Open Source Publishing:
design tools for designers

Much of the work (graphic) designers do, happens with software. And not just any software - the set of programs used is often limited to Adobe In-Design, Adobe Photoshop and Adobe Illustrator; maybe you've got Macromedia Dreamweaver and Macromedia Flash installed too.

Since Adobe Inc. bought Macromedia, the standard working suite of designers anywhere in the world can be purchased through one single company. And even if Adobe continues to develop brilliant packages, it is discomforting that one single party is responsible for the development of virtually all digital design tools available.

When I watched a friend laying out a document, fluently navigating from toolbox to property window and back, changing the size of a box in passing, tweaking a curve, layering an image, adjusting word spacing and placement, I started to understand that "becoming one with your extension" is in this case not science fiction, but practice. Like everyone else, we use computer programmes to write, read, listen, publish, edit and play. But more than often we do all these things at the same time and in connection to each other. It has become our natural habitat. We use software until we in-corporate its choreography. We make it disappear in the background. A seamless experience.

My physiotherapist used the following analogy to explain how humans use tools to negotiate the space around their bodies:

"If you prepare a sauce?" she said, "and stir it with a wooden spoon? you will be able to feel at which moment exactly the starch starts to burn to the bottom of the pan".

A wooden spoon might not be the kind of glamour and glitter we post-human-cyborgs are looking for, but I think it is in this unspectacular way

our daily operations with software help to make sense of our environment. Software produces culture at the same time as it is produced by culture. It is shaped through and locked into economic models of production and distribution. This is obviously as much true for a wooden spoon as it is for making animations in Flash but in proprietary software this "lock" is apparent in the crudest way possible.

> *Do You Have The Right Plug-in Installed?*
> *It Looks Like You Are Writing A Letter!*
> *You Have Unused Icons On Your Desktop?*
> *PLEASE READ THIS SOFTWARE LICENSE AGREEMENT CAREFULLY BEFORE DOWNLOADING OR USING THE SOFTWARE.*

With each possibility opened up by an operating system or software package, the space within which your design practice can take place, is circumscribed. Software is never politically neutral, nor are its aesthetics without colour: Each product prescribes use, and results in specific forms, sounds and shapes. It is therefore crucial that designers question those tools from the inside out, and investigate their workings as much as they can.

But how could we understand what software does to our work and working patterns without being able to step away from it? What if our work is not only made with, but also by software? Can we think ourselves outside it? What if we would want to adjust, reinvent, change, alter our tools when those forms of use are prevented by extremely restrictive licenses? How could we even understand what software does to design aesthetics and working patterns without being able to step away from them to try out different ways of making things?

Using Open Source software is not an as evident answer to those kind of questions for designers, as it might be for programmers or even DJ's. Most of us are trained to be consumers and not comfortable with actively interacting with software. Besides that, or as a result of that, Open Source software for design did not mature at the same pace as for example operating systems, networking- and sound editing tools did. As its evolution

is for a large part based on voluntarely input from programmers, the projected use of those programs and interfaces reflects another culture of design altogether.

> "*I'm laying out a technical book on small wind turbines in Scribus. There's a very dynamic thing that happens when small turbines furl out of the wind, and it's difficult to understand -- so we had the crazy idea of making a 'flip book' movie in the lower right hand corner...flip through the pages and watch the turbine furl.*
>
> *It seems to me I can -- extract frames from the video and sequentially number the photo files. Insert an image frame at the lower right hand corner of every right hand page, on the master page R. Write the script to increment the filename, and insert the image in the frame, then go to the next righthand page.*"
>
> *(from the Scribus mailinglist)*

It might be, that the implementation of OSX helps to blur those lines. With the help of X-11 it is now relatively easy to use Linux software on Apple Macintosh computers, so that designers can work with various Open Source packages without having to leave their familiar environment behind.

But let's take it a step further. Many of the tools Unix developers built for the production of technical documentation and scientific publications (softwares such as LaTeX for example) could be even more powerful in the hands of skilled designers. Picture the engineers' turbine furl flipbook... and you can imagine a continuum starting from a database back end (MySQL, PGSQL), to filtering with the help of XSLT resulting in automatically and dynamically produced documents in various formats: pdf, postscript, html, sgml etc.

Graham Harwood described The Gimp (Open Source image processing software) as "Photoshop with its guts hanging out", painting a graphic

image of what software can be more, than a user-friendly tool seamlessly doing its job. Open Source tools are not always "user friendly" in the usual sense of the word. Partially, because "user-friendliness" might mean something else all together depending on the expectations of its users, and partially because most Open Source software is "work in progress" and this means it's cut-off points are not necessarily concealed.

