



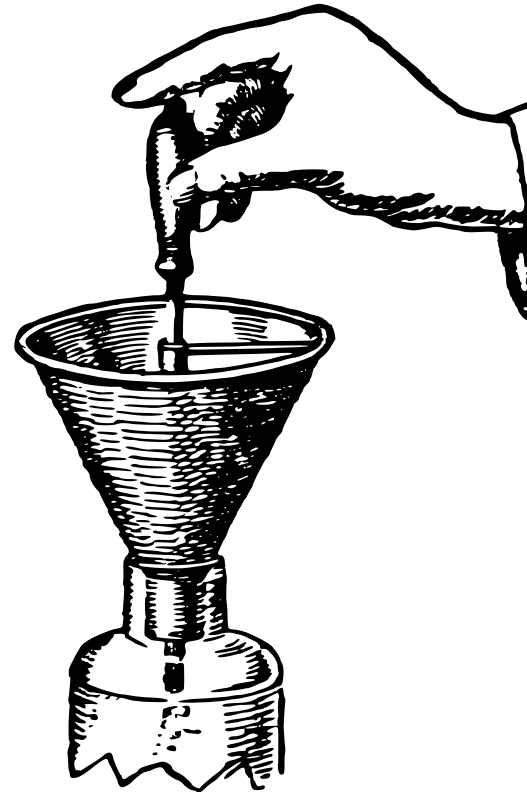






# Index

- 4 Masthead
- 6 Editor's Letter
- 7 Production Colophon
- 9 New Releases
- 10 Upcoming Events
- 12 Copyleft Business Dave Crossland
- 14 The heritage of our pixels Eric Schrijver
- 18 Coding Pictures Ricardo Lafuente
- 22 Setting a book with Scribus Pierre Marchand
- 24 Best of SVG
- 26 Desktop Pierros Papadeas
- 28 Interview with Oxygen's Nuno Pinheiro
- 35 Showcase
  - 36 Allison Moore Papercut
  - 38 Antonio Roberts What Revolution?
- 40 Making your workflow work for you Seth Kenlon
- 43 On being a Unicorn: the case for user-involvement in Free/Libre Open Source Software Libre Graphics Meeting Special
- 44 Talking about our tools Libre Graphics Meeting Special
- 46 AdaptableGIMP: user interfaces for users ginger coons
- 51 Resource List
- 55 Glossary 1.2





# Masthead

## Editorial Team

Ana Carvalho [ana@manufacturaindependente.org](mailto:ana@manufacturaindependente.org)  
ginger coons [ginger@adaptstudio.ca](mailto:ginger@adaptstudio.ca)  
Ricardo Lafuente [ricardo@manufacturaindependente.org](mailto:ricardo@manufacturaindependente.org)

## Publisher

ginger coons

## Administrative Partner

Studio XX <http://studioxx.org>

## Community Board

Dave Crossland  
Louis Desjardins  
Aymeric Mansoux  
Alexandre Prokoudine  
Femke Snelting

## Contributors

Dave Crossland, Seth Kenlon, Pierre Marchand, Allison Moore,  
Pierros Papadeas, Nuno Pinheiro, Antonio Roberts, Eric  
Schrijver.

Printed in Montréal by Mardigrafe on recycled paper.  
<http://mardigrafe.com>

Licensed under a Creative Commons Attribution-Share Alike  
license (CC-BY-SA). All content should be attributed to its  
individual author. All content without a stated author can be  
credited to Libre Graphics Magazine.

## Contacts

Write us at [enquiries@libregraphicsmag.com](mailto:enquiries@libregraphicsmag.com)  
[HTTP://LIBREGRAPHICSMAG.COM](http://LIBREGRAPHICSMAG.COM)

### Images under a CC Attribution Share-Alike license

Cover, inside cover, index and showcase separator page illustrations by Manufactura Independente based on images by Flickr user fdctsevilla.  
Photo of ginger coons by herself.  
Photo of Ricardo Lafuente by Ana Carvalho.  
Photo of Ana Carvalho by Luis Camanho.  
Photo of Dave Crossland by Mary Crossland.  
Photo of Eric Schrijver by himself.  
Illustrations in "Coding pictures" by Joana Estrela, Lidia Malho, Telmo Parreira, Sofia Rocha e Silva, Fábio Santos, Edgar Sprecher.  
Illustration in "Setting a book with Scribus" by Manufactura Independente based on "Book 8" by Flickr user brenda-starr.  
Signs in "Best of SVG" by Wikimedia Commons.  
Screenshots in "Desktop" by Pierros Papadeas.  
Photos in "Interview with Oxygen's Nuno Pinheiro" by Manufactura Independente.  
Icons and wallpaper in "Interview with Oxygen's Nuno Pinheiro" by the Oxygen project.  
Photos in "AdaptableGimp: user interfaces for users" by ginger coons.  
Screenshots in "AdaptableGimp: user interfaces for users" by Ben Lafreniere.  
All images in the Showcase section can be attributed to the creators mentioned therein. All are licensed CC BY-SA.  
GIMP, Inkscape, Kdenlive, MyPaint, Scribus and Web Font Downloader screenshots in "Resources" by Manufactura Independente.

## *A Reader's Guide to Libre Graphics Magazine*

In this magazine, you may find concepts, words, ideas and things which are new to you. Good. That means your horizons are expanding. The problem with that, of course, is that sometimes, things with steep learning curves are less fun than those without.

That's why we're trying to flatten the learning curve. If, while reading *Libre Graphics* magazine, you encounter an unfamiliar word, project name, whatever it may be, chances are good there's an explanation.

At the back of this magazine, you'll find a glossary and resource list. The glossary aims to define words that are unique to the world of Libre Graphics. The resource list provides valuable information about tools, licenses, whatever items we may be mentioning.

Practically, this means that if, for example, you're reading an article about Scribus (see pages 22 to 23), you can always flip to the back of the magazine, look up Scribus in the resource list and become quickly informed about it. This provides some instant gratification, giving you the resources you need to understand, in a moment, just what we're talking about.

We hope you like our system.

### Images under other licenses

Signs in "Best of SVG" are all in the Public Domain. They can be found in:  
[http://commons.wikimedia.org/wiki/File:UK\\_motorway\\_matrix\\_fog\\_nuvola.svg](http://commons.wikimedia.org/wiki/File:UK_motorway_matrix_fog_nuvola.svg)  
[http://commons.wikimedia.org/wiki/File:Ohituskielto\\_p%C3%A4%C3%A4ttyy\\_352.svg](http://commons.wikimedia.org/wiki/File:Ohituskielto_p%C3%A4%C3%A4ttyy_352.svg)  
<http://commons.wikimedia.org/wiki/File:Norwegian-road-sign-406.0.svg>  
[http://commons.wikimedia.org/wiki/File:Raitiovaunun\\_pys%C3%A4kki\\_533.svg](http://commons.wikimedia.org/wiki/File:Raitiovaunun_pys%C3%A4kki_533.svg)  
[http://commons.wikimedia.org/wiki/File:K%C3%A4velykatu\\_575.svg](http://commons.wikimedia.org/wiki/File:K%C3%A4velykatu_575.svg)  
[http://commons.wikimedia.org/wiki/File:Zeichen\\_238.svg](http://commons.wikimedia.org/wiki/File:Zeichen_238.svg)  
<http://commons.wikimedia.org/wiki/File:Laser-symbol.svg>  
[http://commons.wikimedia.org/wiki/File:PN\\_Pcha%C4%87\\_aby\\_otworzy%C4%87.svg](http://commons.wikimedia.org/wiki/File:PN_Pcha%C4%87_aby_otworzy%C4%87.svg)  
<http://commons.wikimedia.org/wiki/File:GHS-pictogram-explos.svg>  
Blender screenshot in "Resources" by Wikipedia user DingTo. GNU General Public License.  
[http://en.wikipedia.org/wiki/File:Blender3D\\_2.4.5-screen.jpg](http://en.wikipedia.org/wiki/File:Blender3D_2.4.5-screen.jpg)

### General

Advertisements, with the exception of those representing Libre Graphics Magazine, are not necessarily covered by the blanket CC BY-SA license. It is best to check with the projects they represent before reusing them.



---

## **April**

---

**11, 12**

.....  
Your wonderful event.

**22-26**

.....  
Relevant conference we  
haven't heard of.

---

## **May**

---

**8-10**

.....  
Amazing conference  
you're attending.

---

## **June**

---

**What,  
there's more?**

.....  
Why didn't you tell us?  
Please drop us a line at

[events@libregraphicsmag.com](mailto:events@libregraphicsmag.com)

We would love to help  
spread the word.





# Getting used to misuse

*ginger coons*

Use cases, at their core, are about the way users proceed through a system in order to achieve an outcome. Normally, there are lots of diagrams and small details involved in creating a use case. But we're not here to go over technical detail. Instead, we're here to talk about that core, the idea of looking at paths of use and interaction.

Then there are affordances, the features of a thing, its possibilities, the ways in which it might come to be used.

Clearly, then, we're talking about the way things are used and, more specifically, the way things are designed to be used.

As designers, artists, makers, builders, we make things that are of use, in one way or another. At the same time, we make use of the productions of others. We do both of those things on an almost constant basis, in our lives, our vocations, our work.

A graphic designer may design a poster which serves the use of informing viewers about that which it promotes. That same designer uses a set of tools, however diverse, to fashion the poster. Thus, the builder is built for. Both the poster and the tools of the designer have affordances and potential use cases. What, after all, is the proper use of a poster? Is it to be read? Is it to be attractive? Is it to be taken off the wall and folded into a paper airplane? To be stolen, only to be hung on another, more private wall?

Our software tools, in their affordances and potential use cases, define for us, to a certain extent, what we may and may not do. Those decisions are put in place by the people who design the tools. Together, as users, developers and all areas between the

two extremes, we boil in a constantly reconfiguring sea of use possibilities, material and mental affordances.

Which is why, in issue 1.2 of *Libre Graphics magazine*, we're looking at the interconnecting topics of use cases and affordances. We can look at it from a technical perspective but, perhaps more productively, we can also look at it philosophically. It's about the idea of the affordances of the work, who it's for, what it can do.

That applies both to the work designers do for others and also to the work of others, as it is employed by designers.

Use, misuse and happy accidents are all areas we're keen to discuss and explore in this issue. We look, this time around, at glitch art, smart workflows, the history of the pixel and its adoption, user interfaces designed to work for instead of against you and any number of other exciting topics.

We hope you'll stick with us as we wander through the diverse meanings of what it is to use and be used.

*ginger coons is a member of the Libre Graphics Magazine editorial team.*

# Versions under control

Ana Carvalho & Ricardo Lafuente



When working on a project, it helps to have a proper workflow set up, one which can help us do away with boring and repetitive tasks through as much automation as possible. One of the crucial parts in such a workflow is version control.

The most popular proprietary design software tools haven't yet incorporated the latest improvements in version control — in many cases it's totally absent from software suites and is usually provided by third-party commercial plug-ins. The consequence of this is that regular users are forced to adopt very crude ways of managing the versions of their work, usually with awkward numbering and notes on the filenames themselves. Things like “illustration7-final\_version3-PRINT-FINAL-SRSLY\_THIS\_IS\_IT-final2.jpg” should be familiar to more than a few designers.

On the other hand, the Free/Libre and Open Source software (F/LOSS) world is very much in touch with version control and other project management strategies. These strategies quite often come from the domain of software development. Thus, the thought of using a version control system (vcs) for the production of this magazine came up early. There is a wide array of choice for this purpose. The most popular options are Subversion, Git, Darcs, Bazaar and Mercurial. It should be said that there's some heavy argument about which vcs is the best and this discussion is nearing the status of a holy war. We decided not to waste too much time weighing choices and instead run with one and see how it fared — and Git was what we stuck with.

Cliché as it might sound, version control is one of the things that once you pick up, you can't figure out how you ever managed to do without. Not only do we get a full log of every

change that has been made, using version control makes collaborative work much more straightforward, giving us the ability to always know what the status of our project is and, if necessary, revert to older versions of any file without hassle.

Nevertheless, version control systems require some learning and hand-holding to get comfortable with. Among all the technical jargon — learning the meanings and effects of committing, reverting, staging, branching, merging, rebasing, pruning, annotating — we are slowly becoming familiar with this way of working, and are definitely seeing the advantages.

## PROPCOURIER 1.2

The ever-evolving typeface for this magazine, PropCourier Sans, benefitted from some tweaks for issue 1.2. Most of the work dealt with punctuation, softening the weight of the most used punctuation glyphs. We also began working on kerning, the headache of choice for type designers. In order to preserve our neurons, we decided to kern as we go: after typesetting 1.2, we looked at printed proofs for the most glaring kerning problems (as in “F/LOSS”) and fixed them. Each issue, we'll be adding more kerning pairs as we find the need for them.



# **HAVE YOUR SAY.**

**SUBMIT YOUR WORK, ESSAY,  
ARTICLE OR RANDOM PIECE.  
YOU'RE THE COMMUNITY,  
YOU'RE OUR FEATURE.**

**[SUBMISSIONS@LIBREGRAPHICSMAG.COM](mailto:SUBMISSIONS@LIBREGRAPHICSMAG.COM)**

**WE WANT YOUR FEEDBACK.  
TELL US WHAT YOU THINK.  
WHERE WE COULD DO BETTER.  
WHERE YOU COULD HELP.**

**[FEEDBACK@LIBREGRAPHICSMAGAZINE.COM](mailto:FEEDBACK@LIBREGRAPHICSMAGAZINE.COM)**

**DISTRIBUTE THE MAGAZINE.  
GET YOUR LIBRARY TO ORDER A SUBSCRIPTION.  
SPREAD THE WORD.  
HELP US REACH OUT.**

**[ENQUIRIES@LIBREGRAPHICSMAG.COM](mailto:ENQUIRIES@LIBREGRAPHICSMAG.COM)**

## New releases

### Reclaim your tools

by Jakub Szypulka

<http://vimeo.com/18568225>

Documents the slow beauty and diversity of activity to be found at even the most hectic meeting of software contributors. In this case, documenting Libre Graphics Meeting 2010. Made using Kdenlive and Audacity.

---

### AdaptableGIMP

<http://adaptablegimp.org>

A new version of GIMP, which allows users to make easy customizations. Read more about it on pages 46-50.

