Libre Graphics Meeting

Adobe, Corel, Quark *et al* had better look over their shoulder – free software is catching up with them. Nick Veitch went to the first Libre Graphics Meeting, a landmark event for open source.

Linux. For most people the word conjures up images of the desperately high-tech world of the internet. Visions of blade servers, clusters, *Apache* and streams of code immediately spring to mind. And penguins. Back in the late nineties it became almost inevitable that free software was going to dominate the technologies driving the internet – not just Linux, but the whole technology stack that was driving the net forward relied on open source and open standards.

But Linux is far from being a platform fit for just a single purpose, and this was clearly illustrated at the inaugural Libre Graphics meeting, hosted by the Ecole Supérieur Chimie Physique Electronique (CPE) in Lyon, southern France.

The conference was organised by Dave Neary in his guise as a *Gimp* developer. What he initially thought would just be a meetup for other *Gimp* coders soon grew into an event that embraced all sorts of other graphic tools – developers from other projects seized the opportunity to commune with their peers, exchange ideas and discuss common problems.

For three days the venue was packed with programmers, users and interested bystanders. Presentations in the main theatre were well attended and featured new technologies, project updates and tutorial-like sessions on how best to contribute.

As well as project updates, there were useful tutorials on how best to contribute.

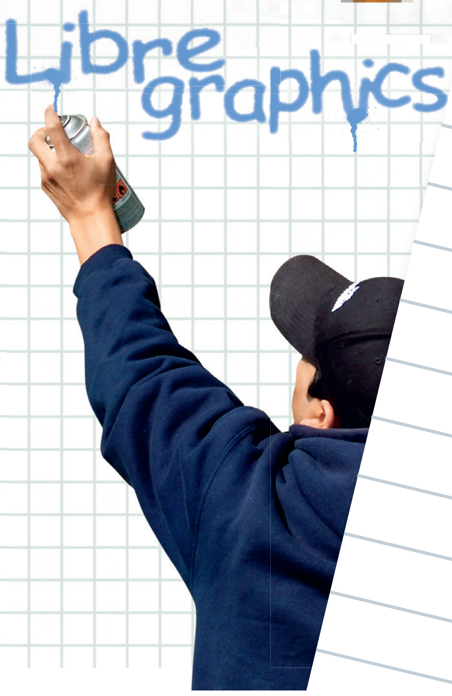
The breakthrough

While all this was very useful, it was the interaction of attendees in the breaks between sessions that probably moved things forward the most. In corridors, on smoking breaks outside, in the lunch venues, and of course at dinner later, the talk was of deeper co-operation, of pooling knowledge and resources to tackle common problems, and friendly advice being exchanged on everything from which libraries to use to best practices.

This wasn't just a nerdy geekfest though – many of the projects represented are on the cusp of being very serious graphics tools indeed. And most of the major graphics tools represented are cross-platform – *Inkscape*, *Gimp*, *Scribus*, *Blender*...

Over the next few pages we're profiling some of the applications driving the use of open source as a choice for professional graphics and design. The established market belongs to the likes of Adobe, but there is very definitely a new kid on the block. And just maybe, someday in the not too distant future, this meeting will be seen as the turning point when open source graphics came of age...

Developers and users turned up from all over.



```
LXF Libre Graphics Meeting interview
Date: 17 March 2006, 11.49 am
Location: In front of CPE building
Subject:Dave Neary,
conference organiser
```

Linux Format: So: why did you organise the LGM?

Dave Neary: Well there are other conferences, but they don't have the same focus. Things like Fosdem, which is very general, or Akademy. But *Gimp*, and all graphics applications, are a little outside that – *Blender* doesn't use *GTK* or *Qt*; *Gimp* is a Gnome application, but it isn't really tied to the project.

People use *Gimp* on KDE, people use *Scribus* on Gnome. The applications represented here are cross-platform – we have more of a need for a specific Linux conference. Also, graphics is a bit odd: we have the orange sunglasses people to think about. The people we are interested in, our users, are graphic designers and art directors – they are outside of the sphere of free software. We need to do something particular to reach out to that type of person.

LXF: Is this an event for persuading people that they should be using open source tools?

DN: We've got two roles: the first is for the people who contribute to the projects to get together and figure out a common future – a coherent roadmap for all of these applications.

The second is to show off free software to a public that perhaps doesn't know about it yet, people who are working in graphics. We need to show that, [even though] maybe we aren't fulfilling all their needs straight away, we have applications that are capable, that are stable and that are full of promise. And that we have a community around those applications that is very welcoming. We need more artists involved in free software.

LXF: What are the barriers to people using free software for serious graphics?

DN: There are a few. There is a code of conduct, which is unwritten and can sometimes be a bit mysterious. That's a community barrier.

There are technology barriers. There are things that are covered by 'intellectual property.' That's a great catch-all that I hate to use but there are things covered by copyright and trademarks and patents. Things like the Pantone colour list, which is under copyright.

The third barrier is manpower – there are things that we just don't do. We're talking about artists – why are artists not using it more. There are things that the *Gimp* doesn't do that artists and film-makers and pre-press people need.

