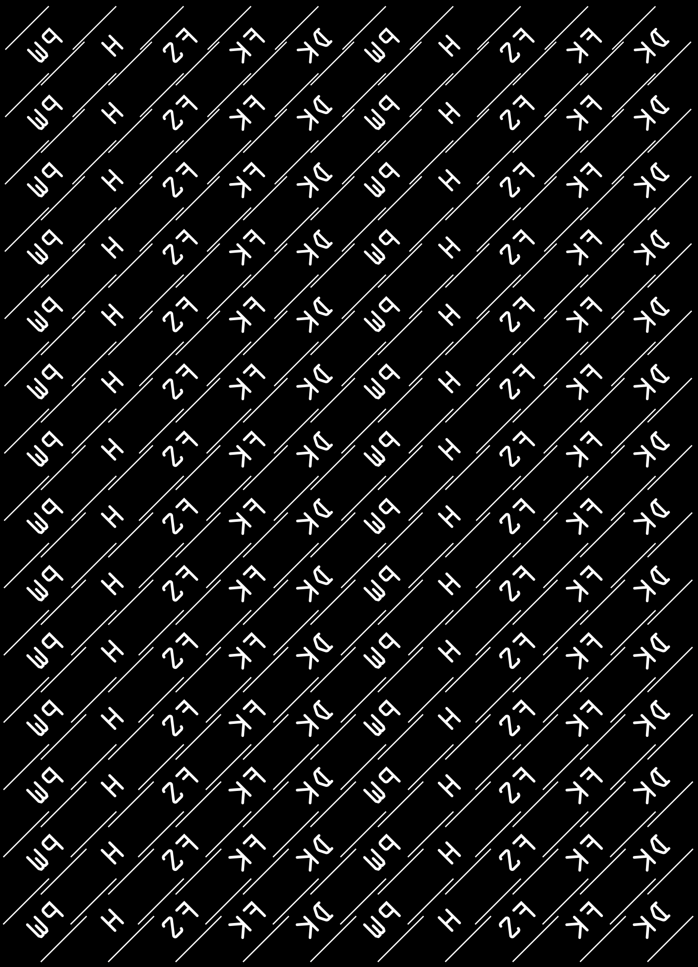
▬ *And do you see a difference in how interface design is done in free software projects, and in proprietary software?*

✕ Well, I have been mostly involved in the research community, so I don't have a lot of exposure to design projects. I mean, in my community we are always trying to look at generating new knowledge, and not necessarily at how to get a product out the door. So, the goals or objectives are certainly different. I think one of the dangers in your question is that you sort of lump a lot of different projects and project styles into one category of 'Open Source'. 'Open source' ranges from volunteer driven projects to corporate projects, where they are actually trying to make money out of it. There is a huge diversity of projects that are out there; there is a wide diversity of styles, there is as much diversity in the Open Source world as there is in the proprietary world. One thing you can probably say, is that for some projects that are completely volunteer driven like GIMP, they are resource strapped. There is more work than they can possibly tackle with the number of resources they have. That makes it very challenging to do interface design, I mean, when you look at interface code, it costs you 50% or 75% of a code base. That is not insignificant, it is very difficult to hack and you need to have lots of time and manpower to be able to do significant things. And that's probably one of the biggest differences you see for the volunteer driven projects, it is really a labour of love for these people and so very often the new things interest them, whereas with a commercial software company developers are going to have to do things sometimes they don't like, because that is what is going to sell the product.


DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW
DK FK FS H PW DK FK FS H PW

# Why you should own the beercompany you design for

Franziska Kleiner
Peter Westenberg
Femke Snelting
Dmytri Kleiner
Harrison



# Why you should own the beercompany you design for

Franziska Kleiner
Peter Westenberg
Femke Snelting
Dmytri Kleiner
Harrison

In **2007**, OSP met with venture communist Dmytri Kleiner and his wife Franziska,[1] late at night in the bar Le Coq in Brussels. Kleiner had just finished his lecture *InfoEnclosure-2.0* at Verbindingen/Jonctions and we wanted to ask what his ideas about peer production could mean for the practice of designers and typographers. Referring to Benjamin Tucker, Yochai Benkler, Marcel Mauss and of course Karl Marx, Kleiner explains how to prevent leakage at the point of scarcity through operating within a total system of worker owned companies. Between fundamentals of media- and information economy, he talks about free typography and what it has to do with nuts and bolts, the problem of working with estimates and why the people that develop Scribus should own all the magazines it enables.

DK First of all we have to be clear, our own company is very small and doesn't actually earn enough money to sustain itself right now. We sustain our company at this point by taking on other projects; for example we are here for a project that has really little to do with Telekommunisten, where we're helping a recruiting company in Canada, I'm in the UK for a very different reason than Telekommunisten, doing independent software development for a private company. So we're still self-funding our company. So we haven't yet got to a stage where our company can actually sustain itself from our own peer production, which is our goal. But how we plan to realize that goal, is through peer production. To start we can sketch out a simple economic model, to understand how the economics work. Economics work with the so called factors of production: you have land, labour and capital. Land is natural resources, that which occurs naturally, that which nobody produces, that just sort exists. Land, electromagnetic frequencies, everything which naturally exists. Labour is work, something that people do. Capital is what happens when you apply labour to land, and you create products. Some of these products have to be consumed, and some of those products are to be used in further production, and that's capital. So capital

[1] editor for a German publishing company

Dmytri Kleiner

Femke Snelting

is the result of labour applied to land that create output that is used for further production, and that's tools, machines and so forth. This system produces commodities which are consumed in the market. In this system the dominating input in the production owns the final product, and all of the actual value of the products is captured at that stage. So whoever sells the product in the marketplace captures the full value of that product, the full marginal value, or use value. All of the inputs to that process can never make anymore than their own cost of reproduction, make their own subsistence cost. So if as a worker you're selling your labour to somebody else who owns the product, you're never going to capture anymore than your subsistence cost.

FS *Could you make that sort of concrete?*

DK Well, the reason that people need design is because there's some product that in the end requires design as an input. For instance, a simple case is obviously a magazine, in which design is a major input. The value is always going to be captured by the people selling the magazine. All of the inputs to that magazine, including design, journalism, layout, administration, are never going to capture more than their reproduction costs. So in order for any group of workers to really capture the value of their labour, they have to own the final product. Which means that they can't just simply be isolated in one field, like design. It means that the entire productive cycle has to be owned collectively by the workers. The designers, together with the journalists, together with the administrators, have to own the magazine, otherwise they can't capture their full value. As a group of designers this is very difficult, because as a group of designers you're only selling an input, you're not at the end owning a product. The only way to do this is by forming alliances with other people, and not based on wages, not based on them giving you an arbitrary amount of money for that input, which will never be higher than reproduction cost, but based on owning together the final product. So you contribute design, somebody else contributes journalism, somebody else contributes administration and together you all own this magazine. Then it is this magazine that is sold on the market that is your wage, the value of the magazine on the market. That is the only way that you can capture the marginal value of your labour. You have to sell the product, not the input, not labour. Marx talks about labour being itself a commodity, and that means that you can never capture its marginal contribution of production, you can only capture its reproduction cost. Which means what it would cost to sustain a designer. A designer needs to eat, a designer needs a place to live, to have a certain lifestyle to fit in the design community and that's all you get by selling your labour. You won't get anymore because there is no reason for the owner of the product to give you anymore. The only way you can get more is if you own the product itself, collectively with the other labour inputs. And I know that's a bad answer, nobody wants to hear that answer.

FS *Haha!*

DK This estimate is at the start in the possibility. Because the whole point of a creative project is that you're doing something that hasn't been done before. And we have all struggled with this before. There's two things you don't know at the beginning of a contract. The first is how long it will take and the second is what the criteria of being finished will be. You don't know either of those two things, and, since you don't, determining the value upfront of that is a complete guess. Which means that, when you agree to a fixed-price term, you are agreeing to take on yourself the risk of the delivery of the project. So it's a transfer of risks. Of course the people that are buying your labour as commodity want to put that risk back on you. They don't want to take the risk so they make you do that, because they can't answer the question of how much does it cost and how long it will take. They want a guarantee of a fixed price and they want you to take all the risk. Which is very unfair because it's their product in the end; the end product is owned by them and not by you. It's a very exploitative relationship to force you to take the risk for capitalizing their product. It's a bad relationship from the beginning. If you're good at estimating and you know your work and your limits and the kind of work you can do, you can make that work, and make a living by being good at this estimates; but still first of all you're taking all the risk unfairly, and second you can't make anything more than a living. While if we're going to build any kind of movement for social change with these new forms of organization, we have to accumulate. Because the political power is an extension of economic power. So if we actually think that our peer production communities are going to have political power and ultimately change society, that can only happen to the degree that we can accumulate. Which means capturing more than the reproduction costs of our labour input, it means actually capturing the full value of our labour's products. The Benjamin Tucker quote I mentioned before is a good way to keep it in mind. *The natural wage of labour is its product.* The natural wage

of labour isn't 40€ an hour, it isn't some arbitrary number. *The natural wage of labour is its product.*
In our case the product is making phone calls. And we don't offer our labour in the form of software development, we are putting together a collective that can do everything, develop a software and bring it to the market. It is actually the consumer making telephone calls that will pay for it. As I said, with it we are not actually making a sustainable living from it right now. We are only building this. We are still making most of our sustenance by selling our labour.

FS *Yeah.*

DK That's where we are starting from. But because we are going for a model where the end product is sold directly to the consumer, there is not mediation. There is no capitalist owners that are buying our labour and owning the product and then selling the product for it's value to the market. We are selling the product directly to the consumers of the product, so there is nothing in-between. And all of the workers that contribute to the making of this product, whether they are programmers or into administration or designers, together own this product and own this company. If you're not selling the product, then what you're selling is behavioural control. If you're not paying for the magazine directly, it is paid for with the money coming from lobbyists or from advertisers that want to control the behaviour of the people perceiving that media, by making them buy some things or vote in a certain way or have a certain image of a certain state department or the role of the state. In the economical model where the actual magazine isn't being sold, where the media is free, in the way television is free, the base of that model is what Dallas Smythe calls 'audience power'. Smythe is one of the main writers about the politically economy of communications, and this is sort of referred to in his 'audience commodity' thing, which is very degraded and unfundamental discourse, but it's related. 'Audience power', ultimately, is just behavioural control. There is money to be made by changing the behaviours of others. And this is the fundamental source of media funding, sometimes it is commercials to sell an actual product by ads and sometimes it is more subtle, like legitimizing a political system or getting people to think favourably about a party or a state department or a government.
All the artists and the designers of the poster and the people that come to the event, they have all kinds of motivations, use value. But the exchange values, where the money comes from, the people buying the checks, what

DK

FS

they are buying is behavioural control, is to be represented in this context. Through their commercial or political or legitimation purposes. The state has legitimation needs, the state needs to be something that is thought of as positive by people. And it does this by funding things that give a legitimacy, like art, culture, social services. What it is buying, is this legitimation. It is behavioural control. When an advertiser sponsors an art show or an event or a television program what they are buying is the chance to make people buy their product. So it is not that every single person, every single artists in the show was thinking about how to manipulate the audience. Not at all, they are just making art … But where the money comes from, what they are actually selling on the market, is behavioural control. It is the so called 'audience power'.

FS *How does that change the work itself you think?*

DK It changes the way you work, a lot. There are so many restrictions and limitations when you work on this model, on capital finance, because the medium is constantly subverted and subjugated by the mediation, the mediation is the message to make it a catch phrase. If you know that your art show is being funded by a certain agency, you're going to avoid talking critically about that agency, because obviously that is going to deny you funding further on. It's clear that the sources of funding affect the actual message that is delivered at the end. It's not possible to have SONY Records sponsor an art show that then tells you how SONY is evil. It is very unlikely that it is going to be funded again, maybe you can trick them once, but it's not going to be sustainable. We were joking before about how my use of anarchist and socialist terminology actually gets the most flak from other people in my own field. That's because they are trying to portray what we do in Free Software development and peer production as being unpolitical. With my saying that no, it's actually quite political, explaining why, they feel like I'm blowing their cover. Like I'm almost outing them as being leftist radicals and they don't want this image because they actually think they can fool this system. Which I think is delusional, I don't think you can fool this system. But that's a very clear example how it does actually change the context and change the message. Because you are always self-conscious of how you're going to pay your rent and how you're going to pay your bills. It's impossible to separate yourself from this context and if the funding is coming from these directions you're always going to self-censor and it's going to affect what you talk about in your choices that you make.

Dmytri Kleiner

Franziska Kleiner

Femke Snelting

What to present, what not to present, where to place the emphasis where not to place the emphasis, it will always be modified by the context you are producing in. And if what you're being paid for is essentially to make people like SONY or make people like the state then it's going to change the way you present what you are doing.

Yochai Benkler used the term 'commons-based peer production' and of course took great pains to avoid talking about communism and try to limit this only to information production. He's very clear, for him this is not for real material production. Because he's a liberal lawyer, working for a major university, in the states … so this is how he presents his work.

But what this means, commons-based production, means that the instruments of production are actually collectively owned but controlled by the direct producers, which means that nobody can actually earn money simply by owning the instruments of production. You can only earn money by employing the instruments of production in actually making something. So, commons-based peer production. You have common things like instruments of production, land and capital, they're are commonly controlled and commonly owned, and individual labour of peers is applied to that shared commons and the results of that labour is then owned by the actual producers. None of that product is owned by the people who are simply owning instruments of production. That is what is meant by commons-based peer production. But that's exactly what the anarchist and the socialist call communism. There is no actual difference. Communism in a text book example is the state less, property-less society. And that's what it means, commons-based peer production is a neologism, a modern way of saying communism because for political reasons, post-war rhetoric, these words are verboten and you can't say them. So people invent new words, but they're saying exactly the same things. The point is that producers require land and capital to produce. If certain private interest controls all of the access of direct producers to land and capital, then those private interests can extract the surplus value. Another great quote from Benjamin Tucker is *whenever one person earns without sweating* … ehm sorry, *whenever one person earns without sweating, another person sweats without earning* and that's fundamentally true. If anybody is earning revenue simply by owning instruments of production, that means that people actually producing are not capturing the value of their labour. And that's what commons-based peer production is. The idea that we have a commons which is all of our property, nobody controls our instruments of production, they're all our property together. Each of us

have our labour and we apply that to the commons and we produce something and whatever we produce, that is ours. It's our own, provided that we are not taking anything away from anybody else, provided that we are not taking any exclusive control of the commons.
In the case of Free Software development, the Free Software itself is a commons. But things that you might make with Free Software are not part of the commons, they're your own. But the problem with software itself is that because software is immaterial and therefore has no reproduction costs, it can be reproduced with no costs, it also has no exchange value. So in order to convert it to exchange value you always have to apply other forms of property: land, capital, hard fixed property … And so, as commons-based peer producers in the Yochai Benkler world, we have our little internal communism, but we can neither live in it nor feed ourselves with it. So in order to actually sustain ourselves, to actually capture our material subsistence, we then have to deal with people that own land an capital; fixed, scarce properties, and we have no leverage in that negotiation. The only things we can get back from the people that consume the output of our labour, is our reproduction costs and nothing more, while they continue to capture and accumulate the extra value. Again, how that applies to design is another thing, I don't think you can isolate one kind of worker from the overall thing. The point is you have to think of where is the value coming from, what are you really selling? Because you're not really selling design, design is an input. What are you really …

FS *What do you mean with 'design is an input'?*

DK Design is an input. The average consumer doesn't buy design. Nobody goes to a store and says *I'd like a design*. They only want the design because they want another product that has design as an input of that product. If you're making beer and you need a label, you find a designer to make the label. But what you're selling is beer, you're not selling design. So you always have to think about what are you really selling. What is the actual product that people is exchanging for, what is the source for the exchange value. And once you identify the source of the exchange value, you have to figure out how to create a direct relationship with all the other producers that are involved in the production cycle.

FK Seems incredibly difficult …

DK If it was easy then capitalism would have been overthrown centuries ago …

Dmytri Kleiner

Franziska Kleiner

Harrison

Peter Westenberg

FK … You're now owning a magazine already with a couple of people. The next person asks you to design a beer label …

DK You have to own the beer factory!

FK … And I think next you should own the paper company that makes …

H *And then you need people and say* I know how to make design, I need some people who know how to make beer. *So then we have a beer factory.*

PW *And then you need people who drink the beer! Who's going to make the people that drink the beer?*

H *Haha.*

FK But wait, there must be a little bit of difference, a modified option to this. For example …

DK In the scenario of commons-based peer production it's not that the designers have to own the beer factory, it's just that there can't be any capitalist in the middle that owns the land, it's enough if the designers and the beer makers both own the land together and the capital together …

FK So if the beer company is also worker-owned and you come to an arrangement … Isn't it the idea of shares? Applying labour and therefore having shares on something …

DK Yes, but it has to be equal. Shares in a capitalist system are unequal. That's the idea of copy-far-left. It's the idea of a public license that allows free use for non-alienated forms of production and denies free use for alienated forms of production. In the case of software, for instance, which is not the greatest application of copy-far-left, but is a good example to understand, the software would be usable by a workers' cooperative for free but a private corporation employing wage labour and private capital couldn't use it for free. They would have to either not use it at all or negotiate a different set of terms under which they could use it. So the question is how do we remove coercive property relationships. If you really have a situation of commons-based peer production, or communism, where there is no state, no property, the instruments of production are collectively owned, people just work together in a very kind of free way, than it could certainly work. But that's not the world we are living in, so we have to be defensive of our commons and how we produce in order for it to grow. We have to think about where the exchange value is and think about where the use value crosses into exchange value and make sure that the point is within our boundary. If we can do that, that's enough. If we have a worker-owned design collective that works with a worker-owned beer company, that's as good as together owning a beer company. But only if they also live on land and apartments that are also worker-owned, because otherwise the landlord will simply capture value; you have to look for the point of leakage. Even with a workers' design company and a workers' beer company living in Brussels renting from capitalist, then the people that own the apartment and the land will simply capture all the surplus value. The surplus value will always leak at the point of scarcity, so the system has to be complete, what Marcel Mauss calls a 'total system'. It has to be a total system, if it is not, if the entire cycle of production doesn't go through commons-based peer production hands, then it's going to leak at the first point of scarcity. Then whoever privately controls the one scarce resource through which all this cycle of production goes through, will capture all the surplus value.

Again, back to our very basic model. The price of anything is its reproduction cost, so the price of something that is immaterial is zero. So, since the beginning of mechanical reproduction, property-based interest groups have tried to create artificial barriers to production. When you have artificial barriers to reproduction the immaterial assets start to behave like material assets; this is where copyright and intellectual property come from. It's the desire of property groups, to make immaterial assets behave price-wise the same as material assets, the only way to do that is creating barriers to reproduction.

Typography obviously comes from this culture, like a lot of other media culture. There is rules about how you can reproduce it, and it creates the opportunity for the owners of these things to capture exchange value. Because the reproduction costs are no longer zero, because of artificial costs of reproduction. But in certain things the capitalists are not homogeneous, there's not just one group of capitalists. There is many different capitalists. Even though some make their living from typography, many more capitalists make their living by using typography, so with typography as an input. From the point of view of those capitalists, the ones trying to restrict the reproduction of typography are a problem. So if they can hire their own staff and develop free typography with other companies, they're not selling typography, that's just an input for them. Like for standardized nuts and bolts, one time this was true too, bolt-makers would make their nuts and bolt not fit, in the sense that if you wanted to use a nut from one company

and a bolt from another you couldn't do so. They tried to create a barrier from this, but since the nuts and bolts industry is not the biggest in capital, because capital itself need nuts and bolts, the other companies got together and said wait a minute, let's just have standardized nuts and bolts, we don't want to make our money from nuts and bolts, we want to make our money off-stream, from the product we make from nuts and bolts. Typography falls into the same system. I imagine most of the people that are creating free typography work for companies and they have their salary paid by companies that use typography, not companies that sell typography. Companies that actually use typography in other production, whether it's publishing or whatever else they're making, so the reproduction costs of the typographers is paid for by not controlling the typography itself, but by employing it in production and using it in another field. The people that are still trying to hold on to typography as a product, as an end product that they capture from intellectual property, are being pushed out.
In other things this is not just the case. If you look at the amount of money that publishing companies spend on QuarkXpress, that's not really a big deal. From their point of view, they can hire some programmers and they can make their own QuarkXpress and work with five other publishing companies, but the amount of money that they spend on QuarkXpress overall, isn't that high …

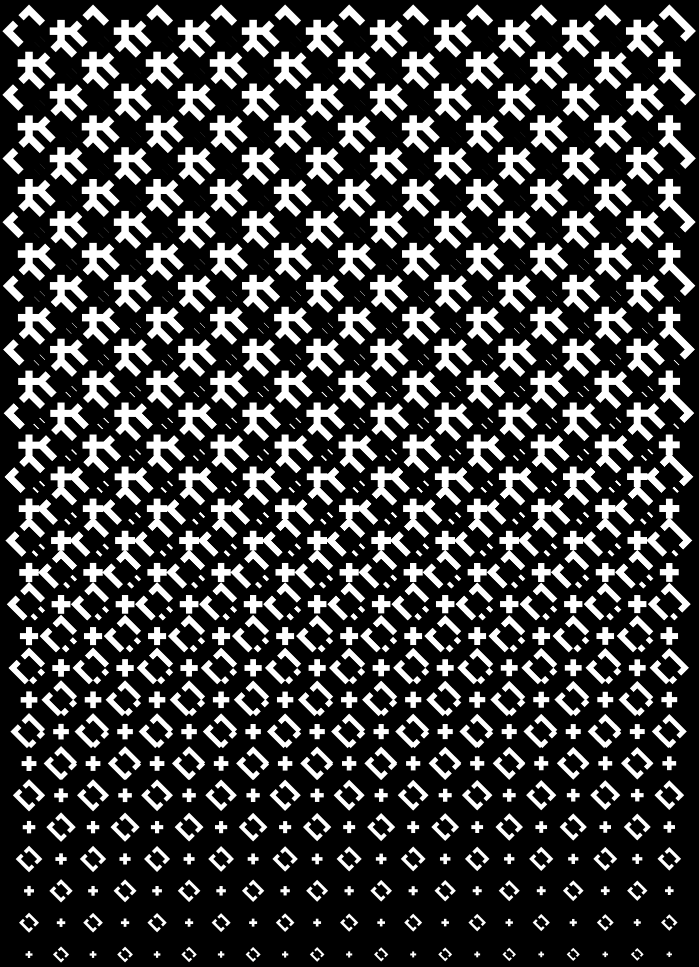
H *Haha.*

DK So the same economy of scale doesn't apply. This is why commercial software is still hanging on in these niche markets where there isn't a broad enough market. It's not a broad enough input so that freedom is supported by the users of it. Typography is a very general input. It's like a nut or a bolt, while QuarkXpress is pretty specific. Franziska was saying that in her publishing company all they really need is two copies, or maybe one even, of the software, and the whole company can work with it. They just go to the computer with it when they need to do the layout, overall it's not a huge cost. They don't need it every time they publish a book. Whether if they had to pay for the font they used and every time they wanted to use a different font, and they had to pay for it again, that would be a problem, so they'd rather use a free font, and if that means hiring somebody to drop the pixels down for a new font once and then having it free forever, it can all make sense. That's why typography is different from software. And so the Scribus project has gone really far but the reason

DK

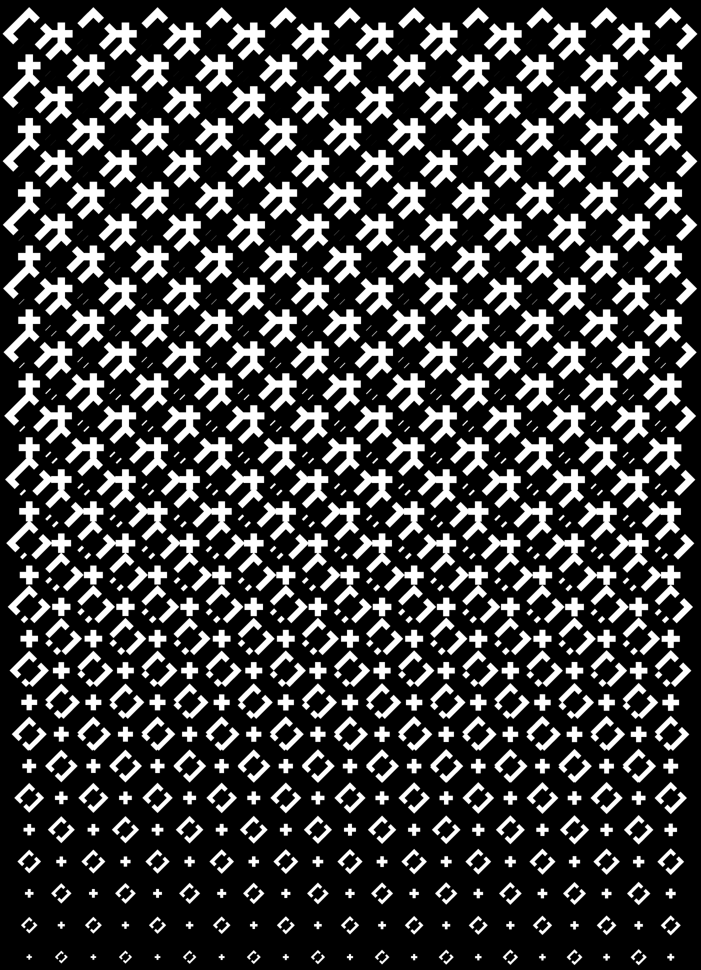
H

it's obscure is because except from the ideological case, they don't have a business case they can make for the publishers. Because for publishers they want a piece of software that works and if it costs 400$ once, who cares. It doesn't really affect their business model. You have to make the case for the publishers that if you form an association of all the publishers and you together develop some new Free Software to do publishing, that would be better and cheaper and faster. Then maybe eventually this case would be made and something like this would exist, but it's not like an operating system or a web browser, that is really used everywhere all the time, and would be really inconvenient to pay for every time. If companies had to pay every single time they put a web browser on their computer, that would be very inconvenient for them. Even Microsoft doesn't dare to charge money for Internet Explorer, cos they know people would just say *Fuck off.* They're not going to buy it. In more obscure areas, like publishing, 3D animation, film and video, it doesn't make so much of a difference. In those business models, for instance 3D animation, one of the biggest companies is Pixar. They make the movies! They don't make the software, they go all the way through the process and they make the movie! So they completely own everything. For that reason it makes sense for them, since they capture the full value of their product in the end, because they make the movies, that their software enables them to make. And this would be a good model for peer production as well, except obviously they're a capitalist organization and they exploit wage labour. But basically if Scribus really wanted to have a financial base, the people that develop Scribus would have to own a magazine that is enabled by Scribus. And if they can own the magazine that Scribus enables then they can capture enough of that value to fund the development of Scribus, and it would actually develop very quickly and be very good, because that's actually a total system. So right from the software to the design, to the journalism, to the editing, to the sale, to the capture of the value of the end consumer. But because it doesn't do that, they're giving Free Software away … To who? Where is the value captured? Where is the use value transferred into exchange value? It's this point that you have to get all the way to, and if you don't make it all the way there, even if you stop a mile short, in that mile all of the surplus value will be sucked out.



# Just Ask and That Will Be That

Asheesh Laroia
Femke Snelting



# Just Ask and That Will Be That

Asheesh Laroia
Femke Snelting

This conversation took place in Montreal at the last day of the **Libre Graphics Meeting 2011**. In the panel *How to keep and make productive libre graphics projects?*, Asheesh had responded rather sharply to a remark from the audience that only a very small number of women were present at LGM: *Bringing the problem back to gender is avoiding the general problem that F/LOSS has with social inclusion.* Another good reason to talk to him were the intriguing 'Interactive training missions' that he had been developing as part of the OpenHatch.org project. I wanted to know more about the tutorials he develops; why he decided to work on 'story manuals' that explain how to report a bug or how to work with version control. Asheesh Laroia is someone who realizes that most of the work that makes projects successful is hidden underneath the surface. He volunteered his technical skills for the UN in Uganda, the EFF, and Students for Free Culture, and is a developer on the Debian team. Today, he lives in Somerville, MA. He speaks about his ideas to audiences at international F/LOSS conferences.

The interactive training missions are really linked to the background of the OpenHatch project itself. I started working on it because to my mind, one of the biggest reasons that people do not participate in Free Software projects, is that they either don't know how or don't feel included. There is a lot you have to know to be a meaningful contributor to Free Software and I think that one of the major obstacle for getting that knowledge, and I am being a bit sloppy with the use of the term maybe, is how to understand a conversation on a bug-tracker for example. This is not something you run into in college, learning computer science or any other discipline. In fact, it is an almost anti-academic type of knowledge. Bug tracker conversations

are 'just people talking', a combination of a comment thread on a blog and actual planning documents. There's also tools like version control, where close to no one learns about in college. There is something like the culture of participating in mailing lists and chatting on IRC … what people will expect to hear and what people are expecting from you.
For people like me that have been doing all these things for years, it feels very natural and it is very easy to forget all the advantages I have in this regard. But a lot of the ways people get to the point where I am now involves having friends that help out, like *Hey, I asked what I thought was a reasonable question on this mailing list and I did not get any answer* or *what they said wasn't very helpful.* At this stage, if you are lucky, you have a friend that helps you stay in the community. If you don't, you fall away and think *I'm not going to deal with this, I don't understand.* So, the training missions are designed to give you the cultural experience and the tool familiarity in an automated way. You can stay in the community even when you don't have a friend, because the robot will explain you what is going on.

✖ *So how do you 'harvest' this cultural information? And how do you bring it into your tool?*

There is some creative process in what I call 'writing the plot'; this is very linear. Each training mission is usually between three and fifteen minutes long so it is OK to have them be linear. In writing the plot, you just imagine what would it take a new contributor to understand not only what to do, but also what a 'normal community member' would know to do. The different training missions get this right to different extents.

✖ *How does this type of knowledge form, you think? Did you need to become a kind of anthropologist of Free Software? How do you know you teach the right thing?*

I spend a lot of time both working with and thinking about new contributions to Free Software. Last September I organized a workshop to teach computer science students how to get involved in Open Source. And I have also been teaching interpersonally, in small groups, for ten or eleven years. So I use the workshops to test the missions and than I simply ask what works. But it is tough to evaluate the training missions through workshops because the workshops are intended to be more interpersonal. I definitely had positive feedback, but we need more, especially from people that have been two or three years involved in the Free Software community, because they understand what it feels like to be part of a community but they may still feel somewhat unsure about whether they have everything and still remember what was confusing to learn.

✖ *I wasn't actually asking about how successful the missions are in teaching the culture Free Software… I wanted to know how the missions learn from this culture?*

So far, the plots are really written by me, in collaboration with others. We had one more recent contribution on Git written by someone called Mark Freeman who is involved in the OpenHatch project. It did not have so much community discussion but it was also pretty good from the start. So I basically try to dump what is in my head?

✖ *I am asking you about this, thinking about a session we once organized at Samedies, a woman-and-Free-Software group from Brussels. We had invited someone to come talk to us about using IRC on the command-line and she was discussing etiquette. She said:* On IRC you should never ask permission before asking a question. *This was the kind of cultural knowledge she was teaching us and I was a bit puzzled… you could also say that this lack of social interfacing on IRC is a problem. So why replicate that?*

In Debian we have a big effort to check the quality of packages and maintaining that quality, even if the developer goes away. It is called the 'Debian QA project' and there's an IRC channel linked to that called #debian-qa. Some of the people on that channel like to say hello to each other and pay attention when other people are speaking, and others said *stop with all the noise*. So finally, the people that liked saying hello moved to another channel: #debian-sayhi.

✖ *Meaning the community has made explicit how it wants to be spoken to?*

The point I am trying to make here, is that I am agreeing to part of what you are saying, that these norms are actually flexible. But what I am further saying, is that these norms are actually being bent.

✖ *I would like to talk about the new mission on bug reporting you said you were working on, and how that is going. I find bug reports interesting because if they're good, they mix observation and narration, which asks a lot from the imagination of both the writer and the reader of the report; they need to think* *themselves in each others place: What did I expect that would happen? What should have happened? What could have gone wrong? Would you say your interactive training missions are a continuation of this collective imaginary work?*

⊏ A big part of that sort of imagination is understanding the kinds of things that could be reasonable. So this is where cultural knowledge comes in. If you program in C or even if you just read about C, you understand that there is something called 'pointers' and something called 'segfaults' and if your program ends in that way, that is not a good thing and you should report a bug. This requires an imagination on the side of the person filing the bug. The training missions give people practice in seeing these sorts of things and understand how they could work. To build a mental model, even if it is fuzzy, that has enough of the right components so they can enter in discussion and imagine what happened.
Of course when there are real issues such as groping at conferences, or making people feel unwelcome because they are shown slides of half-naked people that look like them … that is actually a gender issue and that needs to be addressed. But the example I gave was: *Where are the Indians, where are the Asians in our community?* This is still a confusing question, but not awkward.

✖ *Why is it not awkward?*

⊏ (*laughs*) As I am an Indian person … you might not be able to tell from the transcription?
It is an easy thing to do, to make generalizations of categories of people based on visible characteristics. Even worse, is to make generalizations about all individual people in that class. It is really easy for people in the Free Software community to subconsciously think there are no women in the room 'because women don't like to program', while we know that is really not true. I like to bring up the Indian people as an example because there are obviously a bunch of programmers in India … the impression that they can't program, can't be the reason they are excluded.

✖ *But in a way that is even more awkward?*

⊏ Well, maybe I don't feel it is that awkward because I see how to fix it, and I even see how to fix both problems at the same time.

In Free Software we are not hungry for people in the same way that corporate hiring departments are. We limp along and sometimes one or two or three people join our project per year as if by magic and we don't know how and we don't try to understand how. Sometimes external entities such as Google Summer of Code cause many many more show up at the doorstep of our projects, but because they are so many they don't get any skills for how to grow. When I co-ran this workshop at the computer science department at the University of Pennsylvania on how to get involved in Open Source, we were flooded with applicants. They were basically all feeling enthusiastically about Open Source but confused about how to get involved. 35% of the attendees were women, and if you look at the photos you'll see that it wasn't just women we were diverse on, there were lots of types of people. That's a kind of diversity-neutral outreach we need. It is a self-empowerment outreach: 'you will be cooler after this, we teach you how to do stuff' and not 'we need you to do what we want you to do', which is the hiring-kind of outreach.

