Digital Artists' Handbook

By admin
Published: 09/26/2007 - 13:35

The Digital Artists Handbook is an accessible source of information that introduces you to different tools, resources and ways of working related to digital art.

The goal of the Handbook is to be a signpost, a source of practical information and content that bridges the gap between new users and the platforms and resources that are available, but not always very accessible.

The Handbook is filled with articles written by invited artists and specialists, talking about their tools and ways of working. Some articles are introductions to tools, others are descriptions of methodologies, concepts and technologies. All articles were written and published between 2007 and 2009.

When discussing software, the focus of this Handbook is on Free/Libre Open Source Software. The Handbook aims to give artists information about the available tools but also about the practicalities related to Free Software and Open Content, such as collaborative development and licenses. All this to facilitate exchange between artists, to take away some of the fears when it comes to open content licenses, sharing code, and to give a perspective on various ways of working and collaborating.

- Graphics
- Working with sound

Graphics ›

Graphics

By admin
Published: 10/04/2007 - 08:26

Jon Phillips , December 2007

Image reigns supreme. From the thousands of films churned out each year from Nollywood, to the persistent recording of images by security cameras in London to the scaling of windows on your desktop computer, you are already a pixel pusher. But, how can you reign supreme over images? How can you become an active participant in the creation of graphics and move beyond passive consumption. While the distinction between amateur and professional is erased in the Youtube-record-a-video-get-rich-generation, the focus upon high-quality content controlling tools is key. What is the point of mastering verion 3.5 of Killer Graphics App 97's fuzz filter [1] if you don't have a use, or you have become locked into a niche application that costs 2000 Euros for each new version? The focus of this chapter is about exploring free and open source tools that empower you to do what you want to say and if the tools aren't working out, you are allowed to change them from the inside-out! The major tools in this chapter to be discussed are bitmap editor Gimp, vector drawing tool Inkscape, 3d graphics with Blender, and algorithmic graphics creation with Processing [2] . By the way, these tools are free! They have huge constructive communities around them waiting to help you with your tasks, adding new features and supporting vibrant actively producing pixel pushers.

In working with any graphics application, it is important to understand the difference between vector and pixel graphics. Vector graphics describe the use of geometrical primitives such as points, lines, Bezier curves, and polygons to represent images in computer graphics [3] . It is used in contrast with the term raster graphics (bitmap graphics), which is the representation of images as a collection of pixels (dots). Vector graphics are math equations to define curves, generally have a smaller file size than raster graphics and also, can be scaled from tiny to massively huge billboards with no loss in quality. The letterforms for these words I'm typing are vector graphics. Bitmaps on the other-hand, are the types of images that a camera outputs, for example. Maybe your camera is a 5-megapixel camera, meaning it can record 5 million pixels per image. With bitmap graphics the more data about an image, then generally the better quality image, and thus a larger file size.

Gimp is one of the oldest free and open source applications. It is now 10 years old and is on par with major closed-source [4] applications. Gimp is primarily a tool you can use to edit all aspects of a bitmap image, from color retouching of photos, to painting on a canvas to fixing blemishes on a face, Gimp is chock full of tools. Its vector-based sibling is Inkscape, an Open Source drawing tool. With it you can make complex typography, make huge billboards, draw architectural plans and make lovely charts. This powerful tool implements the World Wide Web consortium's Scalable Vector Graphics specification (SVG) and points out another strength of Open Source graphics tools in supporting free and open standards that won't just vanish because a company closes shop, or locks down a format under proprietary patents.

Another important concept for graphics is the difference between two (2D) and three-dimensions (3D). Most graphics applications, including Gimp and Inkscape, are two-dimensional, meaning they deal with height and width of graphics, also called X and Y coordinates. Think of 2D graphics as a piece of paper. 3D graphics, like those operated on by the free software editor, Blender, add depth (Z-axis) to 2D graphics. This is what you see in the famous Pixar movies like Toy Story and Cars [5] .

These typical 3D animations also add a fourth dimension (4D), time. While Blender does handle the fourth dimension by allowing 3D creators to animate, for these chapters, the concept of 4D also includes the concept of graphics through time and interactivity. When Casey Reas and Ben Fry developed Processing, a simple Java-based language and runtime for creating generative graphics [6] , the tools for creating graphics primarily relied upon manual creation with Gimp and Inkscape, or more sophisticated knowledge of graphics programming in C/C++. Processing lowered the barriers for participation in creating interested graphics from code, and also allowed for these graphics to take on a life of their own through user interaction. It should also be noted that Inkscape, Gimp and Blender all offer forms of scripting and automation as well to enable creators to be extended quickly. The main difference between these three apps and Processing, is that Processing generates standalone applications which can be run anywhere. This is great for artists who are making interactive installations, but way too much manual controls for simple photo retouching.

In addition to these great free and open source tools that exist, there are projects as well, which focus on the community of graphics creation and on connecting together graphics applications into a coherently focused suite. The Open Clip Art Library encourages the upload and remix of public domain vector graphics under the guise of “clip art” and the Open Font Library goal is to build the world's largest free and open collection of fonts [7] . The Open Clip Art Library has approximately 15,000 pieces of high quality public domain clip art, meaning anyone can do anything they want with these resources. The Open Font Library is still a fairly new project with ~40 high quality fonts that are either in the public domain or under the new SIL Open Font License [8] . The most notable font on the system is by famed kottke.org blogger, Jason Kottke. He created the super-popular font Silkscreen, a small bitmap-looking font used everywhere on the web. He recently licensed it under the Open Font License and uploaded it to the collection, signally clearly to other font creators that they can build upon it and make it better.

While all these projects exist in the free and open source software universe, the projects did not talk very much until two key projects developed. The first is the Create Project, whose goal is to provide a third-party space for creation applications to work together on standards, shared resources (palettes, brushes, patterns, keyboard mappings), and to encourage inter-project communication [9] . The other key development is the annual Libre Graphics Meeting [10] which is the major event where artists and developers come together to work on making free and open source tools better, seeing what is possible by artists, and massive amounts of cross-pollination to create the future for graphics pixel pushers.

The major difference to closed source proprietary drawing apps is that you can't reign supreme over images. You can't become a true pixel pusher. You can only be the pixel pusher that someone else wants you to be. By using Gimp, Inkscape, Blender, Processing or one of the many other free and open source applications, you can dig deep into the code and communities of these projects. You can even shape the direction of these projects by joining in the discussions, filing bugs about problems with the tools, and showing off how much you reign supreme over images pixel pusher.

Notes

[1] Please note, this is vast satire over learning tools rather than having a reason to use them. Also, please note, this should be called the cheese filter.

[2] See http://gimp.org , http://inkscape.org , http://blender.org , http://processing.org

[3] My Vector Graphics definition is based on http://www.webopedia.com/TERM/v/vector_graphics.html because of the object-oriented reference.

[4] I dare not link to the various Adobe applications you all know I'm referring to: Adobe Photoshop for the GIMP, Adobe Illustrator for Inkscape

[5] The irony of this is that 3D graphics are rendered back into a 2D representation onto a screen, or in a movie theater.

[6] See http://processing.org

[7] See http://openclipart.org and http://openfontlibrary.org

[8] See http://sil.org/openfontlicense (CHECK THIS URL)

[9] See http://create.freedesktop.org

[10] See http://libregraphicsmeeting.org

" >

span-->

- Blender: Working with 3D
- Working with graphics: Processing

‹ Digital Artists&#039; Handbook up Blender: Working with 3D ›

Blender: Working with 3D

By jennie
Published: 05/21/2009 - 14:34

Sumit Sarkar , May 2009

Once upon a time to work with 3d software you'd need a small fortune to get you started – a few thousand Euros for the software, and a few more to partake in a premium rate course or two to learn the basics. Or you could save your pennies and buy a mute tutorial book, which would invariably go out of date as newer versions of your program would surface, and keyboard short cuts, terminology and layout would change.

Now the world of 3D computer graphics has opened up and its anyone's game. Blender [1] potentially replaces the industry standards of Maya, Cinema4D and 3DS Max, happily quelling the age old dilemma of 'which 3D program is best / should I fork out on?' ('try Blender, its free and its great'). Your premium rate teacher is replaced by the hoards of helpful souls wandering Blender forums, waiting and eager to help, and to get you started you can search for the array of video tutorials on YouTube and beyond, lovingly created to make learning easy.

In the year and a half I have been learning and working with Blender, several new versions have already been programmed, further bridging the gap between open source and industry standard. And the world's knowledge of the program has equivalently improved. Ask a question and it is answered often in minutes. If the answer isn't known, you can bet your bottom dollar that the collective will be working out the answer. As such, any 3D task I've wanted to do, I've been able to.

Blender for Sculpture

The main reason for my foray into 3D work came from the need to create a series of sculptures, digitally designed, and physically made via rapid prototyping (RP) techniques [2] .

I had initially looked into clay work, but I found the processes cumbersome, and nearly impossible to realise the detail, complexity and finish I was after. Thus the solution seemed to be to go digital with the 3D work, in the same way I moved away from analogue painting and design to digital with my 2D artwork, needing techniques infinitely more speedy, controllable and precise.

To be an artist, you now no longer need a studio space, expensive materials, tools and equipment - just a computer and an internet connection.

Work Flow

Your method of working will be down to you - indeed even for simple modelling, there are many ways to skin your cat. Some will want to prepare storyboards and sketches before embarking on a sculpture or animation, others dive in and follow trains of thought. Either way, it is the idea that drives the work, and that can come at any point in your project.

Interface

Many comment on the unusual interface of Blender, initially most noticeably in the way it uses a left click where most programs use a right. While you could change the settings so it is the opposite, I would recommend leaving it as is – once you get used to Blender's interface, it really feels very logical and intuitive. Indeed Blender has the most malleable interface of any program I have used – every window can be set to display the toolbar, panel or working area of your choice and you can even customise the overall look of the interface changing colours, sizing and fonts.   Once you get used to Blender, you'll soon be wishing other programs followed its user friendly interface logic.

Edit Mode and Object Mode

I would highly recommend Apollos' finite Blender Basics video tutorials [3] as your first point of call. I have found following video tutorials the easiest and most successful way of learning. Watch, copy, rewind, repeat, master.

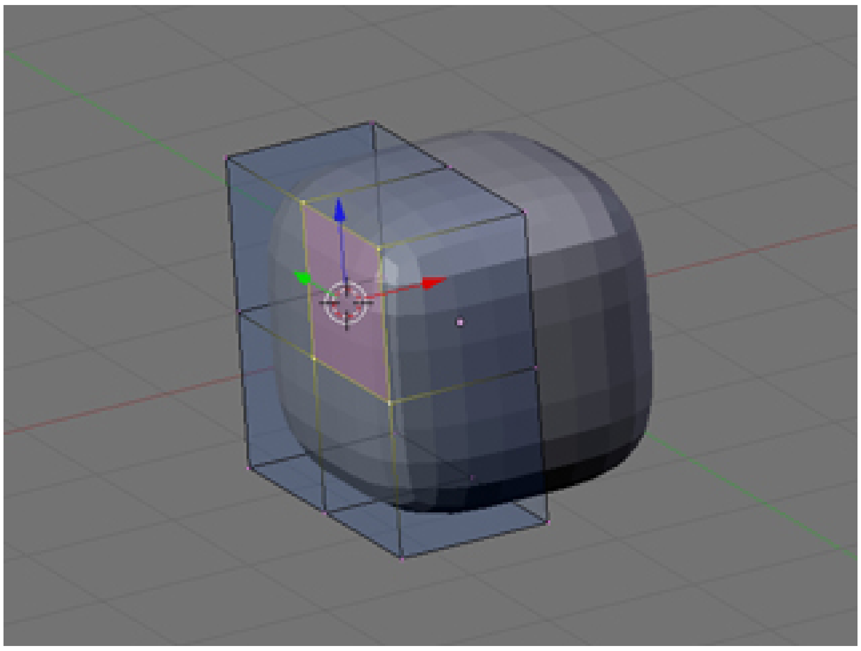
These will give you an insight into the two main modes you will use, Object and Edit Mode. Object mode allows you to work with all the objects in your scene, including such things as lamps and cameras. Click on an object and tab into Edit Mode and you'll then be able to work precisely on the individual object. There are too many vital Blender processes and their keyboard short cuts for it to be relevant to list just a few here – spend an hour or so with Apollos' tutorials and you'll know them all.

Having learnt the basics of Blender's Edit and Object Mode, you could pretty much make anything. If a sculpture can be seen as being a collection of co-ordinates that define its form, once you know how to create, move and join vertices (the individual points that make up your model), the only tricky bit becomes knowing what to make. Certainly patience and commitment are key attributes for any sculptor, digital or otherwise.

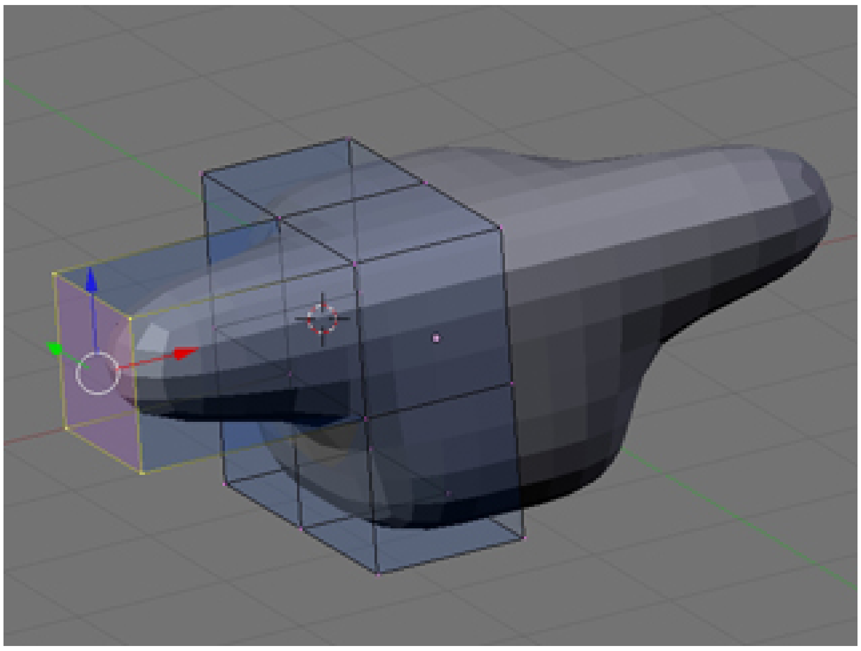
Modifiers are an important tool for modelling, and are equivalent to filters or effects in a 2D art application. Adding a modifier to an object applies that effect to the object, and can be removed, made invisible and modified without affecting the original mesh (the net form of all your connected vertices), You can add multiple modifiers to any given object. In many cases you will not need to apply your modifier at all but if you do make sure you do not apply it too soon, as this can result in meshes with a high number of vertices (which will push the power of your pc) and once applied you can't go back. For example adding a Subsurf modifier will make your object appear smooth from the few vertices of your mesh – apply it and it will make the modifier 'real', i.e. create enough vertices to ensure that smoothness (I always save my files with a different name every time I make a pivotal change so that there is 'going back').

A popular method for modelling is what is known as box modelling. Starting with Blender's default cube, subdividing it, adding a mirror modifier and a subsurf modifier, has you very quickly creating complete forms (subdividing adds extra vertices to your mesh, a mirror modifier means whatever you do on the left side is mirrored on the right, and the subsurf modifier adds extra subdivisions to each surface, making a few simple vertices appear curved and smooth.).

Add to this the extrusion (see image below) of vertices and faces (a face is made up of either 3 or 4 vertices) and you could easily make a humanoid form in 15 steps. This is the very essence of 3D modelling – starting with a simple form, adding more vertices to work with (e.g. by subdividing and extruding), and fine tuning.

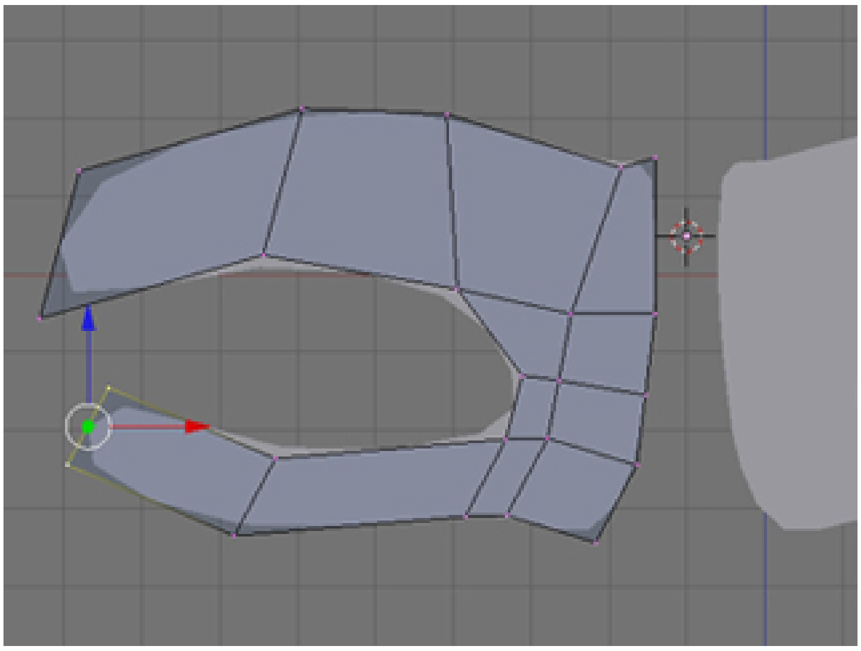


Subsurf and Mirror modifier

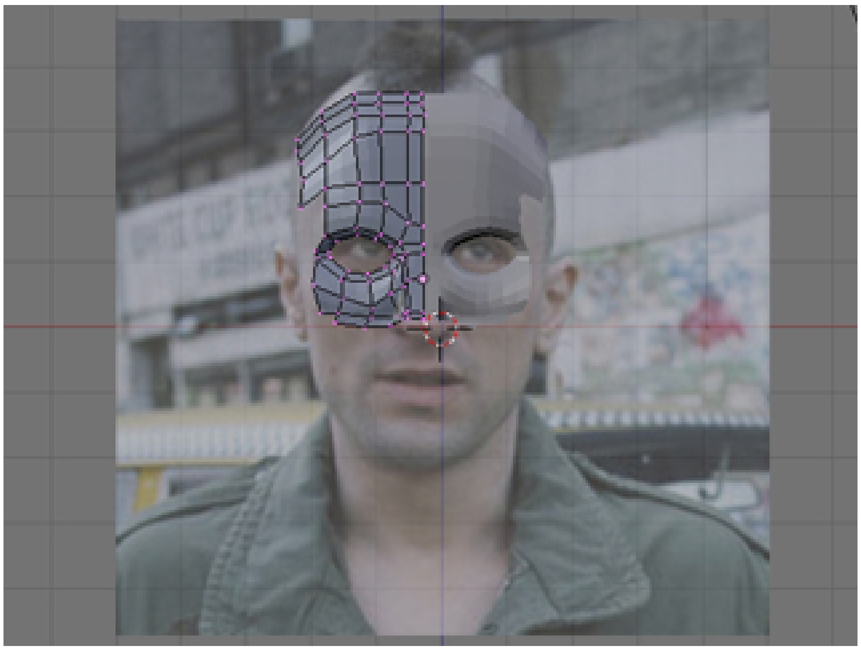


Extruding a face

Alternatively, you can start your model from scratch, which is often the technique used for making human faces, or objects that aren't easily simplified into geometric shapes. Starting with a vertex, edge or face, extrude and reposition away until your form starts to take shape. Many people find it useful to use a background image as reference.