---

### ArtistX 1.0

<http://www.artistx.org/site3>

A version of GNU/Linux which bills itself as able to turn a 'common computer into a full multimedia production studio.' Based on Ubuntu and designed for multimedia artists.

---

### CrunchBang 10 Statler

<http://crunchbanglinux.org>

CrunchBang is version of GNU/Linux notable for its community of users who actively share screenshots of their modifications to the desktop. They share not only screenshots of their modifications, but also instructions for replicating their results.

*What's new with you? We're always eager to find out what designers, artists and others using and working with F/LOSS are up to.*

*Tell us what you've done lately at [enquiries@libregraphicsmag.com](mailto:enquiries@libregraphicsmag.com)*



## Upcoming events

*We're very pleased to present a calendar of upcoming events which encompass all things graphic design, media art and F/LOSS. Given that there are few events which tackle all three subjects, we aim to offer you events where you can be the agent of change: the F/LOSS designer at a traditional design event, or maybe the designer at a predominantly software developer event.*

**1-3  
APRIL**

---

### **Flourish 2011**

CHICAGO

<http://flourishconf.com/2011>

**30 APRIL  
1 MAY**

---

### **Linux Fest Northwest**

BELLINGHAM, WASHINGTON

<http://linuxfestnorthwest.org>

**10-13  
MAY**

---

### **Libre Graphics Meeting**

MONTREAL

<http://libregraphicsmeeting.org/2011>

**9-13  
MAY**

---

### **Icograda design week**

VILNIUS

<http://icograda.org/events/events/calendar738.htm>

1-3  
MAY

---

**Pica 2011**  
BANFF, ALBERTA

<http://picaconference.ca>

2  
MAY

---

**agIdeas 2011:  
International  
design  
research lab**  
MELBOURNE

[http://agideas.net/agideas-2011/  
design-research-lab](http://agideas.net/agideas-2011/design-research-lab)

7-21  
MAY

---

**CHI 2011**  
VANCOUVER

<http://chi2011.org>

19-21  
MAY

---

**Typo Berlin**  
BERLIN

<http://typoberlin.de/2011>

19-22  
MAY

---

**Live  
Performers  
Meeting**  
ROME

<http://liveperformersmeeting.net>





# Copyleft Business

*Dave Crossland*

Copyleft has a scary reputation among business people because they often do not understand it.

Copyright is easy — it's about what restrictions and freedoms you have to use and redistribute a work. Copyleft is a “pay it forward” feature of copyright licenses that says if you redistribute the work, you must pass it along on the same terms. You are free to take a libre work and improve it. You can take it as a part and combine it with your own parts to make a new, and hopefully better, thing. What makes copyleft powerful — and scary — is that if you choose to do this, the whole thing must be libre. You can stand on the shoulders of others but others can also stand on yours — or you can start from scratch and set your own terms.

Copyleft has been smeared as “viral” and “a cancer” because creators of proprietary software much prefer libre licenses without this bargain. Those licenses allow people to have their cake and eat it by exercising their freedom while denying others that freedom. Including libre parts in a proprietary whole defeats the original point of setting the work free, and copyleft is a good defense against this abuse. Copyleft is central to the most popular libre licenses for programs and creative works, in the GNU GPL and the Creative Commons Attribution-ShareAlike licenses respectively. Copyleft powers the explosive, exponential growth of share-and-share-alike culture. And, as always, fonts are special.

PostScript powered the early days of desktop publishing and it required the redistribution of complete fonts with documents. PostScript (ps) document files *linked* to font files. That was intensely annoying for proprietary font vendors because fonts

were endlessly copied all without license fees being paid. Font Digital Rights Management (DRM) schemes were cooked up over the years and found to be more trouble than they were worth. Being unable to print documents correctly is perhaps only slightly less annoying for designers than having an application crash and taking the work with it.

PDF solved this by combining fonts with documents or, ideally, the minimum parts of fonts needed for that particular document to print reliably. (Scribus has had faultless PDF export as a top priority since the beginning.) But this makes the story for copyleft fonts complicated. A copyleft font may overreach into the documents that use it, unless an exception is made to the normal terms — an additional permission to allow people to combine parts of a font with a document without affecting the license of texts, photographs, illustrations and designs. Most libre fonts today have such a copyleft license — the SIL OFL or GNU GPL with the Font Exception described in the GPL FAQ.

Web fonts return the world to linking documents to fonts. This is extremely unfortunate for the proprietary business world because people can see a font, like it, and figure out how to download and save it without paying for a proprietary license. It is, however, extremely fortunate for those doing business with copyleft works, because copyleft distribution is a wealth creation engine for those who know how to drive it. More distribution means more money.

The business of libre fonts is open for designers who can take a libre font and combine it with their own parts to make a custom typeface design for their clients — customers who could not afford to commission totally new typefaces, but who still desire



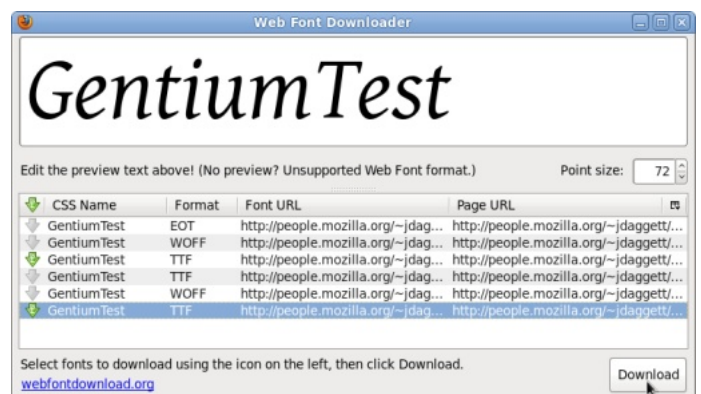
The Web Font Downloader Firefox Add-On delivers the dream, making it easy to download libre web fonts.

fresh typographic identities. Since all businesses will want to use their fonts on their websites, participation in free culture is guaranteed by copyleft. If you see a great typeface on a web page and it has a libre license, you can download and save it and improve it further.

The Web Font Downloader Firefox Add-On delivers this dream, making it easy to download libre web fonts. The next step, improving the font further, highlights the issue of font sources. OpenType has two flavours, one with PostScript-style cubic outlines and the other with TrueType-style quadratic outlines. The PostScript flavor is superior as a font format and looks great on computers using FreeType and on Mac OS X, but lacks the pixel-level control of TrueType needed to look good on most Microsoft Windows computers. This means almost all web fonts are distributed in a format that is a long way from “the preferred form of the work for making modifications to it.” That is the definition of source code in the GNU GPL, and it works very well for programs. I hope one day it will be a tradition for fonts too.

Get the Web Font Downloader Firefox Add-On now from [WWW.WEBFONTDOWNLOAD.ORG](http://WWW.WEBFONTDOWNLOAD.ORG)

*Dave Crossland believes anyone can learn to design great fonts. He is a type designer fascinated by the potential of software freedom for graphic design, and runs workshops on type design around the world.*  
<http://understandingfonts.com>





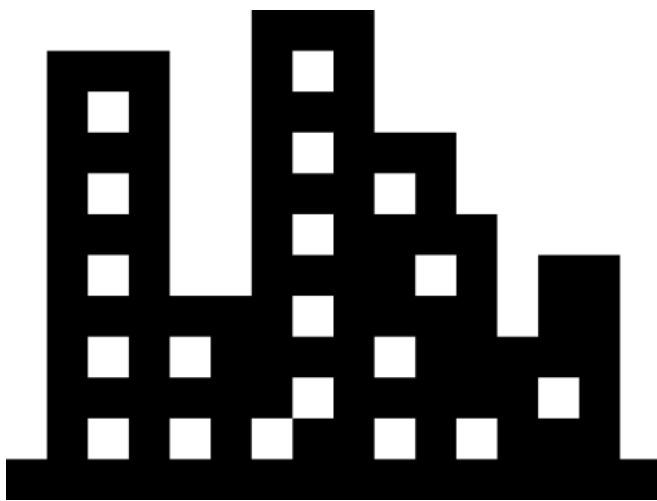


# The heritage of our pixels

*Eric Schrijver*

When John Whitney made his pioneering computer art films as an artist in residence for IBM in 1960, the computer screen he used did not use pixels. Rather, it was a single beam which could be instructed to move across the screen, much in the same way that postscript instructions tell a vector to move.<sup>1</sup>

The graphics in Atari's arcade games, like *Battlezone*, were also drawn with vector lines on an oscilloscope.<sup>2</sup> In the long run, a matrix of points became the preferred method to describe screen output. And it still is today. In fact, we have a more rigid matrix now that we use LCD displays: they have a "native" resolution



A city  
Beautiful abstractions in Anthony's icons

When in doubt, look at your predecessors. Most of our historic design for the computer screen is bitmap-based.

determined by the number of pixel elements — whereas the phosphor dots in a color CRT display bear no relation to pixels or subpixels displayed on them.<sup>3</sup>

So, to get your digital resources out on a computer screen, you have to describe them as a matrix of points. That's easiest when you work with data that itself is a matrix of points. It's even easier when you map the matrix of points directly in the data to the matrix of the points on the screen.

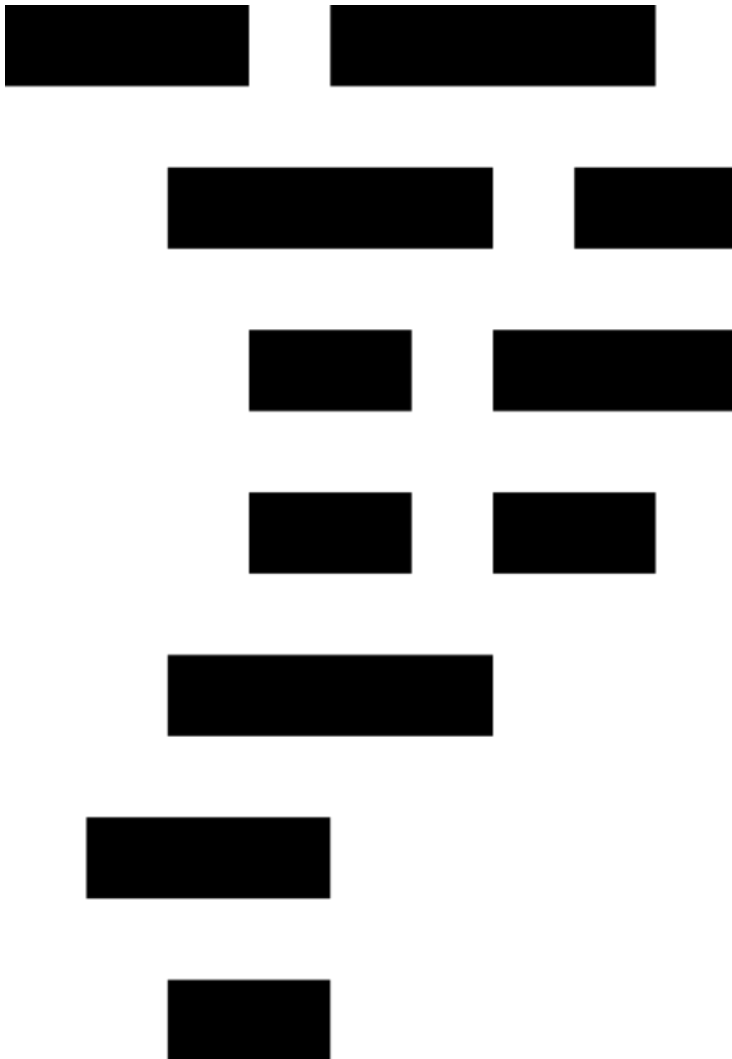
The easiest solution is not the best, in this case. Try to browse the internet on a 24 inch screen and, by default, it will look rather awkward:

singular columns of 960 pixels, with huge

swaths of background image on either side. That is because the layouts are specified in CSS pixels and, by default, the browser makes them correspond with "device pixels".<sup>4</sup> Where this used to be a predicament, now it's just a convention. All modern browsers support zooming in on the content. They're able to make pixel-based layouts smaller or bigger.

On the mobile phone, the rapport between the pixel of the design and the pixel of the screen has been cut entirely. The webpage is initially displayed fully, and subsequently the user can zoom, continuously, in and out on the design.

Scalable user interfaces benefit from vector graphics. After all, vector graphics are supposed to shine in the world of scalable.<sup>5</sup> There's even a vector format that was named after this inherent property: Scalable Vector Graphics. But does that mean we can't use the model of the bitmap in our new designs and interfaces? Not necessarily.

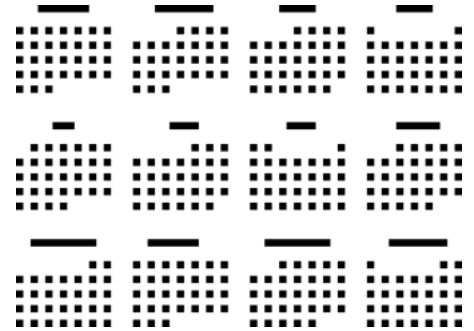


When in doubt, look at your predecessors. Most of our historic design for the computer screen is bitmap-based. I like to look at early pixel-based GUIs for inspiration. There's a library of icons and user interface elements for the X window system, collected by Anthony Thyssen, available online.<sup>6</sup> Because of the limitations inherent in early systems, many of them are really simple, black and white, 16x16 bitmaps. Through tight constraints, they attain a very evocative kind of abstraction. In this they resemble Susan Kare's icon designs for the original Macintosh, which are much better executed than current iterations.

These designs don't deserve to stay locked to the grid of display pixels growing ever tinier. They also don't have to: you could paint these designs with square meter pixels on your wall, with even that rendering making them look great.

But what better way to reinterpret these designs than to convert them to vectors?

Traditional tracing algorithms do no justice to these designs. Looking for the curves underlying the designs ignores that the pixel grid is constitutive of the design. We are not looking for the platonic ideal. In this case, there's nothing to do but make vector pixels:



Above: A calendar.  
Left: A tornado (from Nethack).

a vector square for every pixel! It doesn't even have to be a square. After all, a bitmap is a collection of points, and points have no predefined shapes. It could be circles or any arbitrary shape. You could make the pixels come together in horizontal scanlines, or vertical ones. You could distort the grid on which they are based.

There are many possibilities in the future of rendering and the further we go in exploring them, the closer we come to keeping alive the heritage of our pixels.