✖ *And why do you think Free Software doesn't usually reach out in this way? Why does the F/LOSS community have such a hard time becoming more diverse?*

The F/LOSS community has problems getting more people **and** being more diverse. To me, those are the same problems. If we would hand out flyers to people with a clear message saying for example: here is this nice vector drawings program called Inkscape. Try it out and if you want to make it even better, come to this session and we'll show you how. If you send out this invitation to lots of people, you'll reach more of them and you'll reach more diverse people. But the way we do things right now, is that we leave notes on bug trackers saying: *help wanted*. The people that read bug trackers, also know how to read mailing lists. To get to that point, they most likely had help from their friends. Their friends probably looked like them, and there you have a second or third degree diversity reinforcement problem. But leaving gender diversity and race diversity aside, it is such a small number of people!

✖ *So, to break that cycle you say there is a need to externalize knowledge… like you are doing with the OpenHatch project and with your project 'Debian for Shy People'? To not only explain how things technically work, but also how they function socially?*

## JUST ASK AND THAT WILL BE THAT

I don't know about externalizing … I think I just want to grow our community. But when I feel more radical, I'd say we should just not write 'How to contribute' pages anymore. Put a giant banner there instead saying: *This is such a fun project, come hang out with us on IRC … every Sunday at 3PM.* Five or ten people might show up, and you will be able to have an individual conversation. Quickly you'll cross a boundary … where you are no longer externalizing knowledge, but simply treat them as part of your group.
The Fedora Design Bounties are a big shining example for me. Máirín Duffy has been writing blog posts about three times a year: *We want you to join our community and here is something specific we want you to do. If you get it right, the prize is that you are part of our community.* The person that you get this way will stick around because he or she came to join the community.

✖ *And not because you sent a chocolate cake?*

Not for the chocolate cake, and also not for the 5000$ that you get over the course of a Google summer of code project. So, I question whether it is worth spending any time on a wiki-page explaining 'How to contribute' when instead you could attract people one by one, with a 100% success-rate.

✖ *Writing a 'How to contribute' page does force teams to reflect on what it takes to become part of their community?*

Of course that is true. But compared to standing at a job-fair talking to people about their resume, 'How to contribute' pages are like anonymous, impersonal walls of text that are not meant to create communication necessarily. If we keep focusing on communicating at this scale, we miss out on the opportunity to make the situation better for individual people that are likely to help us.

✖ *I feel that the Free Software community is quite busy with efficiency. When you emphasize the importance of individual dialogue, it sounds like you propose a different angle, even when this in the end has the desired effect of attracting more loyal and reliable contributors.*

It is amazing how valuable patience is.

✖ *You talked about Paul, the guy that stuck around on the IRC channel saying hi to people and than only later started contributing patches after having seen two or three people going through the process. You said:* If we had implied that this person would only be welcome when he was useful … we would have lost someone that would be useful in the future.

The obsession with usefulness is a kind of elitism. The Debian project leader once made this sort of half-joke where he said: *Debian developers expect new Debian contributors to appear as fully formed, completely capable Debian developers.* That is the same kind of elitism that speaks from *You can't be here until you are useful.* By the way, the fact that this guy was some kind of cheerleader was awesome. The number of patches we got because he was standing there being friendly, was meaningful to other contributors, I am sure of it. The truth is … he was always useful, even before he started submitting patches. Borrowing the word 'useful' from the most extreme code-only definition, in the end he was even useful by that definition. He had always been useful.

✖ *So it is an obsession with a certain kind of usefulness?*

Yes.

✖ *It is nice to hear you bring up the value of patience. OSP uses the image of a frog as their logo, a reference to the frog from the fairy tale 'The frog and the princess'. Engaging with Free Software is a bit like kissing a frog; you never know whether it will turn into a prince before you have dared to love it! To OSP it is important not to expect that things will go the way you are used to … A suspension of disbelief?*

Or hopefulness! I had a couple of magic moments … one of the biggest magic moments for me was when I as a high school student e-mailed the Linux kernel list and than I got a response! My file system was broken, and fsck-tools were crashing. So I was at the end of what I could do and I thought: let's ask these amazing people. I ended up in a discussion with a maintainer who told me to submit this bug-report, and use these dump tools … I did all these things and compiled the latest version from version control because we just submitted a patch to it. By the end of the process I had a working file system again. From that moment on I thought: these magic moments will definitely happen again.

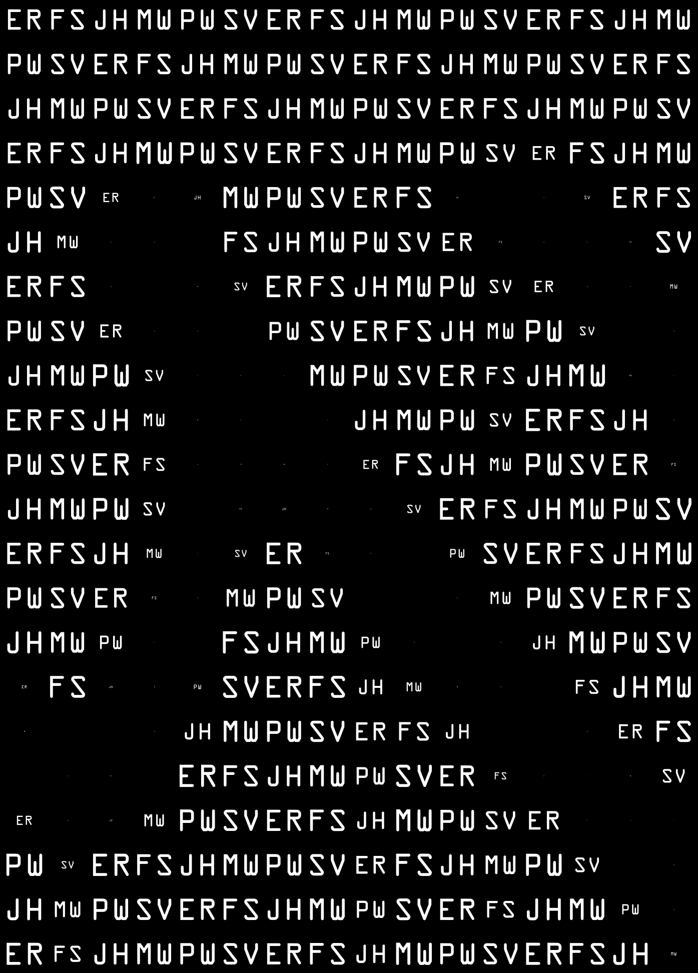
✖ *If you want magic moments, than streamlining the communication with your community might not be your best approach?*

## JUST ASK AND THAT WILL BE THAT

What do you mean by that?

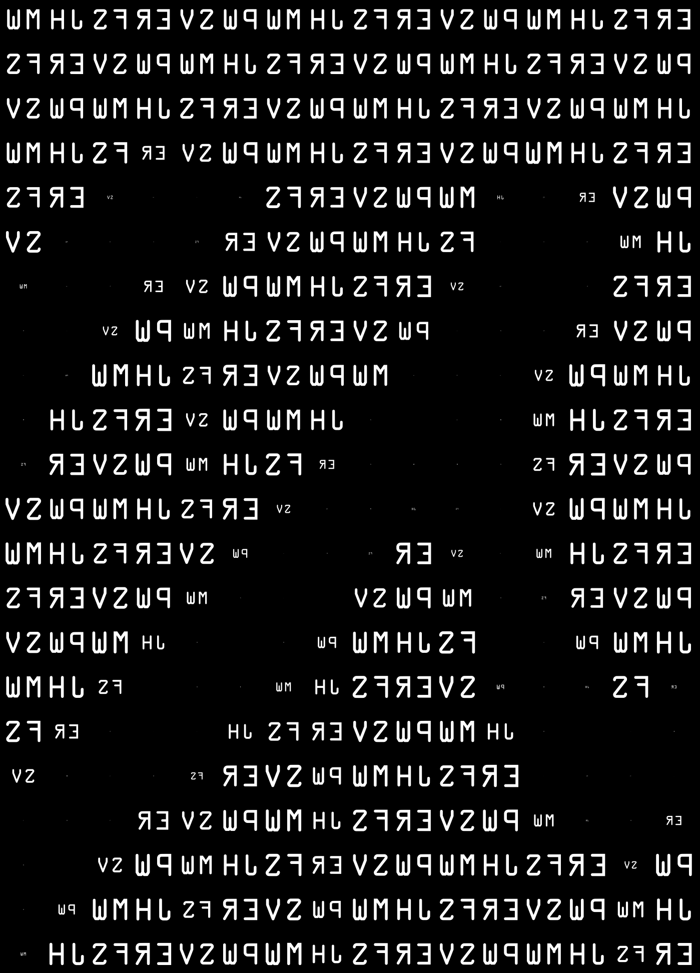
*I was happy to find a panel on the program of LGM that addressed how this community could grow. But than I felt a bit frustrated by the way people were talking about it. I think the user and developer communities around Libre Graphics are relatively small, and all people actually ask for, is dialogue. There seems to be lots of concern about how to connect, and what tools to use for that. The discussion easily drifts into self-deprecating statements such as 'our website is not up-to-date' or 'we should have a better logo' or 'if only our documentation would be better'. But all of this seems more about putting off or even avoiding the conversation.*

Yes, in a way it is. I think that 'conversations' are the best, biggest thing that F/LOSS has to offer its users, in comparison with proprietary software. But a lot of the behavioral habits we have within F/LOSS and also as people living in North America, is derived from what we see corporations doing. We accept this as our personal strategies because we do not know any alternatives. The more I say about this, the more I sound like a hippie but I think I'll have to take the risk (*laughs*).
If you go to the Flash website, it tells you the important things you need to know about Flash, and than you click download. Maybe there is a link to a complex survey that tries to gather data en masse of untold millions of users. I think that any randomly chosen website of a Libre Graphics project will look similar. But instead it could say when you click download or run the software … *we're a bunch of people … why don't you come talk to us on IRC?* There are a lot people that are not in the conversation because nobody ever invited them. This is why I think about diversity in terms of outreach, not in terms of criticizing existing figures. If in some alternate reality we would want to build a F/LOSS community that exists out of 90% women and 10% men, I bet we could do it. You just start with finding a college student at a school that has a good Computer Science program … she develops a program with a bunch of her friends … she puts up flyers in other colleges … You could do this because there are relatively so little programmers in the world busy with developing F/LOSS that you can almost handpick the diversity content of your community. Between one and a thousand … you could do that. There are 6 million thousand people on this planet and the amount of people **not** doing F/LOSS is enormous. Don't wring your hands about 'where are the women'. Just ask them to join and that will be that!



# Tying the story to data

Evan Roth — Stéphanie Villayphiou
Femke Snelting — John Haltiwanger
Peter Westenberg — Michele Walther



# Tying the story to data

Evan Roth  
Femke Snelting  
Peter Westenberg  
Stéphanie Villayphiou  
John Haltiwanger  
Michele Walther

In the **summer of 2010**, Constant commissioned artist and researcher Evan Roth to develop a work of his choice, and to make the development process available in some way. He decided to use a part of his fee as prize-money for The GML-Recorder Challenge, inviting makers to propose an Open Source device 'that can unobtrusively record graffiti motion data during a graffiti writer's normal practice in the city'. In three interviews that took place in Brussels and Paris within a period of one and a half years, we spoke about the collaborative powers of the GML-standard, about contact points between hacker and graffiti cultures and the granularity of gesture.
Based on conversations between Evan Roth (ER), Femke Snelting (FS), Peter Westenberg (PW), Michele Walther (MW), Stéphanie Villayphiou (SV), John Haltiwanger (JH) and momo3010.

## Brussels, July 2010

ER **So what should we talk about?**

FS Can you explain what GML stands for?

ER **GML stands for Graffiti Markup Language [1]. It is a very simple file-format designed for amateur programmers. It is a way to store graffiti motion data. I started working with graffiti writers, combining graffiti and technology back in New York, in 2003. In graduate school, my thesis**

[1] Graffiti Markup Language (.gml) is a universal, XML based, open file format designed to store graffiti motion data (x and y coordinates and time). The format is designed to maximize readability and ease of implementation, even for hobbyist programmers, artists and graffiti writers. http://www.graffitimarkuplanguage.com

**was on graffiti analysis, and writing software that could capture their gestures, to archive motion data from graffiti writers. Back than I was saving the data in an x-y-time array, I was calling them .graph files and I sensed there was something interesting about the data, the visualization of motion data but I had never opened up the project at that time.**
**About a year ago I released the second part of the project, of which the source code was open but the dataset wasn't. In conversation with a friend of mine named Theo[2], who also collaborated with me on the L.A.S.E.R. Tag project[3], he brought up the .graph file again and how we could bring back the file format as a way to connect all these different applications. Graffiti Analysis[4], L.A.S.E.R. Tag, EyeWriter[5] ... so I worked with Theo Watson, Chris Sugrue[6] and Jamie Wilkinson[7] and other people to develop Graffiti Markup Language. It is a simple set of guidelines, basically an .xml file format that saves x-y-time data but does it in a way that is very specifically related to graffiti so there's a drip tag and there's tags related to the size of the brush and to how many strokes you have: is it one stroke or two strokes or three strokes.**
**The main idea is: How do you archive the motion of graffiti and not just the way graffiti looks. There are a lot of people photographing graffiti, making documentaries etc. but there hasn't been a way to archive graffiti in ways of code yet.**

FS What do you mean, 'archive in terms of code'?

ER **There hasn't been a programmatic way to archive graffiti. So this is like taking a gesture and trying to boil it down to a set of coordinate points that people can either upload or download. It is a sort of midpoint between writers and hackers. Graffiti writers can download the software and have how-to guides for how to do this, they can digitize their tags**

2 Theo Watson http://www.theowatson.com

3 In its simplest form, L.A.S.E.R. Tag is a camera and laptop setup, tracking a green laser point across the face of a building and generating graphics based on the laser's position which then get projected back onto the same building with a high power projector. http://graffitiresearchlab.com/projects/laser-tag

4 Graffiti Analysis is a digital graffiti blackbook designed for documenting more than just ink. http://graffitianalysis.com

5 The EyeWriter is a low-cast eyetracking system originally designed for paralyzed graffiti artist TEMPT. The EyeWriter system uses inexpensive cameras and Open Source computer vision software to track the wearer's eye movements. http://www.eyewriter.org

6 Chris Sugrue http://csugrue.com

7 Jamie Wilkinson http://www.jamiedubs.com

**and upload it to an open database. The 000000book-site[8] hosts all this data and some people are writing software for this.**

FS So there are three parts: the GML-standard, software to record and play and than there is the data itself – all of it is 'open' in some way. Could you go through each of them and talk about how they produce uploads and downloads?

ER **Right. It starts with Graffiti Analysis. It is software written in C++ using OpenFrameworks, an Open Source platform designed by artists for visual applications. Right now you can download the recorder app and from that you can generate your own .gml files. And from there you can upload these files into the playback app. In the beginning that was the only Open Source side of the project. Programmers could also make new applications based on the software, which also happened.**
**Last night we met Stéphane Buellet[9] who is developing a calligraphy analysis project and he used Graffiti Analysis as a starting point. I find it exciting when that happens but more often people take the file-format as a starting point, and use it as a jumping-off point for making their own work.**
**Second was the database. We had this file-format that we loosely defined. I worked with Jamie to develop the 000000book site. It is pretty nuts-and-bolts but you can click 'upload' and click on your own .gml files and it will playback in the browser. People have developed their own playback mechanisms, which are some of the first Open Source collaborations that happened around .gml files. There is a user account and you can upload files; people have made image renderers, there are people that have made Flash players, SVG players. Golan Levin has developed an application that converts a .gml file into an auto-CAD format. The 000000book site is basically where graffiti writers connect to developers.**
**In the middle between Graffiti Analysis and database is the Graffiti Markup Language, that I think will have its own place on the web. But sometimes**

8 http://000000book.com. Pronounced: 'Black Book': 'A black book is a graffiti artist's sketchbook. Often used to sketch out and plan potential graffiti, and to collect tags from other writers. It is a writer's most valuable property, containing all or a majority of the person's sketches and pieces. A writer's sketchbook is carefully guarded from the police and other authorities, as it can be used as material evidence in a graffiti vandalism case and link a writer to previous illicit works.'
Wikipedia. Glossary of graffiti — wikipedia, the free encyclopedia, 2014. [Online; accessed 5.8.2014]

9 Stéphane Buellet, Camera Linea http://www.chevalvert.fr/portfolio/numerique/camera-linea

**I see it as one project. One of my interests is in archiving graffiti and all of these things are ways of doing that. It is interesting how these three things work together. In terms of an OS development model it has been producing results I haven't seen when I just released source code.**

FS How do you do that, develop a standard for graffiti?

ER **We started by looking at Graffiti Analysis and L.A.S.E.R. Tag, the apps that were using graffiti motion data. From those two projects I had a lot of experience of meeting graffiti writers as a userbase. When you meet with them, they tell you right away what pieces of the software they think are missing. So from talking with them we developed a lot of features that now are in GML like brushes, drips, line-thickness. Some people had single line tags and some people had multi-line tags so that issue came up because GML tracks both drawing and non-drawing motion so we knew that we needed in the file format to talk about pen up and pen down. I was interested in the connection points between lines also.**
**We tried to keep it very stripped down. From the beginning we knew that people that would participate as developers or anonymous contributors were not going to be the same people that would develop a Linux core. They are students, people just getting into programming or visual programming. We wanted people to be able to double-click a .gml file and than everything should verbally make sense so it is `Begin stroke.` `End stroke.` Anyone with basic programming skills should be able to figure out what's going on.**

FS Did you have any moment where you had to decide: *this does not belong to graffiti* or: *this might be more for calligraphy tracking*?

ER **The only thing that has to be in there is the format in x-y time scenario with some information on drawing and not drawing, everything else is bonus. So if you load an .xml file structured like that, compliant apps will load it in. On top of that, there are features that some apps will want and others not. Keywords are, for example, a functionality that we are still developing applications for. It is there but we are looking for how to use it.**

FS Did you ever think about this standard as a way to define a discipline?

ER **(*laughs*) I think in the beginning it was a very functional conversation. We were having apps running this data and I don't think we were thinking**

**of defining graffiti when we were writing the format. But looking back, it is interesting to think about it.**
**Graffiti has a lot of privacy issues related to it too, right? So we did discuss about what it would mean to start recording geo-located data. There are different interests in graffiti. There is an interest in visuals and in deconstructing characters. Another group is interested in it, because it is a sport and more of a performance art. For this type of interest, it is more important to know exactly where and when it happened because it is different on a rooftop in New York to a studio in the basement of someones house. But if someone realizes this data resulted from an illegal action, and wanted to tie it back to someone, than it starts to be like a surveillance camera. What happens when someone is caught with a laptop with all this data?**

FS Your desire to archive, is it also about producing new work?

ER **I see graffiti writers as hackers. They use the city in the same way as hackers are using computer systems. They are finding ways of using a system to make it do things that it wasn't intended to do. I am not sure graffiti writers see it this way, but I am in this position where I have friends that are hackers, playing around with digital structures online. Other friends are into graffiti writing and to me those two camps are doing the most interesting things right now, but these are two communities that hardly overlap. One of the interests I have is making these two groups of people hang out more. I was physically the person bridging these two groups; I was the nerd person meeting the graffiti writers talking to them about software and having this database.**
**Now it is not about my personal collection anymore, it is making a handshake between two communities; making them run off with each other and having fun as opposed to me having to be there all the time to make introductions.**

FS Is GML about the distribution of signature? I mean: The gestures of a specific person can now be reproduced by a larger community. How does that work?

ER **This is an interesting conversation we should have with the graffiti writers. A tag might be something they have been writing for more than 25 years and that will be very personal to them and the way they write this is because they've written it a million times. So at the one hand it**

**is super-personal, but on the other hand a lot of graffiti writers have no problem sharing this data. To them it is just another tag. They feel like, *I have written this tag a billion times* and so when you want to keep one of them, it is no big deal.**
**I don't think the conversation has gotten as involved as it could have. You set something in motion and cross your fingers hoping that everyone plays nice and things go well and so far that is what has been happening. But you are dealing with people that are uploading something that is super personal to them and I'd be curious to see what happens in the future.**
**The graffiti taxonomy project that I have been doing involves a lot of photos of graffiti. It is a visual studies based on characters, I am shooting thousands of photos of graffiti and I don't have an opportunity to meet with all these writers to ask them if it is OK. So I get e-mails from writers once in a while saying *Hey, you used a photograph of one of my tags* and usually it is them feeling out where my intentions are and where I am coming from.**
**It has taken a long time to gain the trust of the community I am working with. Usually when I am able to explain what I am doing and that everything is released openly and meant to be completely free, so far at least the people I have managed to talk toare OK with it and understand it. Initially when people see something they've made being used by other people, a lot of times it can be a point where a red flag is raised and I am assuming there are more red flags going to go up.**

FS If you upload a .gml file, can you insert a licence?

ER **Not yet. Right now there is not even a 'private mode' on the 000000book site. If you upload, everything is public. There is a lot of interesting issues with respect to the licence that I have been reluctant to deal with yet. Once you start talking too much about it, you will scare off people on either side of the fence. I think that will have to happen at some point but for now I have decided to refer to it as an 'open database' and I hope that people will play nicely, like I said.**

FS But just imagine, what kind of licence would you need?

ER **It might make more sense to go for a media-related licence than for a code licence. Creative Commons licences would lend themselves easily for this. People could choose non-commercial or pure public domain. Does that make sense?**

FS Well, yes but if you look at the objects that people share, we're much closer to code than to a video file?

ER **Functionally it is code. But would a graffiti writer know what GPL is?**

PW I am interested in the apprentice-system you were talking about earlier. Like a young writer learning from someone else they admire. The GML notation of x-y-time might help someone to learn as well. But would you ever really copy someone else's tag?

ER **One of the reasons I think graffiti writing has this history of apprenticeship is because you don't really have a chance to learn otherwise. You don't turn on the TV and see someone else doing it. You only see how it is being written if you see other people actually do it. That was one of the original reasons I started doing graffiti research because, having met with graffiti writers. I thought: it is a dance, it is as much about motion as it is about how the final image is constructed. You can come to a much better understanding about how it is made as opposed to just seeing a photograph of it.**

PW If you want to learn from the person writing, you would need to see more than just the trace of a pen?

ER **Someones tag might look completely different if they had six seconds to make it, they make different decisions. In the first version of the Graffiti Analysis project, I had one camera recorder tracking the pen and another camera behind the hand and another so you could see the full body. But there was something about tracking just the pen tip that I liked. It is an easier point of entry for dealing with the motion data than having three different video feeds.**

FS Maybe it is more about metadata? Not a question of device or application, but about space for a comment.

ER **Maybe in the keywords there will be something like: Rooftop. Brooklyn. Arrested.**
**The most interesting part is often the stories that people tell afterward anyway. So it is an interesting idea, how to tie the story to the data.**
**It is a design problem too. Historically graffiti has been documented many times by outsiders. The movie Style Wars[10] is a good example of**

[10] Style Wars. Tony Silver, 1983. http://www.stylewars.com

**this epic documentary that was made by outsiders that became insiders. Also, the people that have been documenting most of the graffiti are not necessarily graffiti writers.**
**Graffiti has a history with documentarians entering into their community and playing a role but sharing the stories is something writers do internally, not as much to outsiders. How do you figure out a way to get graffiti writers to document their stories into the .gml files themselves, or is it going to take outsiders? How does the format facilitate that?**

FS Do you think the availability of a project like GML can have an impact on the way graffiti is learned? If data becomes available in a community that operates traditionally through apprenticeships and person-to-person sharing, what does it do?

ER **I am interested in Open Source culture being influenced by graffiti, and I am interested in Open Source culture influencing graffiti as well. On a big picture I would love it if the graffiti community got interested in these ideas and had more of a skill-sharing-knowledge-base.**
**KATSU[11], someone I worked with in New York, has acquireda lot of knowledge about how to make tools for graffiti and he initially wasn't so much into sharing them, because graffiti writers tend to save that knowledge for themselves so that their tags are always bigger and better (*laughs*). Talking to him I think I convinced him to write tutorials on how to make some of these tools. On the street art side there is Mark Jenkins[12], he has this technique of making 3D objects that exist within the city and we had a lot of conversations too.**
**There are many ways tech circles and Open Source circles can come together with people that are making things outside, with their hands. I think graffiti can learn from that. In the end people would be making more things outside which would be a good thing.**

FS In a way typography has a similar culture of apprenticeship. Some people enjoy spreading knowledge, and others resist in the name of quality control.

ER **Interesting. I think the work I am doing is such a tangent! In general, for something that is decidedly against the rules, the culture of writing graffiti often has a rigid structure. To people in that community what**

[11] KATSU http://www.flickr.com/search/?q=graffiti+katsu
[12] Mark Jenkins tapesculptures http://tapesculpture.org

**I do is a blip on their radar. I am honored when I get to meet graffiti writers and they are interested in what I am doing but I don't think it will change anything in what is in some ways a very strict system.
And I don't want that either. I like the fact that they found a way to make spraypaint and markers change the way each city in the world looks. They have the tools they need. Digital projectors will not change that. Graffiti writers still like to see their names projected at big scales in new ways but it is not something they really need (*laughs*).**

FS And the other way around? How does graffiti have an influence on Open Source communities?

ER **For the people on the technology side, it is an easy jump. To think about hacking software systems and than about making things outside. I see that with the Free Art and Technology Group[13] that I help run. When they start thinking about projects in the city, it takes little to come up with great ideas. I also see that in the class I teach, Urban Hacking. There is already a natural overlap.**

FS What connects the two?

ER **It is really about the idea of hacking. The first assignment in the class is not to make anything, but simply to identify systems in the city. What are elements that repeat. Trying to find which ones you can slip into. It has been happening in graffiti forever. Graffiti in New York in the eighties was to me a hack, a way to have giant paintings circulating in the city ... There is a lot of room to explore there.**

FS Your experience with the Blender community[14] did not sound like an easy bridge?

ER **Recently I released a piece of software that translates a .gml file and translates it into a .stl file, which is a common 3D format. So you can basically take a graffiti gesture and import it into software like Blender. I used Blender because I wanted to highlight this tool, because I want these communities to talk to each other.
So I was taking a tag that was created in the streets of Vienna and pulling it into Blender and in the end I was exporting it to something that could**

[13] The Free Art and Technology (F.A.T.) Lab is an organization dedicated to enriching the public domain through the research and development of creative technologies and media. Release early, often and with rap music. http://fffff.at

[14] Blender is a free Open Source 3D content creation suite. http://www.blender.org/

**be 3D printed, to become something physical. The video that I posted intentionally showed online showed screenshots from Blender and it ended up on one of the bigger community sites. I only saw it when my cousin, who is a big Blender user, e-mailed me the thread. There is about a hundred dedicated Blender users discussing the legitimacy of graffiti in art and how their tools are used [15]; pretty interesting but also pretty conservative.**

FS Why do you think the Blender community responded in that way?

ER **It doesn't surprise me that much. Graffiti is hard to accept, especially when we are talking about tags. So the only reason we might be slightly surprised by hearing people in the Open Source community react that way, is because intellectual property doesn't translate always to physical property. Writing your name on someone's door is something people universally don't like. I understand. For me the connection makes sense but just because you make Open Source doesn't mean you'll be interested in graffiti or street art or vice versa. I think if I went to a Blender conference and gave a talk where I explained sort of where I see these things overlap, I could make a better case than the three minute video they reacted to.**

FS What about Gesture Markup Language instead of Graffiti Markup Language?

ER **Essentially GML records x-y-time data. If you talk about what it functionally does, it is probably more related to gesture than it is to graffiti. There is nothing at the core specifically related to graffiti. I am interested in branding it in relation to graffiti and to get people to talk about Open Source where it is traditionally not talked about. To me that is interesting. It is a way to get people excited about open data, and popularizing ideas about Open Source.**

FS Would you be OK if it would get more popular in non-graffiti circles?

ER **I am super excited when I see it used in bizarre places. I'll keep using it for graffiti, but someone e-mailed me that they were upset that it only tracks one point. There hasn't been a need to track multiple tags at once. They wanted to use it to track juggling, but how to track multiple balls in the air? I keep calling it Graffiti Markup Language because I think it is a good story.**

[15] http://www.blendernation.com/2010/07/09/blender-graffiti-analysis

PW What's the licence on GML?

ER **We haven't really entered into that. Why would you need a licence on a file format?**

FS It would prevent that anyone could own the standard.

ER **That sounds good. Actually it would be interesting for the project, if someone would try to licence it. Legal things matter, but for the things I do, I am most of all interested in getting the idea across.**

FS I am interested in the way GML stems from a specific practice. How it is different and similar to large, legal, commercial, global standardization practices. Related, how can GML connect to other standard practices? Could it be RDF compliant?

PW **Gesture recognition to help out the police?**

FS Or maps of places that are in need of some graffiti? How to link GML to other types of data?

ER **It is hard for me to imagine something. But one thing is interesting for example, how GML is used in the EyeWriter project. It has not so much to do with gesture, but more with how you would draft in a computer. TEMPT is plotting points, so the time data might not be so interesting but because it is in the same format, the community might pick it up and do something with it. All the TEMPT data he writes with his eyes and it is uploaded to the 000000book site automatically. That allowed another artist called Benjamin Gaulon[16] who I now know, but didn't know at the time, to use it with his Print Ball project. He took the tag data from a paralyzed graffiti writer in Los Angeles and painted it on a wall in Dublin. Eye-movement translated into a paint-ball gun ... that is the kind of collaboration that I hope GML can be the middle-point for. If that happens, things can start to extrapolate on either end.**

FS You talked about posting a wish-list and being surprised that your wishes were fulfilled within weeks. Why do you think that a project like EyeWriter, even if it interests a lot of people, has a hard time gathering collaborators, while something much more general like GML seems to be more compelling for people to contribute to?

[16] Benjamin Gaulon, Print Ball
http://www.eyewriter.org/paintball-shooting-robot-writes-tempt1-tag

ER **I'll answer that in a second, but you reminded me of something else: because EyeWriter was GML based, a lot of the collaborations that happened with people outside of the project were GML related, not EyeWriter related. So we did have artists like Ben and Golan take data drawn by TEMPT and do completely different things which made TEMPT a collaborator with them in a way. The software allowed him to share his work in a format that allowed other people to work with him. The wish-list came out of the fact that I was working on a graffiti related project that had a lot of use but not a lot of innovation. Not so many people were using it in ways I wasn't expecting, which is something you always hope of course. By saying: *Here's the things I really would like to happen*, things started to happen. I have been surprised how that drove momentum. Something similar I hope will happen to the work we will do together in the next months too!**

FS What are you planning to do?

ER **We are planning to make a dedicated community page for the graffiti markup language which is one of the three points of the triangle. The second step would be a new addition to the wish-list, a challenge with a prize associated to it which seems funny. The project I'd like to concentrate on is making the data collection easier so that graffiti writers can be more active in the upload sense. Taking the NASA development model: Can you get into orbit on this budget?**

FS How is that different from the way you record graffiti motion at the moment?

ER **If I go out with a graffiti writer, I'm stuck standing with a laptop and a camera facing the wall and then the graffiti writer needs to have a really bright light attached to the writing device which is a bit counter-intuitive when you are trying to do something without being seen (*laughs*). It could be infrared by the way, that could be the first step but then security cameras would still pick it up. The design I am focusing momentum on is a system that's easier. A system that can work without me there, without having to have a laptop there. The whole idea is that it would be a natural way to get good data, to document graffiti without a red-head holding a laptop following you around the whole time!**

## Paris, December 2010

FS How is it to be the sole jury member?

ER **I tried to get another jury-member on there actually. Do you know Limor Fried? She runs Adafruit Industries.[17] I really like her work. She works with her partner Phil Torrone who runs Make Blog.[18] I invited her to be the second jury-member because she makes Open Source hardware kits; this is her full-time thing. She is very smart and has a lot of background in making DIY kits that people actually build. She is also very straightforward and very busy, so she wrote back and said: this is too much work. No.**
**So ... yeah, I am the only jury member. Hmmm.**

SV Is the contest already over?

ER **It is not over. It was easy to launch; I tried to coincide it with the launch of the website and there were a couple of things going on at the same time. The launch helped spread the word about this file format, and people making projects, and vice versa.**

FS Did you have any proposals that came close to meeting the challenge? Did you consider giving out the prize?

ER **No.**
**There are a couple of people that got really close. The interesting thing that is happening with the challenge is something that is also happening to other high barrier projects: You end up speaking to the people you already work with the most. I have a hard time figuring out to some extent what is really happening, but the things I hear, of people making progress, is people that are close to me. It reminds me of the EyeWriter project where people that are to dip their toes into this, are already in the friend group, or one level removed. They are pretty high level programmers.**
**I didn't really think that actual money would be such an incentive but more that it would make the challenge feel serious, more in the sense of an organization that has some kind of club behind it. If you solved one of the design problems by the Mozilla community you could receive**

[17] Limor Fried, Adafruit Industries http://www.adafruit.com
[18] Phillip Torrone, Makezine http://makezine.com/pub/au/Phillip_Torrone

**kudo's from the community, but if you solved one of my projects, you don't really get kudo's from my community, do you?**
**Having the money associated makes it this big thing. At Ars Electronica and so on, it got people talking about it and so it is out there. That part worked. Beyond that it has been a bit hard to keep the momentum. Friends and colleagues send me ideas and ask me to look at things, but people I don't know are hard to follow; I don't think they are publishing their progress. There is a hackerspace in Porto that has been working on it, so I see on their blog and Twitter that they are having meetings about this and are working on it.**

FS Don't you think having only one prize produces a kind of exclusivity? It seems logical **not** to publish your notes?

ER **Maybe. Kyle [19] has been thinking up ways to do it and I know he wanted to use an optical mouse, and then this a friend Michael [20] has been using sensors, and he ran into a software problem but had the hardware problem more or less solved. And then Kyle, a software expert, has been running into hardware problems and so I kind of introduced them to each other over e-mail so I don't know if they are working on it together.**

FS Would you consider splitting the prize?

ER **I don't care, but I don't know if the candidates would consider splitting the prize! I know Michael has already spent a lot of money because he has been buying Arduinos and other hardware. He wants to make a cheap version to solve the problem and then make another one that costs 150€ on top of the price limitation to make it easier to use. He is spending a bunch of money so even if he wins, it is going to get him only out of the hole and he will not have much left.**
**Actually, Golan [21] had an idea for an iPhone app that he wants to make but I am not sure it solves it.**

FS Why don't you think his app will solve it?

ER **He is really interested in making something where you do not need to meet with the graffiti writer. His idea was that if you could take a photo of it on the wall, and then with your finger you guide it for how it**

[19] Kyle McDonald http://kylemcdonald.net
[20] Michael Auger http://lm4k.com
[21] Golan Levin http://www.flong.com

**was written. It has an algorithm for image processing and that combined with your best guess of how it was written would be backed out in motion data. But it is faked data.**

FS That it is really interesting!