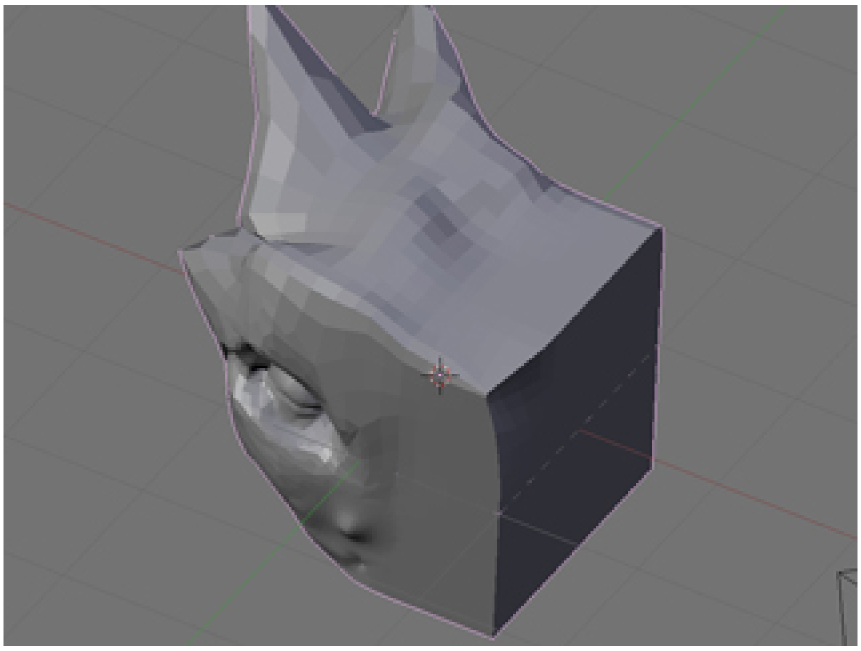
Modelling with extrusions



Using a background image as a guide

Sculpt Mode

Once you have your basic form, switch to Sculpt Mode for a more organic shaping of your mesh.   All physical sculptural   techniques exist here: You can Draw (which adds or 'rubs out'), Smooth, Pinch, Inflate, Grab, Layer and Flatten, all the tasks you could do with clay, without any of the imprecision or mess. You may need to subdivide an area so you can add further sculptural detail. Theoretically you could start with a cube, subdivide it to Hell, and create your model purely by sculpting it.



Sculpt Mode

It is also worthy of note that working in a virtual 3D environment that is essentially 2D from our physical point of view, it takes a bit of time to get used to the way the 3rd Dimension works, i.e. what is in / out, left / right being relative to the angle at which you're viewing the scene. With a bit of use, this quickly becomes instinctive.

Physical Output

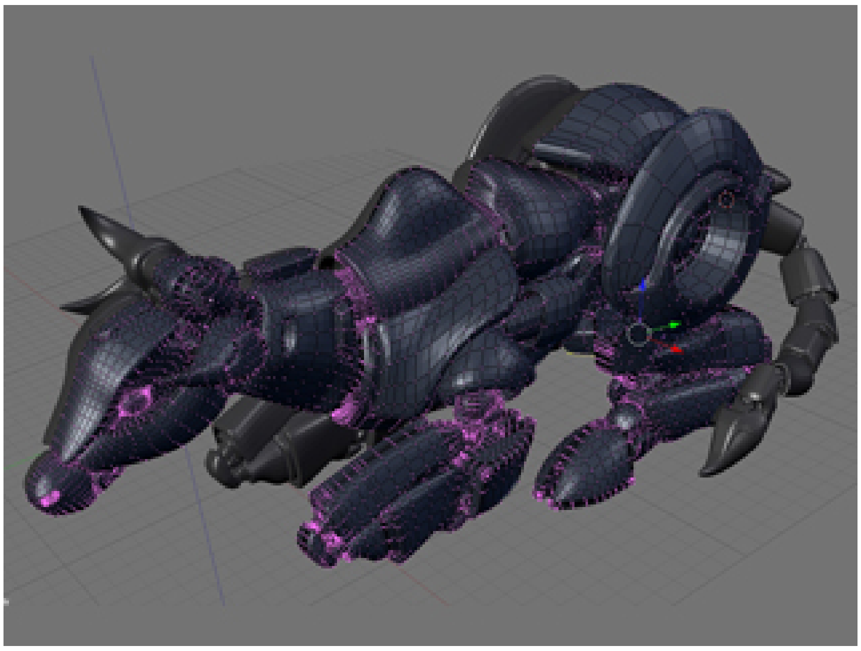
As for outputting to physical form, Blender takes pride in its compatibility, so exporting in the correct format for rapid prototyping is a one click operation.   MeshLab [4] , another open source program, is also a useful tool for working with your .stl's (.stl is the file format commonly used for RP); finishing your meshes for output, sizing your object for the real world, checking for and cleaning up errors and generally seeing what your RP client will see.

One big issue I found when modelling for output was the trickiness in joining two meshes together. For example, if I wanted to merge a cone with a cube, (or a more complicated example being a hand with a sword), it is not merely enough to join the two objects, as all overlapping vertices would remain.   For RP, the model needs to be one continuous outside net.   Blender's Boolean [5] operation is intended for this, though is not quite yet up to scratch, especially when joining complex meshes together.   As such I spent ages manually stitching overlapping meshes together, removing 'inside' vertices, when this could have been done in seconds by your RP client. Don't believe them if they say it will take many design hours to do this - it was only the very honest company I ended up working with that told me it is a one click, two second operation on their industry standard stl programs (which

I'm sure MeshLab and Blender will be able to do before long).

I needed to work with a rapid prototyping company for my sculptures, though this is a very expensive process, the industry catering as it does for industry. However open source solutions are on the up, and that must surely be my next project.   RepRap [6] and Fab@Home [7] are two such open source RP projects that are in development, designing machines so they can be made at home, and in case of the RepRap will even be able to self replicate.   For the time being you can find your local Fab Lab [8] and use their machinery at relatively low costs, and even free in some cases.   Manchester will soon see the UK's first Fab Lab.

It is encouraging that such things are becoming more affordable, and I've heard of 3D printers being introduced to schools and universities.   Given a few years, I'm sure we'll all be able to create sculptures on our computers and 'print' them on our desk top 3D printer.



Screenshot of Blender Sculpt

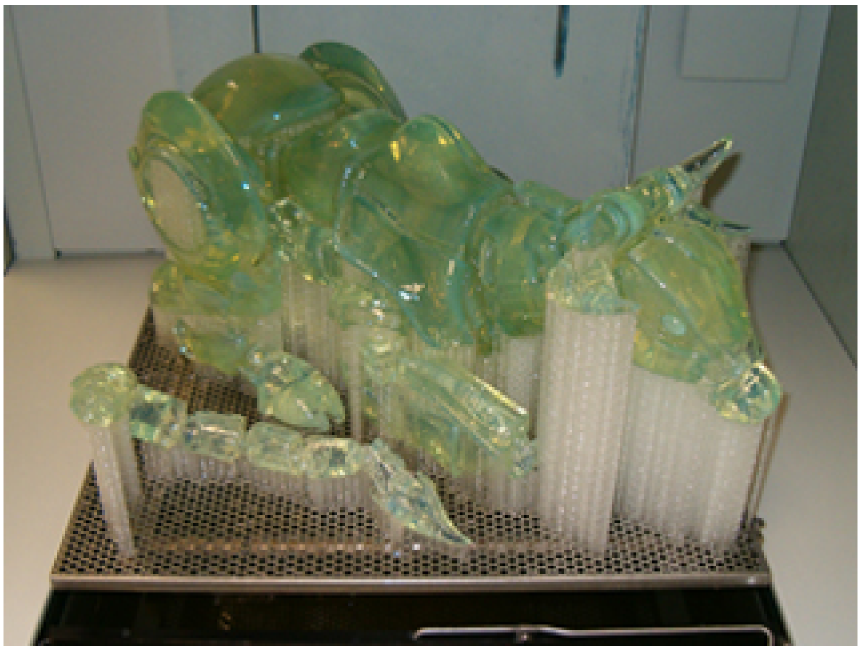


Photo of Rapid Prototyped model

Blender for Animation

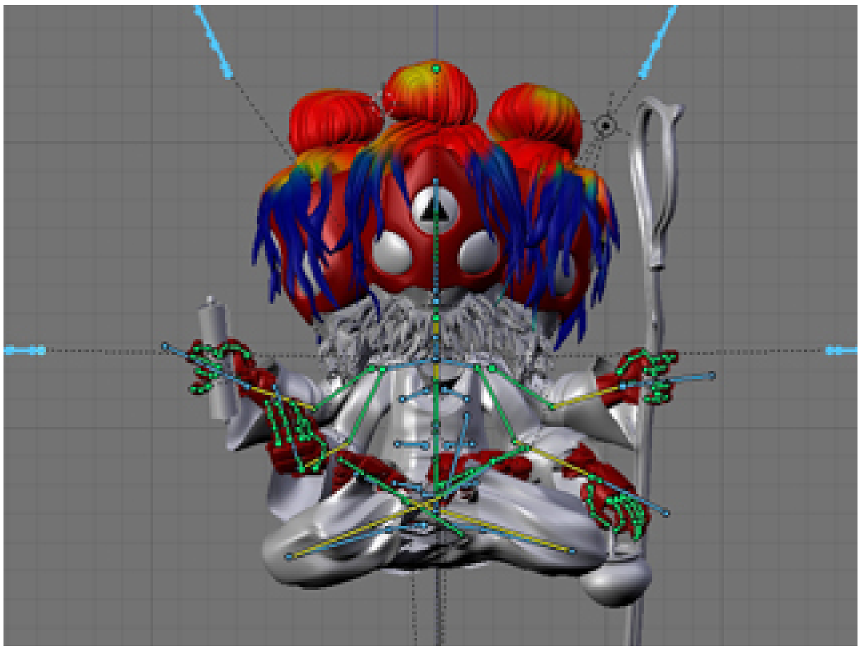
Another main area in which I've used Blender to make artwork is for a series of animations. Modelling for animation is much the same as for sculpture with the main difference being the need to use as few vertices as possible to speed up rendering time. Working with quadrilateral faces (a face made up of 4 vertices) and avoiding triangles makes life easier (you'll see), and it soon becomes like a game, trying to make your model with as few vertices and triangles as possible.

So far you'll have modelled using three dimensions, X, Y and Z.   Animation brings you to the fourth dimension: time. Place a ball at a certain point in your scene, and at a certain point on the timeline (the measure of time for your scene), say 0 seconds, set a key (a marker of an object in time), then move the ball's location and move forward in the timeline, say 4 seconds, set another key, and hey presto, you've animated. The ball will move between those two co-ordinates over 4 seconds.

Learning these basics means you can potentially make any animation you can conceive. Seemingly complicated scenes are purely down to detail and precision, i.e. the movement of a hand from A to B is one action, the movement of a hand with all of its fingers moving realistically is just many more equivalent actions for each section of each finger moving between two or more co-ordinates. Certainly patience and commitment are key attributes for any animator, digital or otherwise.

Get more into Blender use and you'll soon be rigging your character with sets of actions and constraints, so that by moving one object, the whole hand will clench into a fist, or spread its fingers.

Blender's physics engine is another gratifying and hugely useful tool for creating dynamic animations.   A few clicks here and a value entered there, and the hair and clothes of your character will react to gravity, wind and movement, and bellies will bounce.   Google or YouTube-search softbody and cloth, and you're really half way there.



Rigging model for animation

Render of animation

Blender Live

I've also used Blender for live work, that is, creating 3D artwork at clubs or art events [9] . With a computer and a projector and a confident knowledge of all of the above, it is more than possible to create a virtual sculpture in a few hours. It is captivating for an audience to see a blank 'canvas' evolve into a 3D form, more so than the 2D equivalent live work I used to do, and the ease of Blender's interface makes the process fluid and highly watchable.

Rendering

Finally, adding lighting, colour and texture to your scenes is the fun part, and again the only limits to what can be achieved are those of your imagination.

In the vein of such CGI films as The Incredibles and Monsters Inc, there are a growing number of films made entirely in Blender [10] , most famously Elephants Dream, proving Blender (that's you), can do anything the industry standards can. This bodes interesting questions for the future – as personal computers get more powerful, and newer versions of Blender and other open source programs are released year after year, given ten, and a great idea, aren't we all then on the same playing field as Hollywood and multi million Euro budgets?

Still from Elephants Dream

Notes

[1] http://www.blender.org/

[2] http://en.wikipedia.org/wiki/Rapid_prototyping

[3] http://blenderunderground.com/2007/07/18/blender-basics-part-1-video-tut...

[4] http://meshlab.sourceforge.net/

[5] Boolean operations are AND, OR or NOT, or in Blender terms, Union, Intersect and Difference, i.e. You can join two objects as one, work out the area they intersect, or the area they don't, very useful processes in Rapid Prototyping

[6] http://reprap.org/bin/view/Main/WebHome

[7] http://www.fabathome.org/

[8] The Fab Lab program started at MIT's Center for Bits and Atoms (CBA) and spread from inner-city Boston to rural India, from South Africa to the North of Norway. Fab Labs provide widespread access to modern means for invention, and share core capabilities, so that people and projects can be shared across them. http://en.wikipedia.org/wiki/Fab_lab

[9] Live 3D work recorded http://www.youtube.com/kriksix

[10] It's open source so it's all free. Download at http://www.blender.org/features-gallery/movies/

Images

[0-9] by the author

[10] (c) copyright 2006, Blender Foundation, Netherlands Media Art Institute, http://www.elephantsdream.org

" >

span-->

‹ Graphics up Working with graphics: Processing ›

Working with graphics: Processing

By marloes
Published: 09/18/2007 - 13:51

Olivier Laruelle , September 2007

You might have come across the 'made with Processing'   hyperlink on the internet or heard of Processing before. Over the past six years it has become a real phenomenon, allowing creative minds to access the digital world. Based on a rather simple syntax and minimal interface, Processing smoothly drives beginners into the scary world of programming.

This article is not a tutorial, but rather an attempt to give you a global idea of what the programming environment is, looks like and why it was created. Should you decide it is the tool you need, this article will hopefully provide enough pointers to online and offline resources to get you started on the right track.

What is Processing?

Processing was initiated by Benjamin Fry [1] and Casey Reas [2] In Fall 2001, formerly of the Aesthetics and Computation Group [3] at the MIT Media Lab [4] . We can see it as the follow up of Design By Numbers [5], created by John Maeda [6], a programming environment strongly oriented at beginners. It is an open source project free to download and free to use, with already a large community around it.

What and who was it designed for ?

As the Processing web site mentions: “Processing is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool. Processing is developed by artists and designers as an alternative to proprietary software tools in the same domain.” [8]

Those picking it up for the first time will find lots of very well written and archived code samples, a large amount of information on the internet and answers to question on the Processing forum [9] . The people behind Processing [10] have made huge efforts in helping the community to grow and learn. For instance by   illustrating the language reference with clear snippets of code. Thus helping to grasp basic concepts little by little, i.e. image, pixel, array, mouse events, primitive shape drawing etc.

For the more international learner you are pretty sure to find a reference you will understand as volunteers joined forces and translated the reference into 8 different languages. Even if Processing was aimed at beginners, its versatility and ease of development still is relevant to anyone wanting to write creative software pieces.

What have people done with it ?

It is impossible to mention in a short article all the excellent work that has been produced in the past years. Processing has a wide range of possible applications such as data visualisation, generative drawing, print based design, video production, tangible installations etc... Following are some of the most inspirational work made with Processing.

Data visualisation

"We feel fine" [11] - An exploration of human emotion, in six movements - by Jonathan Harris and Sep Kamvar is an impressive web based example of what can be achieved. Processing was used as a creative front end for a database that was constantly fed with comments left by various users on the internet about how they emotionally felt. Based on particle looking interfaces, it's a delight to browse through all the snippets that people left on the internet without thinking that one day it would be collected by an artist, in the look of emotions.

We Feel Fine logo

Artist Aaron Koblin, wondered what it would be like to see one day of aerial traffic over northern America. He collected enough data and used Processing to generate a video revealing a day of aerial traffic in America [12] .

If data visualisation is an area that interests you, http://infosthetics.com/ hosts much more examples. Note that this is not a Processing only focused web site.

Generative drawing

A popular practice within the Processing community is to draw in an algorithmic manner 2d and 3d graphics on screen.

And so does American programmer/artist Jared Tarbell. He creates delightful "paintings" based on pseudo naturalistic behavioural systems. For example, Bubble Chamber and other pieces can be seen on his web site [13] . Exquisite 3d work such as Marius Watz's [14] illumination or Karsten Schmidt's "vox scape" and its intricate 3d mesh fantasy world [15] demonstrates the level of work that has been recently produced.

Bubble chamber by Jared Tarbell

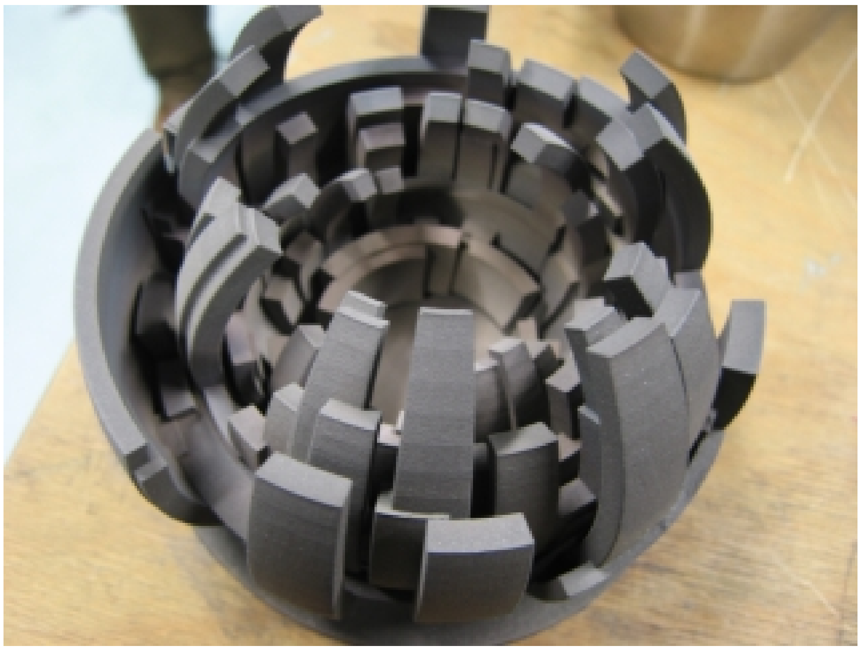
Obviously generative graphics aren't bound to stay within the memory of your computer or on your retina imprint. They can be transposed to other media such as print - like for instance the Lovebytes 2007 generative ident [16] . 3d physical models can be made using rapid prototyping and obviously video can be composited using processing generated imagery such as “Nike One” by Motion Theory [17] .