Within the Open Source Publishing project we try to think out loud about what other tools are possible and what is possible with other tools; to demonstrate ourselves and others what the possibilities and limitations of Open Source software are, how they can be tools to think and how they can be put to work in professional design environments. We try to be disciplined about the less exciting but much needed work of filing bugs and reporting back on our experiences, in order to find common ground with software developers and exchange ideas with the people developing our tools.

We are ultimately interested in making differences, glitches, misunderstandings and hick ups productive. The trick seems to be to simply not expect the same experiences as the ones we are used to.

"*Everything you see I owe to spaghetti.*"
Sophia Loren

How To Print A Booklet In 19 Easy Steps

The focus of this recipe is on the last bit: rearranging pages so that you can easily print out nice booklets. For a quick-and-dirty solution you can use Abiword or OpenOffice for the page-lay out part but Scribus is essential when you want to be precise with typography.

The recipe is based on the How-To posted on the Scribus Wiki:

`http://wiki.scribus.net/index.php/How_to_make_a_booklet.`

To make this recipe, you need to open a terminal, shell or work in the commandline. If you have never done this before, have a look a this tutorial:

`http://linuxcommand.org/`

You can of course print texts of any length, but folding and stapling more than 12 sheets of paper gets really hard so we suggest making booklets of 48 pages maximum.

The tools mentioned are all available in most software repositories, and can be installed using Ubuntu's Synaptic.

sample

Download sample .pdf file; if you simply want to print this document, start the recipe at step 13. The text used in this example is available here:

`http://www.constantvzw.com/cyberf/book?.`

Ingredients

* Linux operating system* [Debian / Ubuntu]
* Browser [Firefox]
* A text available under an open license
* xpdf-utils (includes: pdftotext, pdftops, ps2pdf?)
* Texteditor [Gedit]
* Lay-out software [Scribus 1.3.3.1]
* Font [Bitstream Charter]
* psutils (includes: psnup, psbook)
* Printer
* Paper
* Stapler
* A piece of soft cardboard (side of a box for example)

Print A Booklet In 19 Easy Steps

1. Choose any text that is available under an open license

2. Download the text to your harddisk in .pdf format or copy the text into a text editor

3. If you have downloaded a .pdf file, you need to convert the .pdf to a plain text file using the commandline:
   `~$ pdftotext infile.pdf`

4. Clean up the file as much as possible (remove unneccessary white lines, check whether any other corrections need to be made) in a text editor and save the document as .txt

5. Open Scribus and start a new document with the following options selected: Size: A5, Number of pages: 48, Page Layout: double sided and Automatic Text Frames

6. Import the .txt file in the Automatic Text Frame and do the necessary

lay-out; add page numbers etc.

7. Remove all empty pages so that you end up with a multiple of 4 pages (either 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44 or 48 pages).

8. Save / export the document as .pdf with fonts embedded

9. Using the commandline, convert the pdf file to postscript:
```
~$ pdftops -paper match infile.pdf outfile.ps
```

10. Rearrange the pages so that when printed and folded, each page ends up in the right place (when your booklet has 8 pages, page 1 should be placed opposite of page 8, page 2 opposite of 7 and 4 opposite of 5). n is the amount of pages in your booklet.
```
~$ psbook -sn infile.ps outfile.ps
```

11. Arrange two A5 pages next to each other on one A4 sheet (-2 refers to the amount of pages on the A4):
```
~$ psnup -2 -PA5 infile.ps outfile.ps
```

12. Convert the document back to .pdf format (This seems a redundant step, but without it I had problems with placing, so?)
```
~$ ps2pdf infile.ps outfile.pdf
```

13. Also use the commandline to print first the even pages (myprinter is the name of your printer, n is the amount of copies)
```
~$ lpr -P myprinter -o page-set=even -#1 infile.pdf
```

14. When the even pages are printed, you need to re-arrange the order of the pages so that the first page comes last.

15. Put the pages upside down back in the printer

16. Now print the odd pages
```
~$ lpr -P myprinter -o page-set=odd -#1 infile.pdf
```

17. Fold the pages from A4 to A5

18. Fold the stack back open and place it on the piece of cardboard with the cover facing you. Click open your stapler so you can staple the stack in the middle

19. Gently remove the stack (which is now stuck to the cardboard) and fold the staples back in.

Et voila!