ER **Yes it is and I would love it if he would make it but I am not going to let him win with it (*laughs*). I understand why he wants to do it; especially if you are not inside the graffiti community, your only experience is what you see on the wall and you don't know who these people are and it is going to be almost impossible to ever get data for those tags. If you don't have access to that community you are never going to get the tag of the person that you really want. I like the idea that he is thinking about getting some data from the wall as opposed to getting it from the hand.**

FS Learning by copying. Nowhere near solving the challenge, but interesting. OSP[22] we were discussing about the way designers are invited into Open Source Software by way of contest. Troy James Sobotka[23] got angry and wrote: *We want to be part of this community, we don't want to compete for it.*

ER **With the EyeWriter project, we were thinking a lot about that; how to spur development. I think I would not have done a competition with the EyeWriter. Making it fun, that is what makes it happen. If it would be a really serious amount of money, with people scraping at each other, fighting each other ...**
**For me, the fact that there is prize money makes something that is already ridiculous in itself even more funny. To have prize money for such a small community of people that are interested in coding and in graffiti. I'm not seriously thinking that we can spur development with this kind of money. To use the EyeWriter as an example, we've had money infusions from awards mostly and we had to think about how we could use that money to get from point A to point B. That's also a project where we had very**

---

[22] OSP (Open Source Publishing) is a graphic design collective that uses only Free, Libre and Open Source software. http://ospublish.constantvzw.org

[23] *The very notion of Libre / Free software holds cooperation and community with such high regard you would think that we would be visionary leaders regarding the means and methods we use to collaborate. We are not. We seem to suffer from a collision of unity with diversity. How can we more greatly create a world of legitimate discussion regarding art, design, aesthetic, music, and other such diverse fields when we are so stuck on how much more consistent a damn panel looks with tripe 22 pixel icons of a given flavour?*
http://www.librescope.com/975/spec-work-and-contests-part-two

**definable design goals of what we wanted to reach, especially between the first version and where we are now with the second version.**

FS How did that work?

ER **We are not talking about a ton of money here, 10 to 20.000 €, and we tried to get as far as we could. We got almost no work done between the meetings in LA but if we flew in, it was OK to take a week out of our schedules and really hammer at it. We were trying to think how we could do the same thing for people that we wanted to work with and who we had met in conferences. So that is how we thought of spending that money.**
**The other way we use money in the EyeWriter project is that we buy people kits. We know a few people that are interested in hacking on it but they don't have the hardware. Not that they are so expensive, but Zach wants to buy twenty or thirty unpackaged kits and he has interns working with him in New York helping to build them. So we have these systems ready so as soon as someone wants to get hacking on it, we can mail them a working system that they can just plug in and they don't have to waste their time ordering all these parts from all these websites all over China. And when they are done, they just send it back.**

FS You talked about some things in the challenge that worked and some that didn't.

ER **I think the forum is the obvious thing that did not work. I have friends working on OpenFrameworks, it is headed primarily by Zach and Theo. When you see that forum, it is very involved. It is a deep system, with many different libraries and lots of code flying around. GML is really not large enough.**
**I think what makes sense for this project is when I post news about the project, I see it ripple in Google Alerts. For people working on it, having a place where these things show up is already a lot. The biggest success is the project space, to see all the projects happening.**

FS What happened on the site since we talked?

ER **A project I like, is kml2gml[24] for example. It is done by a friend from Tokyo. He was gathering GPS data riding his bike around various cities, and building up a font based on his path. I like projects like this, where**

[24] Yamaguchi Takahiro http://www.graffitimarkuplanguage.com/kml2GML

**someone takes a work that is already done and just writes an application to convert the data into another format. To see him riding his bike played back in GML was really nice. It is super low barrier to entry, he already did all the hard work. I like that there is now a system for piping very different kinds of data through GML.**

FS But it could also work the other way around?

ER **Yeah. This is maybe a tangent but depending on how someone solves the GML challenge ... I was discussing this with Mike (the person that is developing the sensor based version). He was thinking that if you would turn on his system, and leave it on for a whole night of graffiti writing, you would have the gestural data plus the GPS data. You could make a .gml file that is tracking you down the street, and zoom in when you start making the tag. Also you would get much more information on 3D movement, like tilt and when the pen is picking up and going down. Right now all I am getting is a 2D view through video data. I am really keeping my fingers crossed. But he ran into trouble though.**

FS Like what?

ER **I have my doubts about using these kind of sensors, because 'drift' is a problem. When you start using these sensors too long, it tends to move a little bit. I think he is working within a 0.25 inch margin of error right now, which is right on the edge. If you are recording someone doing a big piece, this is not going to ruin my day too much but if you record a little tag than it is a problem.**
**The other problem is that you need to orient the system before you start tagging. It needs to know what is up and down, you have to define your plane of access. I don't really understand this 100% but he thinks he can still fit it all within the ten second calibration requirement, he's thinking that each time you come to a wall, you tap once, you tap twice and tap a third time to define what plane you are writing on and that calibrates the 3D space. Once you have that calibration done, you can start writing. It is not as easy as attaching a motion sensor. The problem is hard.**

FS So you need to touch the wall before writing on it, feeling out the playing field before starting! It is like working on a tablet; to move from actual movement to instruction; navigation blends into the action of drawing itself.

ER **I like that!**

SV The guy using the iPhone did not use it as a sensor at all?

ER **Theo was interested in using the iPhone to record motion data in GML, but also to save the coordinates so you could try it into a Google Earth or something but he had trouble with the sensitivity of the sensor. Maybe it is better now but you needed to draw on a huge scale for one letter. You could not record anything small.**

FS But it could be nice if you could record with a device that is less conspicuous.

ER **I know. I have just been experimenting with mounting cameras on spray-cans. A tangent to GML, but related. It is not data, but video.**

FS What do you think is the difference between recording video, and recording data? You mentioned that you wanted to move away from documentation the image to capture movement. Video is somehow indirect data?

ER **Video is annoying in that it is computationally expensive. In Brazil[25] I have been using the laptop but the data is not very precise.**
**Kyle thinks he might be able to back out GML data from videos. This might solve the challenge, depending on how many cameras you need and how expensive they are. But so far I have not heard back from him. He said it needs three different cameras all looking at the wall. I mean: talk about computationally expensive! He likes video-processing, he knows some Open Source software that can look for similar things and knows how to relate them. To me it seems more difficult than it needs to be (*laughs*).**

FS It is both overcomplicated and beautiful, trying to reverse engineer movement from the image.

ER **I am getting more into video myself. I get more enjoyment from capturing the data than from the projections, like what most people associate with my work.**

FS Why is it so much more interesting to capture, rather than to project?

ER **In part because it stays new, I've been doing those projections for a while now and I know what happens at these events. For a while it was very new, we just did it with friends, to project on the Brooklyn bridge**

[25] Graffiti Analysis: Belo Horizonte, Brazil 2010 http://vimeo.com/16997642

**for example. Now it has turned into these events where everyone knows in advance, instead of just showing up at at a certain time ate a set corner. It has lost a lot of its magic and power.**
**Michele and I have done so many of these projections and we sort of know what to expect from it, what questions people will ask. When I meet with graffiti writers, that almost always feels new to me. When we went to Brazil, we intentionally tried to not project anything but to spend as much time as possible with writers. Going out with graffiti writers to me always feels right.**

FS Is the documentation an excuse to be taken along, or is the act of documenting itself interesting to you?

ER **To me documentation is interesting. I don't know where all of this is going right now, I am just trying to get the footage; I put these pieces together showing all this movement but I don't really know what the final project is. It is more about collecting data so I am interested in having video, audio and GML that can be synced up, and the sound from these microphones is something to do something with later. This is research for me. I like the idea of having all this data related to a 10 second gesture. I am thinking that in the future we can do interesting things with it. I am even thinking about how the audio could be used as a signal to tell you what is drawing and what is not drawing. It is a really analog way of doing it, but in that way you don't need a button where you are getting true and false statements for what is drawing and what is not drawing; you can just tell by the sound:**
**tfffpt ... tfffpt.**

FS You can hear the space, and also the surface.

ER **I got started doing this because I love graffiti and this is a way to get closer to it again. Like getting back out to the streets and having very personal relationships to the graffiti writers and talking to them, and having them give feedback. I think that is how the whole challenge started. It didn't start because I was projecting, but because I was out on the street and testing the capture, having graffiti writers nearby when it is happening. It feels like things are progressing that way.**

FS Are you thinking of other ways of capturing? You talk about capturing movement, but do you also archive other elements? Do you take notes, pictures? What happens to the conversations you are having?

ER **I have been missing out on that piece. It is a small amount of time we have, and I am already trying to get so much. I am setting up a camera that shoots straight video from a tripod, I am capturing from the laptop and I am also screencasting the application, my head is spinning. One reason I screwed up this footage in the beginning is because with all these things going on I forget to turn on some things. Maybe someone will solve this challenge.**

FS Are you actually an embedded anthropologist?

ER **In the back of my head I am thinking this will become a longer documentary. I like to experiment with documentation, whether that is in code or with video. I do think that there is this interesting connection between documentation and graffiti and how these two things overlap. I am always thinking about documentation. The graffiti writer that was in Vienna[26] showed me a video that was amazing. It was him and a friend going out on a sunny day at 15:30 in the afternoon with two head mounted cameras, bombing an entire train and you hear the birds singing and you only experience it by these two videos that are linked. There are interesting constraints: your hands are already full, you don't want peoples' faces on camera so the head-mounted cameras were smart. Unless you walk in front of a mirror (*laughs*).**

FS Is it related to the dream of 'self documenting code'?

ER **I like that. Even doing the challenge is in a way a reflection on this, how I am fighting to get GML back to the streets somehow, it has a natural tendency to get closer to the browser, to the screen, and my job is to get it back to the street. It is so sexy and fun and flashy and that is important too. My job is to keep the graffiti influence on it as large as the other part.**

FS Is any of this reflected in the standard itself?

ER **I haven't looked at the standard for a while now.**

FS I was thinking again about live coding and notation. Simon Yuill[27] describes notation as a shared space that allows collaboration but also defines the end of a collaboration.

[26] momo3010 http://momo1030.com
[27] Simon Yuill. All problems of notation will be solved by the masses. Mute Magazine, 2008

ER **Maybe using an XML-like structure was a bad idea? Maybe if I had started with a less code-based set of rules? If the files were raw video, it would encourage people to go outside more often? By picking XML I am defining where the thing heads in a way. I think I am OK in the role of fighting that tendency. It is not just a problem in GML but with a lot of work I have been doing with graffiti and technology and even way back with Graffiti Analysis, before GRL (Graffiti Research Lab), the idea was always to keep the research very close to the people doing graffiti. I was intentionally working with people bombing a lot and not with graffiti celebrities. I wanted to work with who's tag was on my mailbox, who's tag do I see a million times when I walk down the street. Since then a lot has happened, like with more popular projects such as L.A.S.E.R. Tag, and it goes almost always further away from graffiti. Maybe that is a function of technology. Technology, or the way it is now, will always drift towards entertainment uses, commercial uses.**

FS Do you think a standard can be subversive? You chose XML because it is accessible to amateur programmers. But it is also a very formal standard, and so the interface between graffiti writers and hackers is written in the language of bureaucracy.

ER **(*laughs*) I thought that there was something funny with that. People that know XML and the web, they get the joke that something so rigid and standardized is connected to writing your name on the wall. But to be honest, it was really just a pragmatic choice.**

SV It reminds me of an interview[28] with François Chastanet who wrote a book[29] about tagging in Los Angeles. He explains that the Gothic lettering is inspired by administrative papers!

FS **I am wondering whether you're thinking about the standard itself as a space for hacking?**

ER Graffiti is somehow coded in-itself. Do you mean it would be interesting to think how GML could be coded in a way for graffiti writers, not for coders?
There would be more space for that when more people start to program at a younger age? When it is more common knowledge. If I would start to do

[28] Interview with François Chastanet http://www.youtube.com/watch?v=ayPcaGVKJHg
[29] François Chastanet, Cholo writing: Latino gang graffiti in Los Angeles. Dokument, 2009

that now, I would quickly lose my small user-base. I love that idea though; the way XML is programmed fits very much to the way you program for the web. But what if it was playing more with language, starting from graffiti which is very coded?

ER **When I was in college, I was always thinking about how to visualize motion in print. I was looking for ways people had developed languages for different ways of writing.**

SV Maybe you could look at the Chinese methods for teaching writing, because the order of the strokes is really important. If you make the stroke from bottom to top, and not from top to bottom, it is wrong.

ER **A friend in Hong Kong, MC Yan, loves the Graffiti Analysis project because it shows the order in which he is writing and he likes to play with that. So he writes words in different order than people are used to and so it changes the meaning. People can not only watch the final result, but also the order which is an interesting part of the writing process. The brush, the angle, direction: depicting motion!**
**In the beginning of the Graffiti Analysis Research project I was very against projection, because I felt that was totally against the idea of graffiti. I was presenting all of these print ideas and the output would be pasted back into the city because I was against making an impermanent representation of the data. In the end Zach said, you are just fighting this because you have a motion project and you want to project motion and then I said alright, I'll do a test. And the tests were so exciting that I felt OK with it.**

FS In what way does GML bridge the gap between digital drawing and hand writing? Could you see a sort of computer-aided graffiti? Could you see computation enter graffiti?

ER **Yeah. When you are in a controlled environment, in a studio, it is easy but the outdoors part always trips me up. That is why the design constraints get interesting, playing in real time with what someone is writing. I think graffiti writers would be into that too. How to develop a style that is unique enough to stand out in an existing canon is already hard enough. This could give someone an edge.**

ER I think the next challenge I'd like to run is about recreating the data outside. I've been thinking about these helicopters with embedded wireless

camera's, have you seen them? The obvious thing to me would be uploading a .gml file to one of these helicopters that is dripping paint on a rooftop. Scale is so important, so going bigger is always going to be better.
Gigantic rooftop tags could be a way to tie it back to the city, give it a reason? I am thinking of ways to get an edge back to the project. The GML-challenge is already a step into that direction; it is not about the prettiest screensaver. To ask people to design something that is tying back to what graffiti is, which is in a way a crime.
I think fixing the data capture is the right place to start, the next one could be about making marks in the city. Like: the first person to recreate this GML-tag on the roof of this building, that would be fun. The first person that could put this 'Hello World' tag onto the Brooklyn bridge and get a photo of it gets the prize. That would get us back to the question of how we leave marks on the surface of the city.

FS **When you capture data of an individual writer in a certain standard, it ends up as typography?**

ER That's another trend that happens when designers look at graffiti, and I've fallen into this too sometimes, you want to be able to make fonts out of it. People have done this actually; there's a project in New York where they met with pretty influential graffiti writers and asked them to write in boxes, the whole alphabet, and I think there's something interesting there.
The alphabet that you saw the robot write was drawn by TEMPT with the EyeWriter and what he did was a little bit smarter than other attempts by graffiti writers to make fonts. He intentionally picked a specific style, the Cholo style, and the format is very tall, vertically oriented, angled. That style is less about letter connections and pen-flow. What graffiti has developed into, and especially tags, is very much about how it is written and the order of the letters. When TEMPT picked this style he made a smart decision that a lot of people miss when you make a font, you miss all the motions and the connections.

SV **What if a programmer could put this data in a font, and generate alternating connections?**

ER That kind of stuff is interesting. It would help graffiti writers to design tags maybe?
To get my feet wet, I designed a tag once, and it was so not-fun to write! I was thinking about a tag that would look different and that would fit

into corners, I was interested in designing something that wasn't curved; that would fit the angles of the city, hard edges. So I had forgotten all my research about drafting and writing. I think I stopped writing in part because the tag I picked wasn't fun o write. For a font to work like writing, it is not just about possible connections between lines. You'd need another level in the algorithm, the way the hand likes to move.

FS **It would be a good algorithm to dream up. It was beautiful to see a robot write TEMPT's letters by the way.**

ER When TEMPT saw the robot writing for the first time, his reaction was all about the order of how the letters were constructed. The order is I think defined by the way he dropped the points in with the EyeWriter software. When he was writing with his eyes, he ended up writing in the same way as he would have written with his hands. When he saw the video with the robot, it freaked him out because he was like: *That's how my hand moved when I did that tag!*

## The Graffiti Markup Field Recorder challenge

An easily reproducible DIY device that can unobtrusively record graffiti motion data during a graffiti writer's normal practice in the city.[30]

### Project Description and Design Requirements:

The GML Field Recorder Challenge is a DIY hardware and software solution for unobtrusively recording graffiti motion data during a graffiti writer's normal practice in the city. The winning project will be an easy to follow instruction set that can be reproduced by graffiti writers and amateur technologists. The goal is to create a device that will document a night of graffiti bombing into an easily retrievable series of Graffiti Markup Language (.gml) files while not interfering with the normal process of writing graffiti. The solution should be easy to produce, lightweight, cheap, secure, and require little to no setup and calibration. The winning design solution will include the following requirements listed below:

– Material costs for the field device must not exceed 300 €.

ER **300€ even felt expensive to me. How can this be a tool that is really accessible? If it goes over a certain price point, it is not the kind of thing that people can afford to make. It is a very small community, a lot of the people that are going to have enough interest to build this are not going to have a background in engineering, and are probably not even a part of the *maker* scene that we know. The audience here might not be people that are hanging out on Instructables. I wanted to make sure that the price point meant that people could comfortably take a gamble to make something for the first time. But I also did not want to make it so small that the design would be impossible.**

[30] GML-recorder challenge as published on: http://www.graffitimarkuplanguage.com/challenges

- Computers and equipment outside of the 300 € can be used for non-field activities (such as downloading and manipulating data captured in-field), but at the time of capture a graffiti writer should have no more than 300 € worth of equipment on him or herself.

ER I was trying to think of how the challenge could be gamed… I did not want to get into a situation where we were getting stressed out because some smart hacker found a hole in the brief, and bought a next generation iPhone that somehow just worked. I didn't want to force people to buy expensive equipment. This line was more about covering our own ass.

- The graffiti writer must be able to activate the recording function alone (i.e., without assistance from anyone else).

FS **Are you going to be out of work soon?**

ER Thinking selfishly, I screw up on documentation a lot because I have too many hats. When I'm going out doing this, I am carrying a laptop, a calibration set up, I also have one video-camera on me that is just documenting, I have another one on a tripod, and I am usually screen capturing the software as it processes the video-footage because it tells another story. I screw up because I forget to hit stop or record. If the data-capture just works, I can go have fun getting good video-footage.

FS **What if it had to be operated by more than one person? It is nice how the documentation now turns the act of writing into a performance-for-one.**

ER If you record alone, the data becomes more interesting and mysterious, right? I mean, no one else has seen it. Something captured very privately, than gets potentially shared publicly and turned into things that are very different. I also thought: you don't want to be dependent on someone else. It is a lot to ask, especially if you are doing something illegal.

– Any setup and/or calibration should be limited to 10 seconds or less.

ER **This came out of me dealing with the current system. It feels wrong that it takes ten to fifteen minutes to get it running. Graffiti is not meant to be that way. This speaks to the problem of the documentation infringing on the writing process, which ideally wouldn't happen. The longer the set-up takes, the more it is going to influence the actual writing. It is supposed to be a fly on the wall.**

FS Does it scale? Does a larger piece allow longer callibration -time?

ER **That's true. But I think this challenge is really about recording tags.**

– All hardware should be able to be easily concealed within a coat with large pockets.

ER A hack to get around that would have been to design a jacket with ten gallon pockets!
I put it there again, to make the device not be intrusive. A big part of graffiti writing is about gaining entry and you limit where you can go depending on how much equipment you have. How bulky it is, what walls you can get up, what holes you can get through.

– The winning solution should be discrete and not draw any added attention to the act of graffiti writing.

ER **It's part of the same issue, but this one also came out from me going out and trying to capture with a system where it requires you to attach a flashlight to a graffiti implement. I didn't want anyone solving the problem and then, Step one is: 'Attach a police siren to a spraypaint can'**

– The resulting solution should be able to record at least 10 unique GML tags of approximately 10 seconds each in length in one session without the need for connecting to or using additional equipment.

ER I wasn't thinking this was going to be an issue in terms of memory-storage, but maybe in terms of memory management. I did not want the graffiti writer to behave as if he was on vacation with a camera that could take only three photos. I wanted to make sure they were not making decisions on what they were writing based and how much memory they had.

– All data recorded using the field recorder should be saved in a secure and non-incriminating fashion.

ER **(*laughs*) If I had to do that one again, I would have put that in Bonus category actually. That's a difficult question to ask. What does secure mean? It seems a bit unfair, because it doesn't fit in to the way graffiti is currently documented. There's not a lot of graffiti writers that currently are shooting encrypted photos and videos, right?**
**But whatever bizarre format comes out from the sensor will help. I don't think that the NYPD will have time or make the effort to parse it. They'd just have a file with a bunch of numbers. Time stamped GPS coordinates would be more dangerous.**

FS What would count as proof?

ER **In most cases it is hard to convict someone on the basis of a photo of a tag that you would tie to another tag. For good reasons, because if it is a crew name for example, all of a sudden you are pinning one tag on a person that could have been written by twenty people. This came up in a trial in DC when an artist named BORF got arrested. He had written his name everywhere, completely crushed DC and his trial was a big deal. This issue came up and they argued that BORF was a collective, not an individual. Who knows if that's true, there were a lot of people around him, but how do you really know?**

FS GML could help balance the load?

ER **You mean it would not be just the image of a tag but more like signing at the bank?**

FS I mean that if you copy and distribute your data, the chance is small that you can link it to an individual.

– The winning design will have some protection in the event that the device falls into the wrong hands.

ER **This again should probably have been a bonus item. Wouldn't it be awesome if you could go home and log in and flip a one to a zero and the evidence goes up in smoke?**
**One graffiti writer friend told me: *If the police comes, just smash the camera as hard as you can!* It's a silly idea, but it shows that they are thinking about it.**

FS Edible SD cards?

ER **That would be a good idea!**

– Data should be able to be captured from both spray cans and markers.

ER Yes.

FS **Are you prepared for tools that do not exist yet?**

ER That was kind of what I was thinking there. Markers are about direct contact, spraypaint is in free space. If it works in those two situations, you should theoretically be able to tie it to anything, even outside of graffiti. If it was too much about spraypaint, it would be harder for someone to strap it to a skateboard.

– System should be able to record writing on various surfaces and materials.

ER **It is something you can easily forget about. When you are developing something in the studio and it works well against a white wall, and than when you go out in the city than you realize that brick is a really weird surface. Or even writing on glass, or on metal or on other reflecting surfaces that could screw up your reading. It is there as a reminder for people that are not thinking about graffiti that much. The street and the studio are so different.**

– Data should be captured at 30 points per second minimum.

ER I was assuming that lots of people were going to use cameras, and I wanted to make sure they were taking enough data points. With other capturing methods it is probably not such a problem. Even at 30 points per seconds you can start to see the facets if you zoom in, so anything less is not ideal.

– The recording system should not interfere with the writer's movements in anyway (including writing, running and climbing).

ER **So this is where Muharrem is going to run into trouble. His solution interferes. Not that much if you are just working in front of your body space. But the way most writers write is that they are shuffling their feet a lot, moving down the wall. Should it have said: *Graffiti writer should retain access to feet functionality*? This point should be at the top almost.**

FS To me it feels strange, your emphasis on the tool blending into the background. You could also see Muharrem's solution as an enhancing device, turning the writer into a tapdancer?

ER **I want to have on record: I love his solution! There's a lot in his design that is 'making us more aware' of what's happening in the creation of a tag. One thing that he is doing that is not in the specs, is that he is**

**logging strokes, like up and down. When you watch him using it, you can see a little light going from red to green when the fingers goes on and off the spraypaint can. When you watch graffiti, it is too small of a movement to even notice but when you are seeing that, it adds another level of understanding of how they are writing.**

– All motion data should be saved using the current GML standard[31].

FS Obvious.

– All aspects of the winning design should be able to be reproduced by graffiti writers and amateur technologists.

ER **It wouldn't be exciting if only ten people can make this thing. This tool should not be just for people that can make NASA qualified soldering connections. Ideally it should not have any soldering. I always thought of a soldering iron like a huge barrier point. I'm all for duct-taped electrical connections.**

FS There's nothing about weather-resistant in the challenge. You're not thinking about rain, are you?

ER **A lot of paint stops working in rain too.**
**I think what you get from this brief though is that the whole impetus for this project is about me trying to steer the ship that clearly wants to go into another direction, back to my interest in what graffiti is rather than anything that people might find aesthetically pleasing. It is not about 'graffiti influenced visuals'.**

[31] http://graffitimarkuplanguage.com/spec

– All software must be released Open Source. All hardware must include clear DIY instructions/tutorials. All media must be released under an Open Content licence that promotes collaboration (such as a Free Art License or Creative Commons ShareAlike License).

ER I didn't want it to be too specific, but there had to be some effort into making it open.

– The recording must be an unobtrusive process, allowing the graffiti writer to concentrate solely on the act of writing (not on recording). The act of recording should not interfere with the act of graffiti writing.

ER **I've been through situations where the process gets so confusing that you can't keep your head straight and juggle all the variables. Your eyes and ears are supposed to tell you about who's coming around the corner. Is there traffic coming or a train? There are so many other things you need to pay attention to rather than: *Is this button on?***
**The whole project is about getting good data. As soon as you force people to think too much about the capture process, I think it influences when and how they are writing.**

**Bonus, but not required:**

– Inclusion of date, time and location saved in the .gml file.

ER Yes. Security-wise that is questionable, but the nerd in me would just love it. You could get really interesting data about a whole night of writing. You could see a bigger story than just that of a single tag. How long did it take to gain entry? How long were they hiding in the bushes? These things get back to graffiti as a performance art rather than a form of visual art.

## Paris, November 2011

FS **Last time we had contact we discussed how to invite Muharrem to Brussels[32]. But now on the day of the deadline, it seems there are new developments?**

ER I think in terms of the actual challenge, the main update is that since we extended the deadline and made another call, I got an e-mail right on the deadline today from Joshua Noble[33] with a very solid and pretty smart proposal that seems to solve (maybe unfortunately for Muharrem) a bit more of the design spec. It does it for cheaper and does it in a way that I think is going to be easier to make also.
His design solution is using an optical mouse and he changed the sensors so it has a stronger LED. He uses a modified lens on top of a plastic lens that comes on top of a mouse, so that it can look at a surface that is a set distance away. It has another sensor that looks at pitch, tilt and orientation, but he is using that only to orient, the actual data gets recorded through the mouse. It can get very high resolution, he is looking at up to a millimeter I guess.

FS **Muharrem's solution seems less precise?**

ER I think he gets away with more because his solution is only for spraypaint and once you are writing on that scale, even if you are off a few centimeters, it might not ruin the data. If you look at the data he is getting, it actually looks very good. I don't think he has any numbers on the actual resolution he is getting but if you were using his system with a pen, I think it would be a different case. I like a lot of his solution too, it is an interesting hack. It is funny that two of the candidates for the prize are both mouse hacks. One is hacking a mechanical mouse and the other an optical mouse.

FS **It goes from drawing on a screen, to drawing on a wall?**

JH And back again!

ER **Yes. When I first was working on graffiti related software, the whole reason I was building Graffiti Analysis as a capture application was be-**

[32] By early October 2011 no winning design-solution had been entered, besides a proposal from Muharem Yildirim that came more than halfway. We decided to use the prize money to fly Muharrem from Phoenix (US) to Brussels (BE) and document his project in a worksession as part of the Verbindingen/Jonctions 13 meetingdays. http://www.vj13.constantvzw.org

[33] Joshua Noble http://www.thefactoryfactory.com/gmlchallenge/

**cause I did not want to hand graffiti writers a mouse (*laughter*). I had done all this research into graffiti and started to be embedded in the community and I knew enough about the community that if you were going to ask them to take part in something that was already weird, you could not give them a mouse and expect any respect on the other end of that conversation. They respect their tools, so the reason I was using camera-input was because I wanted to have a flexible system where they could bring in anything and I could attach a device to it. Now I am coming back to mice finally.**

FS Now the deadline has passed, do you think the passage from wishlist to contest worked out?

ER **I think it was a good experiment, I am not sure how clever it was. To take a piece of culture that a lot of people don't even look at, or look at it and think it is trash, to invest all this time and research and software expertise into it makes people think about the graffiti practice and what it actually is. The cash prize does something similar. It attaches weight to something that most people don't even care about. Even having the name of an organization like Constant attached to it is showing that I am really serious about this. In that sense it is different than a wishlist.**
**I just read the Linus Torvalds[34] biography, and I liked his idea that 'fun' is part of innovation, right? In a programming sense, it is scratching a personal itch. The attachment of a prize is more to underline the fun aspect than anything else.**

FS I am still puzzled about GML and how it is at the one hand stimulating collaboration and sharing, and than it comes back to the proud individual that wants to show off. It is kind of funny actually that now two people are winning the prize.

ER **I understand what you mean.**

FS Also in F/LOSS, under the flag of 'Open' and 'Free' there is a lot of competition. Do you feel that kind of tension in your work?

ER **Even 'Open' and 'Free' are in competition!**
**In a project like White-Glove Tracking for example, the most popular video I had not made and it did not have my name on it but personally I**

[34] Torvalds, Linus; David Diamond (2001). Just For Fun: The Story of an Accidental Revolutionary. New York, New York, United States: HarperCollins.

**still felt a part of it. I think when you are working in open systems, you take pride when a project has wings. It is maybe even a selfish act. It is the story of me receiving some art-finding and realizing that I am not the best toolmaker for the job. Who ever manages to win the prize gets all the glory, but I'm still going to feel awesome about it.**

FS I have been reading the interview that Kyle McDonald did with Anton Marini [35] and at some point he talks about being OK with sharing code and libraries, but when it is too much of a personal style, then it is hard to share.

ER **Yes, I thought that was an interesting point. I've been in similar conversations on listservs with artists in the OpenFrameworks, Processing and visual programming communities. What are the open pieces? It makes sense to share libraries, but if I make a print from a piece of code, do I then have to share the exact source and app for how that exact print was made? What does it mean when I am investing money in a print, and it is a limited series but I'm sharing the code? The art world is still based on scarcity and we're interested in computers that are copy-machines.**
**I see both sides of the argument and I am still trying to see how I fit into it. It gets trickier when you are asked to release a piece rather than a tool. If you are an Open Source artist and you make a toolset, that is easier to share because people use that to make their own things. But then an artist gets asked: how come I can't get the file of that print? I think that is a really hard question.**

FS But isn't the tool often the piece, and vice versa?

ER **I agree. And I haven't solved that question yet. Lately I've been a lot less excited about running workshops for example. A lot of the people that want to take part in the workshops are actually the opposition. Often they own a club and they want to install a cool light-show or they are into viral marketing. I never know which way to go with that. It depends on what side of the curve of frustration I am on at that moment.**

JH Earlier you brought up the contrast between people that were more visually invested and others that are more interested in the performance aspect. I wanted to hear a bit more about the continuum in the culture and how GML fits into that?

---

[35] Anton Marini: *Some personal projects of mine, for example specific effects and 'looks' that I have a personal attachment to, I don't release*
https://github.com/kylemcdonald/SharingInterviews/blob/master/antonmarini.markdown

**ER My focus has been on tags, this one portion of graffiti. I do think there could be cool uses for more involved pieces. It would be great if someone else would come in and do that, because it is a part of graffiti that I haven't studied that much. I would not even be able to write a specs-sheet for it; it requires a lot of different things when you paint these super-involved murals, when you have an hour or more time on your hands a lot more things come into play. Color, nozzles, nozzle changes and so on.**

JH Z-axis becomes important?

**ER Yes, and your distance from the wall, a lot of other things my brain isn't wrestling with. I think tags are always fundamental, even if they are painting murals that take three days to paint, somewhere in their graffiti education they start with the tags. You're still going to be judged by the community based on how you sign your name on the blackbook.**
**Graffiti is funny because it is almost conservative in terms of how a successful graffiti writer is viewed and it is reflected in how graffiti is in some way similar in the world. In some way it is a let down, to travel from Brooklyn to Paris to Brussels and it looks all the same but I think it stems from the fact that the community is so tight-knit. But at the end of the day it comes back to the tag always.**
**In terms of the performance, in a tag the relationship between form and function is really tight. The way your hand moves and how the tag actually looks on the wall is dictated by the gesture you are making. A piece where you have three hours, that tight synchronization isn't there. With a tag, every letter looks the way it does because that's how it needs to be drawn, because it needs to be connected to this other letter. There's a lot of respect for writers that do oneliners, and even if your tag has more than one line, a good graffiti writer has often a one line version. If you don't have to pick up the pen it is a really economical stroke.**

JH It is almost like hacking the limitations of gesture.

**ER It is a very specific design requirement. How to write a name that is interesting to think about and to look at, you have to do it in 5 seconds, you have to do it in one line, you have to do it on each type of surface. On top of that, you have to do it a million times, for twenty years.**

JH In Seattle they call a piece that stays up for a longer time a 'burner'. I was connecting that to an archival practice of ephemera. It is a self-agreed upon archival process, and it means that the piece will not be touched, even for years.

ER **Graffiti has an interesting relationship to archiving. On the one hand, many graffiti writers think: Now that tag's done, but I've got another million of them. While others do not want people painting over them, the city or other graffiti writers. Also if a tag has been up there for a few years, it acquires more reverence and it is even worse when it is painted over.**
**But I think that GML is different, it is really more similar to a photo of the tag. It is not trying to be the actual thing.**

FS Once a tag is saved in GML, what can be done with the data?

ER **I am myself reluctant to take any of these tags that I've collected and do anything with it at all without talking closely to whoever's tag it is, because it is such an intimate thing. In that sense it is strange to have an open data repository and to be so reluctant to use it in a way that is looking at anyone too specifically.**
**The sculpture I've been working on is an average from a workshop; sixteen different graffiti writers merged into one. I don't want to take advantage of any one writer. But this has nothing to do with the licence, it is totally a different topic. If someone uploads to the 000000book site, legally anyone should be able to do anything that they can do under the Creative Commons licence that's on the site but I think socially within the community, it is a huge thing.**

JH There must be some social limits to referentiality. Like beat jacking for DJs or biting rhymes for MCs, there must be a moment where you are not just homaging, but stealing a style.

ER **I've seen cases where both parties have been happy, like when Yamaguchi Takahiro used some GML data from KATSU and piped it into Google Maps, so he was showing these big KATSU tags all over the earth which was a nice web-based implementation. I think he was doing what a graffiti writer does naturally: Get out there and make the tag bigger but in different ways. He is not taking KATSU-data from the database without shining light back on him.**

FS GML seems very inspired by the practice of Free Software, but at the same time it reiterates the conventional hierarchies of who are supposed to use what … in which way … from who. For me the excitement with open licences is that you can do things without asking permission. So, usage can develop even if it is not already prescribed by the culture. How would someone like me, pretty far removed from graffiti culture ever know what I am entitled to do?