LXF: Some people think that cross-platform nature of these tools is a bad thing; that free software ought to be kept on a free platform.

DN: There are people that think that. I don't. I think that our goal is to have everyone using free software, including the OS. The question you have to ask is: "Does porting a GNU application to Windows damage the goal?" The way I see it is that if I am a Windows user, I don't want to change everything in one go. If someone gives me a CD with word processing, sound-editing software, graphics and web browsing, I can use them all on Windows and not worry about switching back if I don't like them. All of a sudden you've got someone whose software is free software, but they are running Windows. The only thing they have to do is change the operating system. You are helping to move people to the path of enlightenment.

NOTE If your workflow needs good colour management, go to www.littlecms.com. and check out the Little CMS project.

Scribus

Open source desktop publishing turns pro.

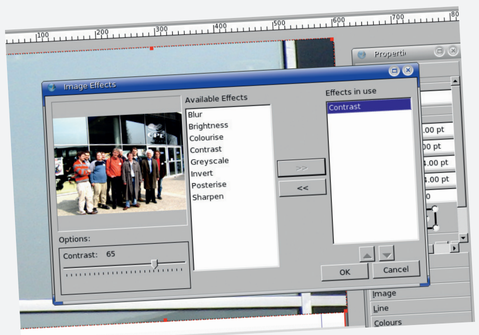
In many ways, *Scribus* is the runaway success story of open source graphics. Yes, *Gimp* is great, *Inkscape* has potential – but *Scribus* set itself the challenge of coming from nowhere to being a solid, professional DTP solution, and by traditional standards of software development, it's done that in next to no time.

In fact, the latest release of *Scribus* (1.3.3) offers features not available as standard in professional DTP software, such as the ability to generate barcodes completely within the software. It would be unthinkable for a commercial DTP product to come from zero lines of code to a reliable, professional tool in five years.

Printing types

As with all of the projects covered here, it seems that having an open source development model has led to some very serendipitous moments for *Scribus*. A story heard so many times is that the right person for a particular task just happened along when it needed doing.

Lucky or otherwise, the *Scribus* team have managed to surround themselves with some very talented and knowledgeable people in the world of print media.



Scribus 1.3.3 already includes some simple image effects, but future versions may use GEGL.

Scribus stared life as a project by self-taught German programmer Franz Schmid. His initial stroke of genius was to build a page layout program entirely around a PDF workflow. Even in 2001 there were good indications that PDF would become the *lingua franca* of printing, and that has proved to be the case. Almost all large publishing companies and printers now deal almost exclusively in PDFs, which are generated, planned and ouptut direct to plate at the print shop.

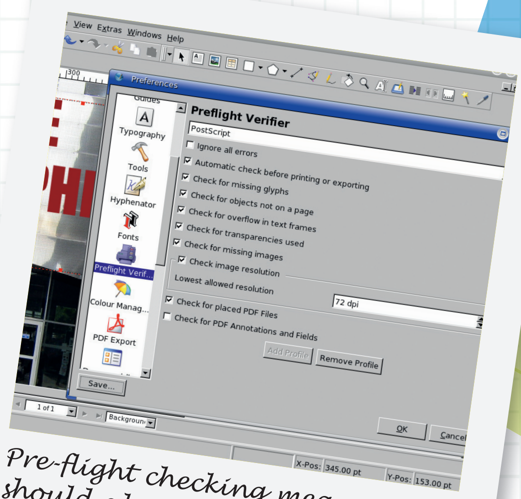
Schmid obviously knew and learned a lot about the PDF specs, because *Scribus* does a fantastic job of generating them – it even handles a lot of the interactive/presentation features that aren't used for print and practically no other software generates apart from the full version of *Adobe Reader*. And hopefully even *Linux Format* helped to popularize Scribus, as we included it in our HotPicks section (it is one of the few projects to have appeared there more than once, in *LXF36* and *LXF50*).

As the software has matured, though, Schmid's small project has become vastly more sophisticated and complex. This in itself has led to a lot of changes – much of the code has had to be rewritten, or at least moved. What made sense for a small project dealing with small documents doesn't work so well now that users are expecting to tap out 200-page books.

Sophistication

With the latest 1.3 series, *Scribus* users can now reap the benefits of spot colours, colour management, proper colour separations and more. In the short term, there is to be a new text-rendering system which will greatly increase the speed of redraws, as well as adding extra support for Unicode. This has benefits that stretch to being able to specify glyph subsets. A lot of people have been clamouring for a command line version of *Scribus* that can convert *Scribus* docs to PDF without needing the GUI, and amazingly, this isn't far off.

The overall feel is that, at least in this corner of the desktop, open source tools are moving on from being curiously useful to becoming professional quality tools. At Libre Graphics, the team were showing off *Le Tigre*, a *Scribus*-produced weekly French newspaper that prints over 12,000 copies. It isn't the first, and won't be the last.