Illuminations 01 by Marius Watz

Generated art forms can also become reactive: to sound like, the beautiful and very popular Magnetosphere [18] iTunes audio visualiser created by Robert Hodgin [19] , or in reverse when forms generate sound as in Amit Pitaru's sonic wire sculptor [20] . Processing doesn't need much extra to be used as part of a physical composition. One easy way to achieve physical interactions with Processing is through camera assisted computer vision. Processing (windows and mac version) is able to read video streams in a very simple fashion. Thus enabling artists to write code for detecting presence, motion, ambient colours etc... Shadow monster [21] by Philip Worthington is a good example. He revisited the shadow puppet theatre mechanic, augmenting it with digital shouting monsters. It is immediate, fun, accessible and self explanatory.

Voxscape3 by Karsten Schmidt

Cameras, keyboards, mouse or sound inputs are not the only interfaces to physically interact with computers. Sensors of all types are also available. Be aware that you will probably need to work in parallel with other programming environments and hardware such as Arduino or Wiring. Physical computing with Processing isn't the focus of this article but, should you be interested in it, you can find references at the end to use as a starting point.



Object #1 by Marius Watz

The environment

Where to start ?

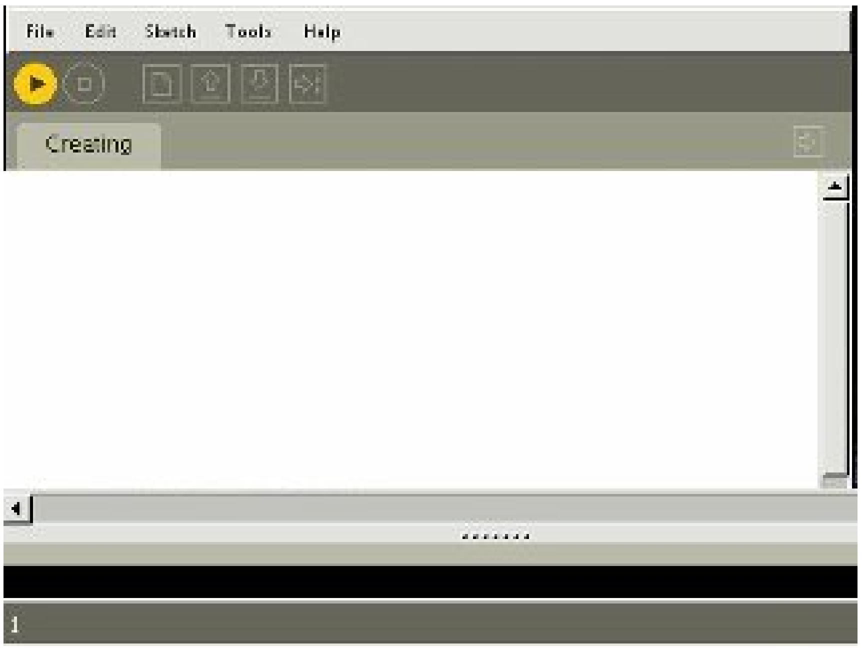
The Processing download page [22] is where it all starts. There you can find the version of Processing corresponding to your operating system. Once downloaded, decompress the file anywhere on your hard drive and open the processing application. It is now ready to run.

Before getting stuck into the code and making things work, it might be useful to take a step back and have a look at what metaphors are used throughout the environment, but also how it basically works.

All in all, the processing environment is a group of 3 things : the IDE (integrated development environment) [23] the core library code, and the external libraries.

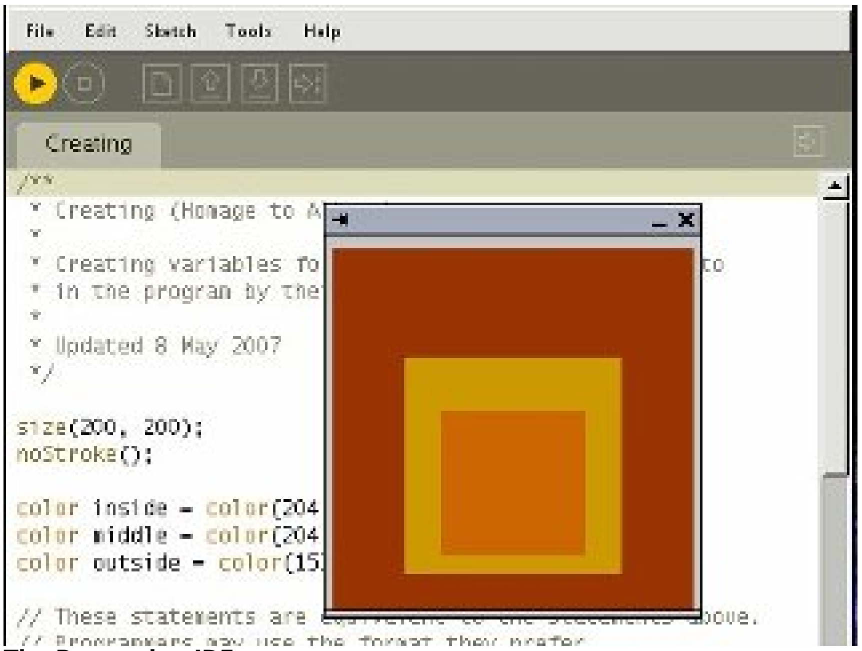
The IDE : what does it look like ?

To make it simple, the IDE is what you first see when you open Processing. Its the area where you write and edit your code, but also contains the buttons and the menus you'll be using. The first thing you see when starting Processing is a blank text editor. This is where to write code that will make beautiful things happen.



The Processing IDE

At the top are a series of buttons. From left to right : run, stop, new, open, save, export. I think their labels are pretty self explanatory in defining their actions. In addition more features are accessible through   contextual menus available at the very top. namely :   file, edit, sketch, tools and help.



```
/**
 * Creating (Homage to A
 *
 * Creating variables fo                                  to
 * in the program by the
 *
 * Updated 8 May 2007
 */

size(200, 200);
noStroke();

color inside = color(204
color middle = color(204
color outside = color(15

// These statements are                               above.
```

The Processing IDE

Sketches and sketchbook

The sketchbook metaphor Processing uses refers to a way of organising programs. A Processing program is a sketch. The application acts like a sketchbook from where you choose or create new sketches. It also comes with its own sketch examples. Reading and running them is a great starting point for becoming familiar with the environment and having a first stab at the language. Just select File > Sketchbook > examples > choose the category you're interested in > a sketch. Then press the 'Run' button. Repeat at will.

Tabs

Code can be split into several files to make it easier to maintain. Tabbing enables you to have all of these files open at the same time, making development quicker than if you had to close and open code sections each time. To do so, you will see a button on the top right. It allows you to create and rename or hide tabs. You can then copy paste them from one project to the other if needed.

Now that you know where to write the code, lets see what the code looks like.

Introduction to the basics

The Processing syntax was made to be easy and flexible to write with while keeping intact the main programming structural logic. This is obviously where it takes an interesting educational dimension. And to get quick visual feedback of your code is another motivational boost to experiment more and learn by the same occasion. Of course it takes a little effort to get into it, but the environment will give you enough freedom to smoothly sail along without pulling your hair every five minutes.

To give you an overview of what Processing code is like, we will go through some of the basic concepts of the language.

Setup and draw

First of all when you start writing your code you need to mention what will be the window display size. So how do we set up our favourite playground boundaries? By using the function setup() ! [24]

```
void setup(){

        size(800, 600);

        background(0);

}
```

If you run the above code it will create a window 800 pixels wide and 600 pixels height with a black background [25] . Processing always calls setup() first and looks at what it contains. It is usually used to invoke global commands relative to the sketch. Like the size, the frame rate, the rendering mode etc...

But drawing an empty window isn't really that exciting right? Where is the action? We want more than creating a window... we want to draw things into that window. How do we do that? Using draw() ! [26]

Add this to your sketch after previous code :

```
void draw(){
        fill(255);

        rect(350, 250, 100, 100);

}
```

And if you run this again you will see a white rectangle drawn in the middle of the screen on a black background. This is how to draw things. Unlike software like Flash or Director, Processing has no timeline, instead draw is constantly called at a regular time interval. So to animate things, you will have to calculate coordinates of your graphic elements from one frame to the other.

To place and draw graphics within the window, Processing uses the Cartesian coordinate system [27] . Like Flash the widow display has its origin at the top left corner. The 'x' being the horizontal axis and 'y' the vertical. So, the further right in your window, the higher the 'x' value, and the lower you are in your window, the higher the 'y' value. This obviously differs from the normal mathematical system with the origin placed at the bottom left of any 'graph'.

So as long as you set things in setup() and draw things in draw() you shouldn't have any problem. Check the reference [28] to learn how to use the syntax, and then check the extended references [29] for more exciting possibilities. Many shapes can be drawn, an infinite number of colour combinations can be used and plenty of animations awaiting to be created.

Note : Processing doesn't limit you to writing procedural code [56] . If you have Object oriented programming knowledge you can apply it too [30] .

Inputs and Outputs

Common Inputs

Processing allows easy access to the common inputs a computer provides. You can capture the mouse coordinate, listen to what key has been pressed, or get the current system time.

File input

You are obviously not limited to drawing primitive shapes in Processing, you can also import all sorts graphics to play with. Processing will look into the data directory of your sketch for anything you want to import. Most bitmap image formats can be read but also SVG vector graphics. You can read video connecting through camera feeds or Quicktime files. It is very easy to read, write and analyse sound data using external libraries such as Sonia, minim or Ess. You can naturally also open and read the content of text or xml files [38] . Finally Processing has its own way of handling fonts, you just need to convert any true type fonts to bitmap versions font before you can use them in your sketch. This is doable manually (Tools>Create Font...) or dynamically using createFont().

Connectivity

The Network library provides the possibility of establishing connections to a network [28] . You can handily use the network connections and OSC to communicate with other applications. Another way to communicate with your program is by listening to the serial port activity (or i/o), perfect solution for connecting to physical interfaces like Arduino or wiring [39] !

Outputs

You can export the result of your computation to many different file formats. The more obvious being exporting an image of what is drawn in the display window using saveFrame() you can also export raw Quicktime for anything in motion (note : not on linux). For printing purposes it is possible to export as .pdf , Svg (proSVG library) or Postscript (SimplePostScript), Collada for 3D or .wav for sound.

Export

With the processing beta version came the possibility to export applications (File > export) application. This will create 3 different types of executable file, one for each operating system, Osx, Windows, Linux. You won't need processing to run those files, they'll be standalone. It is also possible to export an applet to run in a web browser, or finally the present mode (Sketch > Present) straight from the IDE.

The Present mode is particularly interesting when you want to cover the entire screen with your application. It won't change your screen resolution but the area not covered by your sketch will be filled with black.

Renderers

Another thing that might be worth mentioning at this stage is the different renderers you can use to display your sketches. You can do this by defining an extra variable in the size() command. Available renderers are :

By default, if you don't mention anything Processing will use JAVA2D, good for drawing and web but can be a bit slow. Unleash P3D (JAVA3D) (size(800, 600, P3D))and you can exploit its great functionality for drawing in 3d and making things portable onto the web. Faster? Defining OPENGL uses jogl a java bindings for OpenGL API. Very Fast, depending on your graphic card, but not easily portable to the web. OpenGL is almost another topic in itself as you can invoke OpenGL commands from within Processing, hence accessing another programming language. There are numerous posts on the Processing forum on how to hack around Opengl in Processing. Finally you can use PDF to export what the program draws to a pdf file.

Note : More information can be found on the size() reference page.

Now that we have been around the most important functional aspect of the IDE and have an idea of what Processing syntax looks like. The last principal aspect we need to understand is what Processing is doing when we run our beautiful code.

Core

Processing is written in Java and although it has its own syntax, everything written in Processing is converted back into “pure” Java before being compiled (run). “Distinctions between Processing and Java are the Processing graphics library and a simplified programming style that doesn't require users to understand more advanced concepts like classes, objects, or animation and double-buffering (while still making them accessible for advanced users). Such technical details must be specifically programmed in Java, but are integrated into Processing, making programs shorter and easier to read.” [31]

Processing is autonomous. It is shipped with jikes [32] , an open source Java compiler. So there is no need to install Java to make it work. Nevertheless you will need Java Runtime Environment installed if you want to view your applets running in your very own browser [33] .

One advantage of Java is its Cross platform compatibility. It will run on most operating systems. On the other hand Java can be slow compared to some programming languages such as C or C++. But in the context of a teaching software it makes sense to choose ease of use against speed.

Being based on Java also means you have access to the numerous great libraries the language offers. If you can't do what you want using Processing syntax there is probably the answer in the wide world of Java. If you wanted, you could even write Java code into Processing, but that may be the time for you to change IDE (to something like Eclipse [34]) and then use the Processing core as a library.

As it is an open source software development, you can have a look, download and play with the full Processing core code from the development site [35] . On the same web site, you will find lots of useful information on how to help develop Processing.

Libraries

Libraries are add-on code to the Processing core that allow you to extend the capabilities of the language. Some of them are bundled with Processing and so are available to use from when you download the main application. The other are user contributed libraries listed on the Processing site and maintained by their creator. For instance the popular Sonia library by Amit Pitaru allows you to access advanced audio capabilities like doing real time frequency analysis (FFT) and write wav from samples etc. MaxLink [36] Enables communication between Processing and Max/MSP and to name another one JMyron [37] by Josh Nimoy allows you to do advanced motion tracking using a web cam. One's answer to a technical need can then be shared with the rest of the community, resulting in a constantly growing list of libraries [38] .

Processing sister projects.

There are 2 obvious ways of getting Processing to communicate with other devices than your computer. Writing programs for your mobile, or exploring the world of physical computing using Arduino or Wiring.

Mobiles: You can port your Processing applications to Java enabled mobile devices using Processing Mobile. There is a separated web site devoted to it [40] .

Physical computing :

Wiring : Borrows The Processing IDE for programming i/o boards [41] .

Arduino : As Wiring, Arduino shares the Processing programming environment so you won't have to learn another interface, “only” the code will be different [42] .

Resources

Online

The syntax page [28] of the Processing web site is the most useful and obvious online help. It is really the starting point. If you encounter difficulties while developing beyond the syntax, other users can help you on the forum. To get a better chance of getting an answer, make sure to post in the right category (syntax, bug, openGl etc..). make sure to search the forum first see if anyone one encountered and resolved a similar issue. Be specific on what your problem is by posting code for instance, this will help you get a quick answer to your problem. There is no mailing list on the user end, the forum is the heart of Processing user contributed knowledge - it is a beginner friendly place to ask questions so don't be shy.

For clever code tricks look at processinghacks [45] and stellar snippet palace[46]. Make sure you also go through the excellent ”the nature of code” [47] class written by Daniel Shiffman to learn about “programming strategies and techniques behind computer simulations of natural systems”.

To keep in touch with what users individually think and produce, check processingblogs [48] - an aggregation of Processing related blog posts. Flick through the flickr [49] pool, and click around the meta sketchbook buildwithprocessing [50] for examples of interesting applets. Users also appropriately tag their videos so it's easy to find the ones related to Processing on YouTube [51] or Vimeo [52] .

Offline

If you think Processing is for you and you will start creating things with it, then it's probably a good idea to purchase the recently published “Processing: A Programming Handbook” for Visual Designers and Artists [53], written by the very creators of processing Casey Reas and Ben Fry. At this date another book related to learning programming using Processing has been written: Processing: Creative Coding and Computational Art [53] By Ira Greenberg. I personally haven't yet been through those publications. I'll let you browse through the online samples to make up your own mind on what suits your needs best.

Limitations

The choice of the programming environment is a fundamental factor when starting working on a project. Hopefully this article will give you elements to answer this question. Nevertheless, Processing can be limited in some domains. First of all, Java isn't the fastest language around so if you need to process heavy computational tasks it will simply not deliver good results.

For instance, I would not recommend it to do real time sound synthesis. Not that it is impossible, but I would look at specifically designed environments that will provide you with more tools and speed to do so. Real time video manipulation is another treatment that demands lots of resources and if you wanted to access a large bank of clips or run heavy pixel manipulation on a video stream or between streams, probably best looking at some other means.

On a practical side, the IDE is very minimal and can become restrictive when writing lengthy code. It was made for sketches, so if you think about it like painting a fresco, then you might want to look at changing brushes. Eclipse can be used to write Processing code and will give you more flexibility when developing. A quick search on the forum will tell you how.

To end this chapter, something not only technically driven, but related to the whole recent Processing phenomenon and the future of the platform. Karsten Schmidt, a well respected programmer and Processing user, pointed out very pertinent comments about the dangers of having popular, easy to access programming resources within the creative world. The post he did on his blog [54] provoked substantial reactions within the community [55] and I would only encourage everyone serious about creating things with Processing to read through it. A post read 5000+ times can't be a bad post.

Thank you for reading this article. I sincerely hope it will help you in your artistic practice, and should you decide to use Processing, I can't wait to see what you will come up with.

Happy coding!

Notes

[1] Ben Fry : http://benfry.com/

[2] Casey Reas : http://reas.com/

[3] Aesthetic + computation group : http://acg.media.mit.edu/

[4] Wikipedia page about the MIT media lab : http://en.wikipedia.org/wiki/MIT_Media_Lab

[5] Design by Number (DBN) web site : http://dbn.media.mit.edu/

[6] John Maeda : http://plw.media.mit.edu/people/maeda/

[7] Processing Beta release announcement by Casey Reas
http://processing.org/discourse/yabb_beta/YaBB.cgi?board=Collaboration;a...

[8] Processing official web site : http://processing.org/

[9] Processing forum : http://processing.org/discourse/yabb_beta/YaBB.cgi

[10] people behind processing : http://processing.org/people.html

[11] We feel fine, artistic approach to data visualisation onilne : http://www.wefeelfine.org

[12] Flight pattern by Aaron Koblin http://users.design.ucla.edu/~akoblin/work/faa/

[13] Complexification.net, Jared Tarbell's online gallery : http://complexification.net/

[14] Evolution zone, marius Watz's portfolio http://evolutionzone.com/

[15] Karsten Schmidt's (aka toxi) blog : http://www.toxi.co.uk/blog/

[16] Lovebyte generative identiy by Karsten Shmidt and universal everything

http://www.toxi.co.uk/blog/2007_04_01_archive.htm

[17] Nike one video by motion theory : http://www.motiontheory.com:16080/nikegolf/

[18] Magnetosphere iTune audio visualiser http://software.barbariangroup.com/magnetosphere/

[19] Flight 404, Robert Hodgin blog http://www.flight404.com/blog/

[20] Sonic wire Sculptor by Amit Pitaru : http://www.pitaru.com/sonicWireSculptor/framed/

[21] Philip Worthington's Shadow Monster project :
http://www.worthersoriginal.com/viki/#page=shadowmonsters

[22] Processing download page http://processing.org/downlad

[23] IDE Wikipedia page : http://en.wikipedia.org/wiki/Integrated_development_environment

[24] setup reference page : http://processing.org/reference/setup_.html

[25] background reference page : http://processing.org/reference/background_.html

[26] draw reference page : http://processing.org/reference/draw_.html

[27] Cartesian coordinate system Wikipedia page :
http://en.wikipedia.org/wiki/Cartesian_coordinate_system

[28] processing online reference : http://processing.org/reference/index.html

[29] processing online extended reference : http://processing.org/reference/index_ext.html

[30] class reference page : http://processing.org/reference/class.html

[31] Java comparison page : http://processing.org/reference/compare/java.html

[32] jikes web site

[33] Java.com

[34] Eclipse development platform : http://www.eclipse.org/

[35] Processing development web site : http://dev.processing.org/

[36] MaxLink library : http://jklabs.net/maxlink/

[37] JMyron library : http://webcamxtra.sourceforge.net/

[38] Processing libraries page http://processing.org/reference/libraries/index.html

[39] Hardware page (Arduino and Wiring) : http://hardware.processing.org/

[40] Processing mobile web site : http://mobile.processing.org/

[41] Wiring : http://www.wiring.org.co/

[42] Arduino : http://www.arduino.cc/

[45] Processing hacks : http://www.processinghacks.com/

[46] Stellar snippet palace : http://snippet.seltar.org/

[47] Daniel Shiffman nature of code http://www.shiffman.net/teaching/nature/

[48] Processingblogs : http://www.processingblogs.org/

[49] processing flickr pool : http://www.flickr.com/groups/processing/

[50] build with processing : http://builtwithprocessing.org/

[51] youtube processing group : http://youtube.com/group/processing

[52] Vimeo video tagged with processing : http://www.vimeo.com/tag:processing

[53] Processing related books : http://processing.org/learning/books/

[54] Toxi's rant : http://www.toxi.co.uk/blog/2006/01/note-this-article-is-using.htm

[55] Answer to toxi's comments:
http://processing.org/discourse/yabb_beta/YaBB.cgi?board=Collaboration;a...