ER **I have my reasons for which I would and would not use certain pieces of data in certain contexts, but I like the fact that it is open for people that might use it for other things, even if I would not push some of those boundaries myself.**

FS Even when I am sometimes disappointed by the actual closedness of F/LOSS, at least in theory through its licensing and refusal to limit who is entitled and who's not, it is a liberating force. It seems GML is only half liberating?

ER **I agree. I think the lack of that is related to the data. The looseness of its licence makes it less of an invitation in a sense. If the people that put data up there would sit down and really talk about what this means, when they would really walk through all the implications of what it means to public domain a piece, that would be great. I would love that. Then you could use it without having to worry about all the morality issues and people's feelings. It would be more free.**
**I think it would be good to do a workshop with graffiti writers where beyond capturing data, you reserve an hour after the workshop to talk to everybody about what it would mean to add an open licence. I've done workshops with graffiti writers and I talked to everyone: *Look, I am going to upload this tag up to this place where everyone can download them after the workshop, cool?* And they go *cool.* But still, even then, do I really feel comfortable that they understand what they've gotten into? Even if someone has chosen a ShareAlike licence, I would be nervous I think.**
**Maybe I am putting too much weight on it. People outside Free Software are already used to attaching Creative Commons licences to their videos. Maybe I am too close to graffiti. I still hold the tag as primal!**

JH It is interesting to be worried about copyright on something that is illegal, things you can not publicly claim ownership of.

FS **Would you agree that standards are a normalizing practice, that in a way GML is part of a legalizing process?**

ER For that to happen, a larger community would have to get involved. It would need to be Gesture Markup Language, and a community other than graffiti writers would need to get involved.

FS **Would you be interested in legalizing graffiti?**

ER No. That's why I stopped doing projections.

JH **Not legal forms of graffiti, but more like the vision of KRS-One of the Hip Hop city,[36] where graffiti would obviously be legal. Does that fundamentally change the nature of graffiti?**

ER To me it is just not graffiti anymore. It is just painting. It changes what it is. For me, its power stems from it being illegal. The motion happens because it is illegal.

JH **In a sense, but there is also the calligraphic aspect of it. In Brooklyn, a lot of the building owners say: *yeah, throw it up* and those are some of the craziest pieces I know of, not from a tag-standpoint, but more as complex graffiti visuals.**

ER I am always for de-criminalization. I don't think anyone should go to jail over a piece of paint that you could cover over in 5 seconds. And that KRS-One city you mentioned would be cool to see.

JH **It is his Temple of Hip Hop, the idea to build a city of Hip Hop where the entire culture can be there without any external repression. It's an utopian ideal obviously.**

ER Of course I would like to see that. If nothing else, you would totally level the playing field between us and the advertisers. The only ones that would get up messages in the city would be the ones with more time on their hands.

JH **At the risk of stretching coherency, Hip Hop and Free Software are both global insurgent subcultures that have emerged from being kind of thrown away as fads and then become objects of pondering in multinational boardrooms. So I was hoping to open you up to riff on that: zooming out, GML is a handshake point between these two cultures, but GML is a specific thing within this larger world of F/LOSS and graffiti**

[36] KRS-One Master Teacher. AN INTRODUCTION TO HIP HOP . http://www.krs-one.com/#!temple-of-hip-hop/c177q

**in the larger world of hiphop. What other types of contact points might there be? Do you see any similarities and differences?**

ER For me, even beyond technology and beyond graffiti it all boils down to this idea of the hack that is really a phenomenon that has been going on forever. It's taking this system that has some sort of rigidity and repeating elements and flipping it into doing something else. I see this in Hip Hop, of course. The whole idea of sampling, the whole idea of turning a playback device into a musical instrument, the idea of touching the record: all of these things are hacks. We could go into a million examples of how graffiti is like hacker culture.

In terms of that handshake moment between the two communities, I think that is about realizing that its not about the code and in some sense its not about the spraypaint. There's this empowering idea of individual small actors assuming control over systems that are bigger than themselves. To me, that's the connection point, whether its Hip Hop or rap or programming.

The similarities are there. I think there are huge differences in those communities too. One of them is this idea of the hustler from Hip Hop: the idea of hustling doesn't have anything to do with the economy of gift-giving. The idea that Jay-Z has popularized in Hip Hop and that rap music and graffiti have at their core has to do with work ethic, but there's also a kind of braggadocio about making it yourself and attaining value yourself and it definitely comes back to making money in the end. The idea of being 'self-made' in a way is empowering but I think that in the Open Source movement or the Free Software movement the idea of hustling does not apply. It's not that people don't hustle on a day to day basis. You disagree with me?

JH **It's interesting because the more you were talking, the more I was not sure of whether you were speaking about Hip Hop or Free Software or maybe even more specifically the Open Source kind of ideological development. You have people like David Hannemeier Hansson who developed Ruby on Rails and basically co-opted an entire programming language to the point where you can't mention Ruby without people thinking of his framework. He's a hustler du jour: this guy's been in Linux Journal in a fold-out spread of him posing with a Lamborghini or something. Talk about braggadocio! You get into certain levels or certain dynamics within the community where its really like pissing contests.**

ER I like that, I think there's something there. At the instigation of the Open Source Initiative, though: like Linus 'pre-stock option', sitting in his bedroom not seeing the sun for a year and hacking and nerding out. To me they are so different, the idea of making this thing just for fun with a kind of optimistic view on collaboration and sharing. I know it can turn into money, I know it can turn into fame, I know it can turn into Lamborghinis but I feel like where its coming from is different.

JH **I agree, that's clearly a distinction between the two. They are not coming from the same thing. But for me its also interesting to think about it in terms that these are both sort of movements that have at times been given liberational trappings, people have assigned liberatory powers to these movements. Statistically the GPL is considerably more popular than the Open Source licences, but I don't know if you sat everybody down and took a poll which side they would land on, whether they were more about making money than they were about sharing. Are people writing blogposts because they really want to share their ideas or because they want to show how much cooler they are?**

ER You're totally right and I think people in this scene are always looking for examples of people making money, succeeding, good things coming to people for reasons that aren't just selflessness. People that are into Open Source usually love to be able to point to those things, that this isn't some purely altruistic thing.

JH **Maybe you could take some of the hustle and turn it into something in the Free Software world, mix and match.**

ER I think this line of inquiry is an interesting one that could be the subject of a documentary or something. These communities that seem very different until you start finding things that at their core really really similar.

JH **It would be so interesting to have a cribs moment with some gangsta or rapper who came from that, and he's sort of showing off his stuff and he has this machismo about him. Not necessarily directly mysognistic but a macho kind of character and then take a nerd and have them do the same.**

FS Would they really be so different?

JH **Obviously some rappers and some nerds, I mean that's one of the beauties – I mean its a global movement, you can't help but have diversity – but if we're just speaking in generalizations?**

FS There's a lot of showing off in F/LOSS too.

JH **Yeah, and there's a lot of chauvinism. And when you said that self-made thing, that's the Free Software idea number one.**

ER I think that part is a direct connection.

JH **And they're coming from two completely different strata, from a class-based analysis which is absent from a lot of discussion. Even on that level, how to integrate them to me is a political question to some degree.**

ER Right.

FS **Will any features of GML ever be deprecated?**

ER Breaking currently existing software? I hope not.

FS **Basically I'm asking for your long-term vision?**

ER When the spec was being made of course it wasn't just me, it was a group of people debating these things and of course nobody wants things to break. The idea was that we tried to get in as many things as we could think of and have the base stay kind of what it was with the idea that you could add more stuff into it. It's easy enough to do, of course its not a super-rigid standard. If you look at what the base .gml file is, the minimum requirements for GML to compile, its so so stripped down. As long as it just remains time/x-y-z, I don't think that's going to change, no.
But I'm also hoping that I'm not gonna be the main GML developer. I'm already not, there's already people doing way more stuff with it than I am.

FS **How does it work when someone proposes a feature?**

ER They just e-mail me (*laughs*). But right now there hasn't been a ton of that because it's such a simple thing, once you start cramming too much into it it starts feeling wrong. But all its gonna take is for someone to make a new app that needs something else and then there will be a reason to change it but I think the change will always be adding, not removing.

super

Chat with momo3010 ~ november 2011

> momo3010 entered the room...

momo3010: there is one BIG point i want to make
momo3010: graffiti is so easy to do .. u only need a marker or something .. even a pencil is enough and u are in the game .. it takes u 1 min. to buy something to write and start
momo3010: with all the computer stuff the entry barrier is much higher

>>>> ————

momo3010: i dont have to understand graffiti to do it. just get out and do it! with gml i have to have a sort of understanding of xml, i will need a comp, internet ..
momo3010: what i like of gml is the way to document (save) the tag, keeping the original still outside! that is really cool
momo3010: in the gallery i just show the code .. haha
momo3010: ‘keep it simple keep it fresh’
momo3010: one thing i miss too (i think evan is not forgetting this aspect)
momo3010: is the aesthetic of graffitianalysis

>>>> ————

momo3010: yes
momo3010: it is super nice .. it attracts people. u see it and ..
momo3010: wowow

>>>> ————

momo3010: it is really well done .. so this is the aesthetic point which is also very important for a tag

>>>> ————

momo3010: http://www.graffitiresearchlab.de/blitztag
momo3010: the germans have made various brushes .. do not know i like it that much ..
momo3010: this is more trying to look like graffiti brushes but it is not
momo3010: gml is cool to keep it raw

>>>> ————

momo3010: that is why we need data from outside. the way the tag is done is always depending on the outside!

>>>> ————

momo3010: i said to evan: look it is cool the gml recorder .. but if i am in a room my tag looks different then when i am outside
momo3010: it makes a difference when, where, and how to place the tag
momo3010: is the place hidden, do i have time, is it crowded, is it a big wall ..
momo3010: what i like also about all gml is the fact

>>>> ————

momo3010: it opened a whole new direction. combination of digital art with graffiti art .. the two new popular cultures .. i see gml not only as 'x,y, time'. it paved the way to do electronic outdoor stuff.
momo3010: everybody interesting in doing something into his area is somehow connected to gml.

gml

is

FS: Yes, true.

think

FS: you mean the way it plays out?

FS: i love the way it works with speed, and these fireworks when it turns
FS: also the drip is great -- i like that it is not faking paint

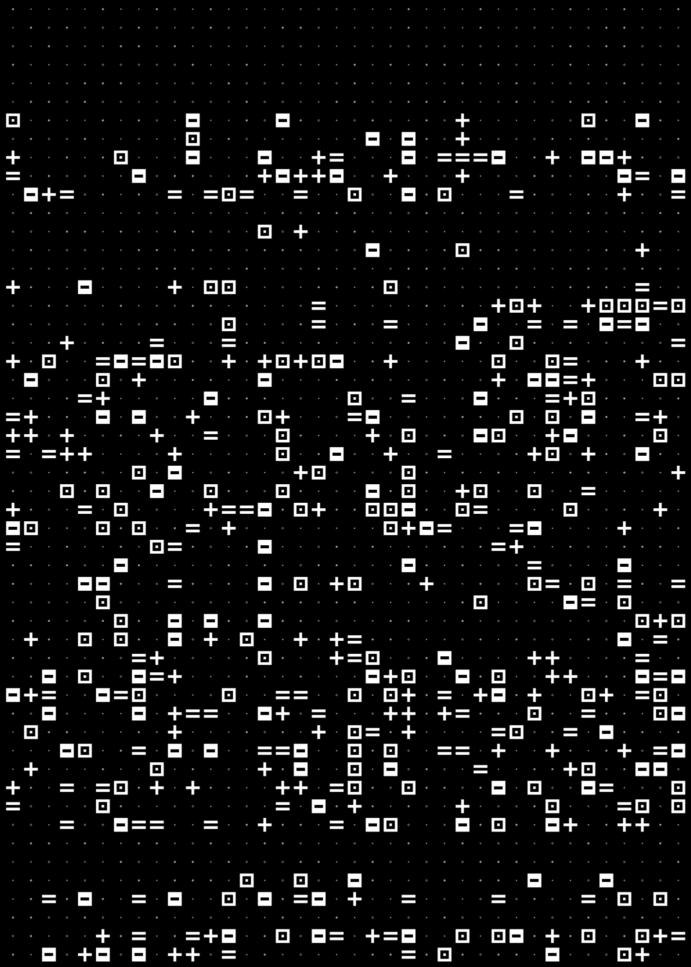
FS: yes. the digital rendering is super precise without trying to be the same. no replacement

FS: yeah. it is sort of legible in the way a tag also talks about how it was done

FS: what do you mean?

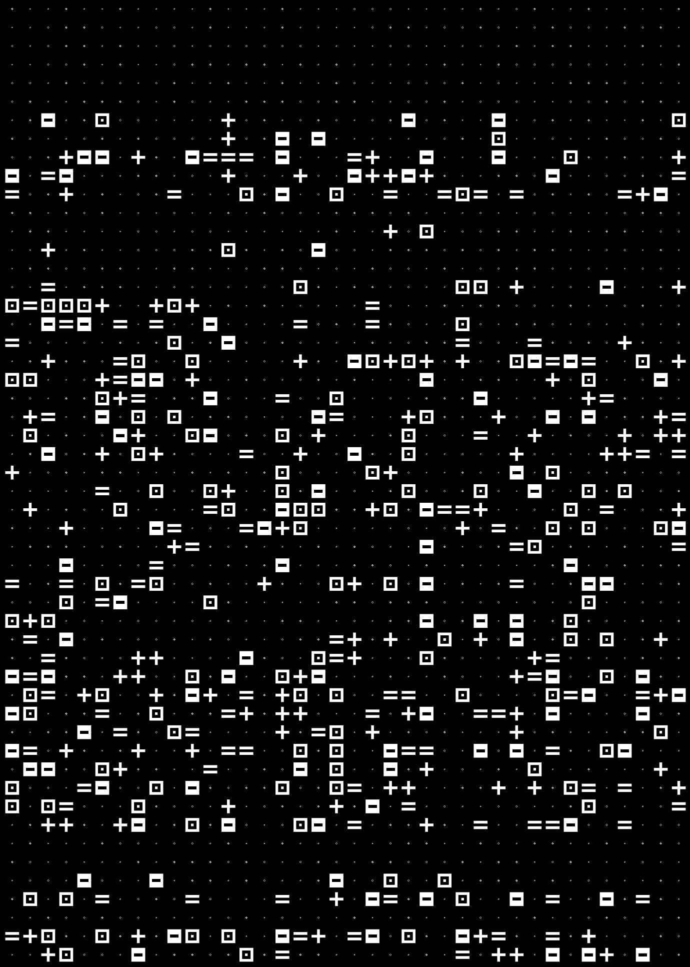
FS: sorry you got disconnected

data



# Unicodes

+ Denis Jacquerye = Femke Snelting
⊡ Pierre Marchand ⊟ Nicolas Malevé



# Unicodes

+ Denis Jacquerye = Femke Snelting
▣ Pierre Marchand ⊟ Nicolas Malevé

The following text is a transcription of a talk by and conversation with Denis Jacquerye in the context of the Libre Graphics Research Unit in **2012**. We invited him in the context of a session called *Co-position* where we tried to re-imagine layout from scratch. The text-encoding standard Unicode and moreover Denis' precise understanding of the many cultural and political path-dependencies involved in the making of it, felt like an obvious place to start. Denis Jacquerye is involved in language technology, software localization and font engineering. He's been the co-lead of the DéjàVu Font project and works with the African Network for Localization (ANLoc) to remove language limitations that exist in today's technology. Denis currently lives in London.This text is also available in *Considering your tools.*[1] A shorter version has been published in *Libre Graphics Magazine 2.1.*

+ This presentation is about the struggle of some people to use typography in their languages, especially with digital type because there is quite a complex set of elements that make this universe of digital type. One of the basic things people do when they want to use their languages, they end up with these type of problems down here, where some characters are shown, some aren't, sometimes they don't match within the font. Because one font has one of the character they need and then another one doesn't. Like for example when a font has the capital letter but not the corresponding lowercase letter. Users don't really know how to deal with that, they just try different fonts and when they're more courageous, they go online and find how to complain about those to developers – I mean font designers or engineers. And those people try to solve those problems as well as they can. But sometimes it's pretty hard to find out how to solve them. Adding missing characters is pretty easy but sometimes you also have language re-

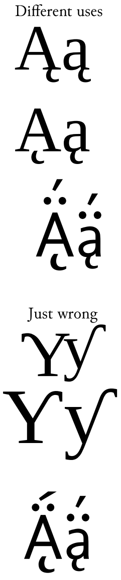
[1] Considering your tools: a reader for designers and developers http://reader.lgru.net

quirements that are very complex. Like here for example, in Polish, you have the ogonek, which is like a little tail that shows that a vowel is nasalized. Most fonts actually have that character, but for some languages, people are used to have that little tail centred which is quite rare to see in a font. So when font designers face that issue, they have to make a choice rather they want to go with one tradition or another, and if they want to go one way they're scattered to those people. Also you have problems of spacing things differently, like a stacking of different accents – called diacritics or diacritical marks. Stacking this high up often ends up on the line above, so you have to find a solution to make it less heavy on a line, and then in some languages, instead of stacking them, they end up putting them side by side, which is yet another point where you have to make a choice.

But basically, all these things are based on how type is represented on computers. You used to have simple encodings like ASCII, the basic Western Latin alphabet where each character was represented by bytes. The character could be displayed with different fonts, with different styles, they could not meet the requirements of different people. And then they made different encodings because they were a lot of different requirements and it's technically impossible to fit them all in ASCII.

Often they would start with ASCII and then add the specific requirements but soon they ended up having a lot of different standards because of all the different needs. So one single byte of representation would have different meanings and each of these meanings could be displayed differently in fonts. But old webpages are often using old encodings. If your browser is not using the right encoding you would have jibbish displayed because of this chaos of encodings. So in the late eighties, they started thinking about those problems and in the nineties they started working on Unicode: several companies got together and worked on one single unifying standard that would be compatible with all the pre-used standards or the new coming ones.

Unicode is pretty well defined, you have a universal code point to represent to identify a character, and then that character can be displayed with different glyphs depending on the font or the style selected. With that framework, when you need to have the proper character displayed, you have to go the code point in a font editor, change the shape of the character and it can be displayed properly. Then sometimes there's just no code point for the character you need because it hasn't been added, it wasn't in any existing

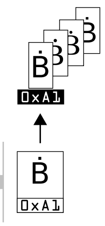


standard or nobody has ever needed it before or people who needed it just used old printers and metal type.
So in this case, you have to start to deal with the Unicode organization itself. They have a few ways to communicate like the mailing list, the public, and recently they also opened a forum where you can ask questions about the characters you need as you might just not find them.
In most operating systems, you have a character map application where you can access all the characters, either all the characters that exist in Unicode or the ones available in the font you're using. And it's quite hard to find what you need, as it's most of the time organized with a very restrictive set of rules. Characters are just ordered in the way they're ordered within Unicode using their code point order: for example, capital A is 41, and then B is 42, etc. The further you go in the alphabet the further you go in the Unicode blocks and tables, and there is a lot of different writing systems … Moreover because Unicode is sort of expanding organically – work is done on one script, and then on another, then coming back to previous scripts to add things – things are not really in a logical or practical order. Basic Latin is all the way up there, and more far, you have Latin Extended A, (Conditional) Extended Latin, Latin Extended B, C and D. Those are actually quite far apart within Unicode, and each of them can have a different setup: for example, here you have a capital letter that is just alone, and here you have a capital letter and a lowercase letter. So when you know the character you want to use, sometimes you would find the uppercase letter but you'd have to keep looking for the corresponding lowercase.
Basically when you have a character that you can't find, people from the mailing list or the forum can tell you if it would be relevant to include it in Unicode or not. And if you're very motivated, you can try to meet the inclusion criterias. But for a proper inclusion, there has to be a formal proposal using their template with questions to answer, you also have to provide proof that the characters you want to add are actually used or how they would be used.

The criterias are quite complicated because you have to make sure that this is not a glyphic variant (the same character but represented differently). Then you also have to prove the character doesn't already exist because sometimes you just don't know it's a variant of another one; sometimes they just want to make it easier and claim it's a variant of another one even though you don't agree. For example, making sure it's not just a ligature as sometimes ligatures are used as a single character, sometimes they exist for aesthetic reasons. Eventually you have to provide an actual font with the character so that they can use it in their documentation.

= ***How long does it take usually?***

+ It depends as sometimes they accept it right away if you explain your request properly and provide enough proof, but they often ask for revisions to the proposals and then it can be rejected because it doesn't meet the criterias. Actually those criterias have changed a bit in the past. They started with Basic Latin and then added special characters which were used: here for example is the international phonetic alphabet but also all the accented ones … As they were used in other encodings and that Unicode initially wanted to be compatible with everything that already exists, they added them. Then they figured they already had all those accented characters from other encodings so they're also going to add all the ones they know are used even though they were not encoded yet. They ended up with different names because they had different policies at the beginning instead of having the same policy as now. They added here a bunch of Latin letters with marks that were used for example in transcription. So if you're transcribing Sanskrit for example, you would use some of the characters here. Then at some point they realized that this list of accented characters would get huge, and that there must be a smarter way to do this. Therefore they figured you could actually use just parts of those characters as they can be broken apart: a base letter and marks you add to it. You may have a single character that can be decomposed canonically between the letter **B** and a colon dot above, and you have the character for the dot above in the block of the diacritical marks. You have access to all the diacritical marks they thought were useful at some point. At that point, when they realized they would end up having thousands of accented characters they figured with this way where we can have just any possibility, so from now on, they're just going to say if you want to have an accented character that hasn't been encoded already, just



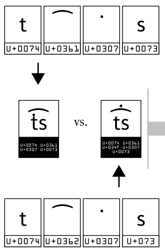
use the parts that can represent it. Then in 1996, some people for Yoruba, a spoken language in Nigeria, made a proposal to add the characters with diacritics they needed and Unicode just rejected the proposal as they could compose those characters by combining existing parts.

= ***Weren't the elements they needed already in the toolbox?***

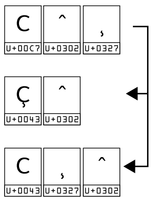
+ Yes, the encoding parts are there, meaning it can be represented with Unicode but the software didn't handle them properly so it made more sense to the Yoruba speakers to have it encoded it in Unicode.

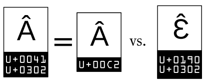
= ***So you could type, but you'd need to type two characters of course?***

+ Yes, the way you type things is a big problem. Because most keyboards are based on old encodings where you have accented characters as single characters, so when you want to do a sequence of characters, you actually have to type more, or you'd have to have a special keyboard layout allowing you to have one key mapped to several characters. So that's technically feasible but it's a slow process to have all the possibilities. You might have one whic is very common so developers end up adding it to the keyboard layouts or whatever applications they're using, but not when other people have different needs.
There is a lot of documentation within Unicode, but it's quite hard to find what you want when you're just starting, and it's quite technical. Most of it is actually in a book they publish at every new version. This book has a few chapters that describe how Unicode works and how characters should work together, what properties they have. And all the differences between scripts are relevant. They also have special cases trying to cater to those needs that weren't met or the proposals that were rejected. They have a few examples in the Unicode book: in some transcription systems they have this sequence of characters or ligature; a **t** and a **s** with a ligature tie and then a dot above. So the ligature tie means that **t** and **s** are pronounced together and the dot above is err… has a different meaning (*laughs*). But it has a meaning! But because of the way characters work in Unicode, applications actually reorder it whatever you type in, it's reordered so that the ligature tie ends up being moved after the dot. So you always have this representation because you have the **t**, there should be the dot, and then there should be the ligature tie and then the **s**. So the **t** goes first, the dot goes above the **t**, the ligature tie goes above everything and then the **s** just goes next to the **t**. The way they

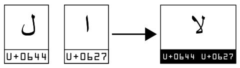

t
s
U+0074
U+0361
U+0307
U+0073
ts
U+0074 U+0361 U+0307 U+0073
vs.
ts
U+0074 U+0361 U+034F U+0307 U+0073
t
s
U+0074
U+0362
U+0307
U+073

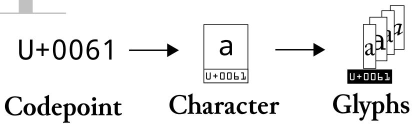
explain how to do this is supposed to do the **t**, the ligature tie, and then a special diacritical mark that prevents any kind of reordering, then you can add the dot and then you can do the **s**. So this kind of use is great as you have a solution, it's just super hard because you have to type five characters instead of… well… four (*laughs*). But still, most of the libraries that are rendering fonts don't handle it properly and then even most fonts don't plan for it. So even if the fonts did anyway the libraries wouldn't handle it properly. Then there are other things that Unicode does: because of that separation between accents and characters and then the composition, you can actually normalize how things are ordered. This sequence of characters can be reordered into the pre-composed one with a circumflex or whatever; you have combining marks in the normalized order. All these things have to be handled in the libraries, in the application or in the fonts.
The documentation of Unicode itself is not prescriptive, meaning that the shape of the glyphs are not set in stone. So you can still have room to have the style you want, the style your target users want. For example if we have different glyphs: Unicode has just one shape and it's the font designer's choice to have different ones. Unicode is not about glyphs, it's really about how information is represented, how it's displayed. Or you have two characters displayed as a ligature: it is actually encoded as one character because of previous encodings. But if ever it would be a new case, Unicode wouldn't stake the ligature as a single character.





**more than just glyphs**





So all this information is really in a corner there. It's quite rare to find fonts that actually use this information to provide to the needs of the people who need specific features. One of the way to implement all those features is with TrueType OpenType and there are also some alternatives like Graphite which is a subset of a TrueType OpenType font. But then, you need your applications to be able to handle Graphite. So eventually the real unique standard is TrueType Opentype. It's pretty well documented and very technical because it allows to do many things for many different writing systems. But it's slow to update so if there's a mistake in the actual specifications of OpenType, it takes a while before they correct it and before that correction shows up in your application. It's quite flexible and one of the big issue it that it has its own language code system, meaning that some identified languages just can't be identified in OpenType. One of the features in OpenType is managing language environment. If I'm using Polish, I'd want this shape; if I'm using Navajo, I'd want this shape. That's very cool because you can make just one font that's used by Polish speakers and Navajo speakers without them worrying about changing fonts as long as they specify the language they're using. But you can't use this feature for languages which aren't in the OpenType specifications as they have their own way of describing languages than Unicode. It's really frustrating because, you can find all the characters in Unicode, not organized in a practical way: you have to look all around the tables to find the characters that may be used by one language, and then you have to look around for how to actually use them. It is a real lack of awareness within the font designer community. Because even when they might add all the characters you need, they might just not add the positioning, so for example you have a … when you combine with a circumflex, it doesn't position well because most of the font designers still work with the old encoding mindset when you have one character for one accentuated letter. Sometimes they just think that following the Unicode blocks is good enough. But then you have problems where, as you can see in the Basic Latin charts at the beginning, the capital is in one block and its lowercase in a different block. And then they just work on one block, they just don't do the other one because they don't think it's necessary, but yet, two blocks of the same letter are there, so it would make sense to have both. It's hard because there's very few connections between the Unicode world, people working on OpenType libraries, font designers and the actual needs of the users.

⊡ *At the beginning of the presentation you went for the code point of the characters, all your characters are subtitled by their code points; it's kind of the beauty of Unicode to name everything, every character.*

+ Those names are actually quite long. One funny thing about this. Unicode has the policy of not changing the names of the characters, so they have an errata where they realized that *oh, we shouldn't have named this that, so here's the actual name that makes sense, and the real name is wrong.*

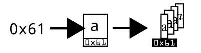
= ***Pierre refers to the fact that in the character mappings that each of the glyphs also has a description. And those are sometimes so abstract and poetic that this was a start of a work from OSP, the Dingbats Liberation Fest, to try to re-imagine what shapes would belong to those descriptions. So 'combining dot above' that's the textual description of the code point. But of course there are thousands of them so they come up with the most fantastic gymnastics …***

⊟ **So when people come in a project like DéjàVu, they have to understand all that to start contributing. How does this training, teaching, learning process takes place?**

+ Usually most people are interested in what they know. They have a specific need and they realize they can add it to DéjàVu, so they learn how to play with FontForge. After a while, what they've done is good and we can use it. Some people end up adding glyphs they're not familiar with. For example we had Ben doing Arabic: it was mostly just drawing and then asking for feedback on the mailing list; then we got some feedback, we changed some things, eventually released it, getting more feedback (*laughs*) because more people complained … So it's a lot of just drawing what you can from resources you can find. It's often based on other typefaces therefore sometimes you're just copying mistakes from other typefaces … So eventually it's just the feedback from the users that's really helpful because you know that people are using it, trying it, and then you know how to make it better.

## Character Encodings

ASCII



ISO 8859
2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16

VISCII EUC
CN, JP, KR
ArmSCII JIS X
0201, 0208, 0212
GOST 10859 ISCII
...

Unicode

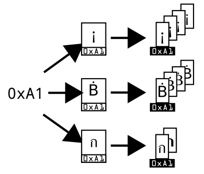
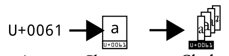


Fig. 2

Fig. 5



**Codepoint** **Character** **Glyphs**

**Communications**

Mailing list | Forum
Ask questions
General discussions
Before proposals

Addition process is
**complex and selective**

Formal proposal
Many criterias:
change over time
not glyphic variant
not unified with existing
not a ligature
evidence of use
encoding or font solution?
In cycles

**Documentation**

Book and charts
Description of Unicode
Character properties
Script properties
Special case

Data
Character properties
Script properties
Errata
CLDR
Sample UDHR
....

UTS
Unicode Technical Standard
Definitions
Algorithms
...

shapes of glyphs
**not prescriptive**

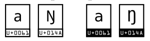
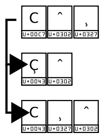


Fig. 4

**Normalization**



**more than just glyphs**

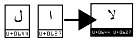
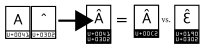


Fig. 6

**Composition, compatibility**



## Standards

TrueType
OpenType
Alternative?

**Specifications**
http://www.microsoft.com/typography/otspec/
Well documented
Very Technical
Slow to update
Flexible to a point
Own language codes

**Features**

Positioning
Kerning
Mark placement
....

Substitution
Ligatures
Language variants
....

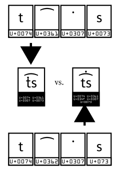
**ISO language codes?**

## Developers

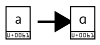
Fig. 3

Libraries

Support of Standards?



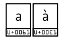
API
for features?



Applications

Discoverable
Character map
Font selector
Legacy standards mindset

Search and replace?



## Usage

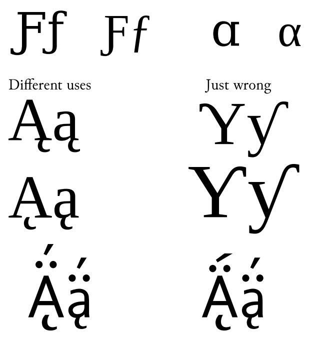
Printing tradition
Writing tradition
Digitalized?

Difference between character and glyph

Fig. 1

Different uses

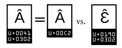
Just wrong



## Designers

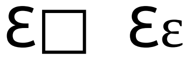
Fig. 7

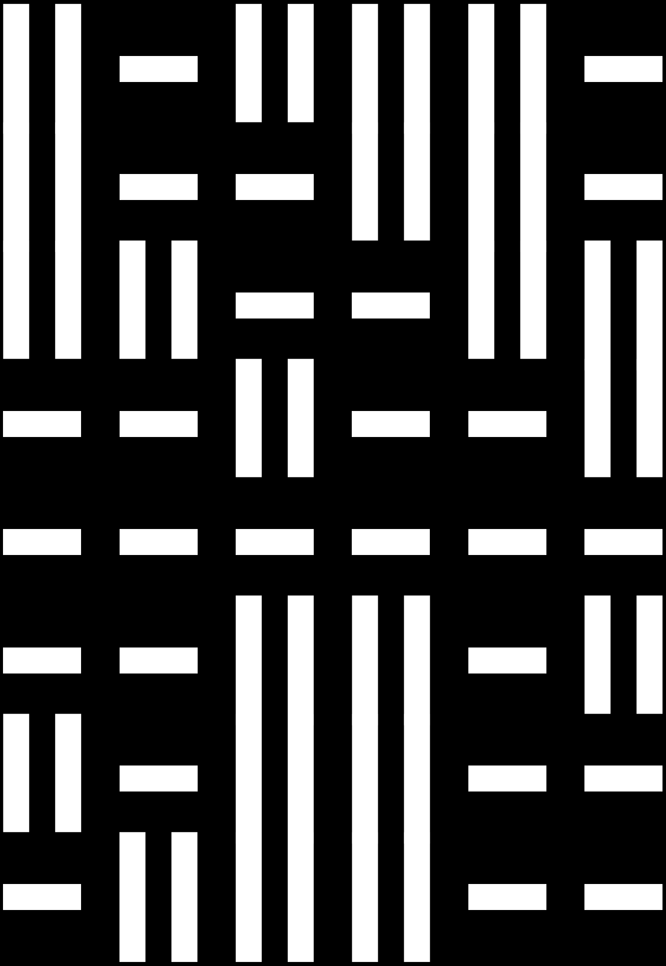
Awareness



Common practices
or mistakes

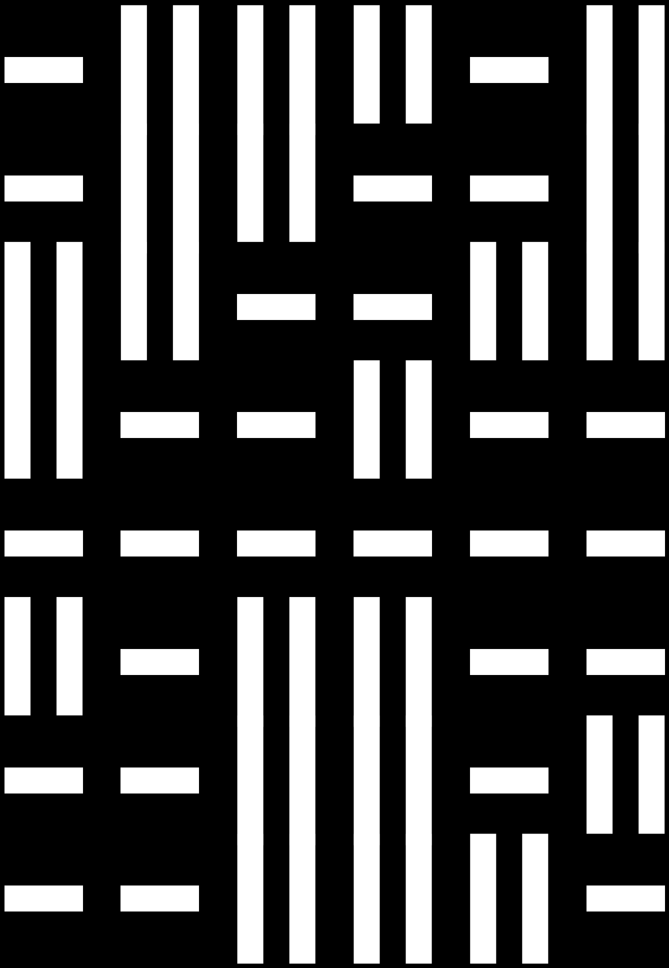
Old encodings
Poor understanding of Unicode
Blindly following Unicode
Following other fonts





If the design thinking is correct,
the tools should be irrelevant

‖ Pedro Amado
- Femke Snelting



If the design thinking is correct,
the tools should be irrelevant

‖ Pedro Amado
– Femke Snelting

(Type) designer Pedro Amado is amongst many other things initiator of TypeForge [1], a website dedicated to the development of 'collaborative type' with Open Source tools. While working as design technician at FBAUP [2], he is about to finish a MA with a paper on collaborative methods for the creation of art and design projects. When I e-mailed him in **2006** about open font design and how he sees that developing, he responded with a list of useful links, but also with:

```
Developing design teaching based on
Open Source is one of my goals, because
I think that is the future of education.
```

This text is based on the conversation about design, teaching and software that followed.