---

1. Thanks to Joost Rekveld for his classes, introducing these works amongst others  
2. Form and Code, In Design Art and Architecture: Casey Reas, Chandler McWilliams, LUST; Princeton Architectural Press 2010  
3. <http://en.wikipedia.org/wiki/Pixel>  
4. <http://webkit.org/blog/55/high-dpi-web-sites>  
5. Actually, there are quite some pixel based scaling algorithms too:  
[http://en.wikipedia.org/wiki/Pixel\\_art\\_scaling\\_algorithms](http://en.wikipedia.org/wiki/Pixel_art_scaling_algorithms)  
6. My reissue available at  
<https://github.com/codingisacopingstrategy/Alcons>

*Eric Schrijver (Amsterdam, 1984) is a visual artist who makes installations and performances. Eric teaches Design for new media at the Royal Academy of Art in The Hague. He is inspired by open source and programming culture.*  
<http://ericschrijver.nl>

Want to make your own vector pixels? Follow these (relatively easy) steps to generate your own vector pixel icons.

The following instructions should work just fine on either Linux or Mac.

Grab the code: Either type it in by hand, copying the code [on the right] or go to the assets sections of our website ([HTTP://LIBREGRAPHICSMAG.COM/ASSETS](http://libregraphicsmag.com/assets)) and download the vector pixel pack we've prepared for you.

If you're copying out the code manually, enter it into a text editor and call the file `vectorpixel.py`.

Find an image: If you're doing it on your own (instead of using the assets we've provided), find a simple image. Make sure it has very few colours (you're going to have to strip all the colour out of it). Simple logos, warning signs and similar types of images work well. Open it up in your favourite raster image editor (we used GIMP).

Strip out the colour by doing things like increasing the contrast as much as possible and posterizing. You're aiming to have an image with only black and white. While you're at it, resize the image to a very small size. 50px by 50px works well.

WARNING! We're serious about the small image size. If it's too big, the resulting SVG will be very, very big and may just crash your image viewer.

Save your image (as something like a PNG, JPG or other basic raster format). Make sure to flatten while you're at it. Layers will only cause trouble in this case. Make sure you save it in the same directory as your `vectorpixel.py` file.

Point the script: Take another look at `vectorpixel.py`. On the 8th line, you'll find something that looks like this: `SOURCEIMAGE = 'city.png'`. If you've made an image of your own, you'll want to change `city.png` to whatever the name of your file is. Then save `vectorpixel.py` again. Now, when you run it, it'll be looking for the right image.

Convert it: Open up your terminal (for more on using the terminal, check out the detailed instructions and explanation on pages 22-23). Navigate to the directory containing `vectorpixel.py` and your image.

At the prompt, type: `python vectorpixel.py > city.svg`

If you've provided your own image, you can change that last bit. For example, if your source file is called `attention.png`, you can sub in `attention.svg`. All this does is set up a destination file.

Hit enter. It'll look a little like nothing has happened. However, if you go and take a look in your folder, you'll find a new file, called `city.svg` (or whatever you've named it). Take a look at it. It should be made up of lots of little vector pixels.

You've just made a vector pixel icon!



```
#!/usr/bin/env python
""" Generates vectorpixels based on 2-bitmaps (2 color pictures).

TODO: use element tree for XML; implement Floyd-Steinberg
dithering for color and greyscale images; implement vertical
and horizontal scanlines """

import Image

SOURCEIMAGE = 'city.tiff'

class vectorpixel:
    def __init__(self, image):
        self.i = Image.open(image)
        self.px = self.i.load()
        self.constructed = False

    def construct(self, grid=24, line=1, rounded=4, test=(lambda x: x == 0)):
        self.grid = grid
        self.line = line
        self.rounded = rounded
        self.width = self.height = self.grid - 2 * self.line
        self.test = test
        self.fill = '#000000'
        self.constructed = True

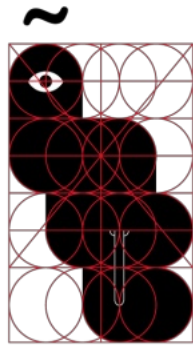
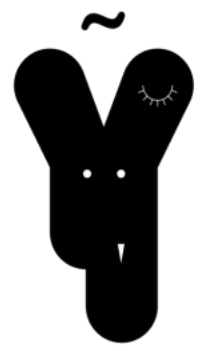
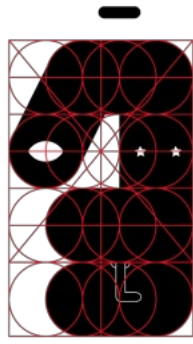
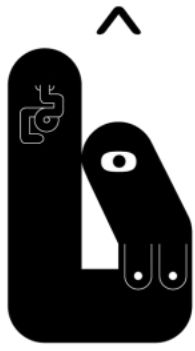
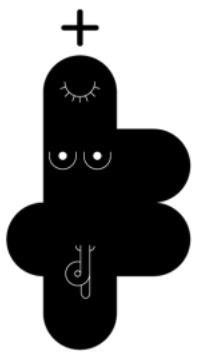
    def _yieldlocations(self):
        for x in range(self.i.size[0]):
            for y in range(self.i.size[1]):
                if self.test(self.px[x,y]):
                    yield (x,y)

    def _mkelements(self):
        for l in self._yieldlocations():
            yield "<rect x='%s' y='%s' width='%s' height='%s' rx='%s' fill='%s' />" % (
                self.grid * l[0] + self.line, self.grid * l[1] + self.line, self.width, self.height, self.rounded, self.fill)

    def _format(self):
        output = '<svg xmlns="http://www.w3.org/2000/svg" width="%s" height="%s">\n' % (self.i.size[0] * self.grid, self.i.size[1]
        * self.grid)
        for e in self._mkelements():
            output += e
            output += '\n'
        output += '</svg>'
        return output

    def generate(self):
        if not self.constructed:
            self.construct()
        return self._format()

if __name__ == "__main__":
    v = vectorpixel(SOURCEIMAGE)
    print v.generate()
```



# Coding pictures

*Ricardo Lafuente*

At the Fine Arts Faculty of Porto University, we built up an introductory class focusing on procedural strategies inside a graphic design context. In less stuffy terms, the purpose was to introduce design students to code. However, this required some thought on what subjects to teach (and which to leave out), which pitfalls to avoid, and which approach would work best to introduce an alien, cold and logical subject such as programming to creative people.

Designers are inevitably involved with computers, which are present in most stages of a graphic designer's workflow, from initial sketches to printing finished pieces. Yet there's a dearth of formal education on the technical and social workings of computers and digital media in general.

Nevertheless, attention has been paid to this field during the last decade, which saw the birth and growth of creative-oriented applications, spearheaded by its most popular example, Processing. Among other creative coding tools, we find Pure Data, Context Free, Drawbot, Nodebox, Shoebot, Supercollider and Fluxus. The overwhelming majority of these tools are F/LOSS.

Learning to code is becoming more and more of an obvious choice for designers. The rising popularity of the web has created a huge demand for skilled coders, but designers are also a key part of any serious venture. A designer who can implement his own ideas, instead of just handing over mockups to a web developer, ends up with a big advantage. Becoming acquainted with digital logic, the workings of computers and their bells and whistles is also a way to liberate oneself from being a software operator, and be able to think for and with the machine.

## TOOLS AND STRATEGIES

In our semester-long class, we focused solely on still, static output, meaning that animation and interactivity were left out. This gave room to explore the basic commands and features, as well as combining them with creative strategies that the digital medium enables, such as repetition and randomness.

Processing was considered as the application for this class, but Nodebox/Shoebot were picked because they work natively with vector graphics, which was a crucial factor when considering that the created designs should be meant for print. The fact that they're based on Python, whereas Processing is based on Java, also played a part. Python is one of the most appropriate



languages for introducing people to programming, due to its clear, readable syntax, which almost resembles the pseudocode used to explain abstract programming concepts. It also hides away (or puts sugar on) much of the complexity behind programming concepts, allowing us to focus on properly wording our orders to the computer.

Nodebox and Shoebot provide a sketchpad for writing small Python scripts. When running them, the program will create graphical output—an image. This instant visual feedback is a big plus for teaching code to creative-minded people, since it allows for swift tinkering and borrowing from the provided example scripts, and was crucial in easing design students into the coder way of thinking.

### **PRACTICAL EXAMPLE : CHARACTER GENERATOR**

One major assignment in this class was to design an identicon generator. Identicons are icons commonly used in blogs, especially inside comment sections, which identify the commenter through a graphical representation of their IP address. This is done by combining different possible parts into one final image. MonsterID and Wavatar provide icons in the shape of quirky monsters, whereas Identicon generates abstract, geometric shapes. The goal of this assignment was to think up the design for an identicon, freely choosing the subject, and create a program that could generate different outputs randomly.

The size constraints of a blog icon are a big limitation, one that wasn't forced on the students in order for them to focus on the more relevant creative questions. Many of the students went through the character-design route, though others attempted more daring approaches, such as cake and bug identicons.

The challenge of this assignment was not the coding itself. Most students were already rather comfortable with using randomness, drawing with code and importing external images. The focus was on creating consistent designs which could work with different compositions and still end up as a complete final result, not giving away the fact that it was generated by a program.

The illustrations running alongside this article are some of the results of this assignment.

#### Image Credits:

Page 18: Fábio Santos; Edgar Sprecher; Joana Estrela.

Page 21: Sofia Rocha e Silva; Telmo Parreira; Lídia Malho.

Nodebox: <http://nodebox.net>

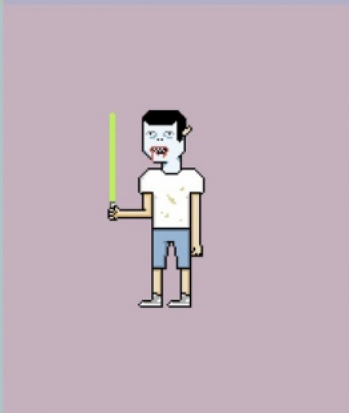
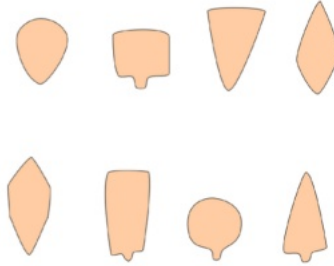
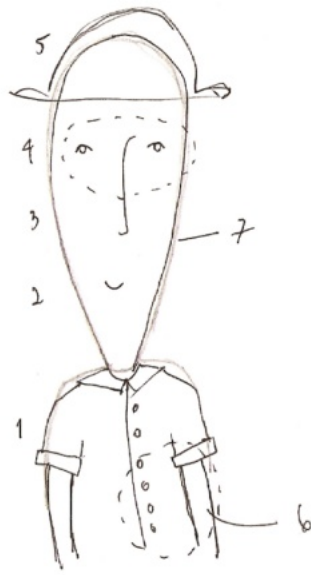
Shoebot: <http://shoebot.net>

Pure Data: <http://puredata.info>

Drawbot: <http://drawbot.com>

Context Free: <http://contextfreeart.org>

Fluxus: <http://pawfal.org/fluxus>



# Setting a book with Scribus

Pierre Marchand

I remember my own first time, the first serious one, was a bookletization of a famous, amongst aficionados, little parody by Pierre Louys under the title of *Manuel de civilité pour les petites filles à l'usage des maisons d'éducation*. With its typical late 19th century French style, it was natural to associate it to a didone font. I ended up using the Didot shipped with the Mac OS I owned at this time.

Here is the crux of this story: at this point I hadn't yet read the paper by René Ponot<sup>1</sup> convincingly establishing that it was not a good idea to use ligatures with the Didot typeface. I wanted to use them! But in this instance of the typeface, the ligatures and old numerals were outside the charmap, intended to be accessed only by means of OpenType substitution or glyph index. That, in itself, was an adventure.

This time, though, was also my first time going deeply into font technologies and Scribus code. Along my journey in these fields I came to read Theotiste Lefevre. His amazing *Guide pratique du compositeur d'imprimerie* helped me realize how much, even if still non-trivial, the making of a book has become within everyone's reach with desktop publishing and personal printers. For now, forget Louys and Didot and go for a book in a minute!

The recipe is as follows. First be modest and grab some text fallen into the public domain at [gutenberg.org](http://gutenberg.org). If you attempt to write your own material, you will definitely not be able to do a

book in a minute. Next, run a bit of Perl magic powder onto it like `perl -n -e 's/(\S)\r\n/\1 /ms; print $_;' original.txt > withoutlinebreak.txt` to let Scribus do its work at line breaking. (See below for more detailed instructions.)

Now you can create a new double-sided Scribus document with a bunch of pages and automatic text frames turned on. Import the text into the first text frame. Set the default paragraph style to something that looks like a book, serif typeface at 10 points, justified, etc. Et voilà!

Well, it isn't exactly ready to serve to your friends, but you get enough of the taste of an actual book to open the door and start to work. If you think not, click on the eye at the right bottom of the Scribus window to turn on Preview mode.

While writing these few lines, I'm doing the same as I did years ago and am still amazed by what Scribus brought to us — by what it allows us to do and the opportunity it gives to learn about desktop publishing. We have the opportunity to do publishing work as Scribus exposes its internal representation of graphic objects through an opened source code and file format.

1. Le Didot a-t-il besoin de ligatures ?  
Cahiers Gutenberg no. 22 (1995), p. 17-41  
[http://cahiers.gutenberg.eu.org/cg-bin/article/CG\\_1995\\_\\_22\\_17\\_0.pdf](http://cahiers.gutenberg.eu.org/cg-bin/article/CG_1995__22_17_0.pdf)

## EXECUTING COMMANDS IN THE COMMAND LINE

No matter what operating system you're using, you've got a command-line interface at your disposal. If you're of a certain age, you may remember fiddling around a little with MS-DOS. Even if you never did, don't worry about it. The command line is friendlier than you may think.

Now, because we're all designers here, chances are good you're using a Mac. Or, if you're like us, Linux. The tips and commands listed below work just fine for both. If you try them in MS-DOS (under Windows), your computer may explode. We're not quite sure, really.

### Open a terminal:

On a Mac: Crack open your Applications

folder and go to Utilities. There's a program there called Terminal. Open it.

On Linux: Normally under your Accessories or System menu, you'll find something called Terminal. Open it.