[56] Procedural programming Wikipedia page : http://en.wikipedia.org/wiki/Procedural_code

Images

[1] Jared Tarbell

Bubble chamber CC Attribution 2.0 Generic (http://creativecommons.org/licenses/by/2.0/deed.en_GB):

http://flickr.com/photos/generated/15448/

[2] Marius Watz

Object #1 (Attribution-Non-Commercial-Share Alike 2.0 Generic)

http://creativecommons.org/licenses/by-nc-sa/2.0/deed.en_GB

http://flickr.com/photos/watz/336839822/

[3] Av 06 - Sage Gateshead - Illuminations 01 (Attribution-Non-Commercial-Share Alike 2.0 Generic)

http://creativecommons.org/licenses/by-nc-sa/2.0/deed.en_GB

http://flickr.com/photos/watz/108738341/

[4] Karsten Schmidt

voxscape3 (Attribution-Non-Commercial-No Derivative Works 2.0 Generic)

http://creativecommons.org/licenses/by-nc-nd/2.0/deed.en_GB

http://flickr.com/photos/toxi/157372173/in/set-72157594145899116/

[5] We Feel Fine logo, image courtesy of Sep Kamvar

http://www.wefeelfine.org/

We Feel Fine copyright, All right reserved

[6] Screenshots of the IDE, images by the author

‹ Blender: Working with 3D up Working with sound ›

Working with sound

By admin
Published: 08/22/2007 - 14:15

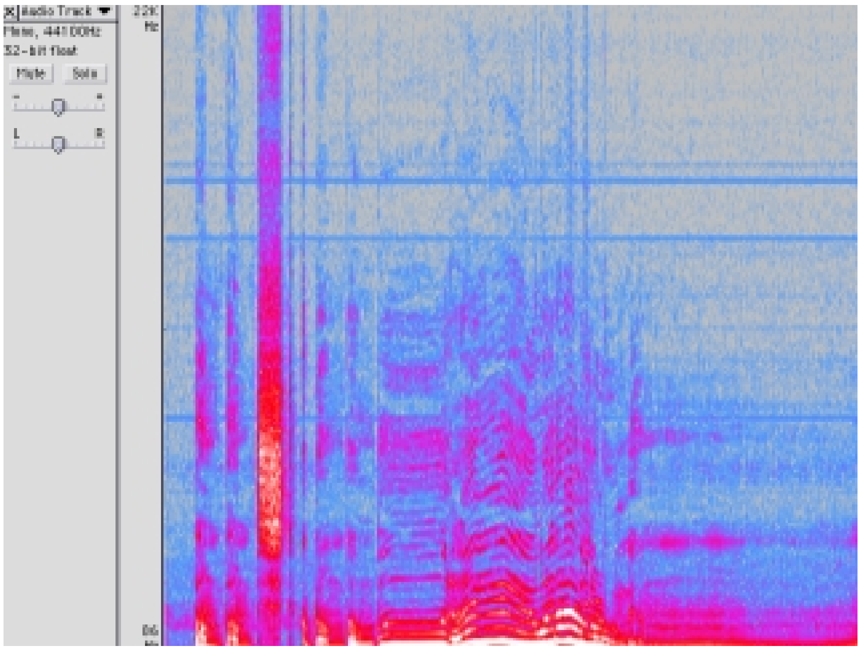
Thor Magnusson , November 2007

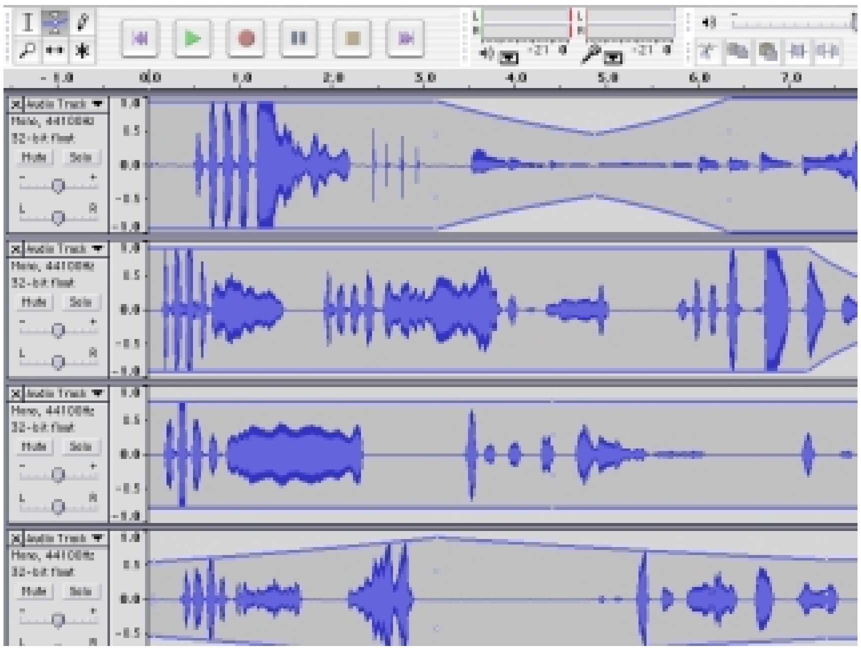
FLOSS Audio Tools... How fantastic! There is a whole world of free and exciting instruments at our disposal for play and work. These instruments try to cater for all the needs that the musician or the sound artist might have. And what is better: if you program computers, you can get the code source of the software and adapt it to your own specific work pattern, thus transforming yourself from being a mere consumer to a co-author of the software. At the moment FLOSS software provides most of what commercial software can provide, and sometimes more.

Normally someone decides upon making a specific software tool when he or she wants to perform a certain task and there is no existing tool that does the job. Software is based upon people's needs and work habits. There might exist a tool that performs a certain task, but not in the same way as is desired, so a new tool is created. If we look at the question of user needs from a very basic overview position, we can define the following categories: **Audio Editing** (for recording and processing sound); **Sequencing** (for layering sound events as tracks on a timeline and perhaps apply effects on the tracks); **Score writing** (creating musical scores on staves or piano roll interface. This is software based upon the old tradition of writing music as notes on paper); **Virtual Instruments** (tools that allow you to generate sounds through events such as commands from a sequencer or input from hardware such as a MIDI keyboard); **Sound Feature Analyser** (for analysing the nature of the sound: its timbre, temporal onsets and amplitude); **Patchers** - **Algorithmic/Interactive/Generative Composition** (for working with formal structures, algorithms and generativity - and that's what music essentially is). In the following sections   we will look at some of the FLOSS tools that are found in each of these categories.

Audio Editing

The most basic need of anyone working with sound is the capability to record it into the computer. This requires a microphone and a soundcard (inbuilt in most computers, but for good quality people buy special soundcards) that digitizes the analog signal from the microphone. Once the sound is digitized (through the soundcard) it can be recorded onto the hard disk with a sound editor and represented graphically in various modalities. The most popular FLOSS audio editor is Audacity [1] . It allows you to record sounds on multiple tracks, process them with virtual effects such as the LADSPA [2] or VST, and export the sound in various known audio formats, such as wave, ogg vorbis [3] (open source compression format) or mp3.





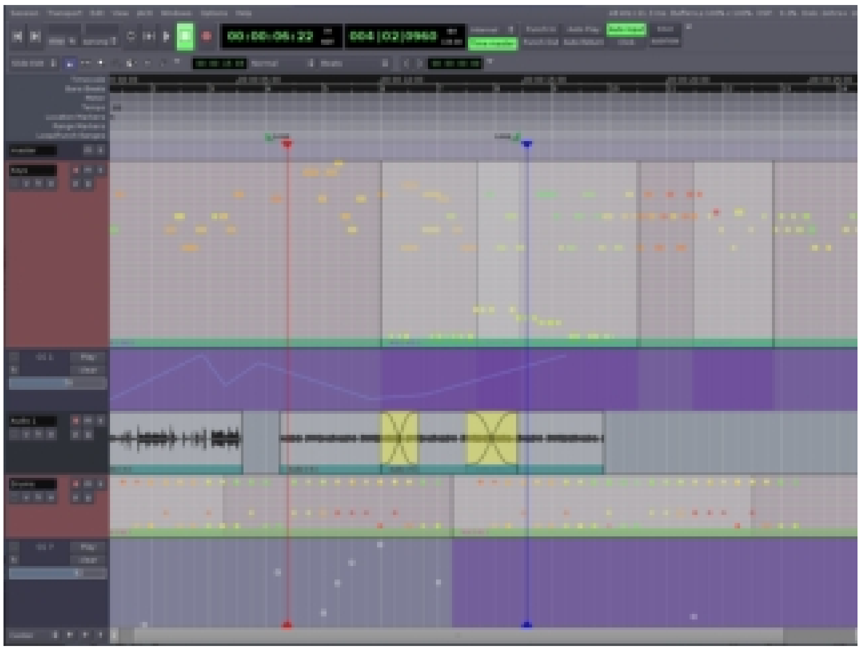
Spectral and waveform view in Audacity

Audio Editors perform tasks that the tape would have done in the pre-digital age, but they add the powers of analysis, graphical representation of the sound (very difficult in the pre-digital age), multiple and easy cutting and pasting, and high quality digital signal processing. Audacity performs tasks that go beyond simple audio editing such as multi-tracking and it has its own scripting language called Nyquist which can be used to generate sound and manipulate or generate MIDI data. Audacity exists for Linux, Mac OS and Windows.

Other editors include Snd [4] and WaveSurfer [5]

Sequencing

Sound Sequencers are basically a digital implementation of the multi-track tape machines that were found in recording studios in the latter part of the 20th century. Sound sequencers can layer sounds into tracks that play at the same time. You can either record directly into the track or import a sound from the hard disk of the computer. Two typical usage situations would be: a) a musician that records sounds in the field or downloads them from the net [see http://freesound.iua.upf.edu]. She manipulates them in an audio editor and then imports them into a sequencer in order to layer them and create a formal structure. b) a band with many instruments creates a demo recording by recording each instrument through a multichannel sound card. Each instrument is recorded live into the respective audio tracks of the sequencer. Later they then process and mix the tracks before bouncing down to a soundfile.



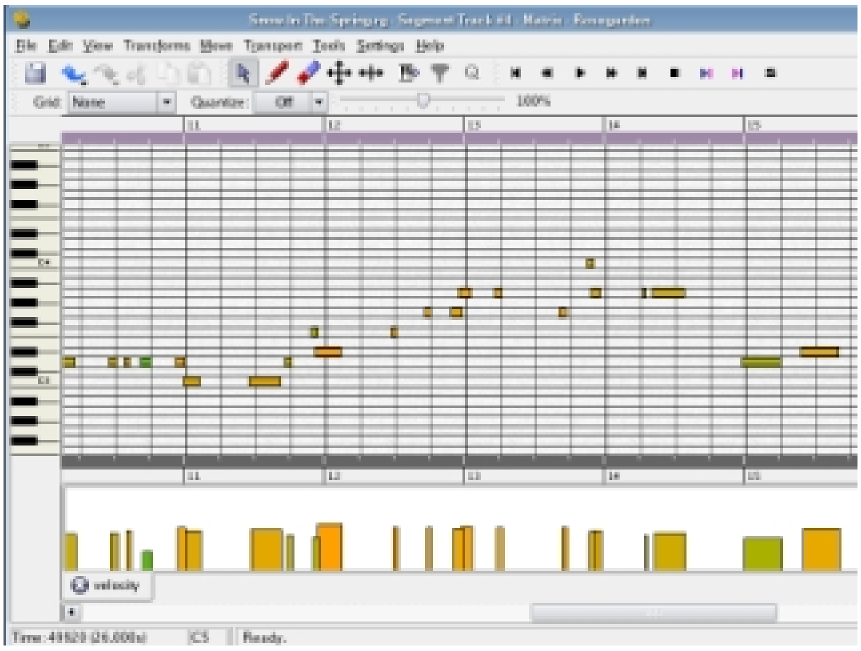
Screenshot of Ardour

Ardour [6] is the most impressive free and open source multi-track sequencer to be found at the moment. It compares to software such as Cubase, ProTools or Logic. It supports multi-track recording, audio processing (using native effects, LADSPA or VST), MIDI recording and manipulation, virtual instruments and post-production. It exists on Linux and Mac OS X.

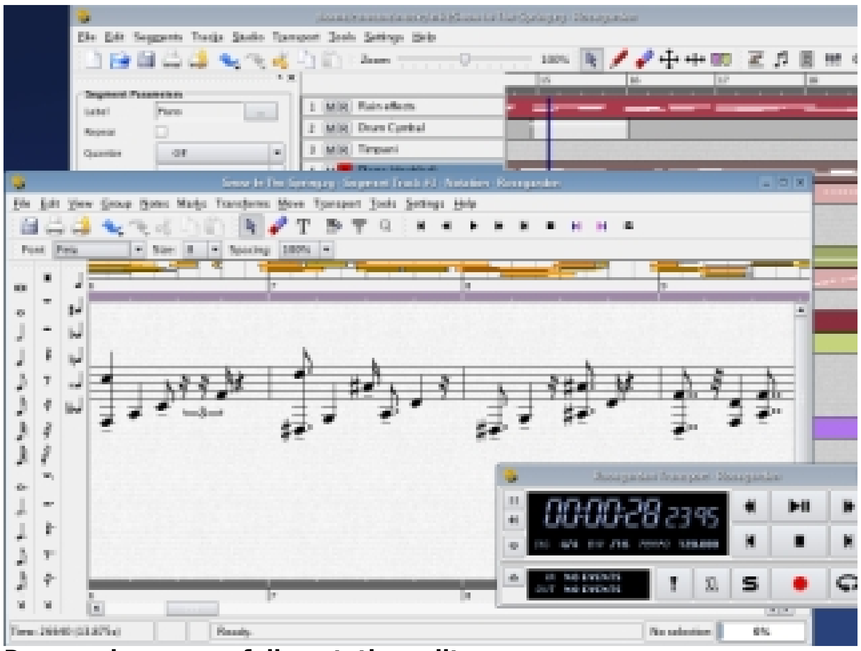
Score Writing

This category overlaps with "Sequencing" but it might have a different user focus. Before computers were powerful enough to deal with real audio (with 16 bit, 44100 sample rate) they were often used to create scores that would be played out through MIDI to hardware synthesizers or samplers. The score-writing tools could often switch between various representational modes such as the piano-roll (where you would see a vertical piano keyboard on the left and then time would be represented horizontally) or the good old 5 line stave. Obviously the piano roll was better suited for people without formal music education. The most popular software in the late 1980s would be Cubase on the Atari computer.

Score writing software is not focusing on recording audio in real-time but is aimed more at the composer who wants to compose music by arranging notes and let the software play the score in order to hear the results. Rosegarden [7] is perhaps the best FLOSS audio software for writing scores. It has to be noted here that Ardour can also be used for arrangement of MIDI notes and Rosegarden in turn records live audio, it's just that the focus of the two tools are different.



Rosegardens Matrix editor


Rosegardens powerfull notation editor

When a MIDI score has been made in Rosegarden the ideal software to set up and print out scores is called Lilypond [8] . It accepts various formats such as MIDI and MusicXML but it also has its own scripting protocol that can be written out from any programming language. Lilypond does great job in writing out beautiful and logical scores for musicians to play.

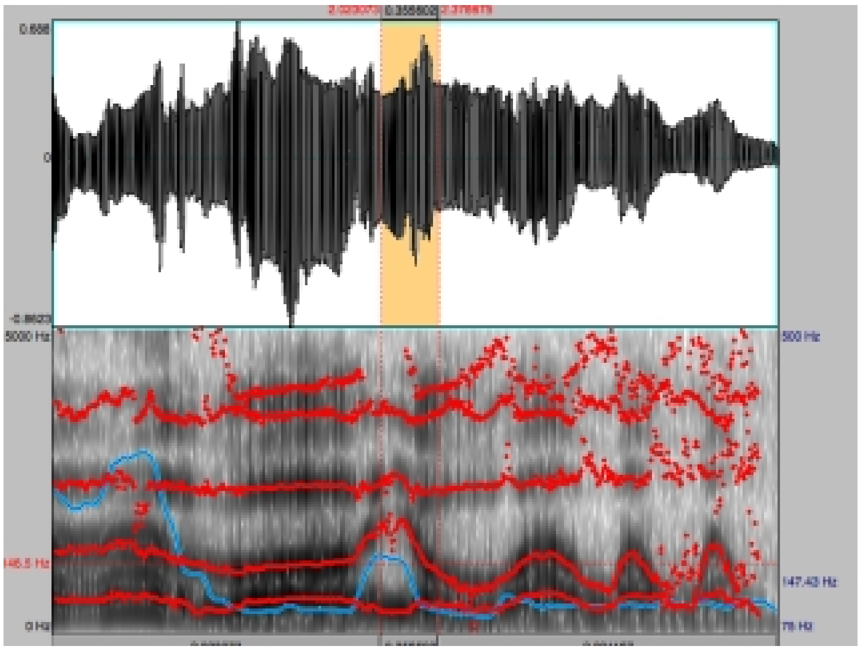
Virtual Instruments

Above we talked about the score sequencer software for arranging notes and how they would send MIDI notes out to external hardware to generate the sound. In the late 1990s affordable computers became powerful enough to do real-time synthesis and virtual instruments and effects were added to the flora of audio tools. Steinberg, the company behind Cubase (then the most popular sequencer), created the VST (Virtual Studio Technology) audio plugin architecture. The software development kit is open and it has resulted in thousands of developers continuously creating new effects and instruments. Apple has created their own open architecture called Audio Units (or AU). Other architectures include FreeST for Linux which makes it possible to use VST plugins on Linux. The native Linux audio plugin architecture is called LADSPA [9] (Linux Audio Developers Simple Plugin API) and it is supported by most of the software mentioned in this article.