Preflight Verifier

PostScript

- [ ] Ignore all errors
- [✓] Automatic check before printing or exporting
- [✓] Check for missing glyphs
- [✓] Check for objects not on a page
- [✓] Check for overflow in text frames
- [✓] Check for transparencies used
- [✓] Check for missing images
- [✓] Check image resolution

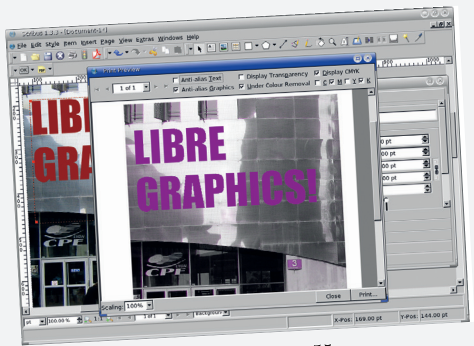
Lowest allowed resolution: 72 dpi

- [✓] Check for placed PDF Files
- [ ] Check for PDF Annotations and Fields

Add Profile | Remove Profile

X-Pos: 345.00 pt Y-Pos: 153.00 pt

Pre-flight checking means you should always get a perfect result from your PDFs.



Yes, Scribus can handle colour separations, using the latest versions of ghostscript.

NOTE Scribus can now set up page margins on your documents automatically, using the functionality of CUPS. From the document setup dialog, click on the Printer Margins button and choose the right printer.

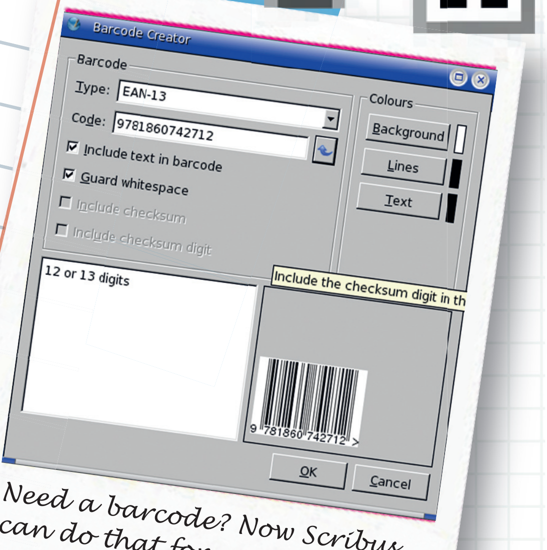
```
LXF Scribus interview
Date: 17 March 2006, 6.14 pm
Location: A large cupboard under
the stairs, CPE building
Subjects: Scribus core developers
```

Linux Format: What's the reason for Scribus's success?
Peter Linnell: On a team level I think we work very well...
Craig Ringer: *Scribus* is a fairly closed development system. There is a core team surrounded by contributors and translators; some contributors that have got to a certain level of proficiency get invited to the core team. It's been rather selective. There is no anonymous access to the private CVS.
PL: We have a private *CVS* for the team. We are quite paranoid about the code. The private *CVS* we can experiment with, so people can trust it for doing real work. We try to have a system where that *CVS* is always buildable and usable. Of course, bugs always slip through, but that's a different thing to being generally usable.

LXF: Even so, *Scribus* has come a long way in a short time – far faster than many other projects.
PL: We're a little older than the usual open source hackers. Craig has significant enterprise experience, Lewis has been in the professional pre-press industry for years. We have quite a few people involved in the printing industry that are able to test various features of the software – either the user interface or the quality of the output, or colour or PDF testing. We also have users that have high-quality printing gear, and either we give them files or they print their own tests. That has helped us to keep a high quality level.
The other thing is that the dev team are very harmonious. It takes some experience in pre-press to know what to do. For example, we won't let you have a false italic [where letters are italicised on-screen rather than by using a specifically-designed italic version] – we won't let you because it's ugly. It shouldn't be done. *Scribus* is very dilligent at checking that fonts are usable – that they are actually there and the glyphs work.

LXF: So you have a low tolerance for bugs, at least serious ones?
CR: With PDF output, we have been very reliable. We haven't had many issues that resulted in creating a bad PDF. But if we do that becomes an absolute focus, and everything stops and it gets fixed.
PL: We found one like that in 1.2. There was this odd bug that if you set up a gradient in a particular way, it would just come out black. When we found out about it, it was fixed in six hours. In 1.4 we'd love to have some sort of automated regression. As the code gets more complex, it becomes harder to really test all the functionality, because there is so much of it. In the old days I could thrash through it in a weekend. I can't do that now. There are more people working on bugs now – finding them, and checking the fixes to make sure they are fixed in the right way.
CR: Through a lot of the cleaning you expose underlying issues covered up by duplicate code, or code that runs twice. I have literally moved thousands of lines of code. Moving towards a different model – being able to convert to PDF from the command line, or run as a server, or run with less memory – a lot of code has existed in places where it was convenient to write it at the time. By cleaning it up, we've been able to make it much faster and smarter.