**– You told me you are employed as 'design technician' ... what does that mean?**

**||** It means that I provide assistance to teachers and students in the Design Department. I implemented scanning/printing facilities for example, and currently I develop and give workshops on Digital Technologies – software is a BIG issue for me right now! Linux and Open Source software are slowly entering the design spaces of our school. For me it has been a 'battle' to find space for these tools. I mean – we could migrate completely to OSS tools, but it's a slow progress. Mainly because people (students) need (and want) to be trained in the same commercial applications as the ones they will encounter in their professional life.

**– How did Linux enter the design lab? How did that start?**

**||** It started with a personal curiosity, but also for economical reasons. Our school can't afford to acquire all the software licenses we'd like. For example, we can't justify to pay approx. 100 x 10 € licenses, just to implement

[1] http://www.typeforge.net/
[2] http://www.fba.up.pt/

the educational version of Fontlab on some of our computers; especially because this package is only used by a part of our second year design students. You can image what the total budget will be with all the other needs … I personally believe that we can find everything we need on the web. It's a matter of searching long enough! So this is how I was very happy to find Fontforge. An Open Source tool that is solid enough to use in education and can produce (as far as I have been able to test) almost professional results in font development. At first I couldn't grasp how to use it under X[3] on Windows, so one day I set out to try and do it on Linux … and one thing lead to another …

**– What got you into using OSS? Was it all one thing leading to another?**

**||** Uau … can't remember … I believe it had to do with my first experiences online; I don't think I knew the concept before 2000. I mean I've started using the web (IRC and basic browsing) in 1999, but I think it had to do with the search of newer and better tools …

**– I think I also started to get into it around that time. But I think I was more interested in copyleft though, than in software.**

**||** Oh … (blush) not me … I got into it definitely for the 'free beer' aspect! By 2004 I started using DTP applications on Linux (still in my own time) and began to think that these tools could be used in an educational context, if not professionally. In the beginning of 2006 I presented a study to the coordinator of the Design Department at FBAUP, in which I proposed to start implementing Open Source tools as an alternative to the tools we were missing. Blender for 3D animation, FontForge for type design, Processing for interactive/graphic programming and others as a complement to proprietary packages: GIMP, Scribus and Inkscape to name the most important ones. I ran into some technical problems that I hope will be sorted out soon; one of the strategies is to run these software packages on a migration basis – as the older computers in our lab won't be able to run MacOS 10.4+, we'll start converting them to Linux.

[3] Cygwin/X is a port of the X Window System to the Cygwin API layer for the Microsoft Windows family of operating systems
Cygwin/x: X windows – on windows! http://x.cygwin.com/, 2014. [Online; accessed 5.8.2014]

**▬ I wanted to ask you about the relation between software and design. To me, economy, working process, but also aesthetics are a product of software, and at the same time software itself is shaped through use. I think the borders between software and design are not so strictly drawn.**

‖ It's funny you put things in that perspective. I couldn't agree more. Nevertheless I think that design thinking prevails (or it should) as it must come first when approaching problems. If the design thinking is correct, the tools used should be irrelevant. I say 'should' because in a perfect environment we could work within a team where all tools (software/hardware) are mastered. Rarely this happens, so much of our design thinking is still influenced by what we can actually produce.

**▬ Do you mean to say that *what we can think is influenced by what we can make*? This would work for me! But often when tools are mastered, they disappear in the background and in my opinion that can become a problem.**

‖ I'm not sure if I follow your point. I agree with *the border between design and software is not so strict* nevertheless, I don't agree with *economy, process and aesthetics are a product of software.* As you've come to say what we think is influenced by what we can make … this is an outside observation …

> A technique is produced inside a culture, therefore one's society is conditioned by it's techniques. Conditioned, not determined. [4]

‖ Design, like economics and software, is a product of culture. Or is it the other way around? The fact is that we can't really tell what comes first. Culture is defined by and defines technology. Therefore it's more or less simple to accept that software determines (and is determined) by it's use. This is an intricate process … it kind of goes roundabout on itself …

[4] Pierre Lévy. *Cyberculture (Electronic Mediations).* University Of Minnesota Press, 2001

**– And where does design fit in in your opinion? Or more precisely: designers?**

**II** Design is a cultural aspect. Therefore it does not escape this logic. Using a practical standpoint: Design is a product of economics and technology. Nevertheless the best design practices (or at least the one's that have endured the test of time) and the most renowned designers are the one's that can escape the the economic and technological boundaries. The best design practices are the ones that are not products of economics and technology … they are kind of approaching a universal design status (if one exists). Of course … it's very theoretical, and optimistic … but it should be like this … otherwise we'll stop looking for better or newer solutions, and we'll stop pushing boundaries and design as technology and other areas will stagnate. On the other hand, there is a special 'school' of thought manifested through some of the Portuguese Design Association members, saying that the design process should lead the process of technological development. Henrique Cayate (I think it was in November last year) said that *design should lead the way to economy and technology in society*. I think this is a bit far fetched …

**– Do you think software defines form and/or content? How is software related to design processes?**

**II** I think these are the essential questions related to the use of OSS. Can we think about what we can make without thinking about process? I believe that in design processes, as in design teaching, concepts should be separated from techniques or software as much as possible.

**– To me, exactly because techniques and software are intertwined, software matters and should offer space for thinking (software should therefore not be separated from design). You could also say: design becomes exceptionally strong when it makes use of its context, and responds to it in an intelligent way. Or maybe I did not understand what you meant by being 'a product of'. To me that is not necessarily a negative point.**

**II** Well … yes … that could be a definition of good design, I guess. I think that as a cultural produce, techniques can't determine society. It can and will influence it, but at the same time it will also just happen. When we talk about Design and Software I see the same principle reflected. Design being the 'culture' or society and software being the tools or techniques that are developed to be used by designers. So this is much the same as *Which came first? The chicken, or the egg?* Looking at it from a designers (not a software developers) point of view, the tools we use will always condition our output. Nevertheless I think it's our role as users to push tools further and let developers know what we want to do with them. Whether we do animation on Photoshop, or print graphics on Flash that's our responsibility. We have to use our tools in a responsible way. Knowing that the use we make of them will eventually come back at us. It's a kind of responsible feedback.

**━ Using Linux in a design environment is not an obvious choice. Most designers are practically married to their Adobe Suite. How come it is entering your school after all?**

‖ Very slowly! Linux is finally becoming valuable for Design/DTP area as it has been for long on the Internet/Web and programming areas. But you can't expect GIMP to surpass Photoshop. At least not in the next few years. And this is the reality. If we can, we must train our students to use the best tools available. Ideally all tools available, so they won't have problems when faced with a tool professionally. The big question is still, how we besides teaching students theory and design processes (with the help of free tools), help them to become professionals. We also have to teach them how to survive a professional relationship with professional tools like the Adobe Suite. As I am certain that Linux and OSS (or F/LOSS) will be part of education's future, I am certain of it's coexistence along side with commercial software like Adobe's. It's only a matter of time. Being certain of this, the essential question is: How will we manage to work parallel in both commercial and free worlds?

**━ Do you think it is at all possible to 'survive' on other tools than the ones Adobe offers?**

‖ Well … I seem not to be able to dedicate myself entirely to these new tools … To depend solely on OSS tools … I think that is not possible, at least not at this moment. But now is the time to take these OSS tools and start to teach with them. They must be implemented in our schools.

I am certain that sooner or later this will be common practice throughout European schools.

**– Can you explain a bit more, what you mean by 'real world'?**

**II** Being a professional graphic designer is what we call the 'real world' in our school. I mean, having to work full time doing illustration, corporate identity, graphic design, etc., to make a living, deliver on time to clients and make a profit to pay the bills by the end of the month!

**– Do you think OSS can/should be taught differently? It seems self-teaching is built in to these tools and the community around it. It means you learn to teach others in fact … that you actually have to leave the concept of 'mastering' behind?**

**II** I agree. The great thing about Linux is precisely that – as it is developed by users and for users – it is developing a sense of community around it, a sense of *given enough eyeballs, someone will figure it out*.

**– Well, that does not always work, but most of the time …**

**II** I believe that using Open Source tools is perfect to teach, especially first year students. Almost no one really understands what the commands behind the menus of Photoshop mean, at least not the people I've seen in my workshops. I guess GIMP won't resolve this matter, but it will help them think about what they are doing to digital images. Especially when they have to use unfamiliar software. You first have to teach the design process and then the tool can be taught correctly, otherwise you'll just be teaching habits or tricks. As I said before, as long as design prevails and not the tool/technique, and you teach the concepts behind the tools in the right way, people will adapt seamlessly to new tools, and the interface will become invisible!

**– Do you think this means you will need to restructure the curriculum? I imagine a class in bugreporting … or getting help online …**

**II** mmhh … that could be interesting. I've never thought about it in that way. I've always seen bugreporting and other community driven activities as part of the individual aspect of working with these tools … but basically you are suggesting to implement an 'Open Source civic behavior class' or something like that?

**– Ehm … Yes! I think you need to learn that you own your tools, meaning you need to take care of them (ie: if something does not work, report) but at the same time you can open them up and get under the hood … change something small or something big. You also need to learn that you can expect to get help from other people than your tutor … and that you can teach someone else.**

‖ The aspect of taking responsibility, this has to be cultivated – a responsible use of these tools. About changing things under the hood … well this I think it will be more difficult. I think there is barely space to educate people to hack their own tools let alone getting under the hood and modifying them. But you are right that under the OSS communication model, the peer review model of analysis, communication is getting less and less hierarchical. You don't have to be an expert to develop new or powerful tools or other things … A peer-review model assumes that you just need to be clever and willing to work with others. As long as you treat your collaborators as peers, whether or not they are more or less advanced than you, this will motivate them to work harder. You should not disregard their suggestions and reward them with the implementations (or critics) of their work.

**– How does that model become a reality in teaching? How can you practice this?**

‖ Well … for example use public communication/distribution platforms (like an expanded web forum) inside school, or available on the Internet; blog updates and suggestions constantly; keep a repository of files; encourage the use of real time communication technologies … as you might have noticed is almost the formula used in e-learning solutions.

**– And also often an argument for cutting down on teaching hours.**

‖ That actually is and isn't true. You can and will (almost certainly) have less and less traditional classes, but if the teachers and tutors are dedicated, they will be more available than ever! This will mean that students and teachers will be working together in a more informal relationship. But it can also provoke an invasion of the personal space of teachers …

**– It is hard to put a border when you are that much involved. I am just thinking how you could use the community around Open Source software to help out. I mean … if the online teaching tools would be open to others outside the school too, this would be the advantage. It would also mean that as a school, you contribute to the public domain with your classes and courses.**

**||** That is another question. I think schools should contribute to public domain knowledge. Right now I am not sharing any of the knowledge about implementing OSS on a school like ours with the community. But if all goes well I'll have this working by December 2006. I'm working on a website where I can post the handbooks for workshops and other useful resources.

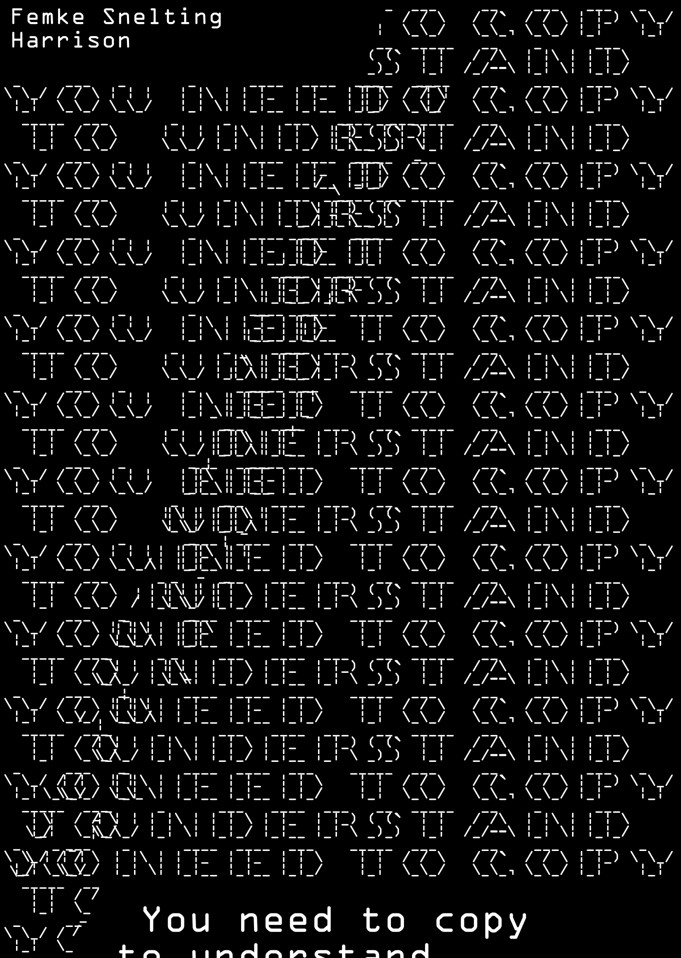
**– I am really curious about your experiences. However convinced I am of the necessity to do it, I don't think it is easy to open education up to the public, especially not for undergraduate education.**

**||** I do have my doubts too. If you look at it on a commercial perspective, students are paying for their education … should we share the same content to everyone? Will other people explore these resources in a wrong way? Will it really contribute to the rest of the community? What about profit? Can we afford to give this knowledge away for free, I mean, as a school this is almost our only source of income? Will the prestige gained, be worth the possible loss? These are important questions that I need to think more about.

**– OK, I will be back with you in 6 month to find out more! My last question … why would you invest time and energy in OSS when you think good designers should escape economical and technological boundaries?**

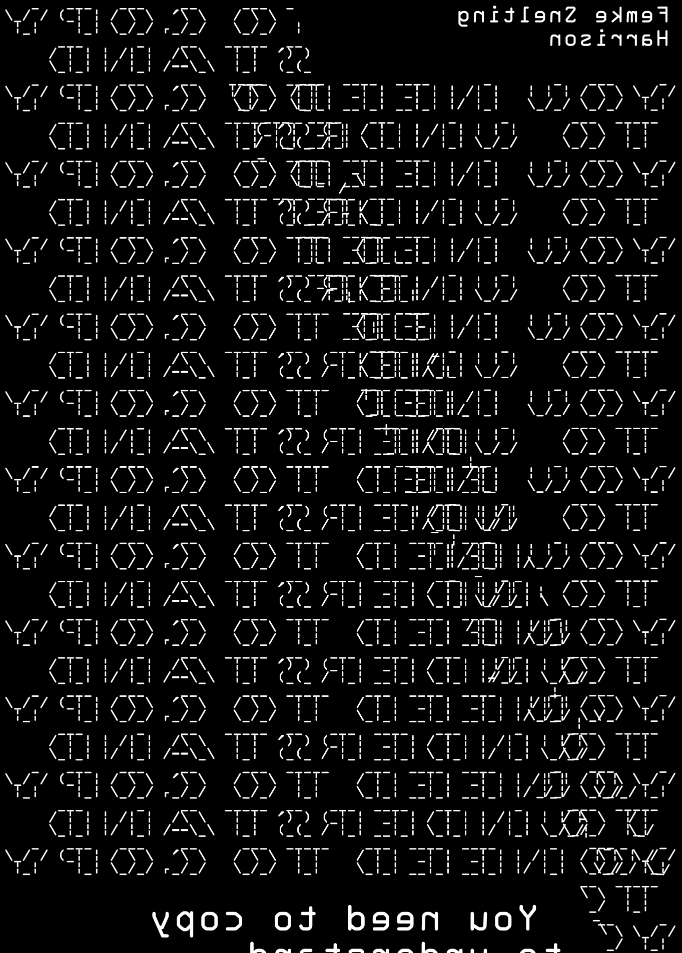
**||** If we invest energy on OSS tools now, we'll have the advantage of already being savvy by the time they become widely accepted. The worst case scenario would be that you've wasted time perfecting your skills or learned a new tool that didn't become a standard ... How many times have we done this already in our life? In any way, we need to learn concepts behind the tools, learn new and different tools, even unnecessary ones in order to broaden our knowledge base – this will eventually help us think 'out of the box' and hopefully push boundaries further [not so much as escaping them]. For me OSS and its movement have reached a maturity level that can prove it's own worth in society. Just see Firefox – when it reached general user acceptance level (aka 'project maturity' or 'development state'), they started to compete directly with MS Internet Explorer. This will happen with the rest (at least that's what I believe). It's a matter of quality and doing the correct broadcast to the general public. Linux started almost as a personal project and now it's a powerhouse in programming or web environments. Maybe because these are areas that require constant software and hardware attention it became an obvious and successful choice. People just modified it as they needed it done. Couldn't this be done as effectively (or better) with commercial solutions? Of course. But could people develop personalized solutions to specific problems in their own time frame? Probably not ... But it means that the people involved are, or can resource to, computer experts. What about the application of these ideas to other areas? The justice department of the Portuguese government (Ministério da Justiça) is for example currently undergoing a massive informatics (as in the tools used) change – they are slowly migrating their working platform to an Open Source Linux distribution – Caixa Mágica (although it's maintained and given assistance by a commercial enterprise by the same name). By doing this, they'll cut costs dramatically and will still be able to work with equivalent productivity (one hopes: better!). The other example is well known. The Spanish region of Estremadura looked for a way to cut costs on the implementation of information technologies in their school system and developed their own Linux Distro called Linex – it aggregates the software bundle they need, and best of all has been developed and constantly tweaked by them. Now Linux is becoming more accessible for users without technical training, and is in a WYSIWYG state of development, I really believe we should start using it seriously so we can try and test it and learn how we can use in in our everyday life (for me this process has already started ... ). People aren't stupid. They're just 'change resistant'. One of the aspects I think that will get peoples' attention will be that a 'free beer' is as good as a commercial one.

Femke Snelting
Harrison



You need to copy
to understand

Femke Snelting
Harrison



You need to copy
to understand

**August 2006.** One of the original co-conspirators of the OSP adventure is the Brussels graphiste going under the name Harrisson. His interest in Open Source software flows with the culture of exchange that keeps the off-centre music scene alive, as well as with the humanist tradition persistingly present in contemporary typography. Harrisson's visual frame of reference is eclectic and vibrant, including modernist giants, vernacular design, local typographic culture, classic painting, drawing and graffiti. Too much food for one conversation.

FS *You could say that 'A typeface is entirely derivative', but others argue, that maybe the alphabet is, but not the interpretations of it.*

H The main point of typography and ownership today is that there is a blurred border between language and letters. So: now you can own the 'shape' of a letter. Traditionally, the way typographers made a living was by buying (more or less expensive) lead fonts, and with this tool they printed books and got paid for that. They got paid for the typesetting, not for the type. That was the work of the foundries. Today, thanks to the digital tools, you can easily switch between type design, type setting and graphic design.

FS *What about the idea that fonts might be the most 'pirated' digital object possible? Copying is much more difficult when you've got lead type to handle!*

H Yes, digitalisation changed the rules. Just as `.mp3` changed the philosophy of music. But in typography, there is a strange confrontation between this flux of copied information, piracy and old rules of ownership from the past.

FS *Do you think the culture of sharing fonts changed? Or: the culture of distributing them? If you look at most licences for fonts, they are extremely restrictive. Even 99% of free fonts do not allow derivative works.*

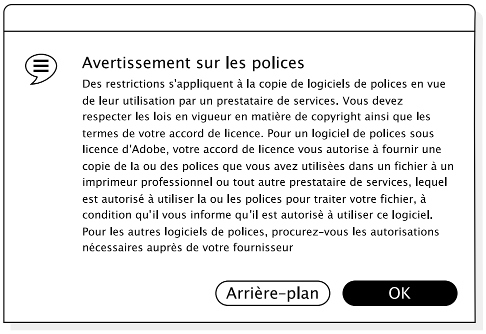
H The public good culture is paradoxically not often there. Or at least the economical model of living with public good idea is not very developed. While I think typography, historically, is always seen as a way to share knowledge. Humanist stuff.

# YOU NEED TO COPY

> The art and craft of typeface design is currently headed for extinction due to the illegal proliferation of font software, piracy, and general disregard for proper licensing etiquette.[1]

H Emigré ... Did they not live from the copyrights of fonts?!

FS *You are right. They are like a commercial record company. Can you imagine what would happen if you would open up the typographic trade – to 'Open Source' this economy? Stop chasing piracy and allow users to embed, study, copy, modify and redistribute typefaces?*



Avertissement sur les polices

Des restrictions s'appliquent à la copie de logiciels de polices en vue de leur utilisation par un prestataire de services. Vous devez respecter les lois en vigueur en matière de copyright ainsi que les termes de votre accord de licence. Pour un logiciel de polices sous licence d'Adobe, votre accord de licence vous autorise à fournir une copie de la ou des polices que vous avez utilisèes dans un fichier à un imprimeur professionnel ou tout autre prestataire de services, lequel est autorisé à utiliser la ou les polices pour traiter votre fichier, à condition qu'il vous informe qu'il est autorisè à utiliser ce logiciel. Pour les autres logiciels de polices, procurez-vous les autorisations nécessaires auprès de votre fournisseur

*Warning message when attempting to embed a font in InDesign*

H Well we are not that far from this in fact. Every designer has at least 500 fonts on their computer, not licenced, but copied because it would be impossible to pay for!

FS *Even the distribution model of fonts is very peer-to-peer as well. The reality might come close, but font licences tell a different story.*

> *I believe that we live in an era where anything that can be expressed as bits will be. I believe that bits exist to be copied. Therefore, I believe that any business-model that depends on your bits not being copied is just dumb, and that lawmakers who try to prop these up are like governments that sink fortunes into protecting people who insist on living on the sides of active volcanoes.*[2]

---

1 http://redesign.emigre.com/FAQ.php
2 Cory Doctorow in http://craphound.com/bio.php

*I am not saying all fonts should be open, but it is just that it would be interesting when type designers were testing and experimenting with other ways of developing and distributing type, with another economy.*

H Yes, but fonts have a much more reduced user community than music or bookpublishing, so old rules stay.

FS *Is that it? I am surprised to see that almost all typographers and foundries take the 'piracy is a crime' side on this issue. While typographers are early and enthusiastic adopters of computer technology, they have not taken much from the collaborative culture that came with it.*

H This is the 'tradition' typography inherited. Typography was one of the first laboratories for fractioning work for efficiency. It was one of the first modern industries, and has developed a really deep culture where it is not easy to set doubts in. 500 years of tradition and only 20 years of computers. The complexity comes from the fact it is influenced by a multiple series of elements, from history and tradition to the latest technologies. But it is always related to an economic production system, so property and 'secrets-of-the-trade' have a big influence on it.

FS *I think it is important to remember how the current culture of (not) sharing fonts is linked to its history. But books have been made for quite a while too.*

H Open Source systems may be not so much influencing distribution, licences and economic models in typography, but can set original questions to this problematic of digital type. Old tools and histories are not reliable anymore.

FS *Yes. with networked software it is rather obvious that it is useful to work together. I try to understand how this works with respect to making a font. Would that work?*

H Collaborative type is extremely important now, I think. The globalisation of computer systems sets the language of typography in a new dimension. We use computers in Belgium and in China. Same hardware. But language is the problem! A French typographer might not be the best person to define a Vietnamese font. Collaborativity is necessary! Pierre Huyghebaert told me he once designed an Arabic font when he was in Lebanon. For him, the font was legible, but nobody there was able to read it.

FS *But how would you collaborate than? I mean … what would be the reason for a French typographer to collaborate with one from China? What would that bring? I'm imagining some kind of hybrid result … kind of interesting.*

H Again, sharing. We all have the idea that English is the modern Latin, and if we are not careful the future of computers will result in a language reductionism.

FS *What interest me in Open Source, is the potential for 'biodiversity'.*

H I partially agree, and the Open Source idea contradicts the reductionist approach by giving more importance to local knowledge. A collaboration between an Arabic typographer and a French one can be to work on tools that allow both languages to co-exist. LaTeX permits that, for example. Not QuarkXpress!

FS *Where does your interest in typography actually come from?*

H I think I first looked at comic books, and then started doodling in the margins of schoolbooks. As a teenager, I used to reproduce film titles such as Aliens, Terminator or other sci-fi high-octane typographic titles.

THE
TERMINATOR

Basically, I'm a forger! In writing, you need to copy to understand. Thats an old necessity. If you use a typeface, you express something. You're putting drawings of letters next to each other to compose a word/text. A drawing is always emotionally charged, which gives color (or taste) to the message. You need to know what's inside a font to know what it expresses.

FS *How do you find out what's inside?*

H By reproducing letters, and using them. A Gill Sans does not have the same emotional load as a Bodoni. To understand a font is complicated, because it refers to almost every field in culture. The banners behind G.W. Bush communicate more than just 'Mission Accomplished'. Typefaces carry a 'meta language'.

# Innovations in Compassion

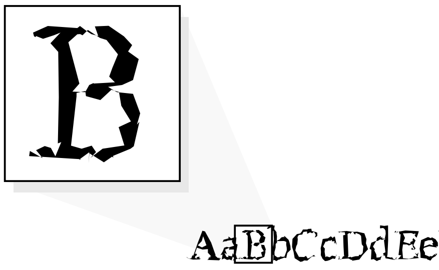
**The Faith-Based and Community Initiative:
A Final Report to the Armies of Compassion**

FS *It is truly embedded content*

H Exactly! It is still very difficult to bridge the gap between personal emotions and programming a font. Moreover, there are different approaches, from stroke design to software that generates fonts. And typography is standardisation. The first digital fonts are drawn fixed shapes, letter by letter, 'outstrokes'. But there is another approach where the letters are traced by the computer. It needs software to be generated. In Autocad, letters are 'innerstroke' that can vary of weight. Letterrors' Beowolf[3] is also an example of that kind of approach. interesting way to work, but the font depends on the platform it goes with. Beowolf only works on OS9. It also set the question of copyright very far. It's a case study in itself.

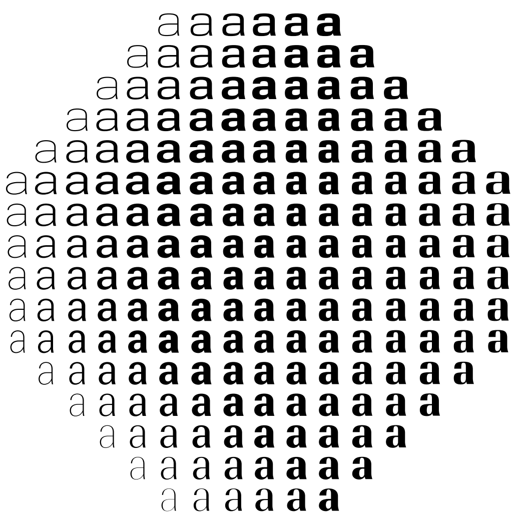
FS *So it means, the font is software in fact?*

H Yes, but the interdependence between font and operating systems is very strong, contrary to a fixed format such as TrueType. For printed matter, this is much more complicated to achieve. There are in-between formats, such as Multiple Master Technology for example. It basically means, that you have 2 shapes for 1 glyph, and you can set an 'alternative' shape between the 2 shapes. At Adobe they still do not understand why it was (and still is) a failure …



*Beowolf by Just van Rossum and Erik van Blokland (1989)*

[3] Instead of recreating a fixed outline or bitmap, the Randomfont redefines its outlines every time they are called for. http://letterror.com/writing/is-best-really-better


*The Metapolator Uinverse by Simon Egli (2014)*

FS *I really like this idea … to have more than one master. Imagine you own one master and I own the other and than we adjust and tweak from different sides. That would be real collaborative type! Could 'multiple' mean more than one you think?*

H It is a bit more complicated than drawing a simple font in Fontographer or Fontforge. Pierre told me that the MM feature is still available in Adobe Illustrator, but that it is used very seldomly. Multiple Master fonts are also a bit complicated to use. I think there were a lot of bugs first, and then you need to be a skilled designer to give these fonts a nice render. I never heard of an alternative use of it, with drawing or so. In the end it was probably never a success because of the software dependency.

FS *While I always thought of fonts as extremely cross media. Do you remember which classic font was basically the average between many well-known fonts? Frutiger?*

H Fonts are Culture Capsules! It was Adrian Frutiger. But he wasn't the only one to try … It was a research for the Univers font I think. Here again we meet this paradox of typography: a standardisation of language generating cultural complexity.

FS *Univers. That makes sense. Amazing to see those examples together. It seems digital typography got stuck at some point, and I think some of the ideas and practices that are current in Open Source could help break out of it.*

H Yes of course. And it is almost virgin space.

FS *In 2003 the Danish government released 'Union', a font that could be freely used for publications concerning Danish culture. I find this an intrigueing idea, that a font could be seen as some kind of 'public good'.*

Univers

Aa Ee Rr

*Aa Ee Rr*

a

**Ausstellung**

abcdefghijklm
nopqrstuvwxyz
0123456789

*Univers by Adrian Frutiger (1954)*

Union regular

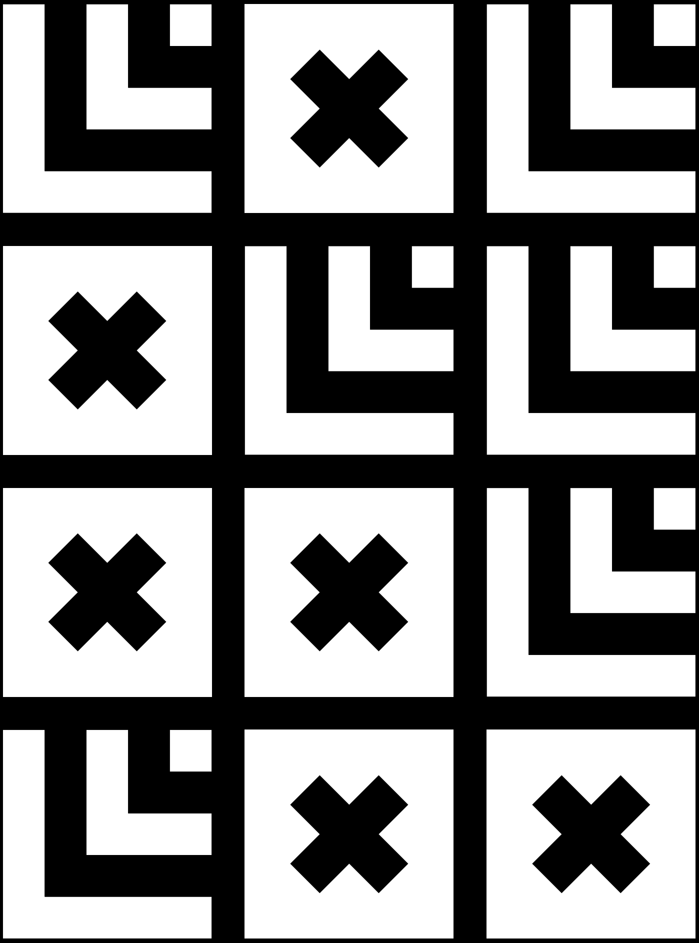
**Union bold**

*Union by Morten Rostgaard Olsen (2003)*

H I am convinced that knowledge needs to be open … (speaking as the son of a teacher here!). One medium for knowledge is language and its atoms are letters.

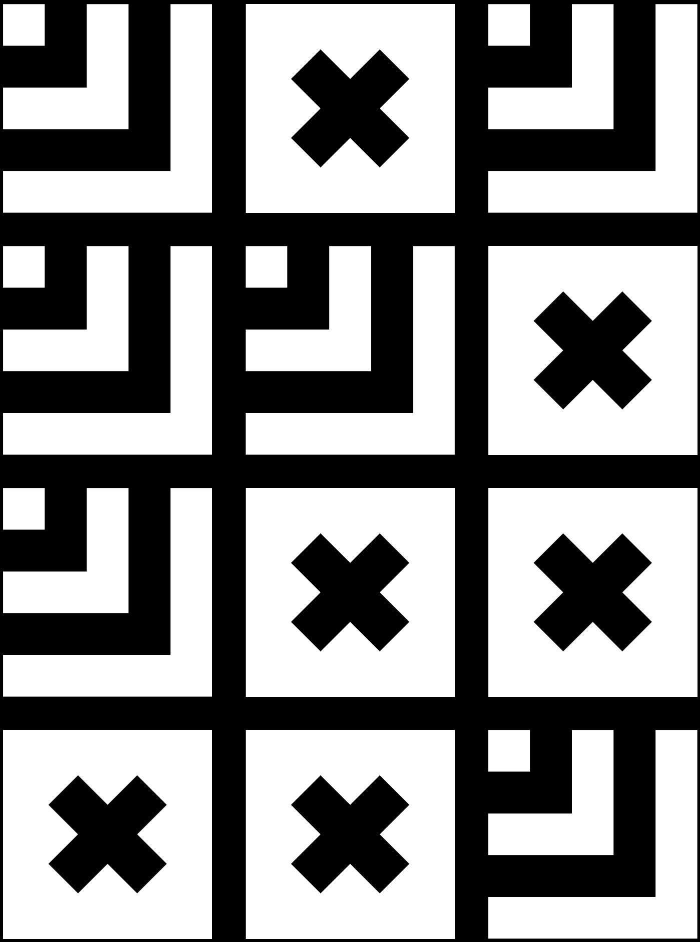
FS *But if information wants to be free, does that mean that design needs to be free too? Is there information possible without design?*

H This is why I like books. Because it's a mix between information and beauty – or can be. Pfff, there is nothing without design … It is like is there something without language, no?