Virtual instruments can be used in various setups. For example you could plug your MIDI keyboard into the computer and use a host program to load up the instruments and effects you want to use. You can create a chain of instruments and effects, typically choosing a sound, say guitar, and then route that through effects such as bandpass filters, reverb, delay or distortion. You could then use the host program to record what you play for later editing. Another usage would be to compose directly by writing notes, perhaps using Rosegarden, and then listen to what you write by playing the score through a virtual instrument.

Sound Feature Analysers

Musicians, artists and scientists often need to analyse the sound they are working with. They might be curious to look at the spectral (the distribution of frequencies) qualities of the sound and change its harmonic structure through manipulating the sound’s partials. Praat [10] is a great application for this purpose. It can do spectral, formant, pitch, intensity and other types of analysis. It is particularly well suited for speech studies and it provides neural network algorithms for learning and synthesis algorithms for speech synthesis.



Screenshot of Praat

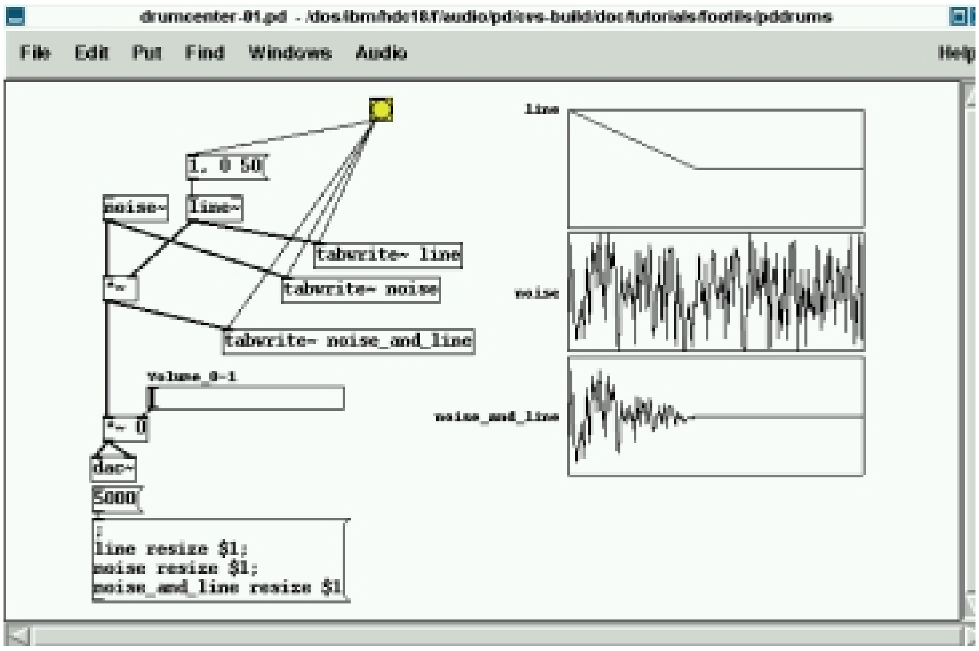
Tapestrea [11] from the productive Princeton Audio Lab is another interesting and fun sound feature manipulator. Like Audacity it has scripting capabilities and it uses the ChucK programming language [20] for scripting.

Other applications include Sonic Visualiser [12] , Baudline [13] and RtFFT [14] or snd-peek [15] .

Patchers - Algorithmic/Interactive/Generative

This category of “patchers” is where FLOSS software blows away the commercial world in quality, ingeniousness and experimentation. The computer does not merely have to imitate old technology from our physical world. It allows us to create our own tools according to our own ideas of how music should be or how tools should behave. For that purpose there are many different patchers out there: basically environments where you can create your own synthesis graphs, control structures and interfaces.

Historically the patchers originate from the Music N languages made by Max Matthews in the 1950s and 60s. The idea here is to create unit generators that generate the sound and then provide a programming environment to control them. This is ideal for sound synthesis and algorithmic composition. From the user interaction perspective, we can divide the patchers into two categories: graphical programming environments such as Pure Data [16] or jMax [17] and textual programming environments such as SuperCollider [18] , CSound [19] and ChucK [20] .

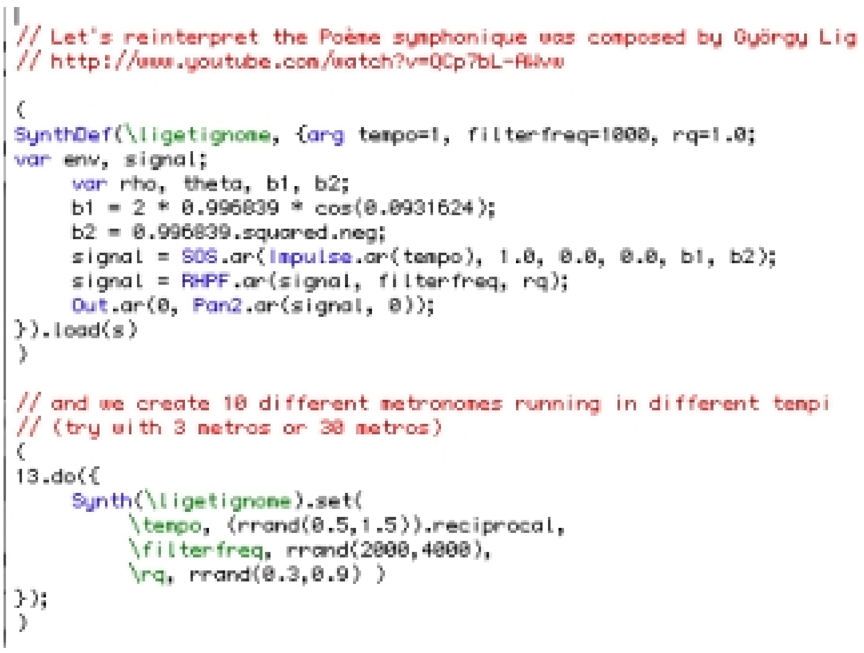


Screenshot of Puredata

The patchers allow you to create your own program, so you could create an algorithmic composition based on rules, a generative composition that plays music that's never the same, sound toys, interactive installations (using sensors, audio, video and motors), musical analysers, a thought platform for music theory, explore sound physics or psychoacoustics and so on... These programming languages are made for sound, but try to shy away from incorporating too much music theory. Therefore you won't find any 16 step sequencers or 12 tone keyboards. It's up to you (and not some software designer) to define the conceptual foundations of your music.

It varies what people prefer when choosing their environment. In open source software, the most important things to consider when choosing a platform to learn (apart from the sound quality and the way you embrace the environment) is the status of documentation, stability, continuity and community. A helpful mailing list community is characteristic of all the above mentioned patchers where more experienced users help less experienced users to understand their way through the theoretical maze one can find they are.

People have different cognitive styles. For some Pd is the ideal platform for composing their music or instruments as the interface and the programming language are one and the same thing. It provides a graphical data-flow representation of the internal functionality of the computer. For others, textual programming languages like SuperCollider can be more efficient for what the goal is. The power here is that of writing classes, compact code, ease of use and different type of representation. Each of these environments have their cons and pros and it is only meaningful to compare them when thinking about some specific task that needs to be performed.



```
// Let's reinterpret the Poème symphonique was composed by György Lig
// http://www.youtube.com/watch?v=QCp7bL-AWvw

(
SynthDef(\ligetignome, {arg tempo=1, filterfreq=1000, rq=1.0;
var env, signal;
	var rho, theta, b1, b2;
	b1 = 2 * 0.996839 * cos(0.0931624);
	b2 = 0.996839.squared.neg;
	signal = SOS.ar(Impulse.ar(tempo), 1.0, 0.0, 0.0, b1, b2);
	signal = RHPF.ar(signal, filterfreq, rq);
	Out.ar(0, Pan2.ar(signal, 0));
}).load(s)
)

// and we create 10 different metronomes running in different tempi
// (try with 3 metros or 30 metros)
(
13.do({
	Synth(\ligetignome).set(
		\tempo, (rrand(0.5,1.5)).reciprocal,
		\filterfreq, rrand(2000,4000),
		\rq, rrand(0.3,0.9) )
});
)
```

Screenshot of some SuperCollider code

Conclusion

All musicians or artists have their own agendas and goals and it is impossible to tell which tools are suitable for each and every person. It is now up to you to download and install these environments and see where they take you. Read the tutorials and subscribe to the mailing lists. There are always people there to help you, and it is easy to unsubscribe again. And remember that people have put their free time into developing these dynamic, free and open source programs. Using them can therefore be exciting experience where you will establish personal relationships with other users of the tools and their developers. New ideas or discussions about the tool are always welcome. There are many good reasons to use free and open source software, but perhaps the most important one is this change of status the user will experience from being a mere “customer” of a company or a “consumer” of software, to a fellow “user” or “co-developer” of it.

Tip: Want to try?

Would you like to try these programs without having to install all of them on your machine? You then have the possibility of running a "live-cd": basically a Linux operating system that runs from a CD. Planet CCRMA [21] , pure:dyne [22] , Ubuntu Studio [23] or 64 Studio [24] all provide you with a Linux distro on a CD that can be run on your computer just by booting up from the CD drive. This way you can explore and try out most of the software that we have covered in this article.

Notes

[1] http://audacity.sourceforge.net

[2] http://www.ladspa.org

[3] http://www.vorbis.com

[4] http://ccrma.stanford.edu/software/snd

[5] http://www.speech.kth.se/wavesurfer/index.html

[6] http://ardour.org

[7] http://www.rosegardenmusic.com

[8] http://lilypond.org

[9] http://www.ladspa.org

[10] http://www.fon.hum.uva.nl/praat

[11] http://taps.cs.princeton.edu

[12] http://www.sonicvisualiser.org

[13] http://www.baudline.com

[14] http://www.music.mcgill.ca/~gary/rtfft

[15] http://soundlab.cs.princeton.edu/software/sndpeek

[16] http://puredata.info

[17] http://freesoftware.ircam.fr/rubrique.php3?id_rubrique=14

[18] http://supercollider.sourceforge.net

[19] http://www.csounds.com

[20] http://chuck.cs.princeton.edu

[21] http://ccrma.stanford.edu/planetccrma/software

[22] https://devel.goto10.org/puredyne

[23] http://ubuntustudio.org

[24] http://64studio.com

Images

Rosegarden screenshots from http://www.rosegardenmusic.com

Puredata screenshot courtesy of Frank Barknecht http://footils.org

All other images courtesy of the author.

- Pure Dataflow - Diving into Pd

‹ Working with graphics: Processing up Pure Dataflow - Diving into Pd ›

Pure Dataflow - Diving into Pd

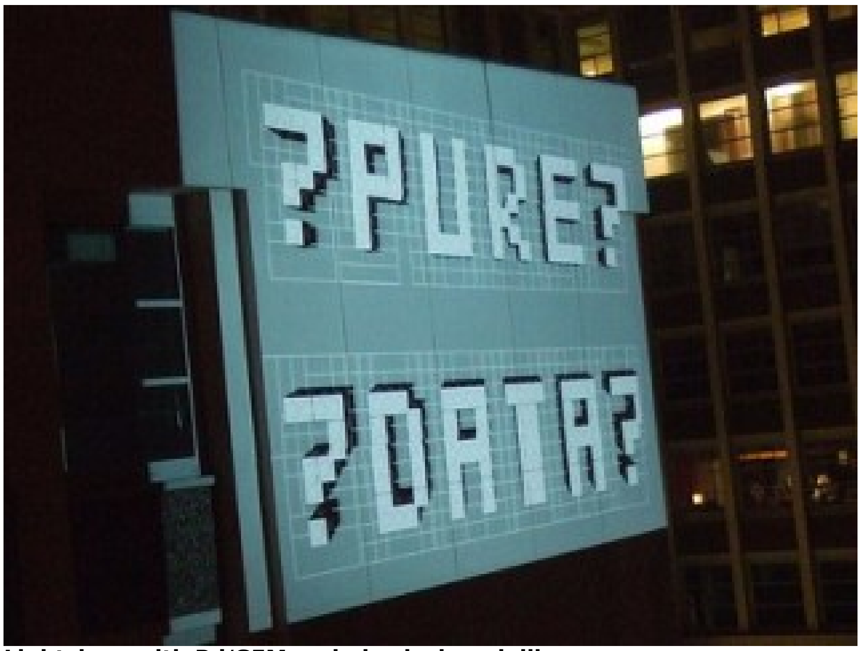
By marloes
Published: 09/18/2007 - 13:43

Frank Barknecht , September 2007

This article introduces the possibilities of the software Pure Data (Pd), explains a bit why it's so popular among artists and shows what Pd can be used for. The goal is to help artists decide if Pd is a tool for their own work.

Intro

Pure Data, or PD for short, is a software written by mathematician and musician Miller S. Puckette [1] . It has become one of the most popular tools for artists working with digital media. Originally conceived in the late 90s as an environment to create sounds and to compose music, it was soon extended by modules to work with video and graphics. Pd is freely available for no cost, and it is Free Software in that the source code can be obtained, modified and distributed without restrictions as well. Pd runs on many operating systems including the big three: Linux, OS-X and MS-Windows.



Lightshow with Pd/GEM and physical modelling

Over the last decade, the user base of Pd has constantly grown and many of these users have also turned into developers who work on the software itself and make their own extensions. The sheer number of available extra packages may make Pd a bit intimidating to beginners, so this article will present Pd itself and give an overview about the various extensions available. The goal is to enable interested artists to make their own choice about whether Pd is a useful tool for their work.

History repeating

What can I do with Pd? That seems to be such an easy question. But actually it is one of these innocent and smart questions, that children ask and parents cannot answer. At its core, Pd is a full-blown programming language and in theory you can write every possible computer program using Pd. So one answer to the question "What can I do with Pd?" could be: You can use Pd to make a computer do everything that a computer is able to do.

However from everyday experience with a computer it is known that a computer often doesn't like to do at all what a human wants it to do.   Why is this printer suddenly refusing to print? I know it can print, it has printed before! Assuming the hardware is working, many issues humans have with computers are based on communication problems. How to make the computer understand what humans mean is a fundamental problem of the digital era and programming languages try to give one solution to it.

Most programming languages today are based on writing text. Text is wonderful: You can read this article and hopefully understand roughly what I'm talking about. (Although I'm not even talking at the moment!) If you don't understand what I write, you can write your own text with your questions and mail it to me. Text is quite easy for a computer to understand as well: It can be divided into single words or characters, that follow each other. Most of the time text is a one-dimensional medium: It connects words left to right (or right to left or top to bottom depending on local conventions).

But text doesn't have to be just left to right, it can be placed in two dimensions as well and still make sense. A spreadsheet is an example of text that is presented in a two-dimensional fashion.

To make a computer understand two-dimensional text, special rules for specifying the connections between words are necessary. In the tables of a spreadsheet, a rule could for example be, that words aligned in the same column have some sort of similar meaning.

Since the early days of the computer, scientists have looked for ways to make computers understand two-dimensionally arranged program text as well. A seminal document in this regard is Bert Sutherland's Ph.D. thesis `"The On-line Graphical Specification of Computer Procedures" [2] from 1966, where a lot of the concepts that are now common in graphical programming languages and dataflow environments like Pd are discussed at a very early stage.

Even though several computer scientists followed Sutherland's pioneering work, somehow graphical programming didn't really take off in the mainstream computer world. Besides the LABView [3] , a tool popular among industrial engineers, there is only one two-dimensional programming language that has found massive use - and that is the paradigm used in the "Max"-family of software which Pd is a member of.

Many of the visual programming languages employ a style of programming called "dataflow". The idea behind this is that changing the value of some variable in the system will make every object that somehow is connected to that variable recalculate its state. During that recalculation of course some other variables may change their values as well, forcing other objects to recalculate their state as well. In the end, changing one value at some place may start a whole torrent of calculations, and that's what gave the principle its name: the changes seem to flow through the system until everything is updated accordingly.

"Open Circuit", an installation transforming the dataflow principle into the physical world - made with Pd

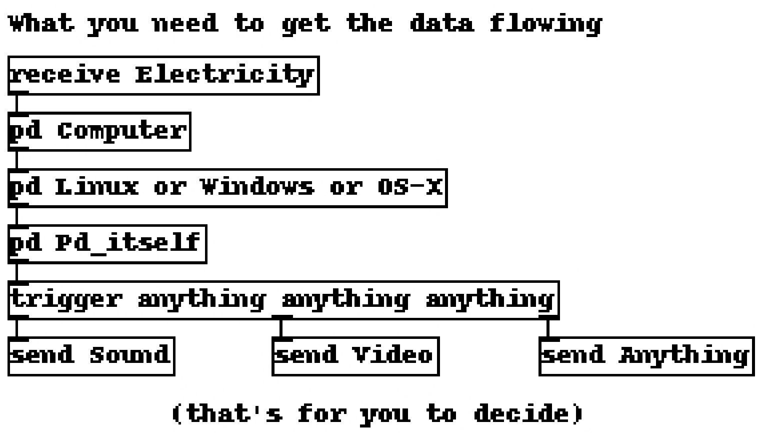
Again the spreadsheet is an example of this principle in action: Often a user is interested in the balance of a complete column of numbers in that spreadsheet, so the software provides a "SUM" function, which can watch many number fields at the same time and will update the sum every time a field changes its value. A modular software synthesizer like Pd constantly has to deliver new audio data to the soundcard, which has to react to new data immediately, for example as soon as a key on a connected keyboard is pressed or released.

While dataflow-inspired software often comes together with visual, two-dimensional programming, there also are pure text based programming languages that support dataflow ideas. Oz/Mozart [5] and Nova [6] are two examples.

Max and his family

Before Miller Puckette wrote Pd he was working at the IRCAM [4] institute in Paris and developed the first versions of Max, named after the "father of computer music" Max Mathews. Max became a very successful program and has many incarnations: Maybe the best known among these is Max/MSP, a commercial software produced by US-Company Cycling'74 [7] that could be yours for about 450 US-Dollar. With the Jitter extension by Cycling'74 Max/MSP also can handle video data and 3D-graphics.

The french IRCAM continues to offer its own version of Max, and IRCAM even made an offspring of it called jMax available as Open Source software some years ago. But jMax didn't enjoy widespread use, maybe because Pd had already filled the niche of a free version of Max. After IRCAM did some restructuring of its Open Source team, development of jMax practically came to a halt. Today jMax is of rather little importance, at least outside of IRCAM.


What you need to get the data flowing
receive Electricity
pd Computer
pd Linux or Windows or OS-X
pd Pd_itself
trigger anything anything anything
send Sound
send Video
send Anything
(that's for you to decide)

Pd is free software and can be used in an almost completely free environment

That leaves Pd as the most popular Max-like software that's available with source code as Free Software. While Max/MSP only runs on the commercial operating systems MS-Windows and Apple-OS, Pd additionally works on Linux and several other systems. (As Pd is developed mainly on Linux, things often even run a bit smoother there, though the other systems are very usable as well.) In the end Pd can make you completely independent from having to buy commercial, closed source software at all. And if you can get a free old PC somewhere the only thing you need to pay for to work with Pd is electricity.