### Get to the right place:

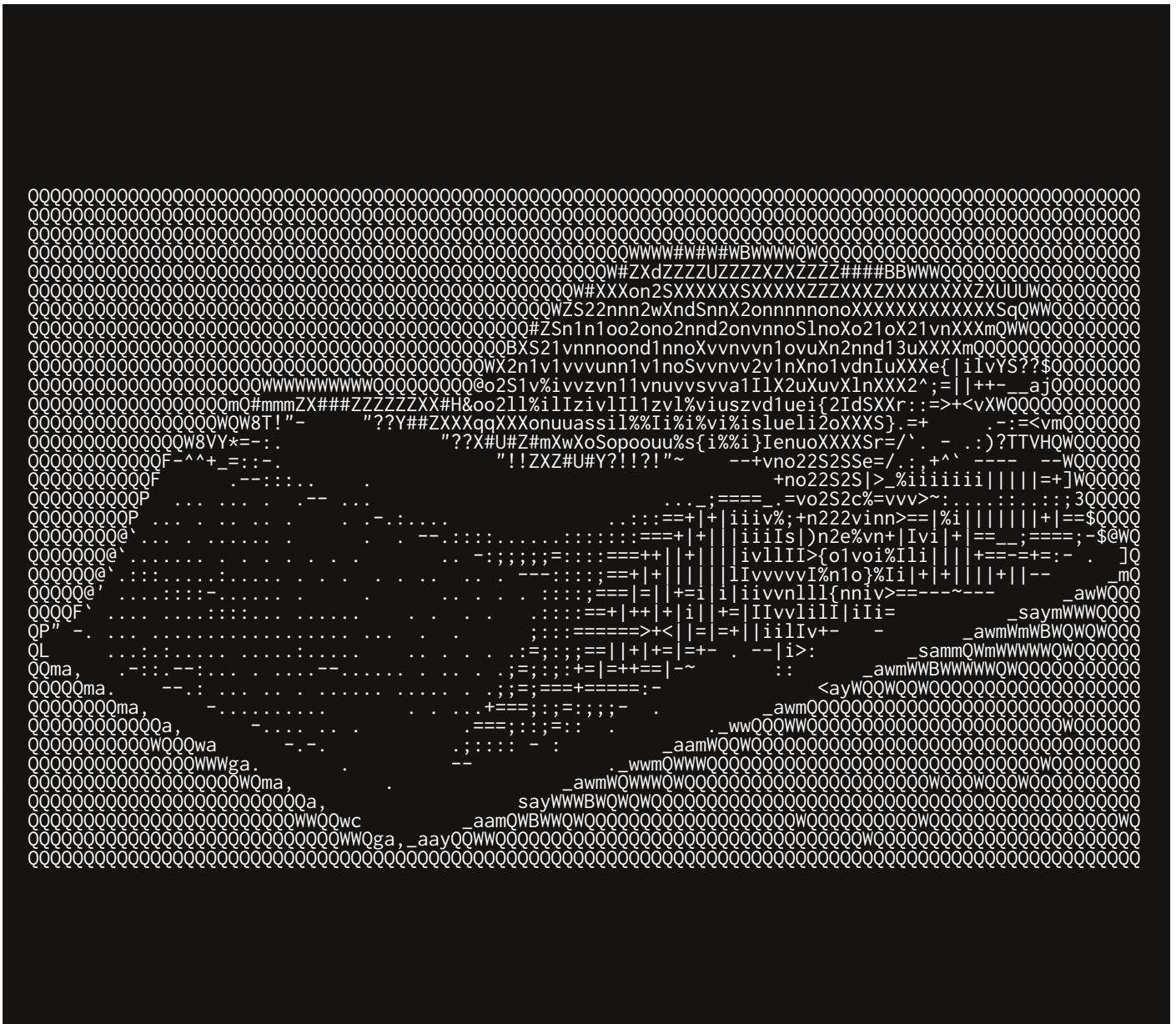
If you've downloaded a book from Project Gutenberg, hopefully as a plain text file (something ending with .txt), great. If not, go back and do that. But make sure to take note of where you've saved it.

When you opened Terminal, it should have started you up in your home directory. To make it easier to find where

you're going, open up your file browser (Finder on a Mac, or on most kind of Linux, just double click on the icon for your home directory). Navigate to where you put your file. Now, take a look at the path leading up to that. For example, if you left it in your Downloads directory, chances are good that it'll only be one directory past home.

Once you have an idea of where you've put your file, go back to the Terminal. To change directories (because that's what you're about to do, unless you've left the file in your home directory), you're going to use the `cd` command. It allows you to (yes!) change directories. Let's say you've left your file in the Downloads directory.





In your terminal, you'd type "cd Downloads" (without the quotation marks). That would take you to your Downloads directory. If, in the Downloads directory, you happened to have another directory, this one called books, for example, you'd then go "cd books" (note that it's case sensitive).

**Looking around:**

Now, we're in our fictional home/Downloads/books directory. Let's take a look at what's there. To get a list of the contents of a directory, just type "ls" while you're in the directory you want to look at. It'll turn up a list of all the files and directories contained within that directory. If you've gotten to the right

place, ls should show you the book you've downloaded.

**Running the script:**

Now you can run the script mentioned in the article. Just copy it and paste it into your terminal. Or, if you're reading this in print, type it. Heck, type it in regardless, just for practice!

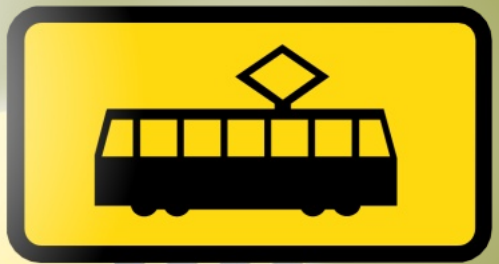
```
perl -n -e 's/(\\S)\\r\\n/\\1 /ms; print $_; ' original.txt >
withoutlinebreak.txt
```

Of course, you'll want to change "original.txt" to reflect the actual file name of the book you downloaded. Then, hit Enter. If all goes well, the next time

you do an ls, you'll find a new file, called "withoutlinebreak.txt" which will be the book you downloaded, without linebreaks and ready to be conveniently typeset in Scribus.

While this may seem like a lot of complicated steps, once you get used to it, you'll find that it's easy, convenient and fast. And it's just the beginning of what you can do with the command line.

—the editors



# Best of SVG

## *Wayfinding and warnings from Wikimedia Commons*



Warning signs, street signs, all kinds of signs: they blend into our environments, letting us know what we need to know with minimal fuss. Rather, that's what they do when well designed. When badly designed, they confuse and jar us.

This time around, Best of svg has collected some of the finest examples of signage Wikimedia Commons has to offer. From warnings of lasers and potential explosions, to incredibly pleasing no passing signs, there's a nice assortment on offer.

We've found that warning and traffic signs are one of the strong points of the Wikimedia Commons collection of svg graphics. Signs and heraldry. But that's a collection for another day. If you don't yet know about Wikimedia Commons, it's well worth checking out. Not only do its graphics feature in the Wikipedia articles we know and love, but it has a pretty nice collection of other media, all under permissive licenses, for your appreciation and re-use. Find it at [commons.wikimedia.org](https://commons.wikimedia.org).

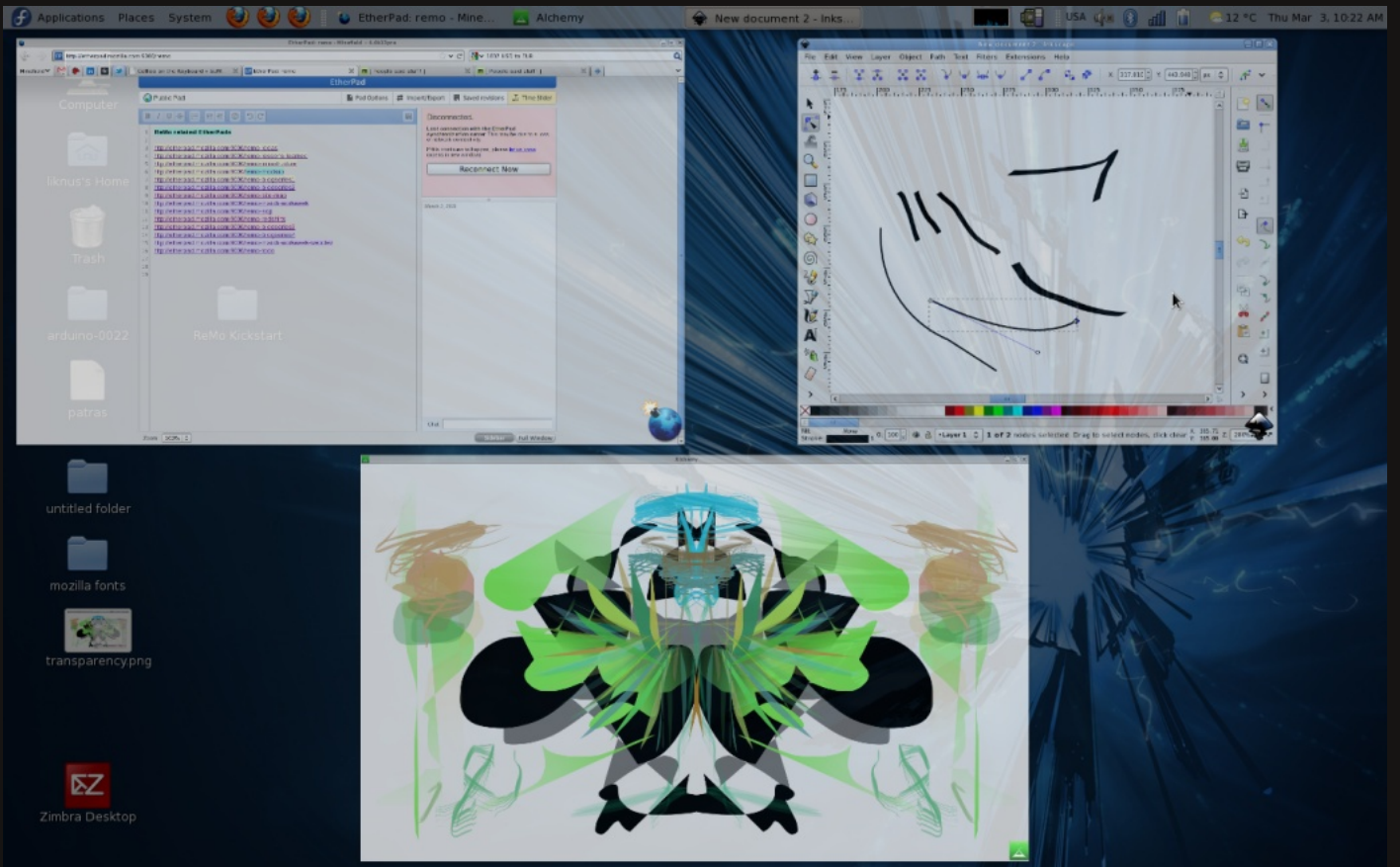
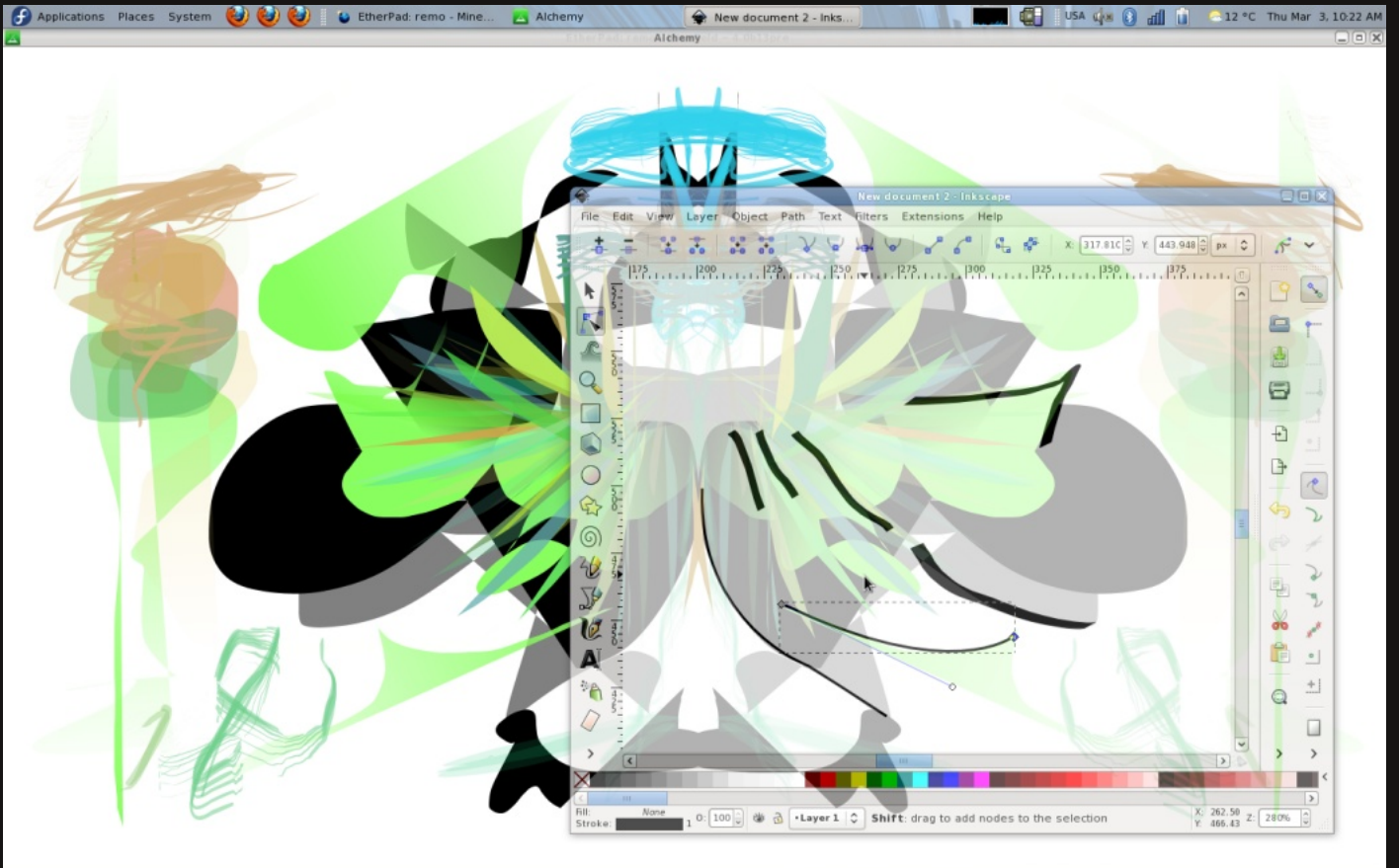
—the editors

*We at Libre Graphics magazine have a thing for open standards. We like their transparency and their interoperability. We like that, with a well documented standard, everyone has an equal chance to play nicely together.*

*That's why we like SVG so much. It's a well developed, well supported standard brought to us by the World Wide Web Consortium (W3C). It's available for implementation by anyone developing software. It shows up in modern browsers, fine vector graphics editors and any number of other places.*

*One thing that's missing, though, is you: the designer, the artist, the illustrator. So put down that .ai file and check out SVG.*





# Desktop

*Pierros Papadeas*

*Desktop features clever hacks, workarounds and customizations used by designers and artists. We look at the way the best and brightest of our peers work.*

*This issue, Desktop features customizations used by Pierros Papadeas, a Fedora Ambassador and Maintainer of the Fedora Design Suite Spin.*

## Transparent Windows

A common drawing problem is layering info from different apps. GNOME with Desktop Effects gives you the ability to make a window transparent. It means being able to trace without having to go back and forth, saving, exporting and importing. This effect can also be achieved with other desktop environments.