# What's the thinking here

Femke Snelting
Matthew Fuller



# What's the thinking here

Femke Snelting
Matthew Fuller

One of the things that is notable about OSP is that the problems that you encounter are also described, appearing on your blog. This is something unusual for a company attempting to produce the impression of an efficient "solution". Obviously the readers of the blog only get a formatted version of this, as a performed work? What's the thinking here?"

This interview about the practice of OSP was carried out by e-mail **between March and May 2008**. Matthew Fuller writes about software culture and has a contagious interest in technologies that exceed easy fit solutions. At the time, he was David Gee reader in Digital Media at the Centre for Cultural Studies, Goldsmiths College, University of London, and had just edited *Software Studies, A Lexicon*,[1] and written *Media Ecologies: Materialist Energies in Art and Technoculture*[2] and *Behind the Blip: Essays on the Culture of Software*.[3]

*OSP is a graphic design agency working solely with Open Source software. This surely places you currently as a world first, but what exactly does it mean in practice? Let's start with what software you use?*

There are other groups publishing with Free Software, but design collectives are surprisingly rare. So much publishing is going on around Open Source and Open Content ... someone must have had the same idea? In discussions about digital tools you begin to find designers expressing concern over the fact that their work might all look the same because they use exactly the same Adobe suite and as a way to differentiate yourself, Free Software could soon become more popular. I think the success of Processing is related to that, though I doubt such a composed project will ever make anyone seriously consider Scribus for page layout, even if Processing is Open Source.

[1] Matthew Fuller. *Software Studies: A Lexicon*. The MIT Press, 2008

[2] Matthew Fuller. *Media Ecologies: Materialist Energies in Art and Technoculture*. The MIT Press, 2007

[3] Matthew Fuller. *Behind the Blip: Essays on the culture of software*. Autonomedia, 2003

OSP usually works between GIMP,[4] Scribus[5] and Inkscape[6] on Linux distributions and OSX. We are fans of FontForge,[7] and enjoy using all kinds of commandline tools, `psnup`, `ps2pdf` and `uniq` to name a few.

*How does the use of this software change the way you work, do you see some possibilities for new ways of doing graphic design opening up?*

For many reasons, software has become much more present in our work; at any moment in the workflow it makes itself heard. As a result we feel a bit less sure of ourselves, and we have certainly become slower. We decided to make the whole process into some kind of design/life experiment and that is one way to keep figuring out how to convert a file, or yet another discussion with a printer about which 'standard' to use, interesting for ourselves. Performing our practice is as much part of the project as the actual books, posters, flyers etc. we produce.

One way a shift of tools can open up new ways of doing graphic design, is because it makes you immediately aware of the 'resistance' of digital material. At the point we can't make things work, we start to consider formats, standards and other limitations as ingredients for creative work. We are quite excited for example about exploring dynamic design for print in SVG, a by-product of our battle with converting files from Scalable Vector Format into Portable Document Format.

Free Software allows you to engage on many levels with the technologies and processes around graphic design. When you work through it's various interfaces, stringing tools together, circumventing bugs and/or gaps in your own knowledge, you understand there is more to be done than contributing code in C++. It is an invitation to question assumptions of utility, standards and usability. This is exactly the stuff design is made of.

*Following this, what kind of team have you built up, and what new competencies have you had to develop?*

The core of OSP is five people[8], and between us we mix amongst others typography, layout, cartography, webdesign, software development, drawing,

[4] image manipulation
[5] page layout
[6] vector editing
[7] font editor
[8] Pierre Huyghebaert, Harrisson, Yi Jiang, Nicolas Malevé and me

programming, open content licensing and teaching. Around it is a larger group of designers, a mathematician, a computer scientists and several Free Software coders that we regularly exchange ideas with.
It feels we often do more unlearning than learning; a necessary and interesting skill to develop is dealing with incompetence – what can it be else than a loss of control? In the mean time we expand our vocabulary so we can fuel conversations (imaginary and real life) with people behind GIMP, Inkscape, Scribus etc.; we learn how to navigate our computers using commandline interfaces as well as KDE, GNOME and others; we find out about file formats and how they sometimes can and often cannot speak to each other; how to write manuals and interact with mailing lists. The real challenge is to invent situations that subvert strict divisions of labour while leaving space for the kind of knowledge that comes with practice and experience.

*Open fonts seem to be the beginnings of a big success, how does it fit into the working practices of typographers or the material with which they work?*

Type design is an extraordinary area where Free Software and design naturally meet. I guess this area of work is what kernel coding is for a Linux developer: only a few people actually make fonts but many people use them all the time. Software companies have been inconsistent in developing proprietary tools for editing fonts, which has made the work of typographers painfully difficult at times. This is why George Williams decided to develop FontForge, and release it under a BSD license: even if he stops being interested, others can take over. FontForge has gathered a small group of fans who through this tool, stay into contact with a more generous approach to software, characters and typefaces.
The actual material of a typeface has since long migrated from poisonous lead into sets of ultra light vector drawings, held together in complicated kerning systems. When you take this software-like aspect as a startingpoint, many ways to collaborate (between programmers and typographers; between people speaking different languages) open up, as long as you let go of the uptight licensing policies that apply to most commercial fonts. I guess the image of the solitary master passing on the secret trade to his devoted pupils does not sit very well with the invitation to anyone to run, copy, distribute, study, change and improve. How open fonts could turn the patriarchal guild

system inside out that has been carefully preserved in the closed world of type design, is obviously of interest as well.
Very concretely, computer-users really need larger character sets that allow for communication between let's say Greek, Russian, Slovak and French. These kinds of vast projects are so much easier to develop and maintain in a Free Software way; the DéJàVu font project shows that it is possible to work with many people spread over different countries modifying the same set of files with the help of versioning systems like CVS.
But what it all comes down to probably … Donald Knuth is the only person I have seen both Free Software developers and designers wear on their T-shirts.

*The cultures around each of the pieces of software are quite distinct. People often lump all F/LOSS development into one kind of category, whereas even in the larger GNU/Linux distros there is quite a degree of variation, but with the smaller more specialised projects this is perhaps even more the case. How would you characterise the scenes around each of these applications?*

The kinds of applications we use form a category in themselves. They are indeed small projects so 'scene' fits them better than 'culture'. Graphics tools differ from archetypal Unix/Linux code and language based projects in that Graphical User Interfaces obviously matter and because they are used in a specialised context outside its own developers circle. This is interesting because it makes F/LOSS developer communities connect with other disciplines (or scenes?) such as design, printing and photography.
A great pleasure in working with F/LOSS is to experience how software can be done in many ways; each of the applications we work with is alive and particular. I'll just portray Scribus and Inkscape here because from the differences between these two I think you can imagine what else is out there.
The Scribus team is rooted in the printing and pre-press world and naturally their first concern is to create an application that produces reliable output. Any problem you might run in to at a print shop will be responded to immediately, even late night if necessary. Members of the Scribus team are a few years older than average developers and this can be perceived through the correct and friendly atmosphere on their mailing list and IRC channel, and their long term loyalty to this complex project. Following its more industrial perspective, the imagined design workflow built in to the tool is

linear. To us it feels almost pre-digital: tasks and responsibilities between editors, typesetters and designers are clearly defined and lined up. In this view on design, creative decisions are made outside the application, and the canvas is only necessary for emergency corrections. Unfortunately for us, who live of testing and trying, Scribus' GUI is a relatively underdeveloped area of a project that otherwise has matured quickly.
Inkscape is a fork of a fork of a small tool initially designed to edit vector files in SVG format. It stayed close to its initial starting point and is in a way a much more straightforward project than Scribus. Main developer Bryce Harrington describes Inkscape as *a relatively unstructured coming and going of high energy collective work* much work is done through a larger group of people submitting small patches and it's developers community is not very tightly knit. Centered around a legible XML format primarily designed for the web, Inkscape users quickly understand the potential of scripting images and you can find a vibrant plug in culture even if the Inkscape code is less clean to work with than you might expect. Related to this interest in networked visuals, is the involvement of Inkscape developers in the Open Clip Art project and ccHost, a repository system wich allows you to upload images, sounds and other files directly from your application. It is also no surprise that Inkscape implemented a proper print dialogue only very late, and still has no way to handle CMYK output.

*There's a lot of talk about collaboration in F/LOSS development, something very impressive, but often when one talks to developers of such software there is a lot to discuss about the rather less open ways in which power struggles over the meaning or leadership of software projects are carried out by, for instance, hiding code in development, or by only allowing very narrowly technical approaches to development to be discussed. This is only one tendency, but one which tends to remain publicly under-discussed. How much of this kind of friction have you encountered by acting as a visible part of a new user community for F/LOSS?*

I can't say we feel completely at home in the F/LOSS world, but we have not encountered any extraordinary forms of friction yet. We have been allowed the space to try our own strategies at overcoming the user-developer divide: people granted interviews, accepted us when we invited ourselves to speak at conferences and listened to our stories. But it still feels a bit awkward, and I sometimes wonder whether we ever will be able to do enough. Does

constructive critique count as a contribution, even when it is not delivered in the form of a bug report? Can we please get rid of the term 'end-user'? Most discussions around software are kept strictly technical, even when there are many non-technical issues at stake. We are F/LOSS enthusiasts because it potentially pulls the applications we use into some form of public space where they can be examined, re-done and taken apart if necessary; we are curious about how they are made because of what they (can) make you do. When we asked Andreas Vox, a main Scribus developer whether he saw a relation between the tool he contributed code to, and the things that were produced by it, he answered: *Preferences for work tools and political preference are really orthogonal.* This is understandable from a project-management point of view, but it makes you wonder where else such a debate should take place.
The fact that compared to proprietary software projects, only a very small number of women is involved in F/LOSS makes apparent how openness and freedom are not simple terms to put in practice. When asked whether gender matters, the habitual answer is that opportunities are equal and from that point a constructive discussion is difficult. There are no easy solutions, but the lack of diversity needs to be put on the roadmap somehow, or as a friend asked: *Where do I file a meta-bug?*

*Visually, or in terms of the aesthetic qualities of the designs you have developed would you say you have managed to achieve anything unavailable through the output of the Adobe empire?*

The members of OSP would never have come up with the idea to combine their aesthetics and skills using Adobe, so that makes it difficult to do a 'before' and 'after' comparison. Or maybe we should call this an achievement of Free Software too?
Using F/LOSS has made us reconsider the way we work and sometimes this is visible in the design we produce, more often in the commissions we take on or the projects we invest in. Generative work has become part of our creative suite and this certainly looks different than a per-page treatment; also deliberate traces of the production process (including printing and pre-press) add another layer to what we make.
Of all smaller and larger discoveries, the Spiro toolkit that Free Software activist, Ghostscript maintainer, typophile and Quaker Raph Levien devel-

ops, must be the most wonderful. We had taken Bézier curves for granted, and never imagined how the way it is mathematically defined would matter that much. Instead of working with fixed anchor points and starting from straight lines that you first need to bend, Spiro is spiral-based and vectors suddenly have a sensational flow and weight. From Pierre Bézier writing his specification as an engineer for the Renault car factory to Levien's Spiro, digital drawing has changed radically.

*You have a major signage project coming up, how does this commission map across to the ethics and technologies of F/LOSS?*

We are right in the middle of it. At this moment 'The Pavilion of Provisionary Happiness' celebrating the 50th anniversary of the Belgian World Exhibition, is being constructed out of 30.000 beer crates right under the Brussels' Atomium. That's a major project done the Belgian way.
We have developed a signage system, or actually a typeface, which is defined through the strange material and construction work going on on site. We use holes in the facade that are in fact handles of beer crates as connector points to create a modular font that is somewhere between Pixacao graffiti and Cuneiform script. It is actually a play on our long fascination with engineered typefaces such as DIN 1451; mixing universal application with specific materials, styles and uses – this all links back to our interest in Free Software.
Besides producing the signage, OSP will co-edit and distribute a modest publication documenting the whole process; it makes legible how this temporary yellow cathedral came about. And the font will of course be released in the public domain.
It is not an easy project but I don't know how much of it has to do with our software politics; our commissioners do not really care and also we have kept the production process quite simple on purpose. But by opening our sources, we can use the platform we are given in a more productive way; it makes us less dependent because the work will have another life long after the deadline has passed.

*On this project, and in relation to the seeming omnipresence in F/LOSS of the idea that this technology is 'universal', how do you see that in relation to fonts, and their longer history of standards?*

That is indeed a long story, but I'll give it a try. First of all, I think the idea of universal technology appears to be quite omnipresent everywhere; the mix-up between ubiquitousness and 'universality' is quickly made. In Free Software this idea gains force only when it gets (con)fused with freedom and openness and when conditions for access are kept out of the discussion. We are interested in early typographic standardization projects because their minimalist modularity brings out the tension between generic systems and specific designs. Ludwig Goller, a Siemens engineer wo headed the Committee for German Industry Standards in the 1920s stated that *For the typefaces of the future neither tools nor fashion will be decisive.* His committee supervised the development of DIN 1451, a standard font that should connect economy of use with legibility, and enhance global communication in service of the German industry. I think it is no surprise that a similar phrasing can be found in W3C documents; the idea to unify the people of the world through a common language re-surfaces and has the same tendency to negate materiality and specificity in favour of seamless translation between media and markets.
Type historian Ellen Lupton brought up the possibility of designing typographic systems that are accessible but not finite nor operating within a fixed set of parameters. Although I don't know what she means by using the term 'open universal', I think this is why we are attracted to Free Software: it has the potential to open up both the design of parameters as well as their application. Which leads to your next question.

*You mentioned the use of generative design just now. How far do you go into this? Within the generative design field there seem to be a couple of tendencies, one that is very pragmatic, simply about exploring a space of possible designs through parametric definition in order to find, select and breed from and tweak a good result that would not be necessarily imaginable otherwise, the other being more about the inefible nature of the generative process itself, something vitalist. These tendencies of not of course exclusive, but how are they inflected or challenged in your use of generative techniques?*

I feel a bit on thin ice here because we only start to explore the area and we are certainly not deep into algorithmic design. But on a more mundane level … in the move from print to design for the web, 'grids' have been replaced by 'templates' that interact with content and context through filters. Designers have always been busy with designing systems and formats,[9] but stepped in to manipulate singular results if necessary.

I referred to 'generative design' as the space opening up when you play with rules and their affordances. The liveliness and specificity of the work results from various parameters interfering with each other, including the ones we can get our hands on. By making our own manipulations explicit, we sometimes manage to make other parameters at play visible too. Because at the end of the day, we are rather bored by mysterious beauty.

*One of the techniques OSP uses to get people involved with the process and the technologies is the 'Print Party', can you say what that is?*

'Print Parties' are irregular public performances we organise when we feel the need to report on what we discovered and where we've been; as anti-heroes of our own adventures we open up our practice in a way that seems infectious. We make a point of presenting a new experiment, of producing something printed and also something edible on site each time; this mix of ingredients seems to work best. 'Print Parties' are how we keep contact with our fellow designers who are interested in our journey but have sometimes difficulty following us into the exotic territory of BoF, Version Control and GPL3.

*You state in a few texts that OSP is interested in glitches as a productive force in software, how do you explain this to a printer trying to get a file to convert to the kind of thing they expect?*

Not! Printing has become cheap through digitization and is streamlined to the extreme. Often there is literally no space built in to even have a second look at a differently formatted file, so to state that glitches are productive is easier said than done. Still, those hickups make processes tangible, especially at moments you don't want them to interfere.

For a book we are designing at the moment, we might partially work by hand on positive film (a step now also skipped in file-to-plate systems). It makes us literally sit with pre-press professionals for a day and hopefully we can learn better where to intervene and how to involve them into the process. To take the productive force of glitches beyond predictable aesthetics, means

[9] it really made me laugh to think of Joseph Müller Brockman as vitalist

most of all a shift of rhythm – to effect other levels than the production process itself. We gradually learn how our ideas about slow cooking design can survive the instant need to meet deadlines. The terminology is a bit painful but to replace 'deadline' by 'milestone', and 'estimate' by 'roadmap' is already a beginning.

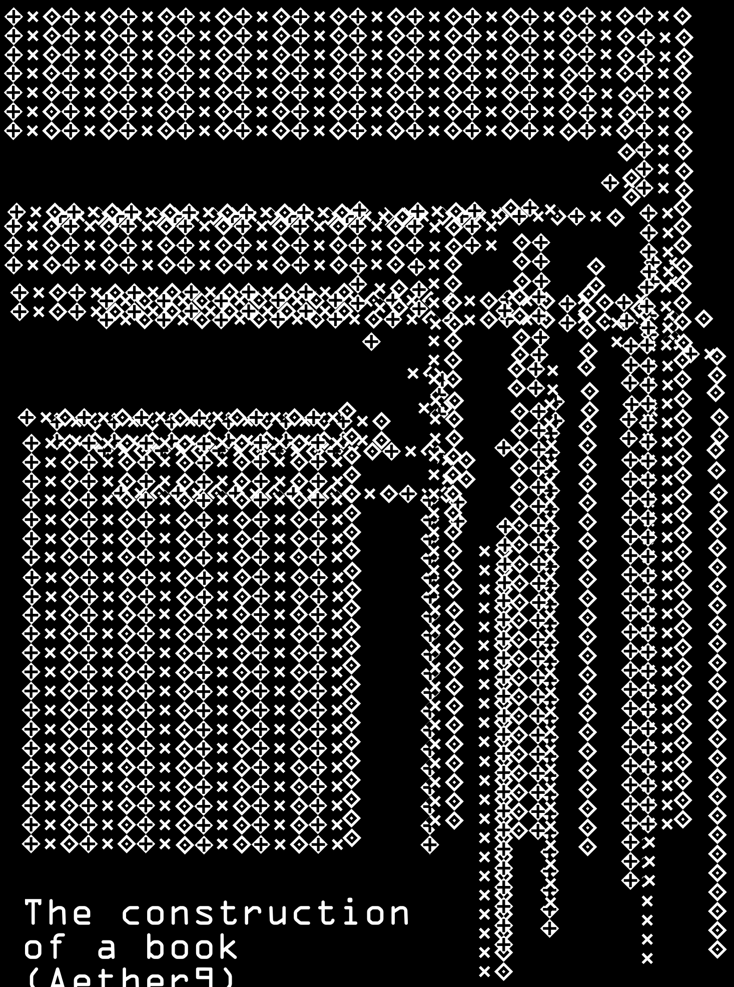
*One of the things that is notable about OSP is that the problems that you encounter are also described, appearing on your blog. This is something unusual for a company attempting to produce the impression of an efficient 'solution'. Obviously the readers of the blog only get a formatted version of this, as a performed work? What's the thinking here?*

'Efficient solutions' is probably the last thing we try to impress with, though it is important for us to be grounded in practice and to produce for real under conventional conditions. The blog is a public record of our everyday life with F/LOSS; we make an effort to narrate through what we stumble upon because it helps us articulate how we use software, what it does to us and what we want from it; people that want to work with us, are somehow interested in these questions too. Our audience is also not just prospective clients, but includes developers and colleagues. An unformatted account, even if that was possible, would not be very interesting in that respect; we turn software into fairytales if that is what it takes to make our point.

*In terms of the development of F/LOSS approaches in areas outside software, one of the key points of differentiation has been between 'recipes' and 'food', bits and atoms, genotype and phenotype. That is that software moves the kinds of rivalry associated with the ownership and rights to use and enjoy a physical object into another domain, that of speed and quality of information, which network distribution tends to mitigate against. This is also the same for other kinds of data, such as music, texts and so on. (This migration of rivalry is often glossed over in the description of 'goods' being 'non-rivalrous'.) Graphic Design however is an interesting middle ground in a certain way in that it both generates files of many different kinds, and, often but not always, provides the 'recipes' for physical objects, the actual 'voedingstof', such as signage systems, posters, books, labels and so on. Following this, do you circulate your files in any particular way, or by other means attempt to blur the boundary between the recipe and the food?*

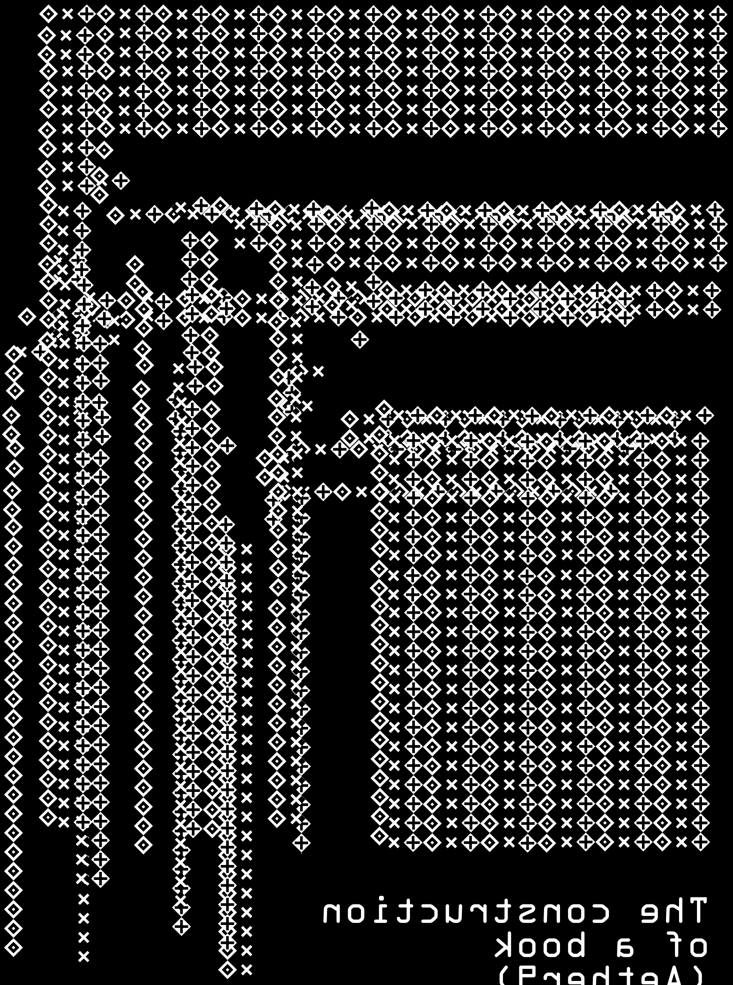
We have just finished the design of a font (NotCourier-sans), a derivative of Nimbus Mono, which is in turn a GPL'ed copy of the well known Courier typeface that IBM introduced in 1955. Writing a proper licence for it, opened up many questions about the nature of 'source code' in design, and not only from a legalist perspective. While this is actually relatively simple to define for a font (the source is the object), it is much less clear what it means for a signage system or a printed book.
One way we deal with this, is by publishing final results side by side with ingredients and recipes. The raw files themselves seem pretty useless once the festival is over and the book printed, so we write manuals, stories, histories. We also experiment with using versioning systems, but the softwares available are only half interesting to us. Designed to support code development, changes in text files can be tracked up to the minutest detail but unless you are ready to track binary code, images and document layouts function as black boxes. I think this is something we need to work on because we need better tools to handle multiple file formats collaboratively, and some form of auto-documentation to support the more narrative work.
On the other hand, manuals and licences are surprisingly rich formats if you want to record how an object came into life; we often weave these kinds of texts back into the design itself. In the case of NotCourierSans we will package the font with a pdf booklet on the history of the typeface – mixing design geneology with suggestions for use.
I think the blurring of boundaries happens through practice. Just like recipes are linked in many ways to food,[10] design practice connects objects to conditions. OSP is most of all interested in the back-and-forth between those two states of design; rendering their interdepence visible and testing out ways of working with it rather than against it. Hopefully both the food and the recipe will change in the process.

[10] tasting, trying, writing, cooking



# The construction of a book (Aether9)

✚ ManuelSchmalstieg
⊡ Pierre Marchand
✖ Ludivine Loiseau



The construction
of a book
(Aether9)

+ ManuelSchmalstieg
⊡ Pierre Marchand
× Ludivine Loiseau

This brief interview with Ludivine Loiseau and Pierre Marchand from OSP was made in **December 2012** by editor and designer Manuel Schmalstieg. It unravels the design process of *Aether9*, a book based on the archives of a collaborative adventure exploring the danger zones of networked audio-visual live performance. The text was published in that same publication.

✚ ***Can you briefly situate the collective work of Open Source Publishing (OSP)?***

✕ OSP is a working group producing graphic design objects using only Libre and/or Open Source software. Founded in 2006 in the frame of the arts organisation Constant[1], the OSP caravan consists today of a dozen individuals of different backgrounds and practices.

✚ ***Since how long are you working as a duo, and as a team in OSP?***

⊡ *3 to 4 years.*

✚ ***And how many books have you conceived?***

⊡ *As a team, it's our first 'real' book. We previously worked together on a somewhat similar project of archive exploration, but without printed material in the end.*[2]

✚ ***Similar in the type of content or in the process?***

⊡ *The process: we developed scripts to 'scrap' the project archives, but it's output was more abstract; we collected the fonts used in all the files and produced a graph from this process. These archives weren't structured, so the exploration was less linear.*

✚ ***You rapidly chose TeX/ConTeXt as a software environment to produce this book. Was it an obvious choice given the nature of the project, or did you hesitate between different approaches?***

✕ The construction of the book focused on two axes/threads: chronology and a series of 'trace-route' keywords. Within this approach of reading and navigation using cross-references, ConTeXt appeared as an appropriate tool.

[1] http://www.constantvzw.org
[2] http://www.ooooo.be/interpunctie/

✚ ***The world of TeX[3] is very intriguing, in particular for graphic designers. It seems to me that it is always a struggle to push back the limits of what is 'intended' by the software.***

⊡ *ConTeXt is a constant fight! I wouldn't say the same about other TeX system instances. With ConTeXt, we found ourselves facing a very personal project, because composition decisions are hardcoded to the liking of the package main maintainer. And when we clash with these decisions, we are in the strange position of using a tool while not agreeing with its builder.*

✕ As a concrete example, we could mention the automatic line spacing adjustments. It was a struggle to get it right on the lines that include keywords typeset with our custom 'traced' fonts. ConTeXt tried to do better, and was increasing the line height of those words, as if it wanted to avoid collisions.

✚ ***Were you ever worried that what you wanted to obtain was not doable? Did you reject some choices – in the graphic design, the layout, the structure – because of software limitations?***

✕ Yes. Opting for a two column layout appeared to be quite tough when filling in the content, as it introduced many gaps. At some point we decided to narrow the format on a single column. To obtain the two columns layout in the final output, the whole book was recomposed during the pdf-construction, through OSPImpose.

⊡ *This allowed us to make micro adjustments in the end of the production process, while introducing new games, such as shifting the images on double pages.*

✚ ***What is OSPImpose?***

⊡ *It's a re-writing of a pdf imposition software that I wrote a couple years ago for PoDoFo.*

✚ ***Again regarding ConTeXt: this system was used for other OSP works – notably for the book*** **Verbindingen/Jonctions 10; Tracks in electr(on)ic fields.[4]** ***Is it currently the main production tool at OSP?***

⊡ *It's more like an in-depth initiation journey!*

✕ But it hasn't become a standard in our workflow yet. In fact, each new important book layout project raises each time the question of the

[3] a software written in 1978 by Donald Knuth

[4] distinguished by the Fernand Baudin Prize 2009

tool. Scribus and LibreOffice (spreadsheet) are also part of our book making toolbox.

✚ ***During our work session with you at Constant Variable, we noticed that it was difficult to install a sufficiently complete TeX/ConTeXt/Python environment to be able to generate the book. Is Pierre's machine still the only one, or did you manage to set it up on other computers?***

⊡ *Now we all have similar setups, so it's a generalized generation. But it's true that this represented a difficulty at some times.*

✚ ***The source code and the Python scripts created for the book are publicly accessible on the OSP Git server. Would these sources be realistically re-usable? Could other publication projects use parts of the code ? Or, without any explicit documentation, would it be highly improbable?***

✖ Indeed, the documentation part is still on the to-do list. Yet a large part of the code is quite directly reusable. The code allows to parse different types of files. E-mails and chat-logs are often found in project archives. Here the Python scripts allows to order them according to date information, and will automatically assign a style to the different content fields.

⊡ *The code itself is a documentation source, as much on concrete aspects, such as e-mail parsing, than on a possible architecture, on certain coding motives, etc. And most importantly, is consists in a form of common experience.*

✚ ***Do you think you will reuse some of the general functions/features of archive parsing for other projects ?***

⊡ *Hard to say. We haven't anything in perspective that is close to the* Aether9 *project. But for sure, if the need of such treatment comes up again, we'll retrieve these software components.*

✖ Maybe for a publication/compilation of OSP's adventures.

✚ ***Have there been 'revelations', discoveries of unsuspected Python/ConText features during this development?***

⊡ *I can't recall having this kind of pleasure. The revelation, at least from my point of view, happened in the very rich articulation of a graphical intention enacted in programming objects. It remains a kind of uncharted territory, exploring it is always an exciting adventure.*

**✚ *Three fonts are used in the book: Karla, Crimson and Consola Mono. Three pretty recent fonts, born in the webfonts contexts I believe. What considerations brought you to this choice?***

✖ Our typographical choices and researches lead us towards fonts with different style variations. As the textual content is quite rich and spreads on several layers, it was essential to have variation possibilities. Also, each project brings the opportunity to test new fonts and we opted for recently published fonts, indeed published, amongst others, on the Google font directory. Yet Karla and Crimson aren't fonts specifically designed for a web usage. Karla is one of the rare libre grotesque fonts, and it's other specificity it that it includes Tamil glyphs.

**✚ *Apart from the original glyphs specially created for this book, you drew the Ç glyph that was missing to Karla ... Is it going to be included to its official distribution?***

✖ Oh, that's a proposal for Jonathan Pinhorn. We haven't contacted him yet. For the moment, this cedilla has been snatched from the traced variant collections.

**✚ *Were there any surprises when printing? I am thinking in particular of your choice of a colored ink instead of the usual black, or to the low res quality (72dpi) of most of the images.***

▣ *At the end of the process, the spontaneous decision to switch to blue ink was a guaranteed source of surprise. We were confident that it wouldn't destroy the book, and we surely didn't take too many risks since we were working with low res images. But we weren't sure how the images would react to such an offense. It was an great surprise to see that it gave the book a very special radiance.*

**✚ *What are your next projects?***

✖ We are currently operating as an invited collective at the Valence Academy of Fine Arts in the frame of a series of workshops named 'Up pen down'. We're preparing a performance for the Balsamine theatre[5] on the topic of Bootstrapping. In April we will travel as a group to Madrid to LGRU[6] and LGM[7]. We also continually work on 'Co-position"', a project for building a post-gutenberg typographical tool.

✚ **Manuel Schmalstieg** ▣ *Pierre Marchand* ✖ Ludvine Loiseau

[5] http://www.balsamine.be/

[6] http://lgru.net/

[7] the international Libre Graphics Meeting: http://libregraphicsmeeting.org/2013/

# PERFORMING LIBRE GRAPHICS

Cornelia Sollfrank
Femke Snelting

SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
S C I H P A R G   E R B I L   G N I M R O F R E P
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP
SCIHPARG ERBIL GNIMROFREP

knarfllloS ailenroC
gnitlenS ekmeF

In **April 2014** I traveled from Leipzig to the north of Germany to meet with artist Cornelia Sollfrank. It was right after the Libre Graphics Meeting, and the impressions from the event were still very fresh. Cornelia had asked me for a video interview as part of *Giving what you don't have*,[1] a series of conversations about what she refers to as 'complex copyright-critical practices'. She was interested in forms of appropriation art that instead of claiming some kind of 'super-user' status for artists, might provide a platform for open access and Free Culture not imaginable elsewhere. I've admired Cornelia's contributions to hacker culture for long. She pioneered as a cyberfeminist in the 1990s with the hilarious and intelligent net-art piece *Female Extension*[2], co-founded Old Boys Network[3] and developed seminal projects such as the *Net Art Generator*. The opportunity to spend two sunny spring days with her intelligence, humour and cyberfeminist wisdom could not have come at a better moment.

## What is Libre Graphics?

Libre Graphics is quite a large ecosystem of software tools; of people, people that develop these tools but also people that use these tools; practices, like how do you work with them, not just how do you make things quickly and in an impressive way, but also how these tools might change your practice; and cultural artifacts that result from it. It is all these elements that come together, I would call Libre Graphics. The term 'libre' is chosen deliberately.

[1] http://postmedialab.org/GWYDH
[2] http://artwarez.org/femext/content/femextEN.html
[3] http://www.obn.org/

It is slightly more mysterious than the term 'free', especially when it turns up in the English language. It sort of hints that there is something different, something done on purpose. And it is also a group of people that are inspired by Free Software culture, by Free Culture, by thinking about how to share both their tools, their recipes and the outcomes of all this. Libre Graphics goes in many directions. But it is an interesting context to work in, that for me has been quite inspiring for a few years now.

## The context of Libre Graphics

The context of Libre Graphics is multiple. I think that I am excited about it and also part of why it is sometimes difficult to describe it in a short sentence. The context is design, and people that are interested in design, in creating visuals, animation, videos, typography… and that is already multiple contexts, because each of these disciplines have their own histories, and their own types of people that get touched by them. Then there is software, people that are interested in the digital material. They say, I am excited about raw bits and the way a vector gets produced. And that is a very, almost formal, interest in how graphics are made. Then there is people that do software. They're interested in programming, in programming languages, in thinking about interfaces, and thinking about ways software can become a tool. And then there are people that are interested in Free Software. How can you make digital tools that can be shared, but also, how can that produce processes that can be shared. Free Software activists to people that are interested in developing specific tools for sharing design and software development processes, like Git or Subversion, those kind of things. I think the multiple contexts are really special and rich in Libre Graphics.

## Free Software culture

Free Software culture, and I use the term 'culture' because I am interested in, let's say, the cultural aspect of it, and this includes software. For me software is a cultural object. But I think it is important to emphasize this, because it easily turns into a technocentric approach, which I think is important to stay away from. Free Software culture is the thinking that, when you develop technology, and I am using technology in the sense that it is cultural as well to me, deeply cultural, you need to take care as well of sharing the recipes, for how this technology has been developed. This produces many different other tools, ways of working, ways of speaking, vocabularies, because it changes radically the way we make and the way we produce hierarchies. It means for example, if you produce a graphic design artifact, that you share all the source files that were necessary to make it; but you also share as much as you can, descriptions or narrations of how it came to be, which does include maybe how much was paid for it, where difficulties were in negotiating with the printer; and what elements were included, because a graphic design object is usually a compilation of different elements; what software was used to make it, and where it might have resisted. The consequences of taking the Free Software culture serious in a design context, means that you care about all these different layers of the work, all the different conditions that actually made the work happen.