Diving into Pure Data

Pd shares the basic workflow with the other programs in the Max family: Starting with a blank page, the user populates this empty canvas with little boxes and connects these boxes with patch cords, through which the boxes can exchange messages and other data. The end result is called a patch in the Max/Pd-lingo.

The object boxes take their functionality from the names they've been given. Pd doesn't use any nifty icons inside these boxes: Although it is a visual programming language, Pd at its heart is a software to work with (spatially arranged) text. Or as the developer of the Pd extension GridFlow [8] , Mathieu Bouchard, once put it: A word says more than a thousand pictures. And Pd knows this.

Pd comes with about 120 built-in object names (also called "classes"). The available objects can be extended either by installing some of the extra packages, that we will see soon, or by writing custom objects yourself. Binary objects for Pd are generally called "externals" while extensions written in the graphical Pd language itself are known as "abstractions".

Miller Vanilla: What the Pd core language can give you

The objects in the core version of Pd as distributed by Miller S. Puckette[1] himself on his website mainly deal with working on messages and numbers and with creating, analyzing and modifying sound. For a beginning user it is crucial to make oneself familiar with the core objects and how they interact. Although their number is relatively small, it is already possible to build powerful applications by using them in a smart and creative way. An advantage of restricting yourself to using core objects is that it's easy to share patches with others without forcing them to install special extensions.

Learning Pd is like a lot like learning a natural language: First one has to built a certain word pool and get familiar with the basic vocabulary of Pd. Then out of these "words" a user has to create "sentences", that is, some small patches and idioms, which instruct the software to fulfill certain tasks. As already mentioned, core Pd excels in making sound and in composing: With a bit of exercise it is possible to record sound, modify it in real-time, synthesize new sounds, apply the timbre of one sound to another and so on. The message and mathematical objects in Pd can be used to write algorithmic compositions that then are played back for example by external synthesizers or other software instruments - or of course in Pd itself.

The core Pd unfortunately doesn't handle video or 3D graphics itself (yet), so video artists will soon need to install one of the graphics extensions. The time invested in learning the core objects is not wasted however, as the graphics objects are connected with each other in the same fashion as the core objects.

Libraries: Important binary extensions

Some extensions for Pd are so useful, that practically every Pd user already has them installed. This includes a number of objects developed at the Institute Of Electronic Music And Acoustics [9] (IEM) in Graz, Austria, collected in libraries like zexy, iemlib, iemmatrix and others. These objects extend Pd by making certain message operations easier, they allow fast matrix calculations or provide excellent audio filters.

Cyclone is another important library, especially for users coming from Max/MSP: It clones many Max objects that are missing in the core Pd and even includes a useful, albeit a bit limited, importer for Max patches. The author of this text has used Cyclone extensively to port the approx 150 objects in Karlheinz Essl's Realtime Composition Library [10] for Max to Pd without having to run Max/MSP even once.

Externals are normally shipped in source code form from the Pd repository website at pure-data.sourceforge.net [12] , so they would need to be compiled before you could use them. While compiling is no big deal on Linux, users of other operating systems are more used to packages, that include binary objects, that they can run directly. The pd-extended package makes this very easy by bundling most of the available extensions with a version of Pd itself, that can be installed quickly.

Abstraction libraries

Besides binary extensions Pd can also load objects written with the Pd language itself. Many of these so called abstractions are collected into libraries as well and can be installed by just copying them into a directory in Pd's search path. Pd-extended includes most abstraction libraries, too, so if you install this, you're ready to roll.

One of the biggest collection of its kind is developed by Pd users in Montreal under the name pdmtl [11] . It includes many synthesizer objects, sequencers and lots of useful small tools. It's a nice touch that the whole pdmtl collection comes with excellent documentation. Every abstraction is explained both with a Pd help patch and on the pdmtl wikisite.

RTC-lib by Karlheinz Essl, now also available for Pd, was already mentioned: This library is especially interesting for composers, as it includes many modules for doing serial or 12-tone music and lots of objects for working with randomness.

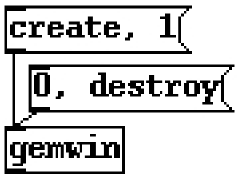
When developing Pd applications, one often has to transform a certain set of data into another realm, a process called "mapping". A library of abstraction with the same name makes these transformations easier. The "mapping" library is especially useful if you work with sensor data.

Abstractions do not only extend the functionality and vocabulary of Pd, they are also a good way to learn more about how Pd works, because it's possible to open an abstraction and look at the way it was built.

Visible Pd: Extensions for GFX and video

GEM

Several extensions to Pd allow working with video and graphics as well. The oldest among these is the "Graphics Environment for Multimedia" GEM [13] . It was first developed by Mark Danks, who later moved on to work in the game industry. IOhannes m zmoelnig (sic!) from Austria took over maintenance of GEM, but as with Pd, GEM is a group effort, where several developers are working as a team.



The most basic GEM patch

GEM's main focus is 3D graphics: For this, GEM uses the OpenGL-library, an industry standard for dealing with 3D graphics (and of course 2D as well). GEM also handles video input with its "pix"-objects: It can read data from a camera or a file, transform and analyse the movies in various ways for example to do motion tracking and display them as animated textures on the screen.

Packets: PDP/PiDiP and pf

"Pure Data Packet" PDP [14] aims at image and video processing and is an ideal Pd extension for VJs. Some objects that were built on top of PDP are collected in the PiDiP [15] library, so to use PiDiP, you first need to install PDP. You can use PDP/PiDiP for real time video processing, blending, mixing or feedback constructs, do motion detection and even transform an image or movie directly into sound.

Tom Schouten, the author of PDP, is currently working on a related project that is going to supersede PDP in the end: PacketForth [16] or "pf". pf is a programming language itself that can be run as a standalone program, but also works inside of a special Pd object. When used as the latter, pf can be used to "clone" the objects of PDP, so that Pd patches written for PDP will continue to work with pf as well.

PDP in action: french artist Benjamin Cadon performing at make art festival 2007

GridFlow

A third popular library for video processing (and more) is GridFlow [8] . Mathieu Bouchard originally wrote it for IRCAM's jMax as well as Pd, but with the descend of jMax he concentrated on Pd and today more or less has dropped jMax support.

Like PDP/pf, GridFlow internally uses a full blown programming language, Ruby. This makes it possible to test and use GridFlow independently from Pd to some extent. GridFlow's objects are rather low level and to develop Pd applications with it, a mathematical background plus knowledge of image processing algorithms are useful if not necessary.

Talk to me: Connecting Pd to other software

Quite often it's necessary to connect Pd to other software, that may not even run on the same machine. To make different programs talk to each other, they have to agree on a certain language that everyone involved can understand. One such protocol comes with Pd itself, it's called FUDI and used in the [netsend] and [netreceive] objects in Pd. FUDI works by connecting two pieces of software with a network connection using the Internet Protocol (IP), and then lets them exchange simple lists of words or numbers that end with a semicolon. Because FUDI uses the Internet Protocol, which has powered the internet for many years, a lot of other programs can be connected to Pd using FUDI. Besides the ending semicolon FUDI is very free form. The players involved must somehow agree on the meaning of the messages sent on their own.

The Open Sound Control OSC [18] specification is simular to FUDI, but it's a bit stricter in regard to how messages need to look. Generally OSC messages have to start with a specifier that looks a bit like an URL or a filesystem path, for example "/trumpet/volume". After this specifier (the "OSC target") one or more values can follow, so that complete messages may look like: "/trumpet/volume 100" or "/player/tom go left". OSC messages are normally transported over IP-network connections as well, though the official specification also allows different kinds of connection.   In recent years, OSC has become a standard that is supported by many programs, among these a lot of commercial, closed-source applications like NI Reaktor or Ableton Live. Pd doesn't include OSC as one of the core objects, instead special externals are used, that already are included if you install the Pd-extended package.

The oldest among the common protocols to make (music) software talk to each other is MIDI. Originally conceived to simplify and standardize the physical connections between hardware instruments, it's also useful to connect programs. Pd includes a complete selection of objects for receiving and sending MIDI data both from hardware and software MIDI instruments.

Shaking hands: Connecting Pd to Hardware

If you want to use external hardware with Pd, again MIDI may be the easiest way, provided you have a piece of hardware that has a MIDI plug, like MIDI keyboards or MIDI slider boxes. Some specialized objects were written to connect so called Human Interface Devices to Pd, that is for example game controllers like joysticks, dance mats or the Wiimote.

Pd also is a good choice if you want to build your own instruments or generally want to interface with custom-made electronic devices. Many of these, like the popular Arduino [17] controller board, connect to the computer through a serial connection. The [comport] object is the low-level building block to activate such hardware in Pd. For Arduino there even exists a special object [arduino] that greatly simplifies working with this board.

Advanced Development

If you're already a programmer, chances are high that you can continue to use your favourite programming language and create your own extensions by writing Pd objects in that language. Pd itself is written in C, so externals written in C or C++ are very common. To get you started, a good tutorial is available. Many other languages are wrapped in special Pd objects: Ruby can be used through GridFlow as already mentioned, Python is possible with the py/pyext objects by Thomas Grill. Other languages like Lua, Scheme, Haskell, Forth or Java can be used as well.

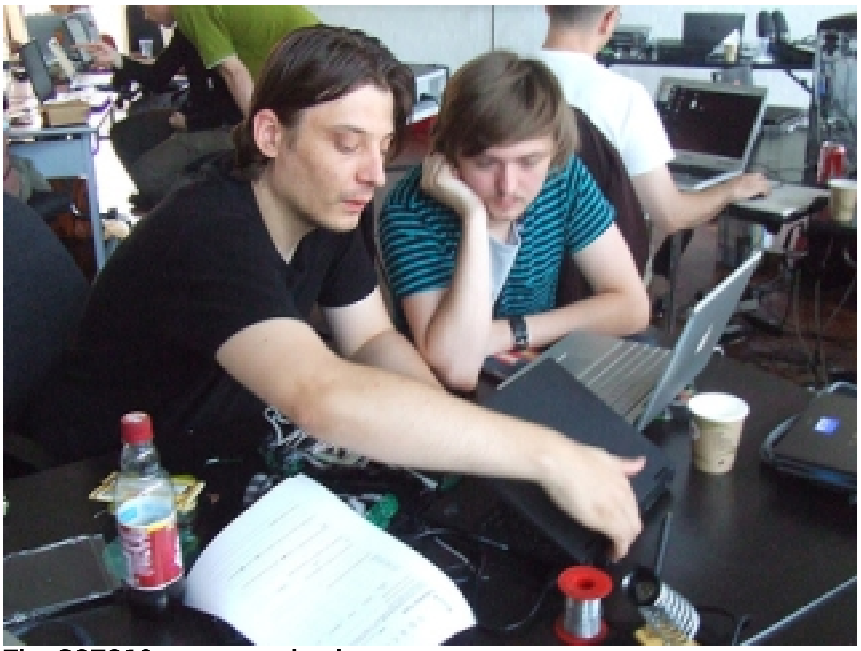
Pd siblings

Besides the version of Pd that Miller Puckette has continued to publish for more than a decade now, several related projects or spin-offs are in active development. They use the - freely available - source code of Pd as a base and built on it. DesireData [19] is one such project where a lot of improvements in the area of usability have been made. For example a lot of functionality can be accessed without having to use the mouse all the time, DesireData is heavily internationalized and generally looks much nicer than the original Pd. On the other hand some functionality is still missing at the time of writing, but DesireData definitly is a project to keep an eye on.

Vibrez [20] by Thomas Grill and others is a spin-off of Pd with many improvements regarding stability and usability. It's available for Windows and OS-X only. Vibrez is based on an older branch of Pd that introduced a reworked audio subsystem and allowed better performance with lower latencies especially on Windows. The work on this branch also kind of found its way into Nova [6] , a Pd-like system developed by Tim Blechmann, who also works on Vibrez. Nova itself doesn't have a graphical patcher system, instead it could be used as the underlying engine for such a system. In fact, at least one such patcher system already is in development.

The Pd community

Since Pd was introduced to the world in the middle of the 90s, it has found hundreds of users all over the world. Many of these communicate with each other using mailing lists, websites or the #dataflow [22] IRC channel. The puredata.info [21] website is a central hub to learn more about these activities and be informed about any news in the Pd scene. The IEM in Graz maintains the important mailing lists [23] : pd-list@iem.at is the list for general discussions on Pd and the first place to ask questions about Pd usage. Most questions are answered in less than a day. Developers of externals and the developers of Pd itself use the pd-dev@iem.at list to discuss and debate more technical issues. Both lists, especially pd-list, can be quite chatty and one should expect to get quite a lot of mails when subscribing. If you're only interested in important announcements, the pd-announce@iem.at list is a low-traffic list for exactly this.


The GOTO10 summer school

In the last years the number of workshops where Pd is taught is on the rise as well. Media centers and artist-run places quite often invite well-known Pd people to spread the knowledge. The programmer/artist collective GOTO10 [24] is famous for its Summer Schools (that sometimes happen in spring or autumn as well), that go on for one or two weeks covering Pd and related topics like physical computing or other synthesis and media software.

With freely available mail support, lots of tutorials online, affordable workshops everywhere, in the end there's no excuse to avoid learning at least a little bit of Pd. Promised: it will come in handy at some point.

Notes

[1] Miller S. Puckette: http://crca.ucsd.edu/~msp

[2]The On-line Graphical Specification of Computer Procedures: http://hdl.handle.net/1721.1/13474

[3] LABView: http://www.ni.com/labview

[4] IRCAM: http://www.ircam.fr

[5] Oz/Mozart: http://www.mozart-oz.org

[6] Nova: http://tim.klingt.org

[7] Cycling'74: http://cycling74.com

[8] GridFlow: http://gridflow.ca/

[9] IEM: http://www.iem.at

[10] Realtime Composition Library: http://www.essl.at/works/rtc.html

[11] pdmtl: http://wiki.dataflow.ws/PdMtlAbstractions

[12] pure-data.sourceforge.net: http://pure-data.sourceforge.net

[13] GEM: http://gem.iem.at

[14] PDP: http://zwizwa.goto10.org/index.php?page=PureDataPacket

[15] PiDiP: http://ydegoyon.free.fr/pidip.html

[16] PacketForth: http://zwizwa.goto10.org/index.php?page=PacketForth

[17] Arduino: http://www.arduino.cc

[18] OSC: http://opensoundcontrol.org/

[19] DesireData: http://artengine.ca/desiredata

[20] Vibrez: http://vibrez.net

[21] puredata.info: http://puredata.info#

[22] #dataflow: http://wiki.dataflow.ws/DataFlow

[23] mailing lists: http://www.puredata.info/community/lists

[24] GOTO10: http://goto10.org

Images

[1]   Lightshow with Pd/GEM and physical modelling. Photo by Chun Lee

[2] "Open Circuit" is an installation transforming the dataflow principle into the

physical world - and it was made with Pd. Photo by Sebastien Vriet

[3] Pd is free software and can be used in an almost completely free environment. Photo by Frank Barknecht

[4] The most basic GEM patch. Photo by Frank Barknecht

[5] PDP in action: french artist Benjamin Cadon performing at make art festival 2007. Photo by Sebastien Vriet

[6] Pay attention, class! At the famous GOTO10 summer school. Photo by Chun Lee

" >

span-->

‹ Working with sound up Working with others ›

Working with others

By admin
Published: 10/04/2007 - 08:32

Phil Chandler , September 2007

It is of course a truism, often repeated, [1] [2] [3] that the Internet has been the basis for a revolution in (remote) interpersonal communications, collaboration and data sharing. It is probably safe to say that there would be very few of the Free/Libre and Open Source (FLOSS) projects that exist today without the collaboration technologies the Internet supports. One of the many effects of the powerful tools FLOSS has put in to the hands of creative people is that it has potentially made them more independent. No longer are they reliant on specialists with access to expensive software and hardware to carry out aspects of their projects for them. Their limitations are now time and knowledge, not the lack of access. It is in fact precisely this issue that the Digital Artists' Handbook seeks to address, by providing authoritative information to guide practitioners in to new fields of endeavour.

The downside of this independence is that many artists find themselves more isolated, working alone at home rather than interacting with others at shared studios or where shared resources were previously found.

The Internet, being fundamentally a communications medium, offers potential solutions to this isolation, but the solutions themselves have, to date, largely dictated that collaboration happens in new ways, shaped by the technology. For some, the thousands of FLOSS coders for example, the tools have made possible projects that would otherwise be virtually inconceivable, but for other artists looking to enhance their existing practice with new digital methods the situation is perhaps more double-edged.

It maybe be useful to step back for a moment and consider what we mean when we talk about working, or collaborating with others. For a start it could be divided in to five broad types of collaboration:

Communication

This is the everyday backwards and forwards of discussion:

email: To most people the original and still the fundamental communication channel on the Internet. Although under constant assault by junk (spam) email, the simplicity, cheapness, robustness and asynchronous nature of the email system are it's fundamental advantages. There are probably few Internet users who are not already familiar with using email, and all internet service providers offer email services to users.

chat/instant messaging/IRC: where synchronous, near real time communication is required these text-based media have proved popular. The common theme is that users type what they want to say in to a chat window, and their words will appear almost instantly in the corresponding windows of those they are connected to. In all cases there are servers out on the Internet that act as hubs for this communication, and a level of complexity is introduced by needing to know which system the person you wish to converse with is using. Chat may be on a one to one basis or in a group situation. Client software is available for all common platforms, sometimes bundled with the operating system. These days systems tend to offer additional services such as the ability to send a file to someone you're chatting to, or even to use voice and video communication. In other words technologies such as chat, VoIP and video conferencing (see below) are converging. There are even websites that allow you to set up a chat session without even installing any software. [4]

Voice over IP (VoIP) / 'Internet telephony': This describes the technologies that use the basic protocols of the Internet to transport voice communications between users. This has allowed computers to be used as telephony devices, and for users to turn their high speed Internet connections in to cheap international telephone lines. Due to the way the Internet presently works it is often hard to achieve equal quality of service compared to a traditional circuit-based telephone connection, however the low cost has encouraged a boom in VoIP's use. It has been popularised by the proprietary Skype software [5] but many open source equivalents exist [6] [7] . As with chat services, you need to create an account on the VoIP service you wish to use, and install the relevant software. Whilst computer to computer connections are generally free, connecting to a traditional phone is usually a paid for service.