## Expose Windows

It's GNOME and Desktop Effects again. Just point your mouse at the upper right corner and you get an "Expose" of all windows. It allows you to easily change between windows and get a quick view of the huge amount of awesomeness you're working on.

*In this instalment of Desktop, Pierros is using Fedora Linux, GNOME desktop environment with Desktop Effects, Inkscape and Alchemy.*







# Interview with Oxygen's Nuno Pinheiro

*Manufatura Independente interviews Nuno Pinheiro*

*Nuno Pinheiro coordinates the Oxygen project, initially a set of icons for KDE which evolved into a design platform comprising 2000+ icons, wallpapers, sound effects and window styles. He's employed as a UI designer at Klarälvdalens Datakonsult AB. Ana Carvalho and Ricardo Lafuente went to ask him about project management, art direction and the history of Oxygen.*

**Manufatura Independente:** *Tell us about Oxygen. What is it? How is it related to the KDE project?*

**Nuno Pinheiro:** Well, Oxygen is considered one of the pillars of KDE. It's a design platform.

Initially, it was created by three people—I wasn't one of them—at this get-together called Appeal Meeting, which took place right after the KDE 3.4 release. Many important KDE people were present in order to discuss and decide where to go from there. KDE had reached a fairly mature state and it was appropriate to find out which next steps to take.

Two people involved in the meeting were Kenneth Wimer and David Vignoni. David is the author of the Nuvola theme which was, back then, one of the most popular alternative themes for KDE. Actually, it was the most used theme. At this meeting, we decided to begin work on a new icon theme, which was to be called Oxygen.

Ken and David then invited me to join the effort of building a completely new icon theme. We had the sponsorship of Novell, which was a nice and cool company back in the day. That's the story behind the creation of Oxygen, which was set to become the icon theme for the fourth version of KDE. We started work on the icons. Novell ended up changing their minds and left the project. We carried on nevertheless.

As we progressed on Oxygen, it became clear that icons were a single aspect of the user desktop experience. The desktop has many things, and it became clear that the user interface (UI)

toolkit (or window decorations) was a significant part of the experience. KDE used Qt, which had its own window decorations. We thought it was appropriate to do our own UI theme. So I started work on that as a sub-project of Oxygen, drawing many mock-ups using Inkscape. I made a full mock-up of the theme, without any code underneath. Then, I approached some developers and they ended up supporting my work. We worked together and did some iterations until we got to the current version.

Now, if you can make an icon theme and a UI theme, you can also make a window theme. So we did that next. If you can make a window theme, you can also make a sound theme. So I talked to Nuno Póvoa, who made it. If you can make a sound theme, you can also make a mouse pointer theme. If you make a mouse pointer theme, you can also make wallpapers. If you can make wallpapers, you can make websites. This way, Oxygen ended up becoming a design platform. Everything in KDE that is design-related is taken care of inside Oxygen (except type).

In the meantime, while making Oxygen, we decided to adopt the freedesktop standard naming spec. Thanks to this icon naming scheme, you can get a KDE icon theme and use it within GNOME, and vice-versa. This means that Oxygen, though closely connected to the KDE project, can be used in many other contexts—in fact, we encourage that other projects use Oxygen. The license is free and the process is open. I get really happy when Oxygen is used in other projects, other places and for other purposes.

*Going from this idea of Oxygen being the design hub for KDE, there's a question we're interested in: what's the decision-making process for aesthetic criteria? In the end, how does it come together? Is it a top-down process, or can anyone propose new directions? Is there any other kind of control?*

It's bidirectional. Actually, I coordinate the project, and I'm the guy who says "Okay, we're going that way" or "We're not going that way." I've got the role of drawing the line when it comes to final design choices.

However, Oxygen is not a young project. There's quite some years behind it. It's sported some different visual tendencies through time, graphically and formally. Every two versions, we



try to slightly change the general concept and message of the theme. This message is not defined by us, but rather by the community. For instance, the message we're working on for 4.6 and 4.7 is about elegance, in its broadest definition. Code can be elegant, user experience can be elegant. So, we took this message and tried to convey it through the theme design, aiming for an elegant experience: elegant wallpapers, elegant sound pieces, and so on. This is the centerpiece of the experience we want to pass on to the user—a global message that Oxygen helps get across.

And this is the most complicated part inside a design project: achieving consistency when we have several people with very different styles and ideas contributing to the same project. It's the challenge of creating a bundle that is smooth and continuous, has an even pace, and speaks the same language. Managing all of this is my task: talking to people and trying to have their work flow into something that's consistent and dynamic, something that goes along with the rest and, at the same time, addresses the core message.

*Regarding your tools of choice, we know you use Inkscape...*

I do use Inkscape. I also work with Blender, Gimp, Krita, scanner, pencil, pen and my imagination.

*Have these been your tools all along?*

When I started, my first tool was Sodipodi, the predecessor of Inkscape. Inkscape is definitely my main design tool.

*Have you ever approached the Inkscape developers to ask for a specific feature?*

To be honest, I'm not close to the Inkscape guys. On the other hand, I do frequent exchanges with the Scribus people. We get along rather well. I'm almost done with their icons! Scribus requires a lot of icons, around three hundred.

*How many icons are there in Oxygen?*

Two thousand and something. It's the largest part of KDE in terms of file size—two hundred or so megabytes. As far as I know, it's the world's most complete icon theme. I'm not aware of any other theme with such an amount of icons. Tango had almost as much, but we're bigger. To give you a point of comparison, Apple only has around eighty base icons, and then each application brings their own set.

*Are there any style guidelines that you set out before starting work on a new theme? Setting a formal style direction is a*

A good designer should incorporate the engineer and the artist, but most of the time the artist wins.

*mainstay of traditional graphic design, usually through corporate identity manuals or interface guides. Our question is, do you follow this tendency, or is the Oxygen style defined through a less formal, more organic way?*

It is organic. To be blunt, I don't believe in those things. I've read several identity and interface guideline manuals, particularly icon style guides. I could get the style guidelines for Windows Vista and create Mac icons following them, and vice versa. This while strictly following their rules.

*And you could end up with something consistent.*

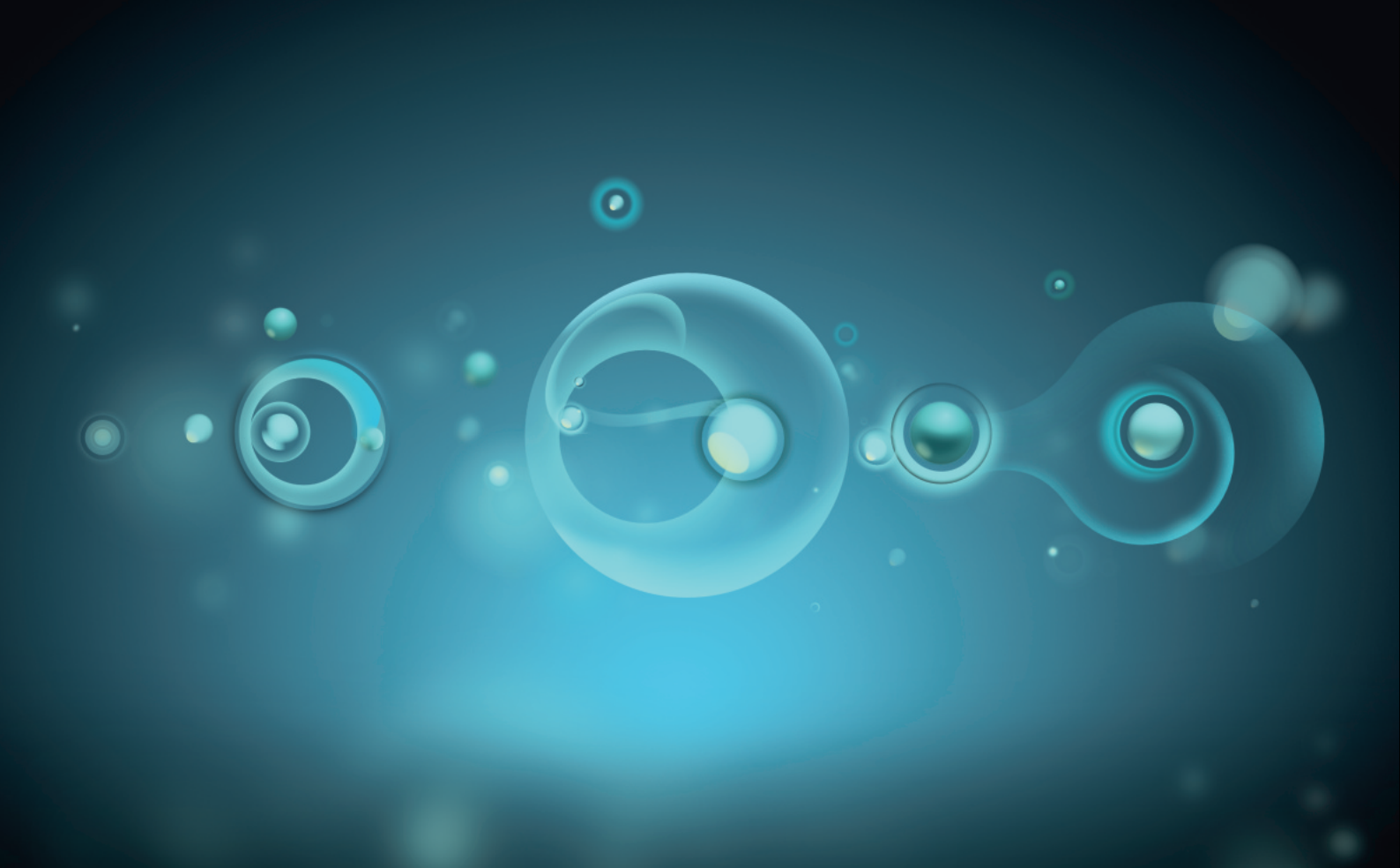
I could! Any designer worth his name can do that. It's very easy for a designer to follow every single rule, and still end up with something that doesn't fit. There's some intangible aspects, a kind of feeling, which you can't turn into logical rules and crystallise on guidelines. Having 42 bullet points that you have to go through in order to achieve X is not something that works in this case. I've heard many dissenting opinions, but I seriously don't agree with this way of doing things. It's my personal opinion. I've started writing basic icon guidelines to help newcomers.

Oxygen could have better documentation, but it's more about having good designers. Every time I have a designer asking for the rules, I tell them to look at the icons. If, after analyzing the icons from any theme, you still have doubts about their graphical and aesthetic rules, you probably shouldn't be working on this. Honestly, it's a language. If it's well written, one should be able to clearly interpret and identify the meaning just by reading it. Something along the lines of "Oh, they're using references to this and that. And I think I get where they're trying to go here." If you need a manual for a language in order to be able to write it, then something failed during the process, I'd say.

*Now, we might be basing this on a historical inaccuracy here, but we're led to believe that KDE pioneered the glossy interface look, with polished looks, clean lines and shiny surfaces. The same approach that has now been made popular by Apple on its recent user interfaces.*







Yes—a great designer, Everaldo Coelho, is to blame for the glossy style. He worked on this theme, Crystal, which is very well known and heavily used on a number of web sites worldwide. It was one of the first Free themes made by a designer, with a rather high quality standard considering the tools available at the time.

The Crystal theme was of very high quality, indeed. But it's the result of what Everaldo is as a designer, a specific kind of stubborn designer with a distinct style. It is glossy, playful, colorful, fun. Its visual style eventually became associated with KDE.

*What we were trying to get at with the previous question was, does the Oxygen team see themselves as trendsetters? Do you think that you're creating a norm, a set of unwritten rules of taste that would motivate others?*

Honestly, with Oxygen I'm aiming to influence my community to get better. I think that there are much more interesting things out there than Oxygen, and I'm not just talking about the desktop. I think that there's incredibly interesting stuff made for the web—which by the way, I don't approve of as a platform, we're sacrificing our future freedom by moving everything to the cloud—but design-wise, there's very exciting things being done for the web today. I try to translate some of those new

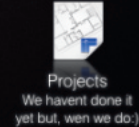
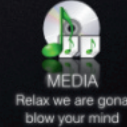
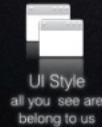
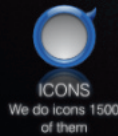
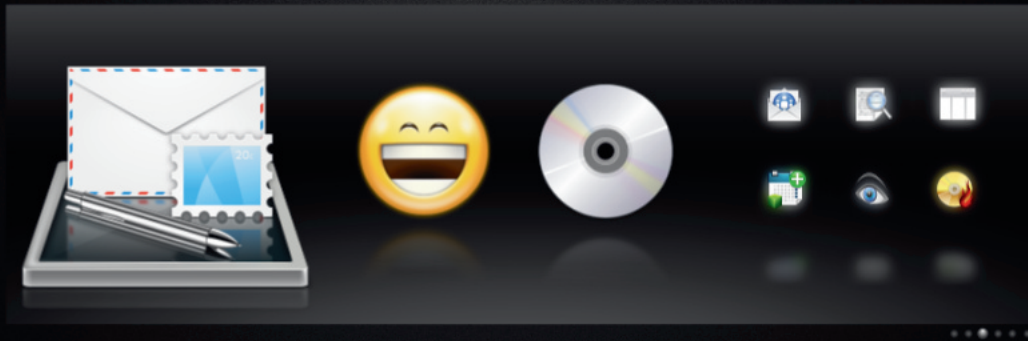
principles to the desktop, and in doing so try to influence design perception inside the community.