LXF: What are you focusing on now?
CR: Cleaning and moving code... has brought a lot of speed improvements. So far we have the new undo system, the pre-flight checker, the colour wheel... We tried to make a few things a lot more user friendly. Currently we have paragraph styles and line styles separated. We have a new style manager coming that will centralise these, plus we'll have character styles.
PL: There's vastly better EPS support. Version 1.3 is the road to 1.4, and *Scribus 1.4* will be a professional-grade tool; it will have all the tools to do almost any kind of serious page layout. I think *Scribus* has broken some barriers. People thought an open source application would never be able to do spot colours, but here we are. One thing that's coming is better long document support for footnotes and things like that. Believe it or not, *Latex* or *OpenOffice.org* are probably the best for long documents at the moment. Also we have resizing the page on the fly, and having dissimilar page sizes in a document. There is some imposition too.
Imposition can just be about someone printing out A3: fold it, staple it and you're done. The next thing is when you get on to books and perfect binding etc – that's a whole different world with some very specialised tools that start at $5,000. We're not going to try that. Not yet.


Barcode Creator
Barcode
Type: EAN-13
Code: 9781860742712
[✓] Include text in barcode
[✓] Guard whitespace
[ ] Include checksum
[ ] Include checksum digit
12 or 13 digits
Include the checksum digit in th
Colours
Background
Lines
Text
OK
Cancel

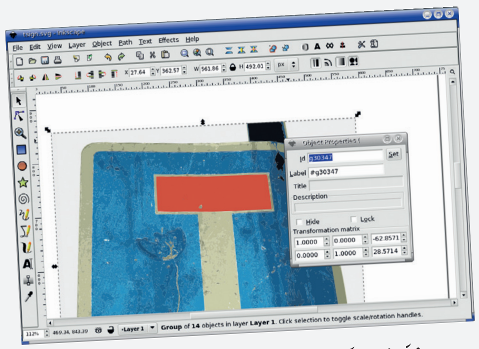
Need a barcode? Now Scribus can do that for you too!

Inkscape

Setting the standards for future graphic design.

Sometimes it pays to make the hard decisions. The SVG standard, hopefully destined to become as ubiquitous for vector graphics as JPEGs are for photos, was never designed as an internal format. Basing the structure on XML made it very flexible and fit for all sorts of purposes. But dealing internally with XML is a pain, and can lead to inefficiency. So it isn't a great surprise that many graphics developers implement SVG support only as a final output option. Some may grudgingly import SVG too, but this is usually little more than a translation layer – try saving the same file out, and it might be subtly different.

But a group of developers decided that what the world needed was a fusion of the traditional programmer tool-style SVG editors (which deal mainly in exposing the XML and allowing programmers to change properties of nodes and such) and the world of drawing tools. The result – *Inkscape* – has been one of the more visible success stories in open source graphics software, and a vindication of the idea that open standards and free thinking can change the world. Well, the graphics world at least.


Inkscape's bitmap tracing is the best currently available for Linux, thanks to the use of other open source code.

The project was started off under the name of *Sodipodi*. At a critical juncture, though, there was a difference of opinions on how to proceed. This is not unusual – the internet phenomenon that is *Firefox* started when just two developers left the Mozilla project. In this case, four of the main contributors to *Sodipodi* decided to fork the code and create the project we know as *Inkscape*.

Inked out

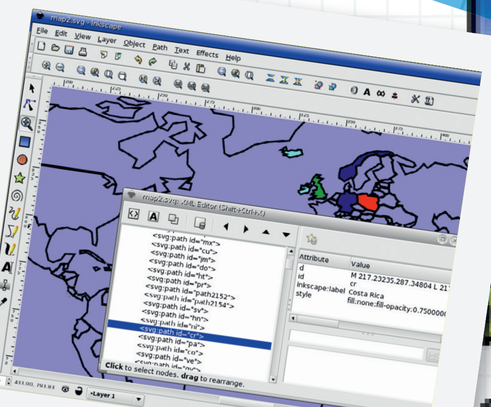
The project has maintained this focus, and for SVG editing it is simply so far ahead of anything else on any platform that it's difficult to imagine it not being the primary tool for creating vector graphics for this standard.

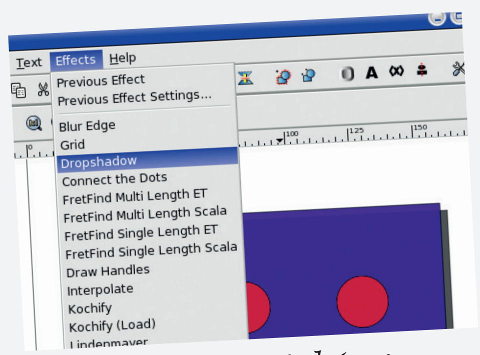
Like most of the software discussed here, *Inkscape* is cross-platform. The same version is available for Linux, Windows and Mac, and in the world of graphics this has great advantages. The incumbent graphics community, certainly for professional design, use the Mac.

Making *Inkscape* available on this platform has not only opened up a market for the software, but also provided very valuable feedback from professional users. While some people still cling to the idea that the only reason people use tools like *Inkscape* is that they are free, this is getting less and less likely to be true. Professional designers can afford the latest kit and the professional tools they need to do their job – that they are choosing to use open tools like *Inkscape* says more about reliability, openness and standards than it does about the lack of a price-tag.

It also means that some of the people most interested in the project, including some of the core developers, think of themselves more as graphics artists than programmers. And while that may suggest that they just code on the stuff that is of interest to them, the *Inkscape* team do have a very solid understanding of the variety of users they are attracting, and the different needs they have.