Content management systems (CMS), wikis and discussion forums: Although these equally fall in to the area of websites and self-publishing, I mention them here due to their use for community and project websites, where registered users can discuss, publicise and share work. One example is the Lancashire Artists' Network site [8] , based on the open source Drupal content management system [9] . CMS' provide a framework where non-technical users can create web content of various types via their browser without needing to understand how to code. Some CMS's support, in addition to text and images, audio, video, as well as specialised areas such as discussion forums. Wikis fit in to a similar model, however tend to be simpler, more focussed tools which are designed to make adding text and linking information together particularly easy. The most famous example of a wiki is probably Wikipedia [10] , however many collaborative projects have a wiki for members to discuss and develop project documentation [11] . Forums are webpages that are designed around the idea of threaded discussions, where users post topics and replies to those topics.

mobilephones/SMS: Of course mobilephone networks are not strictly speaking part of the Internet since they do not use the Internet's protocols, however they are in many countries a ubiquitous communication system. The wide availability of the system is somewhat offset as a collaboration medium by the present general restriction to one-to-one communication and mainly voice and text messages. As bandwidth capacity increases and costs fall it may well become more viable to expand in to video use. A number of projects, for example Twitter [12] , are exploring how to integrate mobile and web-based systems. Whether such projects are serious collaborative tools or toys for the under-employed is still open to debate.

social networking websites: These are essentially highly branded content management systems, allowing users to present their content within fairly tightly controlled home pages. As the market matures various networks are competing on the functionality they offer in terms of managing content and connections to other members. [13] [14] [15] The challenge facing the large networks at present is how to convert their large memberships in to a profit without alienating them. [16]

virtual environments: the natural extension of social networking and online gaming is the 3D virtual environment, the most high profile one at the time of writing being Second Life [17] . Although in essence a relatively immersive chat client, (with voice features now appearing), from a collaboration point of view virtual environments offer some additional features such as the ability to embed, e.g. video content, in virtual objects, to allow sharing and performance of works. As with nearly all online environments, the tools available for use online are prescribed by the environments themselves.

video conferencing: This can cover a wide range of technologies and qualities. At one end there is the low resolution, somewhat jumpy output of cheap webcams combined with domestic broadband connections. As mentioned above such functionality is increasingly being integrated with text based chat and VoIP software. At the other end of the spectrum there are the high quality, high bandwidth options most commonly found in corporations and universities, who have access to the expensive Internet connections required. [18]

Sharing work

Although there is often no clear demarcation, since you can for example send files by email, this refers to technology more explicitly designed around sharing data files:

project websites: Aside from the discursive uses of project websites and wikis, these provide some of the simplest means of widely disseminating digital works. Content management systems often have built in functionality to manage file repositories and control access to the contents, for example restricting downloads to registered users.

file transfer protocol (FTP) sites: The traditional method of disseminating large files over the Internet, this method is perhaps falling from favour compared to the various web-based options. It is nevertheless useful to have a good FTP client [19] in the tool kit to access such sites. Incidentally if you are managing your own web space and need to upload files to your web server, if your provider supports it, a much more secure alternative to FTP is SFTP [20] [21] . SFTP is supported 'out the box' in most Linux file managers.

code management tools such as CVS: For more complex projects, especially where it is important to track what changes have been made to work by multiple collaborators, a code management system is vital. Simon Yuill explores this in detail in his article, however from a new user point of view the first requirement is probably a friendly graphical client to access such systems. You need of course to have the right client for the particular system you're using but examples are [22] [23] [24].

peer-to-peer filesharing: If media and record industry hype was to be believed [25] [26] then peer-to-peer filesharing networks are the font of all evil. However for collaborative projects that need to disseminate very large files, such as major software projects or video sharing, the ability to share the bandwidth load across many users has proven very useful. In essence peer-to-peer networks work by some users downloading files from an original 'seed' server, and then being willing to allow others to download the same files from them. As time goes on more copies proliferate across the Internet meaning that no single server or user takes the bulk of the demand. A technologically advanced example is Bittorrent [27] .

Sharing Ideas

Obviously an idea can be shared by describing it in words, sending a code example or other sample of work. Sometimes though more structured methods aid the collaboration process. One is the idea of design patterns as Simon Yuill discusses in his article, however a more general and widely popular method is that of mindmapping with some good free tools available [28] . Mindmapping is an example of a visualistion tool to allow individuals and groups to share and refine their ideas. These can be coupled with feedback mechanisms such as annotation, voting and messaging [29] .

Project Management

For larger or longer term projects more formal management tools may be appropriate. Some are built in to code sharing sites such as Sourceforge [30] , while others provide a specialist approach, such as dotProject [31] which can be hosted on a website and thus accessible to all members of the project. For many, projects wikis provide the necessary level of co-ordination. The term groupware is sometimes applied to integrated packages of collaboration software including email, instant messaging, and document management functions. [32] [33]

Working in real time

The methods mentioned so far are either a) asynchronous and/or b) essentially for talking about the work, or for making some version of the work available for others to see, try, or perhaps alter themselves. There are simple online creative tools [34] , and even some that allow more than one person to work at the same time [35] . These are obviously not comparable with the tool sets provided by such software as GIMP [36] and Inkscape [37] . It is also noticeable that many of these tools tend to be basic drawing tools. What is still at an early stage are ways for artists to actually work together, remote from each other, on the same piece of work, at the same time. There are interesting experiments going on, such as Furtherfield's Visitors' Studio [38] , however most have been orientated towards developing new tools specially for use on the Internet. But why should we leave the tools we are used to using 'at home' when we go online? There is a new frontier to be explored as to how we make the tools we are comfortable with available to us when we start working with others remotely. Perhaps the area where the most progress has been made so far is in the audio world with technologies such as streaming, Pure:Data and Open Sound Control (OSC) [39] . Going further one idea might be to have an underlying framework that can share sessions between any appropriately adapted software. This is perhaps somewhat analogous to the role JACK [40] plays in the free audio software world, where any piece of software that has JACK support built in can exchange audio data with other similarly equipped applications. In this case what would be shared between two remote instances of a particular application would be information about their state, key presses and mouse clicks etc.

Whatever technical solution is found it is to be hoped that it won't be too long before a new level of working with others becomes available, whereby we can work interactively on the same piece of work, using the full power of the tools we've invested time learning. Perhaps one day we will be able to take our favourite tools in to virtual 3D studios and work closely alongside our chosen collaborators wherever in the (real) world they might be.

Notes

[1] “UN Fights US over internet's future” The New Zealand Herald 2005

http://www.nzherald.co.nz/section/2/story.cfm?c_id=2&amp;ObjectID=10354409 [Accessed 12/12/2007]

[2] “Internet and e-mail policies” ACAS 2006 http://www.acas.org.uk/index.aspx?articleid=808 [Accessed 12/12/2007]

[3] “How the Internet has Changed Our Lives” Neilsen/Netratings 2003 http://www.nielsennetratings.com/pr/pr_031229_uk.pdf [Accessed 12/12/2007]

[4] http://www.gabbly.com

[5] http://www.skype.com

[6] http://www.ekiga.org

[7] http://www.linphone.org/index.php/eng

[8] http://www.lanarts.com

[9] http://drupal.org

[10] http://www.wikipedia.org

[11] http://puredyne.goto10.org

[12] http://twitter.com

[13] http://www.bebo.com

[14] http://www.myspace.com

[15] http://www.facebook.com

[16] http://news.bbc.co.uk/1/hi/technology/7130349.stm [Accessed 12/12/2007]

[17] http://secondlife.com

[18] http://www.videocentric.co.uk/videoconferencing/videoconferencing.shtml#...

[19] http://filezilla-project.org

[20] http://www.winscp.com

[21] http://cyberduck.ch

[22] http://www.tortoisecvs.org/ and http://tortoisesvn.tigris.org

[23] http://sente.epfl.ch/software/cvl

[24] http://www.twobarleycorns.net/tkcvs.html

[25] http://www.theregister.co.uk/2007/12/12/copyrights_and_wrongs

[26] http://www.riaa.com/physicalpiracy.php?content_selector=piracy_details_o...

[27] http://www.bittorrent.com/what-is-bittorrent

[28] http://freemind.sourceforge.net/wiki/index.php/Main_Page

[29] http://en.wikibooks.org/wiki/Collaborative_Networked_Learning:_A_Guide/S...

[30] http://sourceforge.net

[31] http://www.dotproject.net/index.php

[32] http://www.icthubknowledgebase.org.uk/opensourcegroupware

[33] http://www.opengroupware.org/en/applications/index.html

[34] http://selfportraitchallenge.net/2007/02/27/march-online-tools

[35] http://www.imaginationcubed.com/Imagine

[36] http://gimp.org

[37] http://www.inkscape.org

[38] http://www.furtherstudio.org/live

[39] http://www.yourmachines.org/tutorials/collab_stream.html

[40] http://jackaudio.org

" >

span-->

- Collaborative development: theory, method and tools
- Social networking

‹ Pure Dataflow - Diving into Pd up Collaborative development: theory, method and tools ›

Collaborative development: theory, method and tools

By dummy
Published: 09/06/2007 - 23:00

Simon Yuill , September 2007

One of the most significant features of the development of Free Software projects is the immense numbers of people involved. It has been estimated that in the year 2006, almost 2,000 individuals contributed to the coding of the Linux kernel alone, which is just one, admittedly very important, software project amongst the many thousands that are currently in development worldwide [1] . Alongside such numbers, other Free Software projects may be the work of just one or two individuals, or changing sizes of groups who come and go throughout the lifetime of a particular project [2] . One of the reasons for the success of Free Software has been the creation of tools and working practises to support such levels and varieties of involvement. Whilst often tailored for the needs of programming work, many of these tools and practises have spread into other areas and uses, with tools such as the wiki, which provides the technical infrastructure for the online encyclopedia Wikipedia, being one of the best known and most widely used [3] .

Collaborative development emphasises the idea of people working together and this is a key aspect of the Free Software ethos. In writing and talking about Free Software, Richard Stallman repeatedly speaks about the 'Free Software community' and the idea of software as something created by a community and belonging to a community, rather than it being exclusively the work of one individual or corporation [4] . 'Collaboration' in itself, however, does not fully cover the full implications of a Free Software practise. Free Software is about making the work you create available for others to also create from. It is a form of production that enables and encourages others to produce. Although it is often associated with the idea of 'appropriative' art and remix culture (in the sense of re-using material that others have created), what Free Software proposes is quite different. This goes beyond the idea of the simple re-use of materials and is better described as a form of 'distributive' practise [5] . Free Software is distributive because in sharing the source code along with the software, it distributes the knowledge of how that software was made along with the product, thereby enabling others to learn and produce for themselves from that. This is something which neither appropriative art or remix culture do. Free Software is something more than just collaboration, or just sharing. It recognises that invention and creation are inherently social acts that produce their own forms of sociability through this process [6] . The collaborative potential of Free Software therefore arises out of this basic principle of distributive production. Given this basis, it is no surprise that many of the actual tools created by the Free Software community for its own use are inherently sociable in their design and are geared towards a process in which people can build and learn together [7] .

This chapter provides an overview of some of the development tools currently in use in Free Software projects. Each tool is described in terms of how it works and how it might be applied and integrated into different types of project. In some cases it also discusses the background to the tools, how they came into existence and some of the conceptual underpinnings of how they handle their task. This does not cover all of the tools you would use to create a software project however, only the ones that are necessary for people to work together. For that reason there is no discussion about different code editors, compilers or debugging tools. Nor does it cover issues such as designing and planning a software project from scratch or the typical development cycle involved in that. These are all substantial issues in their own right and are covered elsewhere [8] . The focus here is specifically on tools that help you work with other people, and how to make what you do available to others.

Communication

Any project involving a number of people working together relies on efficient communication between its members. Email and internet chat (or messaging) are the two most widely used forms of online communication and play an important role in any collaborative project. Email is best suited for sending documents and detailed information amongst people, whilst chat is best for realtime discussion. When working in a group it is obviously important to keep everybody in the communication loop, so most projects will make use of a mailing list to coordinate emails between people and a designated chat channel for discussion. As it is also useful to keep a record of communication between people, systems which provide built in storage of messages are a good idea.

Mailman is a mailing list system created as part of the GNU project [9] . It is easy to setup and use, and is provided with a web-based interface that allows people to configure and administer multiple lists through a conventional web-browser. One of its best features is a web-based archive system which stores copies of all messages on the list and makes them available online. The archive allows messages to be searched and ordered in different ways, for example, by date, sender or subject, as well as arranged into topic threads. Each thread starts with a new message subject and follows the responses in a structured order that enables you to see how the discussion around it developed. The archive can be made open to the public, so anyone can read it, or protected so that only people who are subscribed to the list can view it.

It is common for a large project to have multiple mailing lists, each one dedicated to a particular kind of discussion area or group of people. For example there may be one for developers who are writing the code, and another for people writing documentation and instruction materials. There may also be one for users who can ask and discuss problems in using the software, and sometimes additional lists for reporting bugs to the developers and putting in requests for new features in the code. Given all this activity, mailing list archives provide an important information resource about the development of a project, how to use the software created in it, or different approaches to programming problems. They can be a valuable resource, therefore, for someone learning to program, or for someone joining a long-running project who wants to understand its background and personal dynamics better. More recently, a number of academics making studies of FLOSS development have also analysed mailing lists as a way of understanding some of the social structures and interplay involved in such projects [10] .

There are many forms of chat and messaging system in use today. IRC (Internet Relay Chat) is one of the oldest systems dating back to the late 1980's and is one of the most widely used. IRC can support both group discussions (known as conferencing) or one-to-one communication, and can also provide a infrastructure for file-sharing [11] . Channels are used as a way of defining distinct chat groups or topics in a way that is analogous with dedicated radio channels. Each channel can support multiple chatrooms which may be created and disposed off on the fly. A chat can be completely open to anyone to join or may require password access. For extra security IRC chats can also be encrypted. IRC is not itself a tool, but rather a protocol that enables online discussions. To use IRC requires a server for hosting the chats and clients for each of the participants. There are many public IRC servers, such as freenode.net, which provide free IRC channels for people, and these are often sufficient for most small-scale projects. Larger projects may wish to have their own dedicated IRC servers.

When working together it is common for programmers to have an IRC client running on their machine where they can ask quick questions to other developers or handle discussions in a more immediate and flexible manner than email permits. IRC can also be an excellent way of taking someone through a process step-by-step remotely. Whilst having a public archive of a mailing list is common, it is less usual to store IRC discussions. As IRC is a sociable medium there can be a lot of ephemeral or trivial content exchanged between people (as in everyday chat between friends) and storing all of this for public consumption is probably not worth the disc space. All good IRC clients, however, have built-in logging capabilities which allow the user to store the discussion in a text file for later reference.

Logo of BitchX, a free IRC client

There are a number of IRC clients for GNU/Linux. ircII is the oldest IRC client still in use, and one of the first to be created. BitchX is a highly popular offshoot of ircII originally written by "Trench" and "HappyCrappy" and has a certain style of its own synonymous with aspects of hacker and l33t culture. irssi is a more recent client written from the ground up to be highly modular and extensible. All of these clients can be scripted to handle various forms of automated tasks, such as chat bots, or integrate with other processes [12] .

Sharing Code

When Richard Stallman first made EMACS available to the world, and Linus Torvalds released Linux, people generally made feedback and contributions to the code via newsgroups and email communication [13]. This is fine for small-scale projects, but as the number of people and amount of work involved in a project grows, this kind of interaction between the code and developers becomes quickly unmanageable. In the late 1980's, Dick Grune, a programmer working on the Amsterdam Compiler Kit (ACK) project, wrote a set of scripts for making changes to source code files in a way that ensured the code could be changed at different times by different people without accidentally losing one person's input [14]. These scripts were called the Concurrent Versions System (CVS), as they enabled different versions of the code to exist side by side as well as be integrated into one another. They were later developed into a self-contained tool that is still one of the most widely used 'version control' or Source Code Management (SCM) systems.

Alongside CVS, a number of other SCMs have become popular, these include Subversion (SVN), Git and darcs (David's Advanced Revision Control System) [15]. Each has its own particular features and the design of a particular SCM system often carries an implicit model of code development practises within it. There are, however, various common features and tasks that are standard across different SCMs. SCMs generally work on the basis that the source code for a given software project is organised into a set of distinct files. The files are stored in a 'repository' which is often simply a normal file directory on a computer containing both the source files and meta-data (background information) about the changes that have been made to them - sometimes the meta-data is held separately in a database. Programmers can work on the source files independently, submitting their changes and additions to the repository and obtaining copies of the files as they are changed by other programmers. The SCM manages this process and enables the different contributions to be combined into a coherent whole.

The main tasks normally performed by a SCM are:

1. to store each new version of a source file without erasing previous versions, and to enable earlier versions to be retrieved. These tasks are often known as 'committing' changes and 'checking out' revisions.

2. to associate specific changes in a source file with a specific programmer. This is usually done by requiring programmers to log into the SCM under a specific username each time they 'commit' their changes to the code.

3. to provide a history of changes in a given source file and enable files to be compared to one another. A comparison is often called a 'diff' after the UNIX commandline tool for displaying differences between two files.

4. to warn or indicate when two or more programmers have attempted to change the same lines of code, known as a 'conflict'.

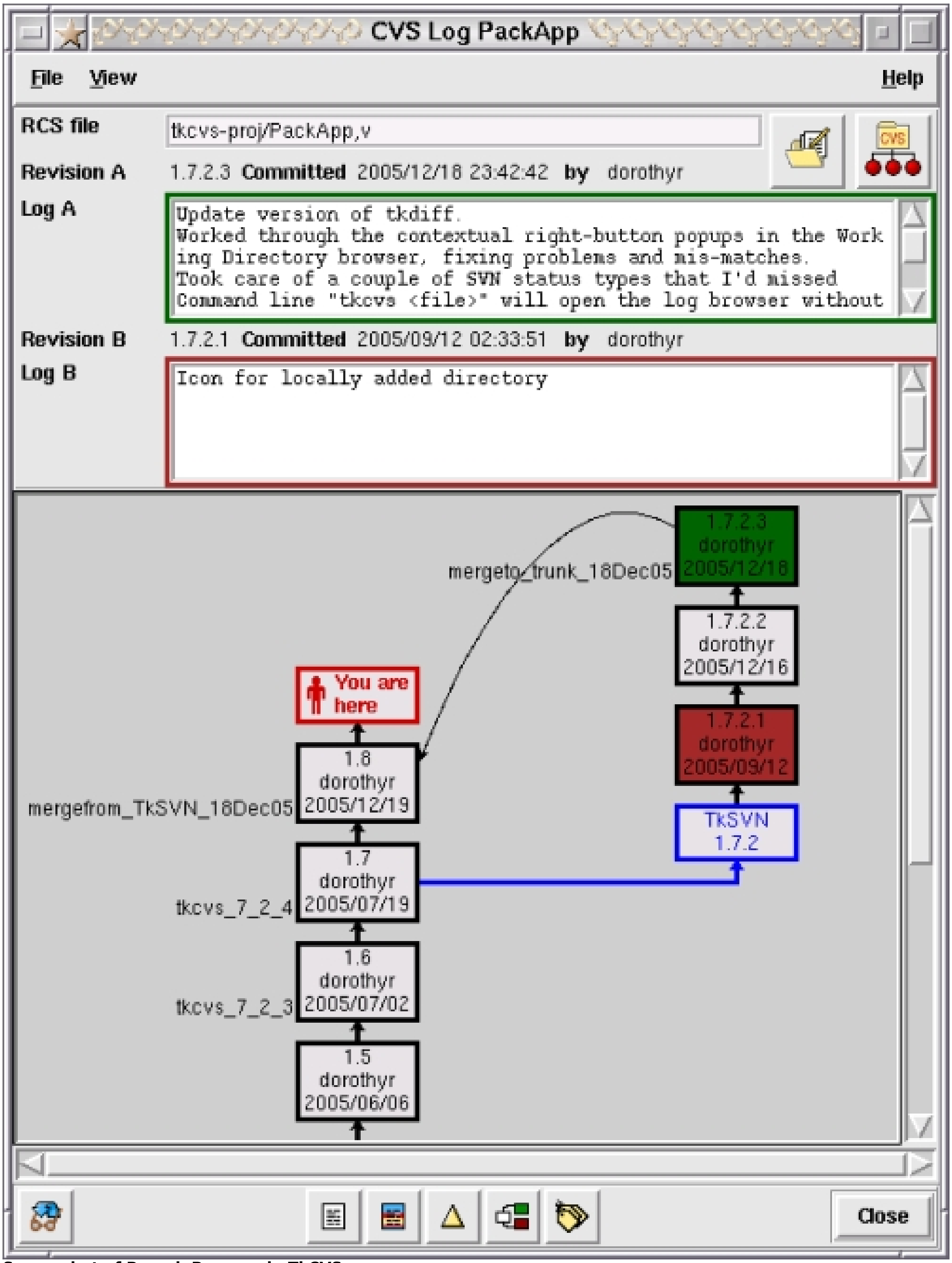
5. to enable parallel versions of the source files to be maintained alongside each other. This is known as 'branching', with each parallel version being a separate branch.