Design is learned, an acquired taste. Just like enjoying wine, or cooking in general. You might go your whole life just eating french fries and burgers; one day, you try a fish course, and the next day you come back. Maybe you'll go on to like fish. Then you try a good wine, and you gradually become able to appreciate wine. I try to influence the community to become aware of these little things.

*Icons are pieces created with the purpose of being used and re-used in different contexts. In the case of Oxygen this becomes more evident because of the Free licensing you employ. Have you ever been surprised by a particular use of the Oxygen icons?*

Oh, many surprises. “A Bola” [a top-selling daily Portuguese sports newspaper] used my icons. Any mention to the license or even attribution is nowhere to be seen. I've stopped worrying about licensing issues.

The license we use is the LGPL. It's not the perfect license for icons. We could have used Creative Commons, but the most permissive CC license is very similar to LGPL. With it, you only have to make sure proper attribution is made; other than that, the icons belong to whoever wants to use them.



*In corporate settings, one can usually find a schism between designers and developers or engineers. In Oxygen's case, does this kind of tension occur?*

Such tensions are nowhere to be found. I'm actually lucky—I'm an engineer. I studied engineering. And I find this is one of the reasons why Oxygen solved many of the issues that can plague other open source projects. I can speak both languages.

I come from a specific background: I'm a civil engineer. Civil engineering implies a crossover between architecture and engineering. My sister is an architect, so I know the battlefield well when it comes to the problems of both theorizing and implementing. The person who theorizes—often the designer—should be comfortable with implementation details, but very often that's not the case. A good designer should incorporate the engineer and the artist, but most of the time the artist wins. I keep trying to make sure I'm in touch with both perspectives.

There was a particular issue in Oxygen regarding the styling of window shadows. Only someone well aware of design and implementation issues could tackle the problem of how to properly anti-alias window corners and make it look good. I knew the implementation pitfalls, knew how the tech worked,

Code can be elegant, user experience can be elegant.

went to the developers and presented a different solution that could elegantly solve the problem. It is very important for the designer to be aware of the technical limitations. However, it's very important for the designer to not be aware of the technical limitations.

That's why coordination is important. I like having designers in the Oxygen project who work with absolute freedom, pure artists. The kind of people who come to me with completely nonsensical ideas and make me say "You're an idiot, this is impossible." But it's very important that they keep pushing me in that direction so that I can go "This is impossible but hey, maybe we can do half of it." Then I go to the developer and he'll say "This is impossible because of this, this and that," and I can suggest "But this and this could be done in this particular way," to which he'll reply "Maybe we can do half of it." This way, things progress according to the artist's vision and the developer's understanding.





## Next steps

Libre Graphics magazine was born from the community and exists for the community.

Issue 1.3 will be dedicated to the subject of collaboration, under the "Collaboration, Collaboratively" tagline. We want to feature your work and your insights on this subject, so why not let us know about what you've been brewing? If you're involved in an event related to Libre Graphics, we'd love to know so that we can feature it in our Upcoming Events section.

Ever considered placing an ad in Libre Graphics magazine? Contact us, and keep an eye on our blog at <http://libregraphicsmag.com/blog>.

We also run pro bono ads at no cost for F/LOSS projects – contact us for details.









# Papercut

*Allison Moore*

Papercut is a Blender-based video game in the style of traditional side-scroller roleplaying games. There is a central character and a landscape to traverse. You are a lumberjack. You must cut down trees with a chainsaw. The game world is designed combining hand drawn illustrations with cut-out scanned textures.

There are two characters to choose from: a Lumber Man and a Lumber Lady. The main character must deforest the landscape. Your only tool is a chainsaw. As you cut down trees you collect points. There is a time limit to each level, and if you meet your tally, you advance to the next level.

Papercut creates a main character with a questionable morality. In traditional gameplay, the main character is definitively good whilst any character blocking the path is definitively bad. Geographical obstacles, woodland creatures and hippies block your path.

The virtual world combines exaggerated representations of the existing world with elements interpreted from my imagination.

I played a lot of games in the 80s and early 90s, so I like vintage/retro games and this is the aesthetic that influences me most. It was hard to wrap my mind around a 3D world, so I decided to make it 2 ½D. I use 2D references of vintage games incorporated in the 3D landscape. The final result is like a paper puppet set, my 2D characters like puppets navigating through a diorama-style set built in 3D. Trees fall like leaves of paper.

<http://www.looper.ca>



## What Revolution?

*Antonio Roberts*

What Revolution? is the first in a series of images challenging the ideas of celebrity and idols. The 1960 photograph of Che Guevara by Alberto Korda has been endlessly mutated, transformed, and morphed. It can be found advertising anything from belts and "hip and cool" t-shirts to health insurance. It is tacked onto political movements without much consideration of the history behind it. One has to ask if his image is still the symbol for change and revolution that it was fifty years ago, when it was furiously distributed throughout Europe by Jim Fitzpatrick in protest of the conditions of Guevara's murder.

The vector image of Che was glitched using a C script written by Garry Bulmer. The script randomises the position and other values of the nodes in the file. The background is a random image found on the Internet tagged with "Revolution," which was then glitched many times using ucnv's Glitchpng Ruby script. To get the sharp colours, I reduced the image from 8 bits to 1 bit using ImageMagick. All of the separate elements were then recomposed in Inkscape.

<http://www.helloatfood.com>





# Making your workflow work for you

*Seth Kenlon*

Everyone works differently, regardless of the task. Every artist has an individual style for getting things done quickly, efficiently, and in such a way that the effort required doesn't ruin the inspiration driving the work in the first place. Whether the motivation is a client or a personal passion, the process that an artist uses to finish the job is generally known by the term "workflow."

Even though everyone tends to be unique in the way they work, much proprietary software enforces a very specific workflow. In fact, deviation from that workflow is discouraged. The nature of the business demands that a proprietary software vendor ensures its product is all an artist needs. In other words, proprietary software, in order to make the greatest sales, seeks to be a monopoly.

Many artists take this for granted because those proprietary software packages are what they learned in school or at work. Some literally do not realize there is any other option. However, on almost any platform there are a host of F/LOSS tools which can enable artists to take control of how they want to work, and what works best for them.

How do you know if your workflow needs refinement? There are a few good indications:

**—If you find yourself using applications to do things that they (technically) can do but were clearly not designed to do, you might find it far more efficient to seek out the right tool for the right job.**

A characteristic of Free/Libre Open Source Software applications is that very few attempt to be everything to everyone. In fact, a basic tenet of F/LOSS, handed down from Unix, an historically easy operating system for which to create custom applications, is that of modularity. This idea is commonly expressed in the mantra "do one thing and do it well."

This means that F/LOSS tends to focus on individual tasks that can then be strung together. Does this sound like the great beginning of a formidable personalized workflow? It is.

Proprietary graphics applications lull users into believing they can do everything, but in reality they do one general set of tasks well and offer heavily pared-down tools for everything else. For instance, a bitmap graphic manipulation software might offer some basic page layout and vector drawing features. The theory, presumably, is that if a user only needs a few basic vector illustration or page layout tools, then those tools will be available. In practise, however, artists become so familiar with this monolithic application that they start using it for everything, cobbling and hacking together entire pieces with one wrong tool. While this does get work produced (a result that is always difficult to argue with), it often does so after far too much unnecessary pain, too many workarounds and speed bumps.

F/LOSS software encourages people to use the tools that are designed for the job. In so doing, the artist is freed to use anything he wants to use. Whatever application an artist finds easiest and most suitable for his art, he is free to use, from the most complex vector drawing program to the most basic paint program. Since F/LOSS is dedicated to interoperability, there aren't as many format problems; the work done in one application can be imported and modified in another. No separate, fancy, confusing bridge application necessary.

In a way, this means an artist might need to learn more applications. Most people find that while learning F/LOSS applications, there is enough internal logic to that application that the learning curve is modest. And certainly the fact that the application is designed to do the task being done helps a lot. There's no hacking around the fact that an application doesn't do the normal things it should do.

**—If you find yourself doing repetitious tasks by hand, again and again throughout a project, then there may be something designed to take that burden from you.**

This idea springs up in many different places within the F/LOSS world. Since none of the code in F/LOSS applications is hidden, scripting these applications is quite simple if you have even

modest scripting skills. However, some people have no scripting skills and don't want them - and for them, there is the Internet. Simple searches uncover myriad scripts to do repetitious tasks with command line applications.

The Image Magick suite, for example, which itself consists of a number of command line tools is one of those applications that no graphic artist should ever be without -- regardless of preferred os.

Now, it often puzzles people to think of graphic work being done from a command line, but it is amazingly useful and flexible. Graphic artists using propriety software might spend an afternoon opening a graphic in a big bulky graphics application just to convert its colourspace. Artists using Image Magick, on the other hand, can issue a simple line command:

```
bash$ convert file.tif -colorspace cmyk fileCMYK.tif
```

and have the job finished in moments. Script that and hundreds of files can be done while you're onto the next task.

**—If you find yourself consistently being stopped or drastically slowed by the same set of small "quirky" problems on every project you do, then you may need a specialized tool to avert that issue.**

Proprietary software typically has two answers to your problems: don't do it, or spend more money to be able to do it. This might apply to a specific file format you want to use, or an effect you want to achieve, or a way of working.

The F/LOSS world is set up differently, because there's no agenda to up-sell you on improved versions of the software and no need to limit what you can do. New tools are being developed every day to meet the demands of artists, and these tools are all free to

download and use. All Free/Libre Open Source Software, by the very nature of having free source code, is extensible and expandable. As new tools are released, they can be integrated into the applications you use.

Do one thing and do it well.

## DESIGNING F/LOSS WORKFLOWS

Whether or not you have an existing workflow based on proprietary software, working on F/LOSS for multimedia is most efficient with a little planning. Without stepping back and looking at the whole project, it's quite likely that you'll reach a critical point and realize you're not prepared for the next step - or even aware of what your next step should be.

The first step in designing your workflow is to identify what raw materials you'll need for production. If you're doing a digital painting, you might want to go out and find brushes and establish a custom color palette. If your work is graphic manipulation, then you might want to find useful textures, patterns, brushes, fonts, stock images, and so on. If your work is a magazine then you'll need articles, images, and fonts.

Having this kind of kit before starting will make the project flow more smoothly during the creation phase. Some proprietary software comes pre-packaged with gigabytes and gigabytes of royalty-free stock content which, among other things, takes up quite a bit of room on your hard drive, mostly will never be used by you, and is stylistically quite identifiable as the corporate, royalty-free stock content that it is. F/LOSS does not ship with this, so you'll have to find your own, but with Creative Commons being the force that it is, this is a trivial matter and one that, in the end, produces a more unique work than the alternative.

A good place to start is the so-called "Great Multimedia Sprint" from <http://slackermedia.info/sprints>. This is a nearly 2GB



collection of Creative Commons licensed content meant to be used as raw materials. More sprints are scheduled for the future, so more content will be available soon.

The next step is to determine what software tasks and compatibility your project requires. If you're working on a magazine, for instance, then you're sure to need both bitmap and vector manipulation programs, a host of fonts and some way to organize and track them, as well as a good layout program. If you're not already familiar with the tools that F/LOSS has to offer for these tasks, investigate and try some of them to determine which one you prefer and which one will actually do the tasks you want to accomplish.

Since you'll potentially be able to break up tasks into smaller applications, you might also want to consider how multiple computers might be put to work for your project. In the studio where I work, an old G4 running Debian Linux has been repurposed with the solitary job of converting music files from one format to another while a G5 converts still frames to video. They aren't the fastest computers, they don't have so much as a monitor connected to them, but they can run these dedicated tasks all day and all night, so that the materials are available when needed.

In the end you should be able to trace in a flow-chart how the work will get done. A graphic might first be converted and scaled with one application, manipulated and customized in another, and laid out in the final work in yet another. Exporting should, as often as possible, be done at maximum quality to result in a "gold master," which can then be modified and compressed into easily-distributed versions. Again, this can easily be done with dedicated line commands that specialize in compressing (Image Magick for graphics including PDFs, pdftk for PDF modification, FFmpeg for video, and so on).

## THE WAY FREEDOM WORKS

The bottom line is that the workflow in F/LOSS is not pre-determined for the artist. While this places the burden of designing a workflow upon the artist, it also frees the artist from a locked-down, inefficient art creation process, and opens a world of possibilities and creativity. And that's something worth working for.

Proprietary software typically has two answers to your problems: don't do it, or spend more money to be able to do it. This might apply to a specific file format you want to use, or an effect you want to achieve, or a way of working.

# On being a Unicorn: the case for user-involvement in Free/Libre Open Source Software

As an artist or designer (or both), you use a range of tools in your everyday work. Even though it's not something you think about, you may be contributing to the growth of these tools without realising. Every time an application crashes and you hit the button, giving permission for it to report, you're contributing a little something. But, if you're interested, there's more. And there's more you can get out of it than just reliable software.

Let's assume that you think of yourself exclusively as a user of design tools. In the same way you don't offer suggestions to the company manufacturing your pencils, you don't consider letting the people making your software know what you think.

And you know what? You're not alone. Not many designers let the people behind their favourite tools know what they think. It's not common for designers and artists to make their voices heard, but it is useful.

Because, you see, it works this way: if you use F/LOSS graphics software, standards and methods in your art or design practice, chances are good you have something interesting to talk to developers about. What you have to talk to them about is the way you use their software. And they want to hear it.

They want the gory details about which specific tools and commands you use, what problems you have, why you use the things you use in your workflow.

There are lots of different opportunities to have these conversations. The one we're going to suggest right now is Libre Graphics Meeting, an annual meet-up of developers and users. The one thing tying everyone together is an interest in F/LOSS graphics. We want to let you know, as a little service to you, the designer or artist using extensively or even just dabbling with F/LOSS graphics software, standards and tools, that it's coming up.

We want to let you know because, as a designer or artist using F/LOSS, you're a bit of a unicorn. By which we mean that you're something kind of rare and beautiful, not often seen by F/LOSS developers, and perhaps even misunderstood. And as something a little out of the ordinary, you're interesting. You've got lots to contribute, so consider joining in with the spirit of the community a little and bringing your own expertise to the table.