Usability is very much the focus of current development work – making the software applicable to different types of designers, and also fit for every task that SVG is used for. With the growth in the SVG format online, on the desktop and in mobile devices, *Inkscape* is poised to be the *Illustrator* of the next generation.


Inkscape includes a powerful XML editor for direct manipulation of elements.



Experimental script-driven effects are now available if you enable them in the application preferences.

NOTE The Inkscape team (and many other developers), stressed how much effort they put in to ensure that their code is buildable: "Never check anything in that breaks the build," say the team.

```
LXF Inkscape interview
Date: 17 March 2006, 2.43 pm
Location: Lyon CPE classroom
Subject: Jon Cruz,
Inkscape developer
```

LXF: How did you get involved in *Inkscape*?

JC: It just happened to break from *Sodipodi* at about the time I wanted to get more involved in open source; I had already helped out with some advice on Win32 tricks and stuff. They said the focus was to be on [building] the best SVG editor – technically and artistically. They wanted to have a community and open developer involvement, and they wanted to structure it so that there was no single point of control. These just seemed to be the things that made sense to my mind, from my years being a professional software developer.

It turned out that in my opinion, the core developers of *Inkscape* took a lot of the right approaches to software development. If you have a good idea, you should write it up and put it in. If there are problems, people will take a look at it. It's a very flat organisation in that way. It's a project, and everyone works on the project – it's not one person's code.

LXF: Don't people just want to concentrate on the things they are interested in?

JC: The main people involved care about making a very solid piece of software. We're writing for the users. Initially, the users were only the developers, but as time goes on the user base has grown.

Where I have some experience is in requirement discovery. People will come in with ideas for how the software can be used. Sometimes people ask for the thing they think they need, but don't explain what the end use they need to achieve is.

LXF: So you're guided by how the users are actually going to use the software, rather than just thinking up features? Extraordinary!

JC: Yes, and I think that's one of the things the team have come under fire for. I think we get criticism like, "They are just software engineers, what do they know?" But some of the developers are graphics artists and they are very good on the interface.

LXF: What features are coming up in the next year?

JC: We're thinking about the properties view – refining that into more of an object browser, and also adding a better layers dialog. That is very much requested. The people working on that are taking their time to get it right. It will be good, but you don't want to put it out half-finished.

LXF: Will you be looking at layers? At the moment *Inkscape* seems to take a similar approach to that of *OpenOffice.org*, but it's only when you come to use the layers that you realise there are things that aren't easy to do.

JC: *OpenOffice.org* and apps like that, they need to do a bit of everything OK. We are focusing on SVG, so we should be able to drill down on that one thing and do it well. We are also doing UI rework and target profiles. For example, if you need to create SVG Tiny, for phones and mobile devices, we can do things to make that easier – when you try to export you will get SVG that is compliant with that. *Firefox* released their new version with SVG enabled. That has made a difference.

LXF: There do seem to be a lot of connected ideas surfacing, like Cairo, for example.

JC: Well, there might be a lot more. For example, say you could use *Inkscape* to target the next Gnome desktop using vectors for anything. If you can create [vector graphics] easily, they will be used more. Right now if you create an XHTML page and embed SVG in it, you can have that working in *Firefox* with scripting and DOM, and you can do complex things. So many things like that are interesting now.

LXF: There aren't many SVG-focused tools out there, though. There are some programming tools, but nothing as artistic as *Inkscape*.

JC: Yes, and we have the focus and the experience. We have a good range of people.. SVG is a very clear standard, so it's easy to make sure things are right. I know that there are people using *Inkscape* day to day for their work – if something breaks, we know about it.

Gimp

Number one in its field – serious graphics power.

If you asked 100 average Linux users to name a graphics application, it's a safe bet that they'll shoot one word straight back at you: *Gimp*.

It's now more than ten years since Spencer Kimball announced the project to create an open source application capable of rivalling the professional (and pricey) alternatives on the Mac and Windows. A lot has happened since then – like the umpteen-thousand plugins that enrich the feature set, and the complete redesign of the UI.

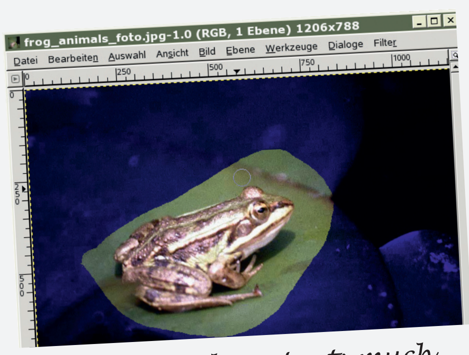
Gimp has come a long way, but it still gets criticised for its lack of some key features, and the slow pace at which development is taking place. Some also claim that *Gimp* is actually holding back development: its amazing hold over bitmap graphics editing in Linux means that there are few rivals worth considering – and this means a lack of choice and competition. In reality though, the problem with *Gimp* is that it is so complicated that it takes a great deal of time and effort to achieve noticeable improvements.