## Free Culture

The relationship from Libre Graphics to Free Culture is not always that explicit. For some people it is enough to work with tools that are released under a GPL, an open content licence. And there it stops. Even their work will be released under proprietary licences. For others, it is important to make the full circle and to think about what the legal status is of the work they release. That is the more general one. Then, Free Culture, we can use that very loosely, as in 'everything that is circulating under conditions that it can be reused and remade'. That would be my position. Free Culture is of course also referred to a very specific idea of how that would work, namely Creative Commons. For myself Creative Commons is problematic, although I value the fact that it exists and has really created a broader discussion around licences in creative practices. I value that. For me the distinction Creative Commons makes for almost all the licences they promote, between commercial and non-commercial work, and as a consequence, between professional and amateur work, I find that very problematic. Because I think one of the most important elements of Free Software culture for me, is the possibility for people from different backgrounds, with different skill sets, to actually engage with the digital artifacts they're surrounded with. By making this lazy separation between commercial and non-commercial, which especially in the context of the web as it is right now, is not really easy to hold up, seems really problematic. It creates an illusion of clarity that I think actually makes more trouble than clarity. So I use Free Culture licences, I use licences that are more explicit about the fact that anyone can use whatever I produce in any context. Because I think that is where the real power is of Free Software culture. For me Free Software licences and all the licences that are around it, because I think there is many different types and that is interesting, is that they have a viral power built in. So if you apply a Free Software licence to, for example, a typeface, it means that someone else, even someone else you don't know, has the permission and doesn't have to ask for a permission, to reuse the typeface, to change it, to mix it with something else, to distribute it and to sell it. That is one part, that is already very powerful. But the real secret of such a licence is, that once this person re-releases the typeface, it means that they need to keep that same licence and it propagates across the network and that is where it is really powerful.

## Free tools

It is important to use tools that are released under conditions that allow me to look further than its surface. For many reasons. There is an ethical reason. It is very problematic I think, as a friend explained last week, to feel that you're renting a room in a hotel. That is often the way practitioners nowadays relate to their tools. They have no right to move the furniture. They have no right to invite friends to their hotel room. They have to check out at eleven, etc. it is a very sterile relationship to the tools. That is one part. The other is that there is little way to come into contact with the cultural aspects of the tools. Some things that I suspected before starting to use Free Software tools for my practice, but has been already for almost ten years, continuously exciting, is the whole, let's say, all the other elements around it. The way people organize themselves in conferences, mailing lists, the kinds of communication that happens, the vocabularies, the histories, the connections between different disciplines ... And all that is available to look at, to work with, to come into contact with; to speak to people that do these tools and ask them, why is it like this and not like that. And that to me seems obvious that artists want to have that kind of layered relationship with their tools, and not just only accept whatever comes out of next door shop. I have a very different, almost different physical experience of these tools, because I can enter on many levels. That makes them part of my practice, not just means to an end. I really can take them into my practice. That I find interesting, as an artist and as a designer.

## Artifacts

The outcomes of this type of practice are different, or at least, let's say, in the kind of work I make, try to make and the people I like to work with. There is obviously also groups of people that would like to do Hollywood movies with those tools. That is kind of interesting, that that happens. For me somehow the technological context or conditions that made a work possible, will always occur in the final result. So, that is one part. And the other is that the product is never the end. It means that in whatever way source materials will be released, will be made available, it means that a product is always the beginning of another product, either by me or by other people. I think that is two things that you can always see in the kind of works we make when we do libre-graphics-my-style. When we make a book, for example, what is already different, is when we start the process, it is not yet defined what tool we will use. There is a whole array of tools you can choose from. I mean, books are basically text on paper, and there are many ways to arrive at that output. For one book we did a few years ago, we decided for the first time, because we had never used this tool before, to use TeX, a typesetting system that is developed by Donald Knuth in the context of academic publishing. That has been around as an almost mythological solution for a perfect typesetting. We were curious about whether we could use that system that is developed in a very specific context for an art catalog that we wanted to make. We had to learn how to use this tool, which meant that we somehow had to learn the vocabulary, understand its sort of perspective; things that were possible or not, get used to the kind of humor that is quite terrible in these manuals; accept that certain things that we thought would be easy, were actually not easy at all; and then understand how we could use the things that were popping up or not working or that were different, how we could use them in our advantage. The final result is a book that is slightly strange, because there are some mistakes that have been left in, deliberately or by accident sometimes. The book contains an extensive description of how it was made. Both visually, like it explains the technical details of how it was made, but also the description of that learning process. Another example of how tools, practice and outcomes are somehow connected, but also the whole politics around it, because often these projects are also ways of teasing out; ways licences, practice and tools somehow interact, is a project called 'Sans Guilt'. It is a play with the 'Gill Sans' which is a famous classic typeface that is claimed to be owned by a company called Monotype. But according to our understanding, they have no right to actually claim this typeface as such. But through their communication they do so. OSP was invited to work in an art academy in London, where they had a lead version. And we decided to play with the typeface. The typeface OSP released has many different versions, not versions as in bold, light etc. but it has different levels of 'licencing risk'. One is a straight scan of the prints that were made at that workshop. Another version is more guilty, in the sense that it is an extraction from a .pdf using the Monotype Gill. Another is a redrawn version that takes the matrix, the spacing of a Monotype Gill, but combines it with a redrawn example. All different variations of this font touch on different elements of licencing problems that might occur with typefaces. We sent our experiment to Monotype, because we wanted to hear from them what they thought. After a few months we received a letter from a lawyer saying, would you please identify yourself. We decided to write back as we are, which is, 25 people from 20 different countries with stable and unstable addresses. This long list probably made that we never heard anything again, and 'Sans Guilt' is still available from our website under an open font licence. What the is important, the typeface is different, in the sense that the specimen is not much about showing off how beautiful it will look in any context, but has the description of the process, the motivation of why we did it, the letter we sent to Monotype, the response we got, … The whole packaging of the font becomes then a way of speaking about all these layers that are in our practice.

## Libre fonts

A very exciting part of Libre Graphics is the Libre Font movement, which is strong and has been strong for a long time. Fonts are the basic building blocks of how graphics come to life. When you type something, it is there. And the fact that that part of the work is free, is important on many levels. Things you often don't think about when you speak English and you stay within a limited character set, is that, when you live in let's say India, the language you speak is not available as a digital typeface, meaning that when you want to produce a book in the tools that are available or publish it online, your language has no way of expressing itself. That has to do with commercial interests, laws, ways the technical infrastructure has been built. By understanding that it is important that you can express yourself in the language and with the characters you need, it is also obvious that that part needs to be free. Fonts are also interesting because they exist on many levels. They exist in your system; they're almost software because they're quite complicated objects; they appear in your screen, they are when you print a document; they are there all the time. We consider the alphabet as a totally accessible and available and a universal right to have the alphabet at our disposal. So it is about 'freeing the A', you know. That's quite a beautiful energy. I think that has made the Libre Font movement very strong. Something that has happened the last years and brings up new problems and potential areas to work on, is fonts available for the web. Web fonts have really exploded the amount of free fonts available. Before, fonts were always, let's say, when they were used, tied to a document, and there was some kind of fantasy about that you could hold them, you could somehow contain them, licence them and keep them in check. With the web that idea has gone. And many people have decided to liberate their fonts to be able to make them usable for a website. Because if you think about it, if you use a font on a website, it means that it has to be able to travel everywhere. Everyone has to be able to look at what the font does, but it is not just an output. It is not just an endpoint. The font is active, it means it is available. In theory, any font that appears on the web is both display and program. By displaying the page, you need to run the font. That means the font needs to be available as a source and as a result. That means you have to publish your font. This has really created a big boom in the last few years in Free Fonts, because that is the easiest way to deal with that problem: allow people to download these fonts, but in a way that keeps authorship clear, that keeps genealogy clear, and also propagates then the possibility of making new fonts based on someone else's work.

## Free artifacts / open standards

It took me a while to figure this out. For me it was obvious that if you would use Free Software, you would produce free artifacts. It seems obvious, but it is not at all the case. There is full-fledged commercial production happening with these tools. But one thing that keeps the results, the outcomes of these projects freer than most commercial tools, is that there is really an emphasis on open document formats. That is extremely important, because first of all, it is very obvious that the documents that you produce with the tool, should not belong to the software vendor. They are yours. And to be able to own your own documents, you need to be able to inspect how they're produced. I know many tragic stories of designers that lost documents because they could never open them again. There is really an emphasis and a lot of work on making sure that the documents produced from these tools remain 'inspectable', are documented, that either you can open them in another tool or could develop a tool to have these files available for you. It is really part and parcel of Free Software culture, that you care about that what generates your artifact, but also the materiality of your artifact. Open standards are important. Or maybe let's say it is is important that file formats are documented and can be understood. What is interesting to see is that in this whole Libre Graphics world there is also a strong tradition of reverse engineering, document activism, I would call it. They claim: *documents need to be free, and we will risk breaking the law to be able to understand how non-free documents actually are constructed*. They are really working on trying to understand non-free documents, to be able to read them and to be able to develop tools for them, that they can be reused and remade. The difference between a free and a non-free document is that, for example, an InDesign file, which is the result of a commercial product, there is no documentation available of how this file works. This means that the only way to open the document, is with that particular program. It means there is a connection between that what you've made and the software you used to produce it. It also means that if the software updates or the licence runs out, you will not have access to your own file. It means it is fixed. You can never change it and you can never allow anyone else to change it. An open document format has documentation. That means that not only the software that created it, is available, and in that way you can understand how it was made, but also there is independent documentation available that whenever a project, like a software, doesn't work anymore, or is too old to be run, or you don't have it available, you have other ways of understanding the document and being able to open it and reuse and remake it. What is important, is that around these open formats, you see a whole ecosystem exists of tools to inspect, to create, to read, to change, to manipulate these formats. I think it is very easy to see how around InDesign files this culture does not exist at all.

## Sharing practise / re-learn

This way of working changes the way you learn, and therefore the way you teach. And as many of us have understood the relation between learning and practice, we've all been somehow involved in education. Many of us are teaching in formal design or art education. And it is very clear how those traditional schools are really not fit for the type of learning and teaching that needs to happen around Libre Graphics. One of the problems we run into, is the fact that validation systems are really geared towards judging individuals. And our type of practice is always multiple. It is always about things that happen with many people. And it is really difficult to inspire students to work that way, and at the same time know that at the end of the day, they'll be judged on what they produced as an individual. In traditional education there is always a separation between teaching technology and practice. You have, in different ways, you have the studio practice, and then you have the workshops. And it is very difficult to make conceptual connections between the two. We end up trying to make that happen, but it is clearly not made for that. And then there is the problem of hierarchies between tutor and student, that are hard to break in formal education, just because the setup is, even in very informal situations, that someone comes to teach and someone else comes to be taught. And there is no way to truly break that hierarchy, because that is the way a school works. For years we are thinking about how to do teaching differently or how to do learning differently, and last year, for the first time, we organized a summer school. Just like a kind of experiment to see if we could learn and teach differently. The title, the name of the school is Relearn. Because the sort of relearning for yourself but also to others, through teaching learning, has become really a good methodology, it seems.
If I say 'we', that's always a bit uncomfortable, because I like to be clear about who that is, but when I'm speaking here, there is many 'wes' in my mind.

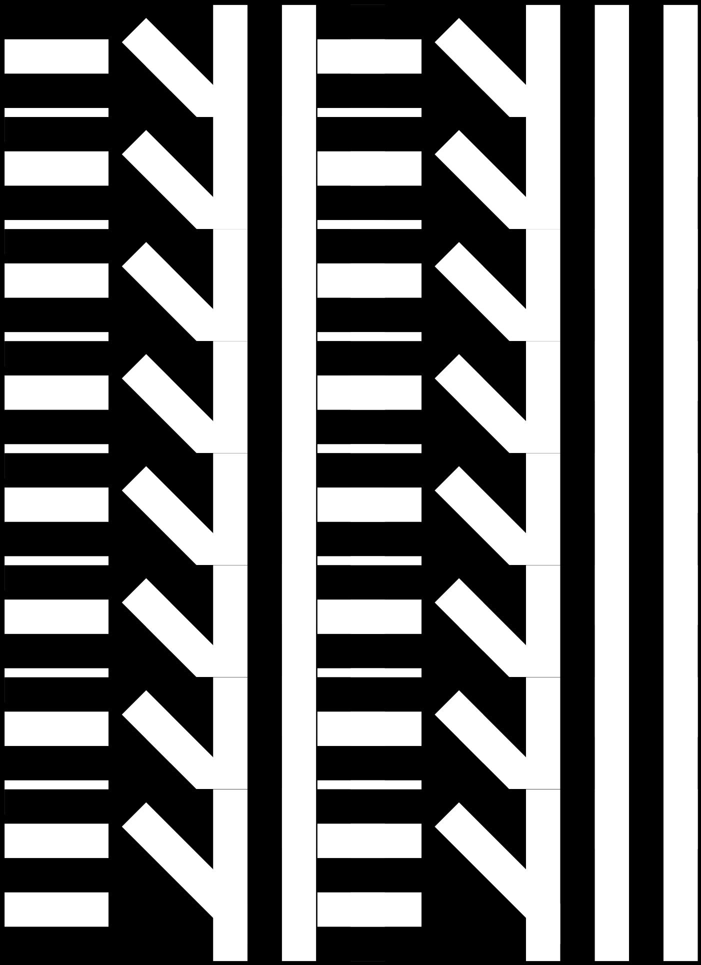
There is a group of designers called OSP. They have started in 2006 with the simple decision to not use any proprietary software anymore for their work. And from that this whole set of questions and practices and methods developed. Right now, that's about twelve people working in Brussels, having a design practice. I am lucky to be honory member of this group. I'm in close contact with them, but I'm not actively working with the design group. Another 'we', an overlapping 'we', is Constant, an association for arts and media active in Brussels since 1996. Or 1997 maybe. Our interest is more in mixing Copyleft thinking, Free Software thinking and feminism. In many ways that intersects with OSP but they might phrase it in a different way. Another 'we' is the Libre Graphics community, which is even a more uncomfortable 'we'. Because it includes engineers that would like to conquer the world … and small hyper intelligent developers that creep out of their corners to talk about the very strange worlds they're creating. Or typographers that care about universal typefaces, or … I mean there is many different people that are involved in that world. I think for this conversation, the 'wes' are: OSP, Constant and the Libre Graphics community, whatever that is.

## Libre Graphics annual meeting Leipzig 2014

We worked on a Code of conduct, which is something that seems to appear in Free Software or tech conferences more and more. It comes a bit from US context. We have started to understand that the fact that Free Software is free, doesn't mean that everyone feels welcome. For long there have been and there still are large problems with diversity in this community. The excitement about freedom has led people to think that people that were not there would probably not want to be there and therefore had no role to be there. For example, the fact that there are not a lot of women active in Free Software, a lot less than in proprietary software, which is quite painful if you think about it. It has to do with this sort of cyclical effect of *because women are not there, they will probably not be interested, and because they're not interested, they might not be capable or feel capable of being active.* So they might not belong. There is also a very brutal culture of harassment, of racist and sexist language, of using imagery that is let's say unacceptable, and that needs to be dealt with. Over the last two years I think, documents like Codes of conduct have started to come up from feminists that are active in this world, like Geek feminism or the Ada initiative, as a way to deal with this. And what it does, is it describes … it is slightly pompous, in the sense that it describes your values. But it is a way to acknowledge the fact that these communities have a problem with harassment, first. That they explicitly say *we want diversity*, which is important. That it gives very clear and practical guidelines for what someone that feels harassed can do, who he or she can speak to, and what will be the consequences. Meaning that it takes away the burden, at least as much as possible, from someone that is harassed to defend actually the gravity of the case.

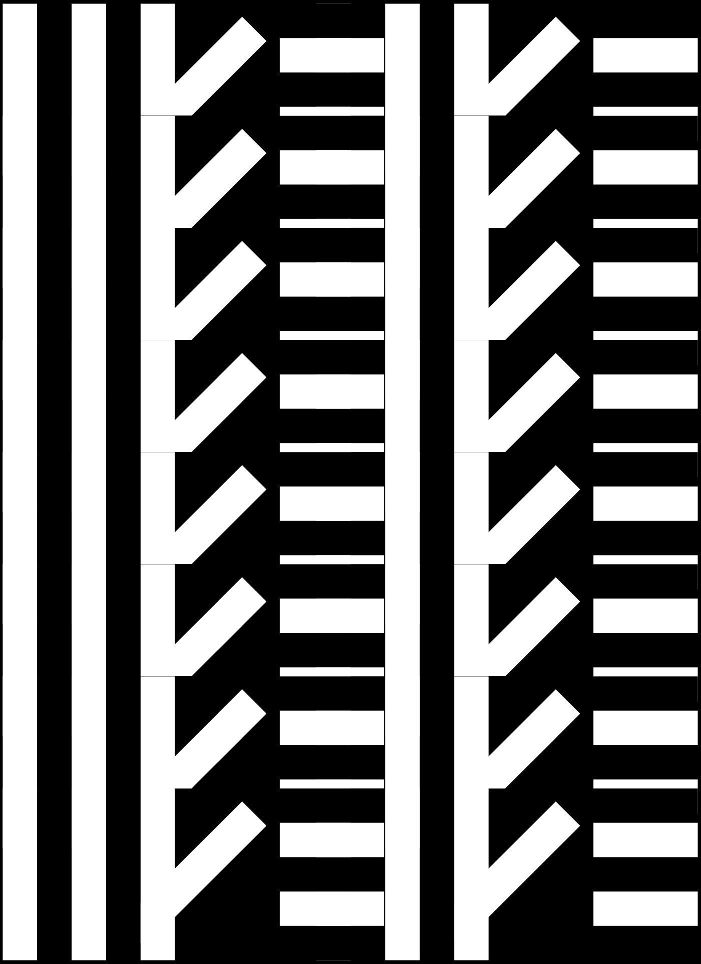
## Art as integrative concept

For me calling myself an artist is useful, is very useful. I'm not busy with let's say, the constitutional art context. That doesn't help me, at all. But what does help me is the figure of the artist, the kinds of intelligences that I sort of project on myself and I use from others and my colleagues, before and contemporary. Because it allows me to not have too many … to be able to define my own context and concepts, without forgetting practice. And I think art is one of the rare places that allows this. Not only allows it, but actually rigorously asks for it. It is really wanting me to be explicit about my historical connections, my way of making, my references, my choices, that are part of the situation I build. And the figure of the artist is a very useful toolbox in itself. And I think I use it, more than I would have thought. It allows me to make these cross connections in a productive way.



# The Making of Conversations

= Christoph Haag
\ Xavier Klein
||| Femke Snelting



# The Making of Conversations

= Christoph Haag
/ Xavier Klein
||| Femke Snelting

The making of *Conversations* was on many levels a process of dialogue, between people, processes, and systems. Xavier Klein and Christoph Haag were as much involved in editorial decisions as they were in creating an experimental platform that would allow us to produce a publication in a way true to the content of the conversations it would contain. In **August 2014** we discussed the ideas behind their designs and the status of the systems they were developing for the book that you are reading right now.

||| ***I wanted to ask you Xavier, how did you end up in Germany?***

\ *It's a long story, so I'll make it short. I benefit from the Leonardo program, a scholarship to do an internship abroad. So I searched for graphic design studios that use Open Source and Free Software. I asked OSP first, but they said* No. *I didn't know LAFKON at this time, and a friend told me:* Hey there is this graphic design studio in Germany, *so I asked and they said* Yes. *So I was happy. (*laughs*)*

||| ***How did you start working on this book?***

= I thought it would be nice to have a project during Xavier's stay in Augsburg with a specific outcome. Something going beyond pure experimentation. So I asked Constant if there were any projects that need to be worked on. And I'm really happy with the *Conversations* publication, because it is a good mixture. There is the technical experiment, how you would approach something like this using Free Software. And there is the editing side. To read all these opinions and reflections. It's really interesting from the content side, at least for me – I don't dare to speak for Xavier. So that's basically how it started.

||| ***You developed a constellation of tools that together are producing the book. Can you explain what the elements are, how this book is made?***

```
-----------------------------_-------------------------------
_|_|_    _   _ _  _ |  o  _  _     _|_  _  _  _   _  _  _ _|_o  _  _  _
 |_| |(7_  |-|-|(_||<| |-|(_|  (_)|-   (_(_)|-|\/(7_|_>(_|-|_|(_)|-|_>
---------------------------_|--------------------------------------
```

= We decided in the beginning to use Etherpad for the editing. A lot of documentation during Constant events was done with Etherpad and I found its very direct access to editing quite inspiring. Earlier this year we prepared a workshop for the Libre Graphics Meeting, where we'd have a transformation from Etherpad pages to a printable `.pdf`. The idea was to somehow separate the content editing and the rendering. Basically I wanted to follow some kind of 'pull logic'. At a certain point in the process, there is an interface where you can pull out something without the need to interfere too much with the inner workings of this part. There is the stable part, the editing on the Etherpad, and there is something, that can be more experimental and unstable which transforms the content to again a stable, printable version. I tried to create a custom markdown dialect, meant to be as simple as possible. It should reduce to some elements, the elements that are actually needed. For example if we have an interview, what is required from the content side? We have text and changing speakers. That's more or less the most important informations.
So on the first level, we have this simple format and from there the transformation process starts. The idea was to have a level, where basically anybody, who knows how to use a text editor, can edit the text. But at the same time it should have more layers of complexity. It actually can get quite complex during the transformation process. But it should always have this level, where it's quite simple. So just text and for example this one markup element for *ok now the speaker changes*.
In the beginning we experimented with differents tools, basically small scripts to perform all kinds of layout task. Xavier for example prepared a `hotglue2svg` converter. After that, we thought, why don't we try to connect different approaches? Not only the very strict markdown *to* TeX *to* `.pdf` transformations, but to think about, under which circumstances you would actually prefer a canvas-based approach. What can you do on a canvas that you can't do or is much harder with a markup language.

||| ***It seems you are developing an adhoc markup language? Is that related to what you wrote in the workshop description for* Operating Systems*:*[1] Using operating systems as a metaphor, we try to imagine systems that are both structured and open*?***

= Yes. The idea was to have these connected/disconected parts. So you have the part where the content is edited in collaboration and you have the transformer script running separately on the individuals' computers. For me this

1 http://libregraphicsmeeting.org/2014/program/

solved in a way the problem of stability. You can use a quite elaborated, reliable software like Etherpad and derive something from it without going to its inner workings. You just pull the content from it, without affecting the software too much. And you have the part, where it can get quite experimental and unreliable, without affecting all collaborators. Because the process runs on your own computer and not on the server.
The markup concept comes from the documentation of a video streaming workshop in Linz. There we wanted to have the possibility to write the documentation collaboratively during the workshop and we needed also to solve problems like *How about the inclusion of images?* That is where the first markup element came from, which basically just was was a specific line of text, which indicates 'here should be this/that image'. If this specific line appears in the text during the transformation process, it triggers an action that will look for a specific file in the repository. If the image exists, it will write the matching macro command for LaTeX. If the image is not in the repository, it will do nothing. The idea was, that the creation of the `.pdf` should happen anyway, e.g. although somebody's repository might be not at the latest state and a missing image would prevent LaTeX from rendering the document. It should also ignore errors, for example if someone mistypes the name of image or the command. It should not stop the process, but produce a different output, e.g. without the image.

||| ***Why do you think the process should not stop when there's an error? Why is that so important?***

= For me it was important to ensure some kind of feedback, even if there might be 'errors' in the output. Not just 'not work'. It can be really frustrating, when the first thing you have to do, is to find and solve a problem – which can be quite hard with this sort of unprofessional scripts – before there's is happening anything at all. So at a certain point, at least something should appear, even if it's not necessarily the way it was originally intended. Like a tolerance for errors, which would even produce something, that maybe different from what you expected. But it should produce 'something'.

||| ***You imagine a kind of iterative development that we know from working with code, that allows you to keep differents versions, that keeps flowing in a way.***

= For example, this specific markup format. It's basically markdown and I wanted some more elements, like footnotes and the option to include citations and comments. I find it quite handy, when you write software,

that you have the possibility to include comments that are not part of the actual output, but part of the working process. I also enjoy this while writing text (e.g. with LaTeX), because I can keep comments or previous versions or drafts. So I really have my working version and transform this to some kind of output.
But back to the etherpash workshop. Commands are basically comments that will trigger some action, for example the inclusion of a graphic or changing the font or anything. These commands are referenced in a separate file, so everybody can have different versions of the commands on their own machine. It would not affect the other people. For example, if you wanted to have a much more elaborated `GRAFIK` command, you could write it and use it within your transformer of the document or you could introduce new commands, that are written on the main pad, but would be ignored for other people, because they have a different reference file. Does this make sense?

**III** ***Yes. In a way, there are a lot of grey zones. There are elements that are global and elements that are local; elements can easily go parallel and none of the commands actually has always the same output, for everyone.***

**=** They can, but they do not need to. You can stick to the very basic version that comes directly from the repository. You could use this version to create a `.pdf` in the 'original' way, but you can easily change it on different levels. You can change the Bash commands that are triggered by the transformer script, you can work on the LaTeX macros or change the script itself. I found it quite important to have different levels of complexity. You may go deeper, but you do not necessarily have to. The Etherpad content is the very top level. You don't have to install a software on your computer, you can just open up a browser and edit the text. So this should make the access to collaboration easier. Because for a lot of experimental software you spend a lot of time to get it even running. Most often you have a very steep learning curve and I found it interesting, to separate this learning curve in a way. So you have different layers and if you really want to reconfigure on a deep level, you can, but you do not necessarily have to.

**III** ***I guess you are talking about collaboration across different levels of complexity, where different elements can transform the final outcome. But if you take the analogy of CSS, or let's say a Content Management System that generates HTML, you could say that this also creates divisions of labour. So rather than making collaboration possible, it confines people to to different***

***files. How do you think your systems invite people to take part in different levels? Are these layers porous at all? Can they easily slip between different roles, let's say an editor, a typographer and a programmer?***

= Up to a certain extent it's like a division of labour. But if you call it a separation of tasks, it makes definitely sense for me. It can be quite hard, if you have to take over responsability for everything at the same time. So it makes sense for me, also for collaboration, to offer this separation. Because it can be good to have the possibility not to have to deal with the whole system and everything at the same time. You should be able to do so, but you should not necessarily have to. I think this is important, because a lot of frustration regarding Free Software systems comes from the necessity to go to the deep level at an early stage. I mean it's an interesting problem. The promise of convenience is quite hard, because most times is does not really work. And it's also fine that it doesn't really work. At the same time it's frightening for people to get into it and so I think, it's good to do this step by step and also to have an easy top level opportunity to go into, for example, programming. This is also a thing I became really interrested in. The principle of the commandline to 'extend usage into programming'.[2] You do not have to have a development environment and then you compile software and then you have software, but you have this flexible interface for your daily tasks. If you really need to go a deeper level, you can, at least with Free Software. But you don't have to … compile your kernel every time.

III ***Not every time! What I find interesting about your work is that you prefer not to conceal any layers. References, commands, markup hints at the existence of other layers, and the potential to go somewhere else. I wanted you to ask about your fascination or interest in something 'old school' as Bash scripting. Why is it so interesting?***

= Maybe at first point, it's a bit of a fascination for the obscure. That normally, as a graphic designer you wouldn't think of using the commandline for your work. When I started to use GNU/Linux, I'd try to stay away from the terminal. Which is basically, as I realised pretty soon, not possible.[3] At some point, Bash scripting became really fascinating, because of the possibility to use automation to correct or add functionalities. With the commandline it's easy to automate repetitive tasks, e.g. you can write a small script that

[2] Florian Cramer. (echo echo) echo (echo): Command Line Poetics, 2007

[3] let's say hard

creates a separate `.svg` file for each layer in a `.svg` file[4], convert this separated `.svg` files to `.pdf` files[5] and combine the `.pdf` files to a multipage `.pdf`[6]. Just by collecting commands you'd normally type on your commandline interface. So in this case, automation helps to work around a missing multipage support in inkscape. Not by changing the application itself, but by plugging something 'on top' of it. I like to think of the Bash as glue between different applications. So if we have a look now at the setup for the conversations publication, we may see that Bash makes it really easy to develop own configurations and setups. I actually thought about prefering the word 'setup' to 'writing software' …

||| ***Are you saying you prefer setup 'over' configuration?***

= Setup or configuration of software 'over' actually writing software. Because for me it's often more about connecting different applications. For example, here we have a browser-based text editor, from which the content is automatically pulled and transformed via text-transform tools and then rendered as a `.pdf`. What I find interesting, is that the scripts in between may actually be not very stable, but connect two stables parts. One is the Etherpad, where the export function is taken 'as is' and you've got the final state of a `.pdf`. In between, I try to have this flexible thing, that just needs to work at this moment, in my special case. I mean certain scripts may reach quite an amount of stability, but not necessarily. So it's very good to have this fixed state at the end.

||| ***You mean the*** `.pdf`***?***

= I mean the `.pdf`, because … These scripts are quite personal software and so I don't really think about other users beside me. For me it's a whole different subject to go to the usability level. That's maybe also a cause for the open state of the scripts. It would not make much sense – if I want to have the opportunity for other people to make use of these things – to have black boxes. Because for this, they are much too fragile. They can be taken over, but there is no promise of … convenience?[7] And it's also important for myself, because the setups are really tailored to a specific use case and

---

[4] using `sed`, stream editor for filtering and transforming text

[5] using `inkscape` on the commandline

[6] using `pdftk`

[7] *… distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.* Free Software Foundation. GNU General Public License, 2007

therefore more or less temporary. So I need to be able to read and adapt them myself.

III ***I know that you usually afterwards you provide a description of how the collage was made. You publish the scripts, and sketches and intermediary outcomes. So it seems that usability is more in how you give access to the process rather than the outcome. Or would you say that software is the outcome?***

= Actually for me the process is the more interesting part of the work. A lot of the projects are maybe more like a proof of concept, than finished pieces of software. I often reuse parts of these setups or software pieces, so it's more collections of 'How to do something' then really a finished thing, that's now suitable to produce this or that.

III ***I'm just wondering, looking at your designs, if you would like that layering, this unstability to be somehow legible in the*** `.pdf` ***or the printed object?***

= I don't think that this unstability is really legible. Because in the process there's a certain point where definitive decisions are taken. It's also part of the concept. You make decisions and that make the final state of the object what it is. And if you want to get back to the more flexible part, then you would really have to get back. So I don't actually think that it is legible in the final output, on the first sight, that it is based on a very fluid working process. And for me that's quite ok. It's also important for me – because I tend not to do so – to take a decision at a certain point. But that's not necessarily the ultimate decision and therefore it's also important to keep the option open to redefine ... 'the thing'.

III ***What you're saying, is that you can be decisive in your design decisions because the outcome could also be another. You could always regenerate the*** `.pdf` ***with other decisions.***

= Yes. For example, I would regenerate the `.pdf` with the same decisions, another person maybe would take different decisions. But that's one step before the final object. For example, if we do not talk about the `.pdf`, but we actually talk about the book, then it's very clear, that there are decisions, that need to be taken or that have been taken. And actually I like the feeling of convenience when things get finished. They are done. Not configurable forever.

III ***(*****laughs*****) That's convenient, if things get done!***

## the making of conversations

***For this specific book, you have made a few decisions, for example your selection of fonts is particular.***
***Xavier, can you say something about the typography of Conversations?***

\ *Huuumn yep, for the typographic decisions … in the beginning we searched for fancy fonts, but in a way came back to use very classic fonts, respectively one classic font. So the Junicode*[8] *for the text and the OCR-A*[9] *for anything else. Because we decided to focus on testing different ways of layouting and use the fonts as a way to keep a certain continuity between the parts. We thought this can be more interesting, than to show that we can find a lot of beautiful, fancy fonts.*

= So in the beginning, we thought about having a different font for every speaker, but sooner or later we realised that it would be good to have something that keeps the whole thing together. Right now, this are the two fonts. The Junicode, which is a font for medievalists, and the OCR-A, which is a optical character recognition font from the early age of computer technology. So the hypothesis was, to have this combination – a very classical typeface inspired by the 16th century and a typeface optimized for machine reading – that maybe will produce an interesting clash of two different approaches. While at the same time providing a continuous element throughout the book. But that still has to be proven in the final layout.

III ***I find it interesting that both fonts in their own way are somehow conversational. They are both used in situations where one system needs to talk to another.***

= Yeah, definitely in a way. They are both optimised for a special usage, which, by the way, isn't the usage of our case. One for the display of medieval texts, where you have to have lot of different signs and ligatures and … that's the Junicode. The other one, the OCR-A, is optimized to be legible by machines. So that are two different directions of conversation. And they're both Free and Open Source fonts …

III ***And for the layout? How are the divider pages going to be constructed?***

= For the divider pages, it's an application 'Built with Processing', done by Benjamin[10]. In a way, it's a different approach, because it's a software with an extensive Graphical User Interface, with a lot of options. So it's different

[8] http://junicode.sourceforge.net/
[9] http://sourceforge.net/projects/ocr-a-font/
[10] Stephan

from the very modular, connective approach. There we decided to have this software, which is directly controlled by the controller, the person who uses it. And again, there is this moment of definitive decision. *Ok, this is exactly how I want the title pages to look.* And then they are put in a fixed state. At the same time, the software will be part of the repository, to be usable as a tool. So it's a very … not a 'very classic' … approach. To write '**your**' software for '**your**' very specific use case. In a more monolithic way …
Just to add this. In this custom markdown dialect, I decided at a point to include a command, which is `INCLUDEPAGES`, where you can provide a `.pdf` file via an url to be included in the document. So the `.pdf` may be stored anywhere, as long as it is accessible over the internet. I found this an interesting opportunity for collaboration. Because if somebody does not want to stick to the grid given by the LaTeX configuration or to this kind of working in general, this person could create a `.pdf`, store it online, reference it and the file will be included. This can be a very disconnected way of contributing to the final book. And that's also a thing we're now trying to test ourselves. Because in the beginning we developed a lot of different little scripts, for example the `hotglue2svg` converter. And right now we're trying to extend this. For example, to create one interview in Scribus and include the `.pdf` made with Scribus. To also test ourselves different approaches.

III ***This book will be both a collage and have a overall, predefined structure provided by the lay-out engine?***

= I'm trying to make pragmatic use of the functionalities of LaTeX, which is used for the final compiling of the `.pdf`. So for example, also ready-made `.pdf` files included into the final document are referenced in the table of contents.