The sixth annual Libre Graphics Meeting is taking place May 10-13 2011 in Montreal. More information is available at [libregraphicsmeeting.org](http://libregraphicsmeeting.org)

# Talking about our tools

## Call for submissions

Let's talk about tools for a moment. We, as humans, distinguish ourselves from other animals by talking about our ability to make and use tools. We make tools, we use tools, we are tools, all at different times and in different amounts.

Tools can be physical things used to manipulate equally physical things. At the same time, they can be digital things, used to shift bits. We can love them or hate them. The one thing we can't manage is to escape them.

As we define what they do, so too do they define what we do. In the shape of a paint brush, the kink of a bezier curve, the change a gaussian filter exerts over an image, they make our work what it is. We are our tools and our tools are us. So let's talk about tools, in the best way we know how, graphically.

Libre Graphics Meeting, Libre Graphics magazine and Mardigrafe are co-sponsoring a juried exhibition of F/LOSS graphics work on the subject of tools. Break out your own F/LOSS graphics tools and design a poster (24"x34") detailing your perception or ideas about tools.

All submissions will be included in an online gallery, presented in conjunction with Libre Graphics meeting. In addition, a jury of designers, thinkers and doers will meet in May. They'll pick 15 posters to be printed by Mardigrafe and displayed during Libre Graphics Meeting in Montreal. The editors of Libre Graphics magazine will pick a further eight to be featured in the showcase section of an upcoming issue.

So get thinking about your tools, what they mean to you and what you mean to them. Then, get designing.

More details and how to submit at [HTTP://LIBREGRAPHICSMAG.COM/TOOLS](http://libregraphicsmag.com/tools)



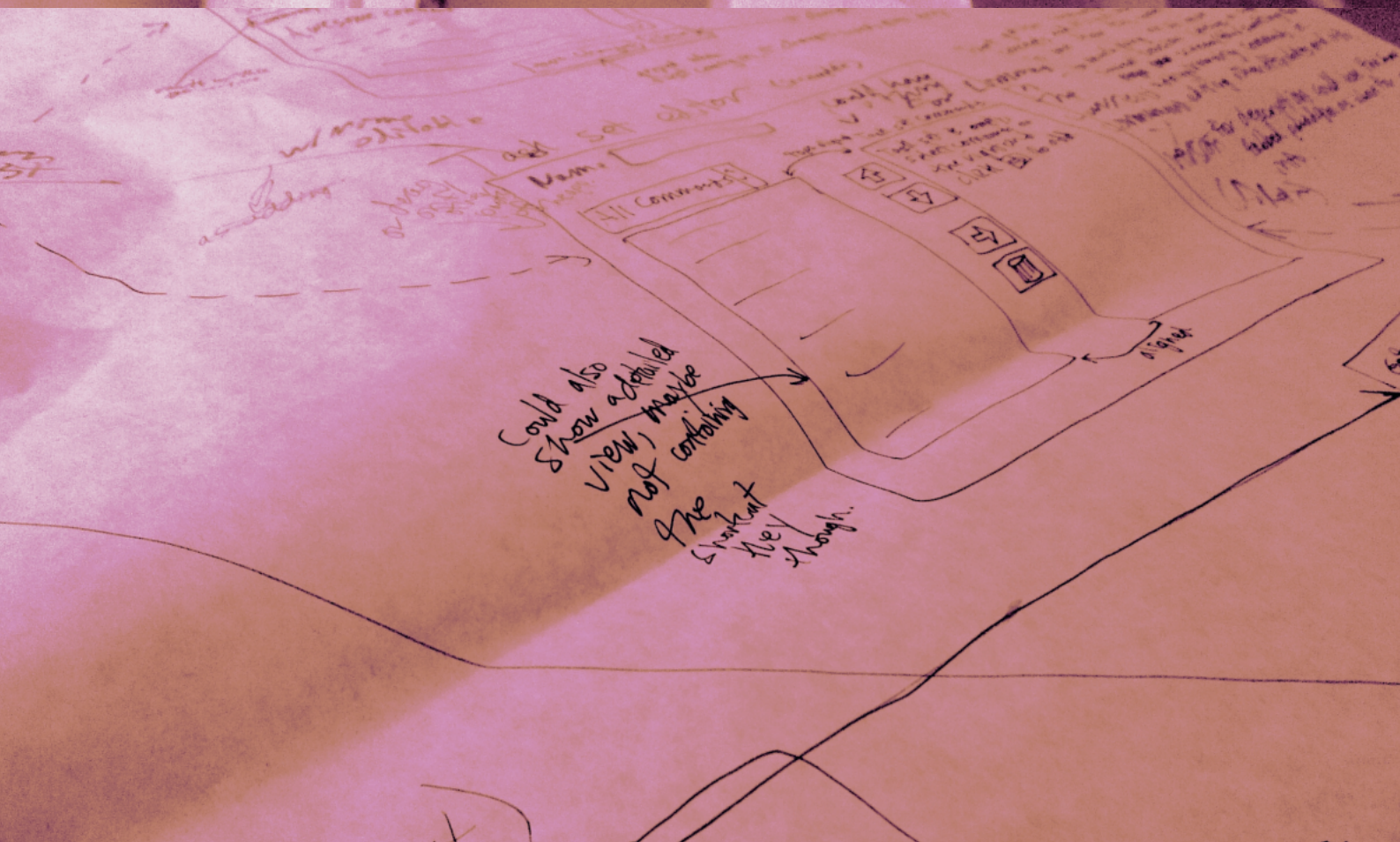


**LIBRE GRAPHICS  
MEETING 2011  
MONTRÉAL  
10-13 MAY**

**JOIN  
SPEAK  
CONTRIBUTE**

**[HTTP://LIBREGRAPHICSMEEING.ORG/2011](http://libregraphicsmeeting.org/2011)**







# AdaptableGIMP: user interfaces for users

*ginger coons*

In 2007, Michael Terry and other members of the University of Waterloo HCI lab set out to learn just what GIMP users actually do. To achieve that lofty goal, they created something called *ingimp*, a variation of GIMP which tracked feature use. Four years later, they have an answer, in a way.

The answer, broadly, is what you might expect. It turns out that different users of GIMP do different things. Ben Lafreniere, a doctoral candidate in Terry's HCI lab, has combed through the data and come up with a more nuanced answer. Usage tends to be focused on small sets of tools, using only a tiny percentage of the actual capabilities of the program. The members of the lab refer to these groups as "corners."

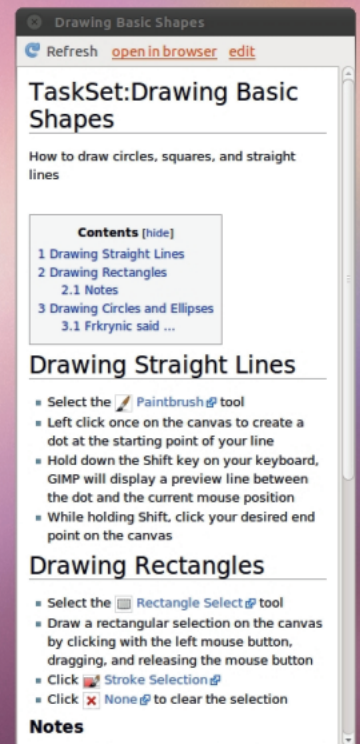
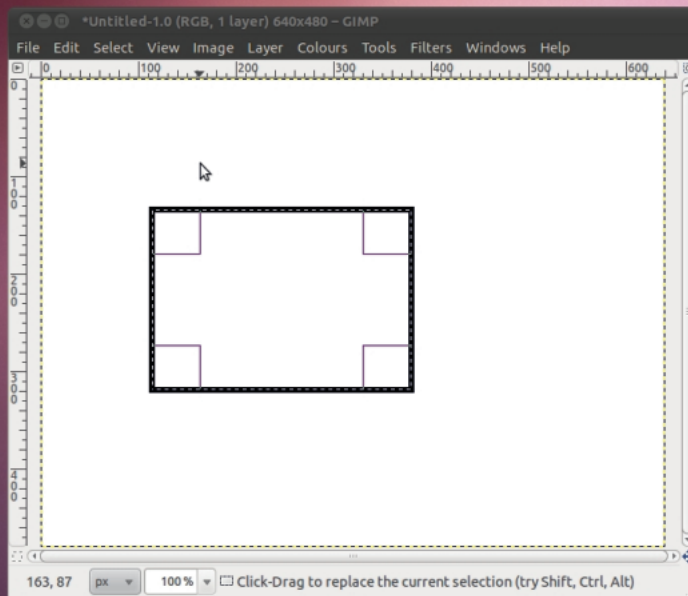
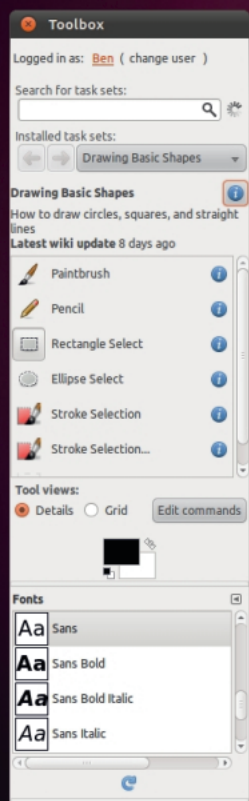
According to Lafreniere, "not only do people use just a small corner of the functionality in the system, but they tend to use fairly distinct corners." Which means that there's no one-size-fits-all answer. With different users making use of small, distinct sets of tools, no one easy interface tweak will suit everyone and make GIMP universally more usable.

But never fear. There's another, far more exciting option. That option comes in the form of AdaptableGIMP. The premise of AdaptableGIMP, another project from the HCI lab, is that not only should users be able to customise the interface of their software, they should be able to share those customisations with others. Or, as Lafreniere puts it, crowdsourcing "the creation of customisations to the entire user community."

To do this, AdaptableGIMP relies on a modified version of MediaWiki. Task sets—customised collections of GIMP commands tailored to a specific use—are stored in a central repository, tied to wiki pages which are capable of both describing and controlling the mix of features in each set.

"It's like an infinite set of overlapping Microsoft ribbons. They try to do the same thing, they're trying to group functionality. But we're saying that it doesn't need to be the six that are defined by the people making the application, there can be a million. You can't only have the paintbrush in one. The paintbrush can be in 500,000 of them."—Filip Krynicki



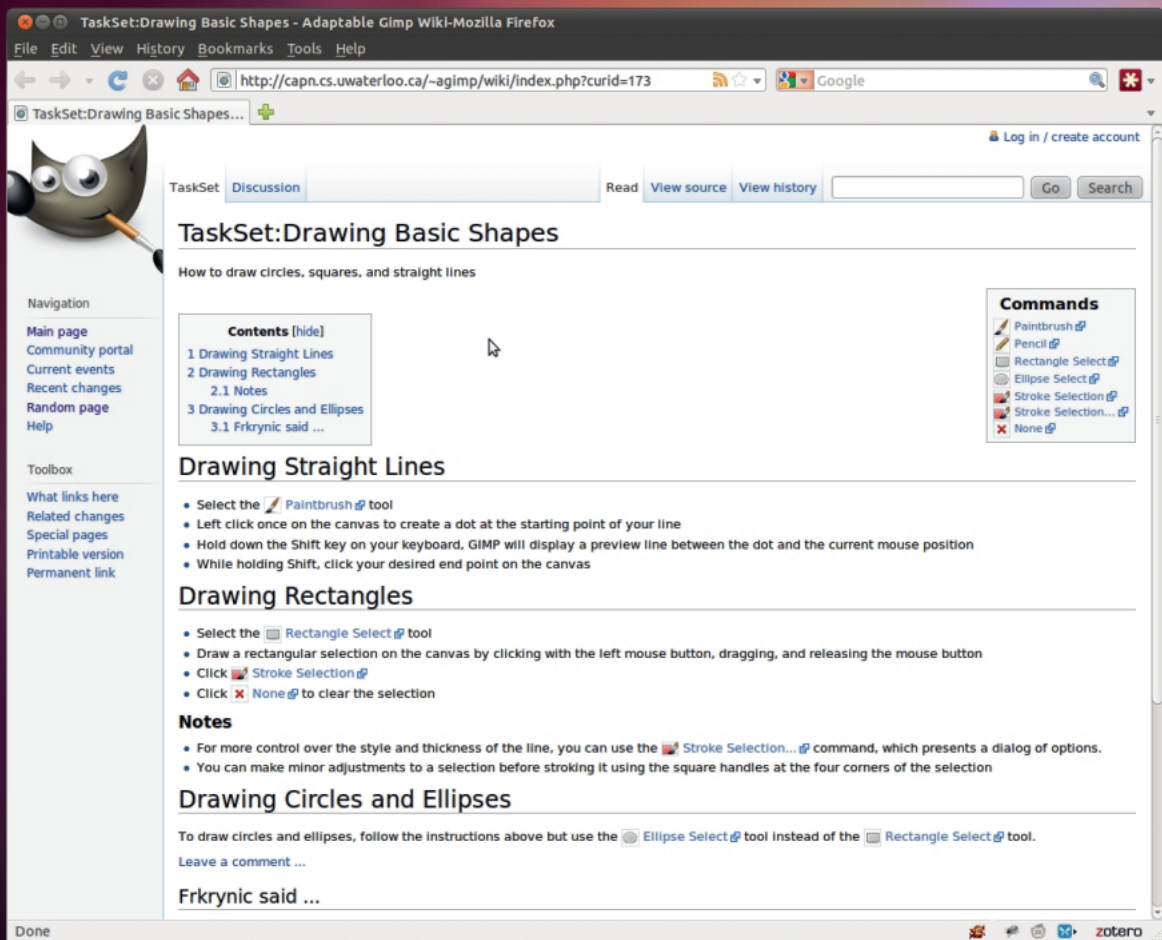


The beauty of this, according to Lefreniere, is that “when anybody creates a customisation to the interface, it's immediately there, available to all the users of the application.” This provides all users with a collection of available task sets, just waiting to be used. Says Lafreniere, the intention is that a user “can sit down at the interface, type a few keywords describing what they want, searching things made by the community, select one, and then immediately have it.”

And who will build those task sets? According to Terry, there's already tangible evidence that some users are more than willing to create documentation, tutorials and other resources. “What we're doing,” he says, “is bringing that practice more directly into the interface.”