Day of the GEGL

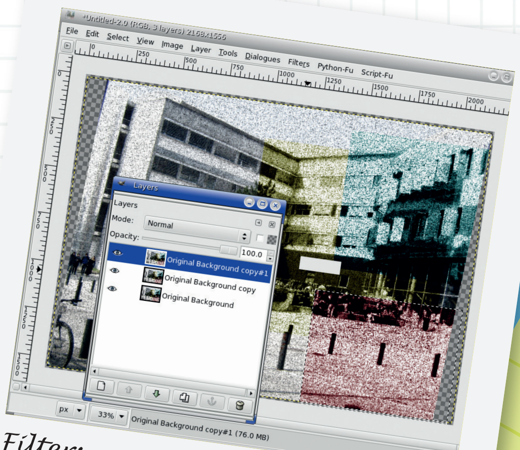
While not part of *Gimp* itself, *GEGL* (the Generic Graphics Library) is going to have a major influence on the direction that *Gimp* takes in the future. *GEGL* is still being developed prior to the first release candidate being made available, and it's being designed to be a standalone, on-demand processing library for graphics work.

The system's most notable feature is that it is designed to be demand-driven. This means that final processing is done at the point it is needed (ie just before display), rather than sequentially through a range of steps. The advantage to this is that it should make it possible to make very good optimisations to the processing at that stage – rather than processing all pixels and throwing some away, transformations will only apply to the relevant areas.

GEGL is also designed to handle different colour depths, so it is useful for a whole range of image-processing needs. Of course, *GEGL* will be very useful for other applications too. The *Scribus* team, for example, was very interested in the potential of *GEGL* and what it might be able to do for image processing within a DTP app.


frog_animals_foto.jpg-1.0 (RGB, 1 Ebene) 1206x788
Datei Bearbeiten Auswahl Ansicht Bild Ebene Werkzeuge Dialoge Filter

SIOX will make cutouts much easier - but use the beta version of Gimp v2.3, not the 2.2 plugin version.


File Edit Select View Image Layer Tools Dialogues Filters Python-Fu Script-Fu
Layers
Mode: Normal
Opacity: 100.0
Original Background copy#1
Original Background copy
Original Background
33%
Original Background copy#1 (76.0 MB)

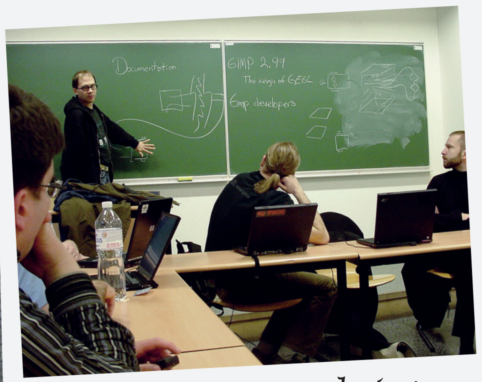
Filters are responsible for most of the new tricks in Gimp until 2.4. arrives.

Plugged in

Most of the work that goes into release versions of *Gimp* is in housekeeping and tidying up before the long-awaited 2.4 release – so we have to look to plugins for our fix of interesting new developments.

We looked at SIOX (Simple Interactive Object Extraction) in *LXF75*, but it's worth pointing out again. This excellent tool is used to extract foreground images from their background with just a few swishes of the mouse. The cutouts generated are every bit as good as if you had painstakingly traced every pixel of the outline yourself.

SIOX will be integrated into the final release of *Gimp 2.4*, but it's available as a plugin now. Because of limitations to *Gimp 2.2*, the plugin is not nearly as effective as it will be in the final release – if you really want to try it out you should run the development versions of *Gimp* (found on the coverdisc) until version 2.4 is released. You can also find more on SIOX at **www.siox.org**.



Mitch Natterer explains why GEGL will be great! (pic from Roman Joost)

NOTE Gimp, like most of the tools covered in this feature, is cross-platform - there are versions available for Mac OS X and Windows too!

```
LXF Gimp interview
Date: 17 March 2006, 3.17 pm
Place: Lyon, outside
Subjects: Mitch Natterer and Sven
Neumann, Gimp maintainers
```

Linux Format: Are you happy with the state of Gimp today?
Sven Neumann: I would really like to see it being further down the road. We made some great plans years ago and we haven't managed to accomplish them yet. A lot of the stuff people are asking for – like support for a better colour depth or a different colourspace like CMYK – that has been on our to-do list for some time already. We don't have enough resources to implement these things.

A lot of work is being done on fixing bugs and doing small improvements. We want to make the features that are already in *Gimp* available to the users, so they can find them and use them without reading a manual – nobody reads a manual.

LXF: A lot of people will be surprised that such an important project only has 30–40 contributors. Has it always been like that, or did you have more people in the early days?
SN: *Gimp* has always been a project that consisted of only few people spending time on it. Those people change all the time though.

One of the problems is that a lot of the long-term contributors that know the code very well have got older – they are not students anymore – they have jobs and a real life. They just can't devote as much time to *Gimp* as they used to a couple of years ago. Perhaps there is some fresh blood needed.

LXF: It must take new contributors a long time to get used to such complicated code.
SN: Well, if people are interested in doing things there is the possibility of work on plugins. A lot of them aren't difficult to understand – they're mostly written in a single C file. So it shouldn't be too difficult – and there are similar plugins and examples to look at.