III ***Can you explain that again ?***

= Separate `.pdf`s, that are included into the final document will be referenced in the table of contents. We can still make use of the automatic generation of page numbers in the table of contents, so there it goes together. There are certain borders, for example since the `.pdf`s are more like finished documents, indexing will probably not work. Because even if you can extract references from the `.pdf`, I didn't find a way until now, how to find out the page number in a reliable way. There you also realise, that you can do much more with the plain text sources than you can do with a finished document.

```
--------------------------------------_----------------------------------
_|_|_        _  _       |  o  _        _|_    _  _   _    _   _  _|_o  _  _
 |_| | (7_  | || | (_| |<| | | (_|   (_) |   (_(_) | | \/ (7_|  _> (_| |_| (_) | | _>
-------------------------------_| ------------------------------------------
                              _|
```

But I think that’s ok. In this case you wouldn’t to have a keyword reference to the `.pdf`, while it’s still in the table of contents …

**|||** ***What if someone would want to use one of these interviews for something else? How could this book becoming source for an another publication?***

**=** That’s also an advantage of the quite simple source format on the Etherpad. It can be easily converted to e.g. simple markdown, just by a little script. I found this quite important – because at this point we’re putting quite an amount of work into the preparation of the texts – to have it not in a format that is not parseable. I really wanted to keep the documents transformable in a easy way. So now you could just have a ~fiveliner, that will pull the text from the Etherpad and convert it to simple markdown or to HTML.

**|||** ***Wonderful.***

**=** If you have a more or less clean source format, then it’s in most cases easy to convert it to different formats. For example, the Evan Roth interview, you provided as a ConTeXt file. So with some text manipulation, it was easy to do the transformation to our Etherpad markup. And it would be harder if the content is stored as an Open Office document, but still feasible. `.pdf` in a way is the worst case, because it’s much harder to extract usable content again, depending on the creator. So I think it’s important to keep the content in a readable and understandable source format.

**|||** ***Xavier, what is going to happen next?***

**** *Right now, I’m the guy who tests on Scribus, Inkscape. But I don’t know if it’s the answer to your question.*

**|||** ***I was just curious because you have a month to work on this still, so I was wondering … are there other things you are testing or trying ?***

**** *Yeah, I think I want to finish the `hotglue2svg.sh`, I mean it’s my first Bash program, I want to raise my baby. (*laughs*) But right now I’m trying to find different ways of layouts. The first one is the one with the big squares, the big unicode characters and all the arrows. So it’s very complicated, but it’s the attempt to find an another way to express a conversation in text.*

**|||** ***Can you say more about that ?***

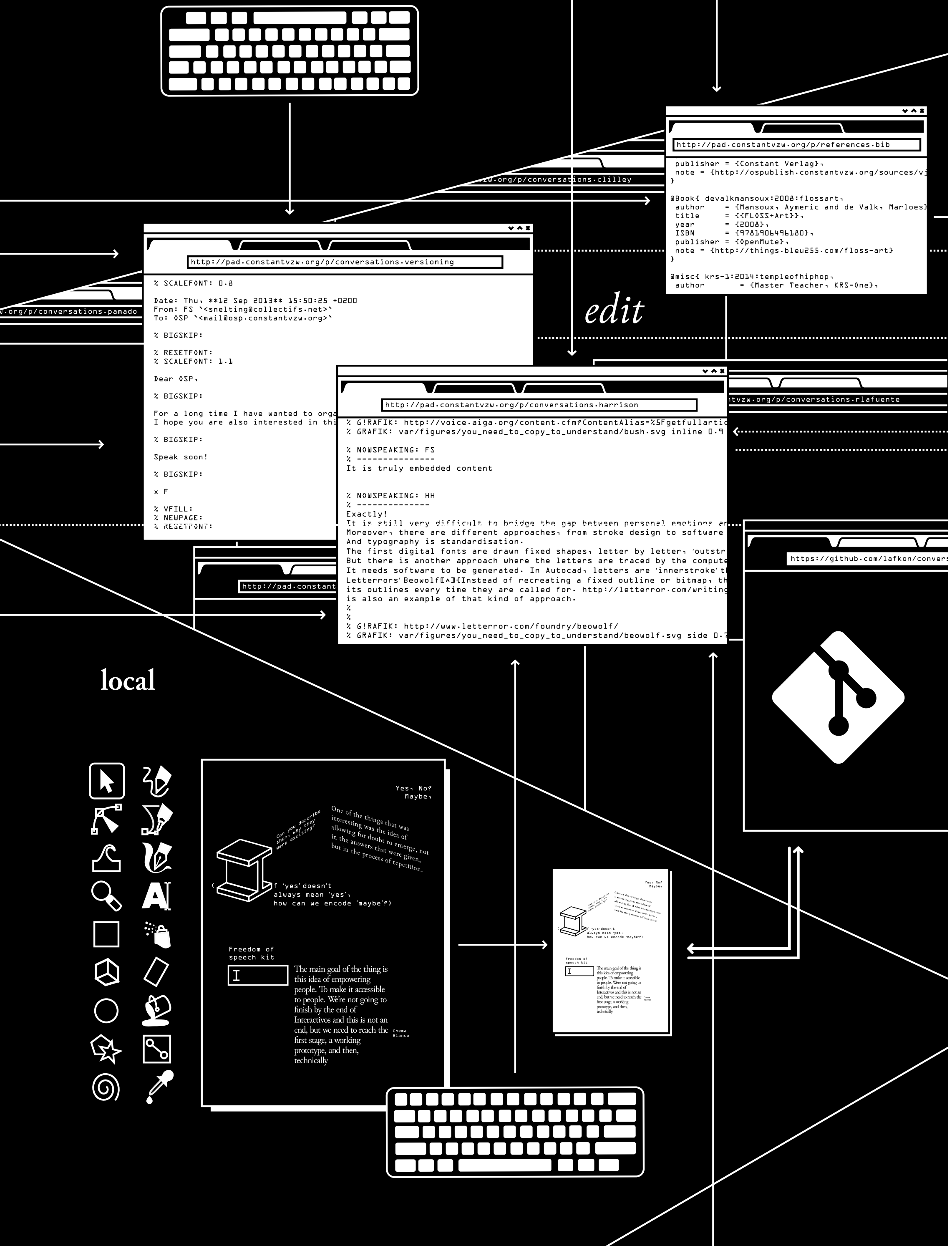
**** *Because in the beginning, my first try was to keep the ‘life’ of a conversation in the text with some things, like indentation or with graphic things, like the choice* *of the unicode characters. If this can be a way to express a conversation. Because it's hard to it with programming stuff so we're using GUI based software.*

= It's a bit coming to the question, what you are doing differently, if you work with a direct visual feedback. So you don't try to reduce the content to get it through a logical structure. Because that's in a way how the markdown to LaTeX transformation is doing it. You set certain rules, that may be in special cases soft rules, but you really try to establish a logical structure and have a set of rules and apply them. For me, it's also an interesting question. If you think of grid based graphic design, where you try to introduce a set of rules in the beginning and then to keep for the rest of the project, that's in a way a very obvious case for computation. Where you just apply a set of rules. With this application of rules you are a lot confronted in daily graphic design. And this is also a way of working you learn during your studies. Stick to certain logical or maybe visual grids. And so now the question is: What's the difference if you do a really visual layout. Do you deal differently with the content, does it make sense, or if you're just always coming back to a certain grid, then you might as well do it by computation. So that's something that we wanted to find out. What advantage do you really gain from having a canvas-based approach throughout the layout process.

III ***In a way the interviews are very similar, because it's always peoples speaking, but at the same time each of the conversations is slightly different. So in what way is the difference between them made legible, through the same set of rules or by making specifics rules for each of them?***

= If you do the layout by hand you can take decisions that would be much harder to translate to code. For example, how to emphasize certain part of the text or the speaker. You're much closer to the interpretation of the content? You're not designing the ruleset but you are really working on the visual design of the content … The point why it's interesting to me is because working as a designer you get quite often reduced to this visual design of the content, at the same it may make sense in a lot of cases. So it's a evaluation of these different approaches. Do you design the ruleset or do you design the final outcome? And I think it has both advantages and disadvantages.

*conversations* setup approximation (***not** the missing manual*)

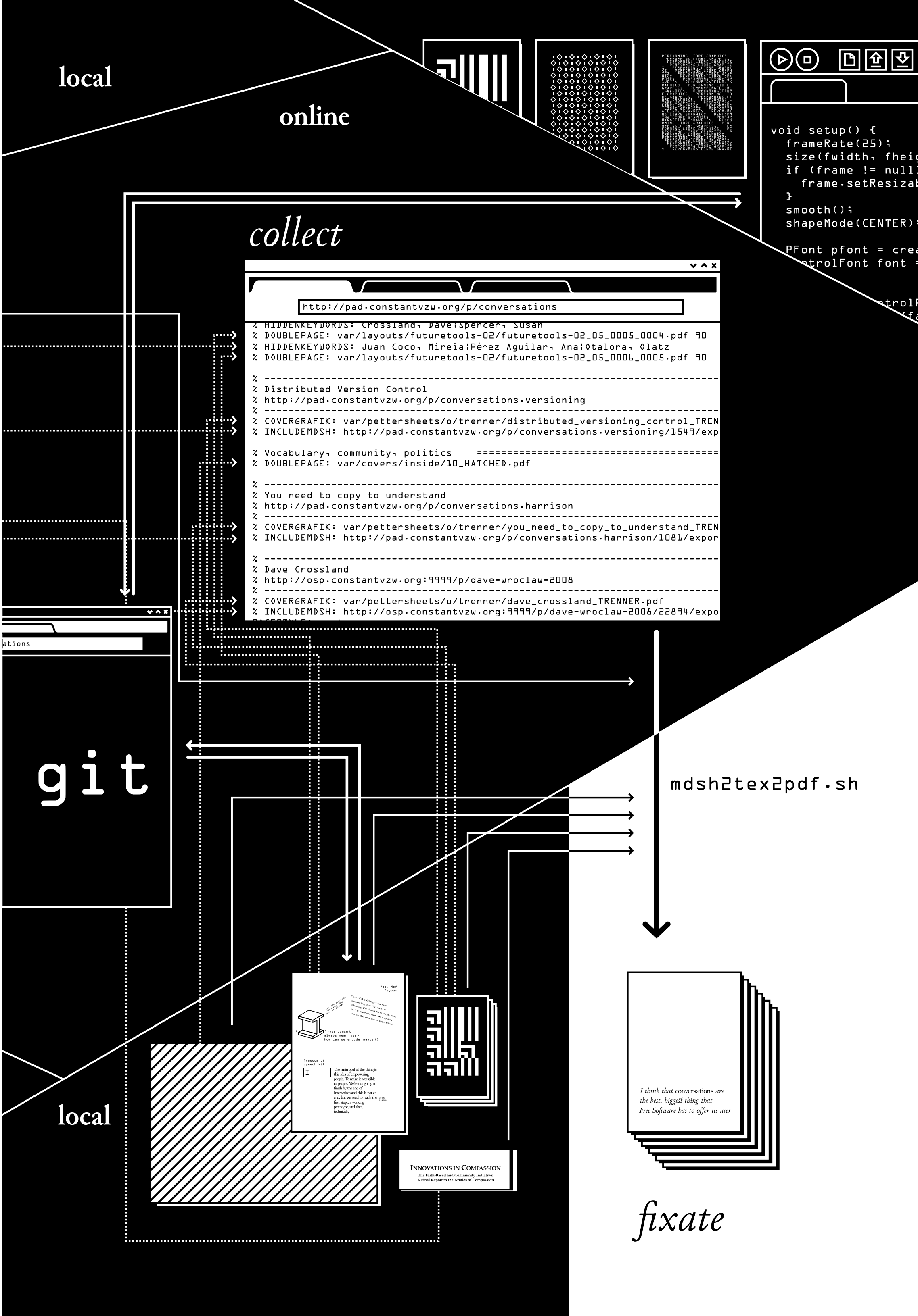


edit
local
http://pad.constantvzw.org/p/conversations.clilley
http://pad.constantvzw.org/p/conversations.pamado
http://pad.constantvzw.org/p/conversations.rlafuente
http://pad.constant
https://github.com/lafkon/convers
Yes, No? Maybe,
Can you describe them: why they were exciting?
One of the things that was interesting was the idea of allowing for doubt to emerge, not in the answers that were given, but in the process of repetition.
(If 'yes' doesn't always mean 'yes', how can we encode 'maybe'?)
Freedom of speech kit
The main goal of the thing is this idea of empowering people. To make it accessible to people. We're not going to finish by the end of Interactivos and this is not an end, but we need to reach the first stage, a working prototype, and then, technically
Chema Blanco

```
http://pad.constantvzw.org/p/references.bib

 publisher = {Constant Verlag},
 note = {http://ospublish.constantvzw.org/sources/vj
}

@Book{ devalkmansoux:2008:flossart,
 author    = {Mansoux, Aymeric and de Valk, Marloes}
 title     = {{FLOSS+Art}},
 year      = {2008},
 ISBN      = {9781906496180},
 publisher = {OpenMute},
 note = {http://things.bleu255.com/floss-art}
}

@misc{ krs-1:2014:templeofhiphop,
 author       = {Master Teacher, KRS-One},
```

```
http://pad.constantvzw.org/p/conversations.versioning

% SCALEFONT: 0.8

Date: Thu, **12 Sep 2013** 15:50:25 +0200
From: FS `<snelting@collectifs.net>`
To: OSP `<mail@osp.constantvzw.org>`

% BIGSKIP:

% RESETFONT:
% SCALEFONT: 1.1

Dear OSP,

% BIGSKIP:

For a long time I have wanted to orga
I hope you are also interested in thi

% BIGSKIP:

Speak soon!

% BIGSKIP:

x F

% VFILL:
% NEWPAGE:
% RESETFONT:
```

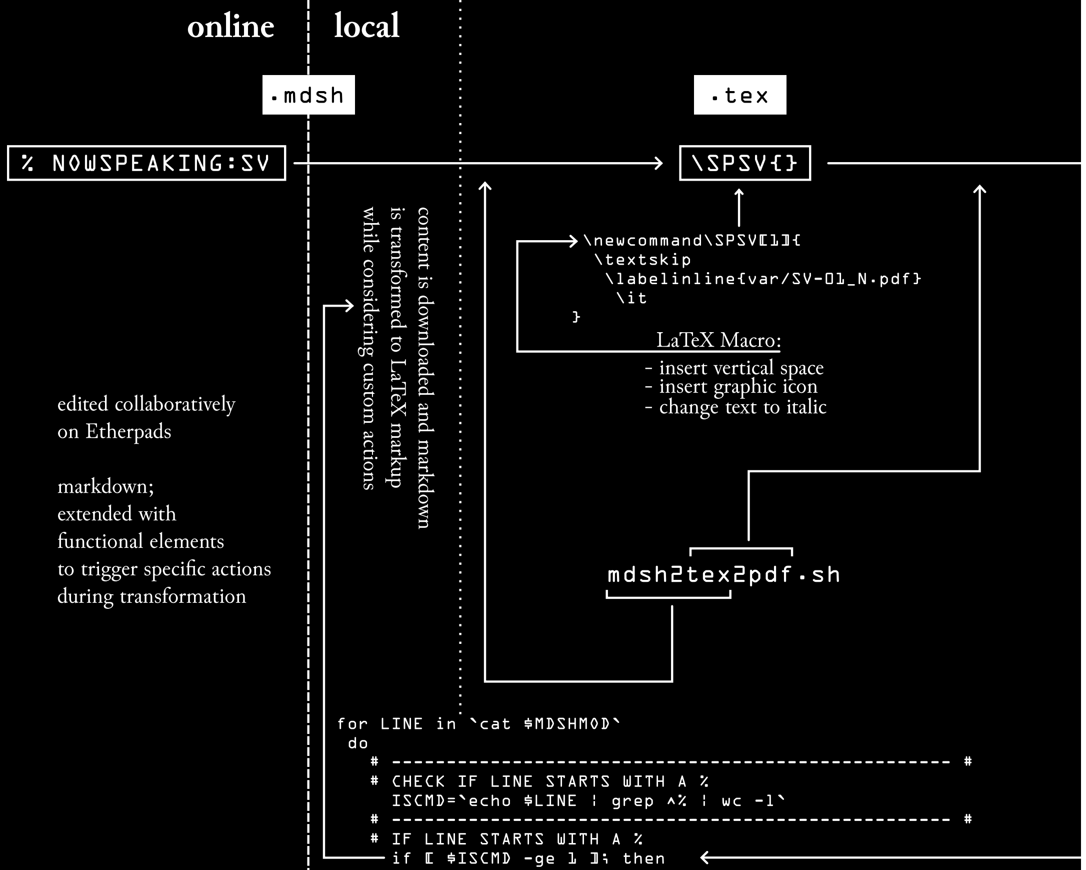
```
http://pad.constantvzw.org/p/conversations.harrison

% G!RAFIK: http://voice.aiga.org/content.cfm?ContentAlias=%5Fgetfullartic
% GRAFIK: var/figures/you_need_to_copy_to_understand/bush.svg inline 0.9

% NOWSPEAKING: FS
% ---------------
It is truly embedded content


% NOWSPEAKING: HH
% --------------
Exactly!
It is still very difficult to bridge the gap between personal emotions an
Moreover, there are different approaches, from stroke design to software
And typography is standardisation.
The first digital fonts are drawn fixed shapes, letter by letter, 'outstr
But there is another approach where the letters are traced by the compute
It needs software to be generated. In Autocad, letters are 'innerstroke' th
Letterrors'Beowolf[^]{Instead of recreating a fixed outline or bitmap, th
its outlines every time they are called for. http://letterror.com/writing
is also an example of that kind of approach.
%
%
% G!RAFIK: http://www.letterror.com/foundry/beowolf/
% GRAFIK: var/figures/you_need_to_copy_to_understand/beowolf.svg side 0.7
```


local
online
collect
git
mdsh2tex2pdf.sh
local
fixate
ations
Yes, No?
Maybe.
Can you describe them, why they were exciting?
One of the things that was interesting was the idea of allowing for doubt to emerge, not in the answers that were given, but in the process of repetition.
(if 'yes' doesn't always mean 'yes', how can we encode 'maybe'?)
Freedom of speech kit
Chema Blanco
INNOVATIONS IN COMPASSION
The Faith-Based and Community Initiative:
A Final Report to the Armies of Compassion
I think that conversations are the best, biggest thing that Free Software has to offer its user

```
void setup() {
  frameRate(25);
  size(fwidth, fheig
  if (frame != null
    frame.setResizab
  }
  smooth();
  shapeMode(CENTER);

  PFont pfont = crea
  ntrolFont font =

                ntrolF
                 (fa
```

The main goal of the thing is this idea of empowering people. To make it accessible to people. We're not going to finish by the end of Interactivos and this is not an end, but we need to reach the first stage, a working prototype, and then, technically

```
http://pad.constantvzw.org/p/conversations

% HIDDENKEYWORDS: Crossland, Dave|Spencer, Susan
% DOUBLEPAGE: var/layouts/futuretools-02/futuretools-02_05_0005_0004.pdf 90
% HIDDENKEYWORDS: Juan Coco, Mireia|Pérez Aguilar, Ana|Otalora, Olatz
% DOUBLEPAGE: var/layouts/futuretools-02/futuretools-02_05_0006_0005.pdf 90

% -----------------------------------------------------------------------
% Distributed Version Control
% http://pad.constantvzw.org/p/conversations.versioning
% -----------------------------------------------------------------------
% COVERGRAFIK: var/pettersheets/o/trenner/distributed_versioning_control_TREN
% INCLUDEMDSH: http://pad.constantvzw.org/p/conversations.versioning/1549/exp

% Vocabulary, community, politics    ====================================
% DOUBLEPAGE: var/covers/inside/10_HATCHED.pdf

% -----------------------------------------------------------------------
% You need to copy to understand
% http://pad.constantvzw.org/p/conversations.harrison
% -----------------------------------------------------------------------
% COVERGRAFIK: var/pettersheets/o/trenner/you_need_to_copy_to_understand_TREN
% INCLUDEMDSH: http://pad.constantvzw.org/p/conversations.harrison/1081/expor

% -----------------------------------------------------------------------
% Dave Crossland
% http://osp.constantvzw.org:9999/p/dave-wroclaw-2008
% -----------------------------------------------------------------------
% COVERGRAFIK: var/pettersheets/o/trenner/dave_crossland_TRENNER.pdf
% INCLUDEMDSH: http://osp.constantvzw.org:9999/p/dave-wroclaw-2008/22894/expo
```



edited collaboratively
on Etherpads

markdown;
extended with
functional elements
to trigger specific actions
during transformation

content is downloaded and markdown
is transformed to LaTeX markup
while considering custom actions

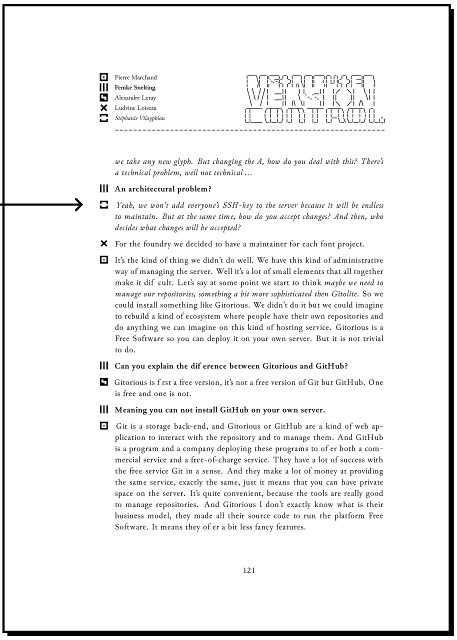
```
\newcommand\SPSV[1]{
 \textskip
  \labelinline{var/SV-01_N.pdf}
   \it
}
```

LaTeX Macro:

- insert vertical space
- insert graphic icon
- change text to italic

```
for LINE in `cat $MDSHMOD`
 do
   # ------------------------------------------------------- #
   # CHECK IF LINE STARTS WITH A %
     ISCMD=`echo $LINE | grep ^% | wc -l`
   # ------------------------------------------------------- #
   # IF LINE STARTS WITH A %
     if [ $ISCMD -ge 1 ]; then
        CMD=`echo $LINE | \
             cut -d "%" -f 2 | \
             cut -d ":" -f 1 | \
             sed 's/\[/ /g' | sed 's/\]/ /g' |\
             sed 's; ;;g'`
        ARG=`echo $LINE | cut -d ":" -f 2-`
   # ------------------------------------------------------- #
   # CHECK IF COMMAND EXISTS
        CMDEXISTS=`grep "^function ${CMD}()" $FUNCTIONS |\
                   wc -l`
   # ------------------------------------------------------- #
   # IF COMMAND EXISTS
     if [ $CMDEXISTS -ge 1 ]; then
        # EXECUTE COMMAND
          $CMD $ARG
     fi
   # ------------------------------------------------------- #
   # IF LINE DOES NOT START WITH % (= SIMPLE MARKDOWN)
     else
   # ------------------------------------------------------- #
   # APPEND LINE TO TEX FILE
     echo "$LINE" >> $TEXBODY
     fi
   # ------------------------------------------------------- #
done
```

.pdf



⊡ Pierre Marchand
**Ⅲ Femke Snelting**
Alexandre Leray
✕ Ludvine Loiseau
*Stéphanie Vilayphiou*

DISTRIBUTED VERSION CONTROL

*we take any new glyph. But changing the A, how do you deal with this? There's a technical problem, well not technical…*

**An architectural problem?**

*Yeah, we won't add everyone's SSH-key to the server because it will be endless to maintain. But at the same time, how do you accept changes? And then, who decides what changes will be accepted?*

For the foundry we decided to have a maintainer for each font project.

It's the kind of thing we didn't do well. We have this kind of administrative way of managing the server. Well it's a lot of small elements that all together make it dif cult. Let's say at some point we start to think *maybe we need to manage our repositories, something a bit more sophisticated then Gitolite.* So we could install something like Gitorious. We didn't do it but we could imagine to rebuild a kind of ecosystem where people have their own repositories and do anything we can imagine on this kind of hosting service. Gitorious is a Free Software so you can deploy it on your own server. But it is not trivial to do.

**Can you explain the dif erence between Gitorious and GitHub?**

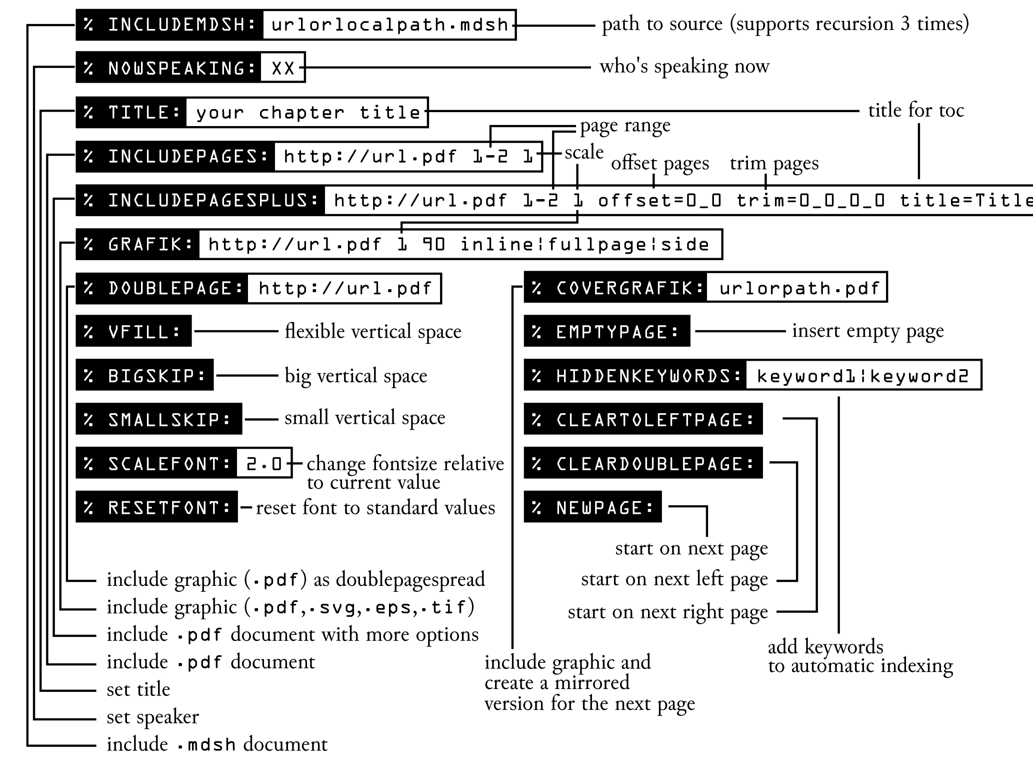
Gitorious is f rst a free version, it's not a free version of Git but GitHub. One is free and one is not.

**Meaning you can not install GitHub on your own server.**

Git is a storage back-end, and Gitorious or GitHub are a kind of web application to interact with the repository and to manage them. And GitHub is a program and a company deploying these programs to of er both a commercial service and a free-of-charge service. They have a lot of success with the free service Git in a sense. And they make a lot of money at providing the same service, exactly the same, just it means that you can have private space on the server. It's quite convenient, because the tools are really good to manage repositories. And Gitorious I don't exactly know what is their business model, they made all their source code to run the platform Free Software. It means they of er a bit less fancy features.

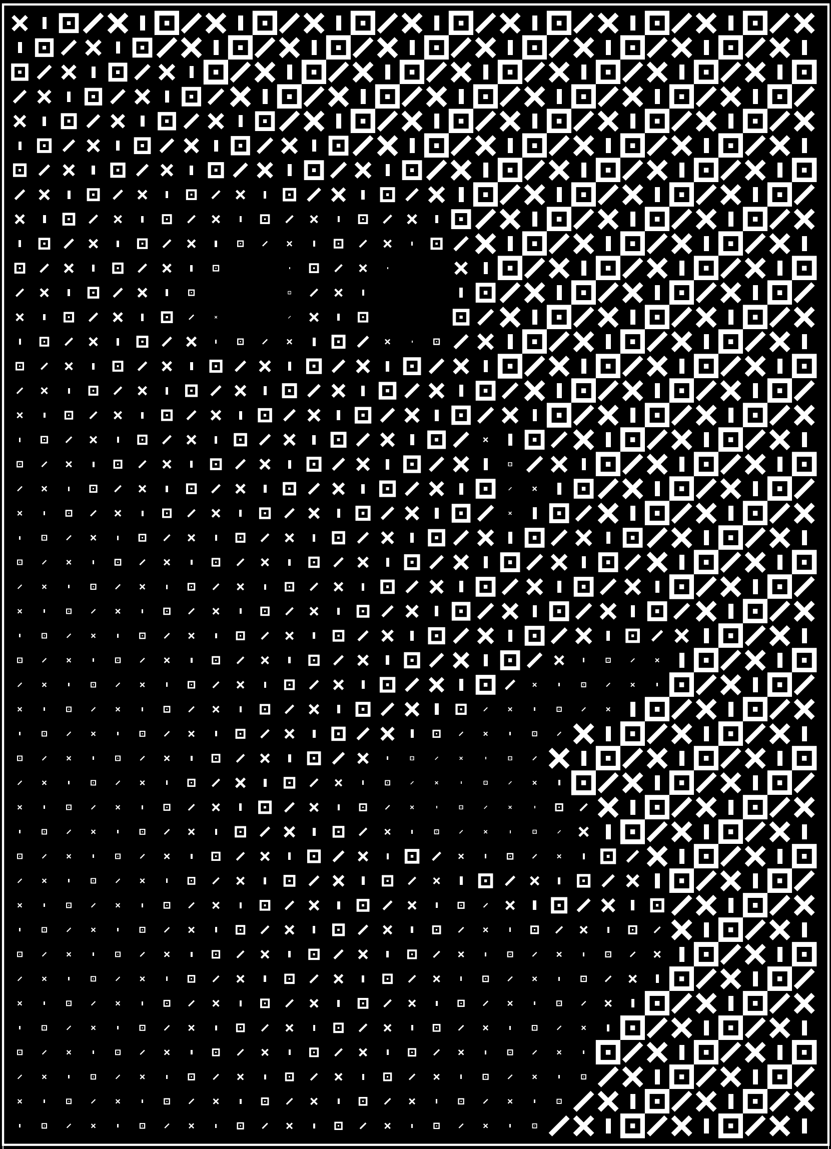
121

MDSH COMMANDS (MAY BE EXTENDED DURING EDITING)



```
% INCLUDEMDSH: urlorlocalpath.mdsh
% NOWSPEAKING: XX
% TITLE: your chapter title
% INCLUDEPAGES: http://url.pdf 1-2 1
% INCLUDEPAGESPLUS: http://url.pdf 1-2 1 offset=0_0 trim=0_0_0_0 title=Title
% GRAFIK: http://url.pdf 1 90 inline|fullpage|side
% DOUBLEPAGE: http://url.pdf
% VFILL:
% BIGSKIP:
% SMALLSKIP:
% SCALEFONT: 2.0
% RESETFONT:
% COVERGRAFIK: urlorpath.pdf
% EMPTYPAGE:
% HIDDENKEYWORDS: keyword1|keyword2
% CLEARTOLEFTPAGE:
% CLEARDOUBLEPAGE:
% NEWPAGE:
```

path to source (supports recursion 3 times)
who's speaking now
title for toc
page range
scale
offset pages
trim pages
flexible vertical space
big vertical space
small vertical space
change fontsize relative to current value
reset font to standard values
insert empty page
start on next page
start on next left page
start on next right page
add keywords to automatic indexing
include graphic and create a mirrored version for the next page
include graphic (.pdf) as doublepagespread
include graphic (.pdf,.svg,.eps,.tif)
include .pdf document with more options
include .pdf document
set title
set speaker
include .mdsh document



☐ CS ■ A4 ☐ A3 ☐ A2 ■ P/L

■ REF ☐ NFO ■ GUIE BG

79 OFFSET

33 XTILENUM (LEFT/RIGHT)

46 YTILENUM (UP/DOWN)

0.0 GLOBAL TRANS X

0.0 GLOBAL TRANS Y

0.0 RELATIVE TRANS X

0.0 RELATIVE TRANS Y

0.0 GLOBAL ROT

0.0 RELATIVE ROT

LINEAR.EASEIN

0.3 GLOBAL SCALE

3.1 RELATIVE SCALE

S R T I X

STYLE

OFFSET: -2 X 1

**In conversation with**: Agnes Bewer, Alexandre Leray, An Mertens, Andreas Vox, Asheesh Laroia, Carla Boserman,Christina Clar, Chris Lilley, Christoph Haag, Claire Williams, Cornelia Sollfrank, Dave Crossland, Dmytry Kleiner, Denis Jacquery, Dmytri Kleiner, Eleanor Greenhalgh, Eric Schrijver, Evan Roth, Femke Snelting, Franziska Kleiner, George Williams, Gijs de Heij, Harrisson, Ivan Monroy Lopez, John Haltiwanger, John Colenbrander, Juliane De Moerlooze, Julien Deswaef, Larisa Blazic, Ludivine Loiseau, Manuel Schmalstieg, Matthew Fuller, Michael Murtaugh, Michael Terry, Michele Walther, Miguel Arana Catania, momo3010, Nicolas Malevé, Pedro Amado, Peter Westenberg, Pierre Huyghebaert, Pierre Marchand, Sarah Magnan, Stéphanie Vilayphiou, Tom Lechner, Urantsetseg Ulziikhuu, Xavier Klein

**Concept, development and design:** Christoph Haag, Xavier Klein, Femke Snelting

**Editorial team:** Thomas Buxó, Loraine Furter, Maryl Genc, Pierre Huyghebaert, Martino Morandi
**Transcriptions:** An Mertens, Boris Kish, Christoph Haag, Femke Snelting, George Williams, Gijs de Heij, ginger coons, Ivan Monroy Lopez, John Haltiwanger, Ludivine Loiseau, Martino Morandi, Pierre Huyghebaert, Urantsetseg Ulziikhuu, Xavier Klein

**Chapter opener:** Built with `petter` by Benjamin Stephan
-> `http://github.com/b3nson/petter`

**Tools:** `basename`, `bash`, `bibtex`, `cat`, Chromium, `cp`, `curl`, `dpkg`, `egrep`, Etherpad, `exit`, `ftp`, gedit, GIMP, ghostscript, Git, `GNU coreutils`, `grep`, ImageMagick, Inkscape, Kate, `man`, `makeindex`, meld, `ne`, `pandoc`, `pdflatex`, `pdftk`, Processing, `python`, `read`, `rev`, Scribus, `sed`, `vim`, `wget`

**Fonts:** *Junicode* by Peter S. Baker, `OCR-A` by John Sauter

**Source Files:**
*Texts, fonts and pdf:* `http://conversations.tools`
*Software:* `https://github.com/lafkon/conversations`
**Published by**: Constant Verlag (Brussels, January 2015)
**ISBN**: 9789081145930

This publication is made possible by the Libre Graphics Community, through the financial support from the European Commission (Libre Graphics Research Unit) and the Flemish authorities.

Printed in Germany. `http://www.online-druck.biz`