This community approach to building and documenting task sets has an added benefit: it makes the efforts of one person useful and valuable to all other users of the software. This means that different types of users can work to their own strengths and preferences, while benefiting from the preferences of others.

“What we're grafting onto the existing interface paradigm, is this task-centric view of computing where you say 'This is what I want to do' and the interface modifies itself to accomodate that particular task.”—Michael Terry



“People are hesitant to stop the current task that they're working on to create a customisation” says Lafreniere. To Filip Krynicky, one of the HCI Lab's co-op students, this is one of the major benefits of the AdaptableGIMP approach. According to Krynicky, “in most interfaces where someone can make a customisation, that's where it stops.” But in the case of AdaptableGIMP, if even one percent of users actually create customisations, all users benefit.

Users creating customisations may see some added incentive, too. Terry suggests that, given AdaptableGIMP's ability to collect usage data, task sets could well come along with information about how many users they've been installed by, how active their development is and even how recently they've been used. To Terry, this gives creators of task sets “some sense of feedback of the utility of the task set.”

## A NEW APPROACH TO INTERFACE DESIGN

Members of the HCI Lab see current interface design as something hierarchical and designed more to contain functionality than to help users accomplish their tasks.

According to Terry, one of the goals of AdaptableGIMP is to help users define their own workflows. This approach contrasts strongly with hierarchical interfaces, which he says are “designed in reaction to the large number of commands that are available and not designed around how people actually sit down and want to use the tool for a particular task.”

This does not mean changing the entire functioning of the program or reinventing the wheel. To Terry, it's a case of “grafting onto the existing interface paradigm,” adding in a “task-centric view of computing where you say ‘This is what I want to do’ and the interface modifies itself to accommodate that particular task.”

Krynicky puts it into contrast with existing tactics: “It's like an infinite set of overlapping Microsoft ribbons. They try to do the same thing, they're trying to group functionality. But we're saying that it doesn't need to be the six that are defined by the people making the application, there can be a million. You can't only have the paintbrush in one. The paintbrush can be in 500,000 of them.”





The future of AdaptableGIMP looks, at very least, exciting. Lafreniere suggests the possibilities presented by a built-in recommendation system, offering complementary task sets based on use patterns or even suggesting task sets which frame commands the user already knows, but to different ends. As Lafreniere puts it, “you know all these commands, you could do this task.”

Of course, it's not just GIMP standing to benefit from this work. Terry hopes to offer a core set of AdaptableGIMP components which would help developers of other software in implementing crowdsourced customisation themselves. Says Terry, “we hope that we can provide a tool set for them that they can plug in and start to use in their own application.”

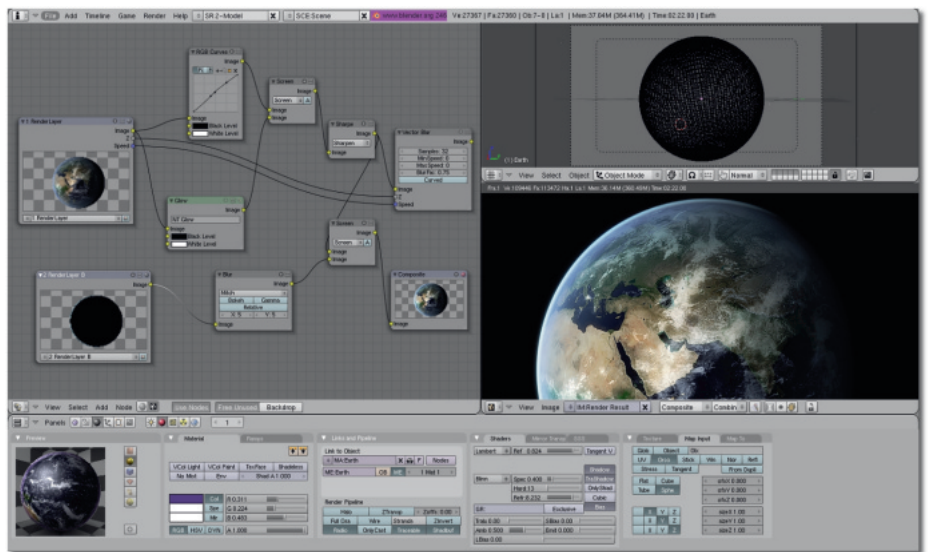
AdaptableGIMP is available now, for users who don't mind compiling from source. Get it at [HTTP://ADAPTABLEGIMP.ORG](http://ADAPTABLEGIMP.ORG).



# Resource list 1.2

## BLENDER

A powerful FLOSS 3D animation application for GNU/Linux, Mac OS X and Microsoft Windows.



## GIMP

A raster based image editor for GNU/Linux, Mac OS X and Microsoft Windows.



## IMAGEMAGICK

A raster image editing, creation and conversion suite for GNU/Linux, Mac OS X, Microsoft Windows and iPhone, among others.

```
# Create a montage from a folder containing various png images
montage -geometry 400x300+0+0 *.png icon-montage.png

# Scale all jpeg images in a folder to a width of 640px
for img in *.jpg ; do convert $img -scale 640 $img; done;

# Rotate a batch of jpeg images 90° and convert them to png
for img in *.jpg ; do convert $img -rotate 90 ${img/.jpg/.png} ; done
```

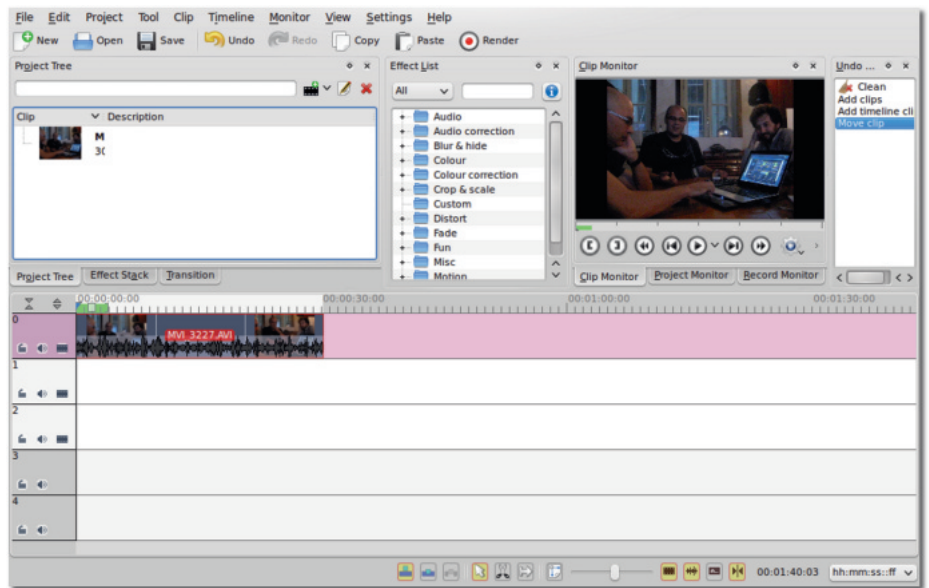
## INKSCAPE

A vector graphics editor for GNU/Linux, Mac OS X and Microsoft Windows.



## KDENLIVE

A video editor for GNU/Linux, Mac OS X, Microsoft Windows and FreeBSD.



## MYPAIN

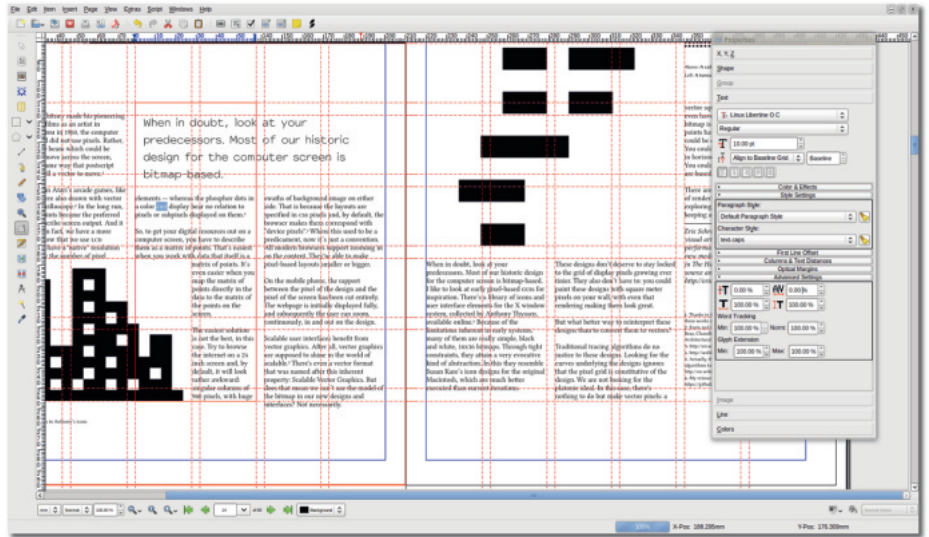
Graphics application focused on natural media simulation. Available for GNU/Linux, Mac OS X and Microsoft Windows.





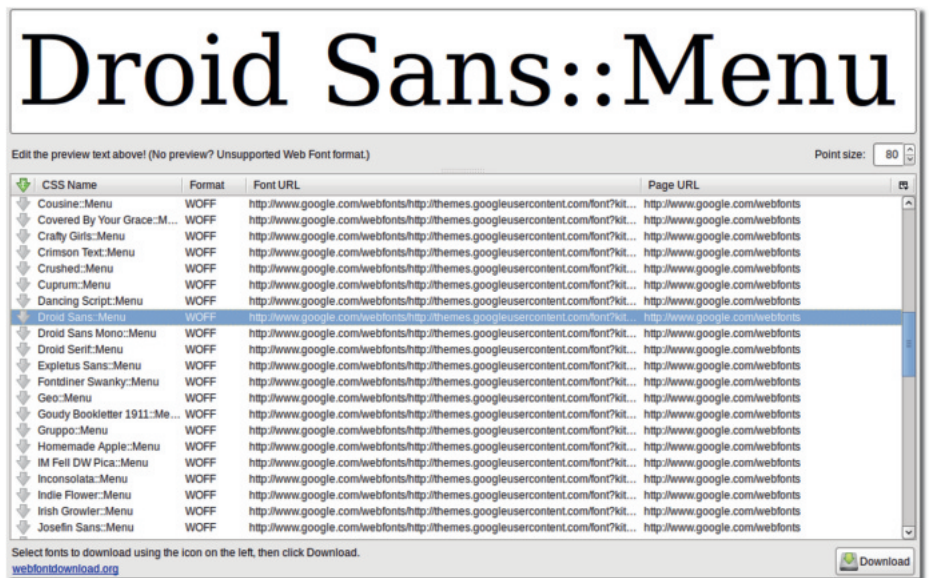
## SCRIBUS

A desktop publishing program for GNU/Linux, Mac os x and Microsoft Windows.



## WEB FONT DOWNLOADER

An extension for Firefox, allowing downloads of embedded web fonts.



# Glossary 1.2

**Alchemy :**

A F/LOSS canvas drawing program, meant to encourage play and exploration. Available for GNU/Linux, Mac OS X and Windows.

**Audacity :**

A F/LOSS sound editing application for GNU/Linux, Mac OS X and Microsoft Windows.

**Blender :**

A powerful 3D animation application for GNU/Linux, Mac OS X and Microsoft Windows.

**command line :**

A text-based interface for controlling a computer.

**desktop environment :**

A collection of tools and interface elements which style the visual and functional aspects of an operating system in a certain way.

**Digital Rights Management (DRM) :**

Technologies (of whatever sort) which prevent users from making certain uses of the larger technologies to which the DRM is applied.

**distro/distribution :**

A specific configuration of GNU/Linux, often designed with a particular purpose in mind.

**Fedora :**

A popular distribution of GNU/Linux, produced by Red Hat, Inc.

**flavour :**

Similar in meaning to distro/distribution, but more general. Simply means a specific version (normally of GNU/Linux).

**Free :**

As in freedom, or often, that which is or is of Free Software.

**Free Culture :**

A general term for activities and artistic works which fall under a similar ideological banner to the Free Software movement.

**freedesktop.org :**

A F/LOSS project which focuses on creating interoperable tools for GNU/Linux and other Unix-type systems.

**Free/Libre Open Source Software (F/LOSS) :**

Software which has a viewable, modifiable source and a permissive license (such as the GNU GPL). It can be modified and redistributed.

**GIMP :**

A raster based image editor for GNU/Linux, Mac OS X and Microsoft Windows.

**Git :**

A popular version control system, originally created to manage development of the Linux kernel.

**GNOME :**

A popular desktop environment for GNU/Linux.

**GNU General Public License (GPL) :**

A license originally intended for use with software, but now used for other applications. Made famous the principle of Copyleft, requiring those using GPL licensed work to license derivatives similarly.

**implement :**

The act of integrating a feature or standard into a piece of software, rendering that software able to (for example) perform a task or use a specific file format.

**Internet Relay Chat (IRC) :**

A popular form of internet-based real-time chat. Has a long history of use and is still popular among groups of developers and users.

**KDE :**

A community project which produces various FLOSS applications, best known as a popular desktop environment for GNU/Linux.

**Libre :**

A less ambiguous adaptation of the word Free. Implies liberty of use, modification and distribution.

**mailing list :**

An email-based forum through which subscribers may receive announcements, view or participate in discussion.

**open standards :**

A standard which is available for viewing and implementation by any party, often at no monetary cost.

**Oxygen :**

A project meant to develop a coherent and attractive visual identity for KDE.

**proprietary :**

A piece of software or other work which does not make available its source, which is not allowed or intended to be modified or redistributed without permission.

**Scalable Vector Graphics (SVG) :**

An open standard for vector graphics, developed by the W3C.

**SIL Open Font License (OFL) :**

A license intended for use with fonts and font related software. Dictates terms which allow modification and redistribution of fonts.

**source code :**

The human readable code on which software is based. Software distributed with its source code can be modified far more easily than software distributed without.

**terminal :**

A program which allows users to perform actions on the command line.

**Ubuntu :**

A particularly popular distribution of GNU/Linux, produced by Canonical Ltd.

**version control :**

Activities which have the effect or intent of distinguishing different versions of a work or body of work from one another.

**Version Control System (VCS) :**

An application/collection of tools designed to facilitate version control. Tracks changes to files and allows a group of collaborators to share their changes as they are made.

**W3C :**

The organization responsible for setting web standards, such as HTML5 and SVG.