Of course, working on the core and making fundamental changes to the UI or introducing new concepts is difficult, because there is a lot of code to deal with. On the other hand, we've significantly improved the code base over the last few years. We've done a lot of cleanup and re-factoring. What used to be several hundred files in a single directory is now divided up, and organised better.

LXF: Some people might have assumed you were all on holiday for a year or so, because the feature set almost stood still for a year while you were doing that.
SN: We wanted to have the 2.4 release out [a long time ago]. We promised to have some features in, and people started work on those. We are now in a position that there are a couple of tools that are simply not in a state that we could release.

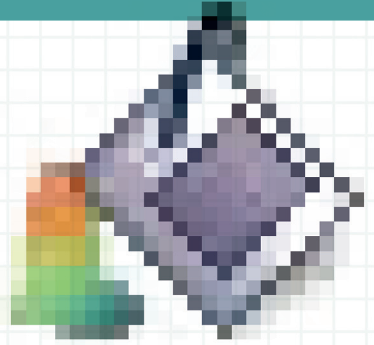
We either have to throw them out, or finish them. One of these is colour management – we decided we really want these features to be in the next release, and so it takes as long as it takes.

LXF: You have mentioned colour management and CMYK – what other features do you think should be available soon?
SN: CMYK will not be available as a mode for storing pixel data. But as soon as we have colour management implemented properly it should give people most of what they have been asking for. I also think that the most important feature missing in *Gimp* at the moment is support for high colour depth. More and more cameras are offering this data, and we shouldn't just throw it away on import.

LXF: Do you think it would be helpful if *Gimp* got sponsorship or was included in some of the bounty schemes that are going around?
SN: That's one of the things I would like to establish in the next few months. Draw up a list of the things that really need to be done in *Gimp* and give advice on how to address it and who to talk to. We would like to publish this, apart from just in Bugzilla. I would like to make it more obvious that we are looking for developers and show what we would like to be done. That could be a way to attract more people.

On the other hand, bounties are sometimes difficult, because for the long-term contributors it can be demotivating to work with people who have just been attracted in by some money. We would have to find a way to make it so everyone is happy. I'm sure it is possible.

Xara

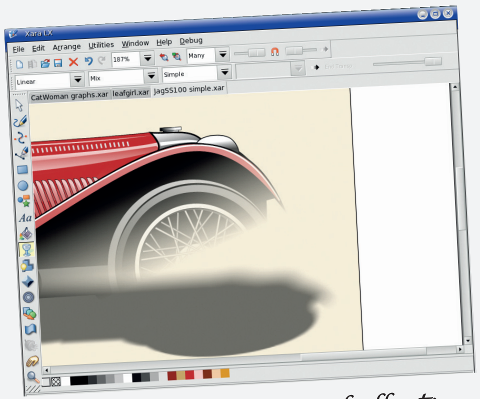


Professional design tool goes open source.

Is there more joy in open source over a proprietary app that lays down the ways of closed code than over ninety-nine apps that need no repentance? If so, there should be a lot of love for *Xara*. Although always developed by a private company in the UK, the *Xara* software was for many years licensed to Corel, who marketed it worldwide under the name *CorelXara*.

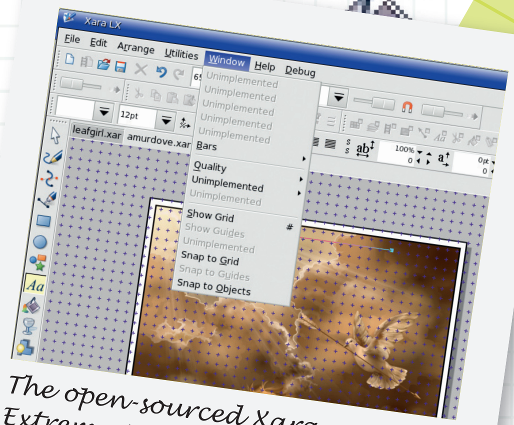
You might assume that all you have to do to 'free' an application is to publish the source code and let everyone get on with it. For some applications, this is indeed the case – but proprietary software, particularly desktop applications, very often make use of toolkits and libraries from third parties, and the code relating to these cannot be released. Stripping this code out can be laborious, and obviously leaves the application incomplete – a few load and save routines here, some algorithms there – never mind the entire UI toolset!

One of *Xara*'s most famous features is the real-time preview engine. Drag a transparency gradient over your work and the screen will update in real time as you're dragging – no more trial and error to get the effect you want! Real-time feedback saves a lot of hassle, and makes the application appear more responsive. This is one area of the software that has not yet been open-sourced. Understandably, the *Xara* developers are taking things slowly and carefully.


Xara's live rendering of effects saves time and makes it a big hit with designers.

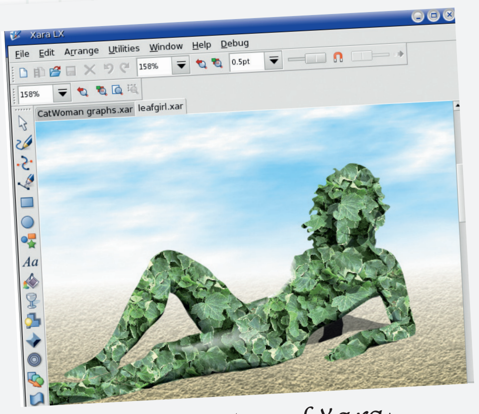
Culture clash?

When *Xara* made the switch, a few people muttered that it may end up competing with *Inkscape* to be the number 1 design app on Linux, but they've missed the point: although there is a lot of crossover, the two projects are largely focused on different things. *Xara* has a long history of being an innovative and versatile tool for illustration and web work, but it doesn't target SVG work as exclusively as *Inkscape*. It seems unlikely that there will ever be an XML editor built in to *Xara* as there is in Inkscape – but now that it's open source, who can say? What is evident is that *Inkscape* and *Xara* developers are keen to co-operate. At the LGM they were very open to an exchange of ideas, and were working on a way to make designs more easily transferable between the two.


The open-sourced Xara Extreme is usable – though many of its features are yet to be enabled. Including 'Save'!

Charles Moir, owner of *Xara*, has also sponsored the development of an 'überconverter'. The idea is familiar to open source advocates – create a 'hub' format for graphics, then a suite of translation tools to convert each format to the hub, and the hub to each format. It sounds simpler than it is, because to be truly effective there needs to be some way to preserve or at least correctly interpret the often unique characteristics of each format. The project is currently called Chromista, and is being developed primarily by Eric Wilhelm. You can check out more about the design goals on his website at **http://scratchcomputing.com/projects/uber-converter**.

As for *Xara* itself, work continues to enable all the features of the proprietary version in the open source code. Regular releases of source and binaries are promised, and the team are working very hard on enabling the 'Save' feature. Promise! LXF


The open sourcing of Xara could bring a great number of new artists to Linux.

NOTE Free at last! The Xara code was released as open source on 17 March. Find the very latest version of the software on your coverdisc!

Libre Graphics Meeting

```
LXF Xara interview
Date: 17 March 2006, 12.14 pm
Place: CPE classroom
Subject: Neil Howe, Xara CTO
```

Linux Format: Why, after more than ten years, are you open sourcing *Xara*?

Neil Howe: The main reason is that we want to see the product take off a lot more than it has. The problem we've got is that we're a very small company with limited resources. People are always telling us how good as a drawing tool *Xara Extreme* is, and we just wanted to get it out there in huge numbers. I think that by making it cross-platform to bring it to the Linux users, and the Mac users, and making it open source, we can do that.

Also the fact that we weren't moving fast enough with the product. I'm sure that every software development company has a wish list that's much bigger than they can ever implement. We're no exception – except that I suspect our wish list is even longer – and with the resources that we've got we didn't feel that we were making progress fast enough.

LXF: So even if you had the best ideas you wouldn't be able to implement them?

NH: Yes, there's not much you can do. We want to see the tool being successful in some form against the likes of Adobe and Microsoft. If we can get support from the open source community we think we can make faster progress.

LXF: Did you have any reservations about open sourcing it?

NH: Yes! It was a big decision for a company like us to have a tool that we've kept the technology and built over so many years – so many man years of effort have gone into it, it's a big step. And that's really the reason why we're not taking a 'big bang' approach – we're not just saying, "Here's the whole thing, it's now open." We're doing it in two stages, so we can see how the first stage goes, get the community built up, get it moving forward and then complete the source release.

LXF: The technology that you've held back is the render engine. Are there any more components you will be keeping back?

NH: Part of that is code for combining shapes – path intersection-type code – and that's all in the same library. So it's the renderer and that piece of code that are in one binary library, which we ship with the source that people have to build it with. I can't tell you exactly, but in terms of the volume of the source code, about 90% is now open.

LXF: What do you expect will be the first rewards from opening that up, and what would you hope them to be?

NH: We'd like to see the community that we've already got. We invited some external developers even in the last few months to do the [Linux] port on a private basis – we just let them have access to the source before it was released publicly. So that sort of started the community off, if you like, and we'd like to see that community grow now and to see those open source developers helping us to complete the port as quickly as possible. That's the first goal, when we've got it as functional and reliable as the Windows version we started off from.

LXF: How long are you expecting that to take?

NH: It's very hard to say. Progress initially was slower than I'd hoped. But that's largely because a lot of the porting is foundation code that doesn't actually do anything itself, it just enables other features. From about mid-January it really started to take off. This week, for example, Alex (one of the developers) switched on two tools in the last two days – the freehand tool and the blend tool. So we hope that the rate of progress is going to continue or get even higher.

LXF: You said you had a huge wish list. At what stage do you think you're going to be able to implement new features beyond the straight port?

NH: We've set the first milestone as being the functionality of *Xtreme*, without things that we can't open source that I mentioned. But open source being open source, we don't expect to tell people what they can and can't work on. If somebody comes along now and wants to work on the product and they write a new tool for example and want to contribute that, we're not going to say, "Sorry, we're only taking bugfixes."

Developers from other projects were very impressed by Xara's capabilities.