6. to combine different parallel versions of a source file into one final version, known as 'merging'.

One of the most significant differences between different types of SCM lies in the kind of repository system that they use. The key distinction being between those that use a centralised repository and those that are decentralised. CVS and SVN both use centralised repositories. In this case there is only one copy of the repository which exists on a specific server. All the programmers working on the project must commit their changes and checkout revisions from the same repository. Git and darcs both use decentralised repositories. With this approach, every programmer has a complete copy of the repository on their own machine, and when updating code commits their changes to that local version. The different repositories are then periodically updated to bring them into sync with one another through use of a networking mechanism. This communicates between the different repositories and merges their changes into one - a process known as 'pulling' and 'pushing' the changes between repositories. The decentralised approach is, in some ways, similar to a situation on a centralised server in which each programmer is working in their own branch. This may seem like a counter-intuitive approach towards maintaining coherence between the different programmers' work, and, perhaps for this reason, centralised systems such as CVS and SVN are often the more popular. Decentralised SCM systems have, however, shown themselves to be extremely effective when dealing with very large projects with many contributors. The Linux kernel, with nearly 2,000 programmers and several million lines of code, is one such large project, and the Git system was specifically written by Linus Torvalds for managing its development. The reason for the effectiveness of decentralised repositories for large projects may be that with the greater the number of people making changes to the code, the greater the amount of 'noise' this activity produces within the system. Within a centralised system this noise would affect everybody and, in doing so, amplify itself in a kind of feedback loop. Within a decentralised system, however, the activity, and resultant 'noise', is spread over many smaller areas thereby making it more contained and less likely to effect others. With a large project furthermore, in which many small groups may be focusing on different areas of its development, having multiple repositories also enables each group to focus on the key issues that it is addressing. In this sense each repository articulates a different 'perception' of the project.

Darcs has a very particular approach to updating code, called the "Theory of Patches", which derives from author David Roundy's background as a physicist [16] . This adopts concepts from Quantum mechanics and applies them to a model of code management in which any kind of change to the source code is abstracted as a 'patch' [17] . A variety of different operations are implemented to apply a patch to a repository. The advantage of Roundy's patch system is that it enables a variety of tightly tuned forms of merging to be performed that other SCM systems do not support, such as only updating replacements of a specific word token within different versions of source files (most SCM systems would change each entire line of code in which the token is found), or applying changes in a given order or combination (called sequential and composite patches).

For the programmer, interacting with a SCM system is generally done through a simple commandline tool. This enables the programmer to more easily integrate the SCM interface with whatever code editor they prefer to work with, and many code editors, such as EMACS, provide a mechanism through which the SCM can be directly accessed from the editor [18] . Various graphical SCM clients have also been developed however. The TkCVS and TkSVN tools provide GUI based interaction with the CVS and Subversion repository systems [19] . In addition they can display visual charts showing the structure of the source files within the repository, and mappings of branches, revisions and merges within a project.



RCS file: tkcvs-proj/PackApp,v

Revision A: 1.7.2.3 Committed 2005/12/18 23:42:42 by dorothyr

Log A:

```
Update version of tkdiff.
Worked through the contextual right-button popups in the Work
ing Directory browser, fixing problems and mis-matches.
Took care of a couple of SVN status types that I'd missed
Command line "tkcvs <file>" will open the log browser without
```

Revision B: 1.7.2.1 Committed 2005/09/12 02:33:51 by dorothyr

Log B:

```
Icon for locally added directory
```

Screenshot of Branch Browser in TkCVS

As with mailing lists, code repositories also provide a great deal of information on how a software project evolves. A number of tools have been developed to generate statistics and visualisations of different aspects of a repository, such as showing which files the most activity is focused around and the varying degrees in which different programmers contribute to a project [20] .

A number of free public code repositories are available on the internet, such as the GNU Project's Savannah repository. Alongside the SCM system, these often provide a packaged set of tools for managing a project. These are discussed in more detail below.

Sharing Ideas

A good piece of software is more than just so many pages of code. A good piece of software is a good set of ideas expressed in code. Sometimes those ideas are expressed clearly enough in the code itself, and there are those who argue that a good programmer is one who can express herself or himself clearly in this way [21] . Sometimes, and perhaps more often with larger projects, the ideas are not so easily contained in a single page of code or are implicit more within its bigger structure or how particular components within a program interact with one another.

One approach to this issue has been to provide comments and documentation notes directly within the text of the program code itself, and most programming languages include a special syntax for doing this. Some languages, such as Forth and Python, have taken this idea a step further and enable the documentation notes to be accessible from within the code whilst the program using it is running [22] . For almost all programming languages, however, a set of tools known as 'documentation generators' are available which can analyse a set of source files and create a formatted output displaying both the structure of the code and its accompanying comments and notes. This output can be in the form of a set of web pages which directly enable the documentation to be published online, or in print formats such as LaTeX and PDF. 'Document generators' are often dedicated to a particular programming language or set of languages. PyDoc, EpyDoc and HappyDoc are all systems designed specifically for Python [23] . Doxygen is one of the most widely used document generators and supports a number of languages including C++, Java, and Python [24] . When used in conjunction with Graphviz (a graph visualisation tool), Doxygen can also create visual diagrams of code structures that are particularly useful for understanding the object-orientated languages that it works with [25] .

Another, quite different, approach lies in the use of Design Patterns. Design Patterns are a concept adapted from the work of architect Christopher Alexander. Alexander was interested in developing a means through which architects and non-architects could communicate ideas of structure and form, so that non-architects, in particular, could envisage and describe various forms of domestic and urban building systems and thereby have greater control over the design of their built environment [26] . As applied to programming, Design Patterns provide a means of articulating common programming paradigms and structures in ways that are separate from actual code and can be explained in ordinary language [27] . They focus on the structure and concepts, the ideas, behind the code, rather than the specifics of how a particular piece of code is written. This allows ideas to be transferred more easily from one project to another even if they do not share the same programming language. In 1995, the Portland Pattern Repository was set up to collect and develop programming Design Patterns through a SCM-like system. This system was the first ever wiki [28] . A wiki is basically a web-based text editing system that provides a simplified form of version control like that used in CVS. The wiki created a simple yet robust way of enabling the content of web-pages to be updated in a collaborative fashion. It was quickly realised that this could be applied to all manner of different topics that could benefit from collaborative input, and the wiki spread to become one of the most widely used content management systems on the internet, with the most famous example being the Wikipedia encyclopedia [29] .

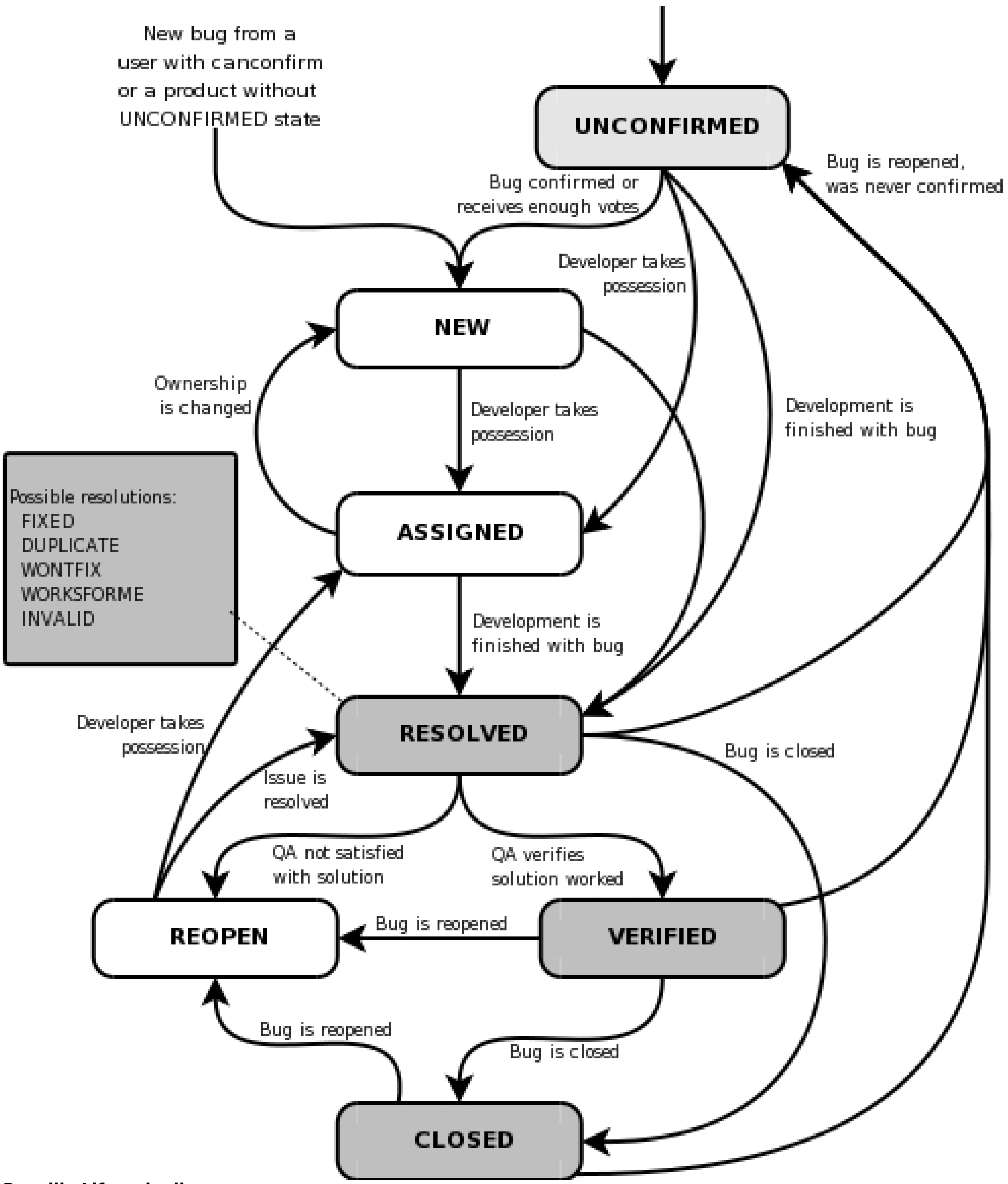
Nowadays, a whole range of different wikis are available, ranging from simple text-only systems to more sophisticated ones supporting images and other media content [30] . Within a programming project wikis provide a simple and effective publishing system for updating information and documentation related to the project. They can augment the primarily technical information produced through a documentation generator to provide more descriptive or discursive information such as user manuals and tutorials.

Project Management

The various tools described so far can all be combined into more comprehensive packages that provide a complete framework for managing a software development project. Often these are available for free through public development repositories such as Savannah and BerliOS, although it is also possible to set up your own system independently. Both Savannah and BerliOS are specifically geared towards supporting fully Free and Open Source projects. Savannah is run by the GNU Project and BerliOS is run by the German group FOKUS, both are not-for-profit organisations [31] . SourceForge provides a similar free service but is run as a commercial enterprise [32] . These sites are often described as 'developer platforms', a platform in this case meaning a web-based service that facilitates or provides an infrastructure for other people's projects [33] .

To use one of these platforms you must first register as a member and then submit a description of your proposed project, each site will provide details of what kind of information they wish you to provide in this description. The project will be reviewed by a selection panel within the organisation or user community that runs the platform and, if accepted, you will be given an account under which your project will be hosted. In addition to resources such as a code repository and mailing lists, all of these platforms also provide tools such as bug tracking systems and web-site facilities for publishing development news and documentation.

Bug tracking systems are tools that enable users and developers working on a project to report bugs they have discovered in the software. These reports can then be parcelled out amongst the developers, who will report back to the tracking system when the bug has been fixed. The status of each bug report is normally published through an online interface. This is particularly useful for any medium-to-large scale project and provides one useful development mechanism that non-programmers can contribute to. One well-known and widely used bug tracking tool is Bugzilla which was originally created for the development of the Mozilla web-browser [34] .


New bug from a user with canconfirm or a product without UNCONFIRMED state
UNCONFIRMED
Bug confirmed or receives enough votes
Bug is reopened, was never confirmed
Developer takes possession
NEW
Ownership is changed
Developer takes possession
Development is finished with bug
Possible resolutions:
FIXED
DUPLICATE
WONTFIX
WORKSFORME
INVALID
ASSIGNED
Development is finished with bug
Developer takes possession
RESOLVED
Bug is closed
Issue is resolved
QA not satisfied with solution
QA verifies solution worked
REOPEN
Bug is reopened
VERIFIED
Bug is reopened
Bug is closed
CLOSED

Bugzilla Lifecycle diagram

A similar service is a 'feature request' system. This enables users and developers of a software project to submit requests and ideas for features that are not currently part of the project but could be implemented at a later stage. As with the bug tracker it provides a good way for non-programmers to contribute to a project. The Python project is one example of a project that has made good use of this facility and has its own system called PEP, short for Python Enhancement

Proposal [35] .

If you wish to set up your own development platform a number of packages are available. Savane is the system used by Savannah and BerliOS [36] . Trac is another system more widely used in smaller projects. It includes a wiki system and is much easier to customise in order to create a self-contained project site with its own look [37] .

Sharing spaces

Whilst online development has been a feature of Free Software, it would be wrong to assume that everything happens exclusively online. The online world supports a rich variety of social groupings many of which have equivalents within the offline world. These often provide strong social contexts that link Free Software practise to local situations, events and issues. Linux Users Groups (LUGs) are one of the most widespread social groups related to Free Software. These groups normally include a mix of people from IT professionals and dedicated hackers, to hobbyists and other people who use GNU/Linux and other forms of Free Software in their daily lives. LUGs are usually formed on a citywide basis or across local geographic areas, and there will most likely be a LUG for the town or area in which you live [38] . They combine a mailing list with regular meetings at which people can share skills, discuss problems and often also give talks and demonstration of projects they are working on or are interested in. Similar groups with a more of an artistic angle to them include Dorkbots, which include all forms of experimental tinkering and exploration of technology, and groups such as Openlab which operate more like an artists collective for people working with Free Software-based performance and installation [39] .

Nerdcore Central, hackmeet

Hacklabs and free media labs are actual spaces, setup and run on a, mostly, voluntary basis. A typical lab will make use of recycled computer hardware running Free Software and providing free public access to its resources. As spaces they can provide a base for developing projects, whether that involves actual coding or a place for the developers to meet. Many labs also provide support for other local groups, such as computers for not-for-profit organisations. Hacklabs generally have a strong political basis to them and will often be linked with local Indymedia and grassroot political groups [40] . They are frequently, though not necessarily, run from squatted buildings. Such action makes a conscious link between the distributive principles of Free Software and the distributive principles of housing allocation promoted by the squatting movement [41] . Hacklabs have developed primarily in Italy and Spain, such as LOA in Milan and Hackitectura in Barcelona, but the Netherlands and Germany also have many active hacklabs, such as ASCII in Amsterdam and C-Base in Berlin [42] . Historically, these have had close associations with the Autonomist and left-wing anarchist groups. Free media labs are usually less overtly political, often with an emphasis on broader community access. Historically they can be seen to have grown from the ethos of the community media centres of the 1970s and 80s, with Access Space, set up in Sheffield in 2000, being the longest-running free media lab in the UK [43] . Free media labs are more likely to use rented or loaned spaces and may even employ a small number of staff. It would be wrong, however, to suggest that there are sharp divisions between hacklabs and free media labs, rather, they present different models of practise that are reflective of the social and economic contexts in which they have emerged. The government-sponsored Telecentros in Brazil offer a different example of such spaces, varying from providing low-cost public computer access to supporting more radical projects such as MetaReciclagem who focus on training people to become self-sufficient with technology [44] . An important offshoot of MetaReciclagem were the Autolabs setup by the Midiatactica group in the favelas of Sau Paulo [45] .

A different use of space to bring people together is found in the many forms of meetings and collective events that are characteristic of the Free Software world. Again these range from self-organised events such as hackmeets to more official festivals and conferences. These may either act as general meeting points for people to come together and exchange contacts, skills and ideas, or as more focused short-term projects that aim to achieve particular tasks such as pulling resources to complete a software project. Examples at the more commercial end of the spectrum include the Debian conference, DebConf, and Google's Summer of Code [46] . The Chaos Computer Club meetings and TransHack are examples from the more grassroots end of things, whilst Piksel and MAKEART are two examples of artist-led events that combine aspects of a hackmeet with an arts festival [47] .

Starting your own project

If you are starting up a Free Software project of your own for the first time, the best thing to do is take a look around at how other projects are run and the tools that they use. Take a bit of time to familiarise yourself with the different tools available and pick the ones that feel best suited to your needs and way of working. A good way to do this is to create a little dummy project on your own machine and test out the tools and processes with it. Another way of building up experience is to join an existing project for a short while and contribute to that, or even just join the developers mailing list to see what kind of issues and ideas come up there. One of the best ways you will find support is through the Free Software community, so make contact with your nearest LUG, see if there are any hacklabs and free media labs in your area and take a trip to the next hackmeet, conference or festival that comes your way.

If you are working with other people, see what the more experienced ones within the group can tell you about how they work and what tools they use. If your entire development team is new then it is probably best to come to a collective decision about what tools you are working with and choose the same set of tools for everyone so that you are learning the same things together. There are certain tools which you will have to share anyway, for example, you will all have to use the same SCM whether it is CVS, SVN, darcs or something else. Other tools can be different among different developers without causing conflicts. You should be able to use different code editors and different IRC clients for example without any problems.

Assuming that your first project will probably not be too ambitious, you may not need to use tools such as bug trackers, however, there is no harm in integrating these into your working process in order to build up a sense of how they operate. If you want to utilise the full range of development tools it is probably best to opt for an account with a public developer platform such as Savannah or BerliOS, or to use an all-in-one package such as Trac.

Not just code - Free Software as artistic method

The early years of internet art were characterised by a number of projects based around large-scale public collaborations often in the form of texts or images that could be added to and altered by different people [48] . These projects emerged at a time when Free Software was only just beginning to gather momentum and so they did not make use of the approaches it offers. There is no reason why a collaborative project that is producing something other than computer code could not make use of the tools and practises described here, however, and it has been suggested that a CVS system can be understood as a form of 'collaborative text' in its own right [49] .

None of these early projects, however, fully realises the idea of a Free Software practise as an artistic method. For all the collaborative nature of their construction, these projects often focus on a self-contained artefact as their sole end. They are collaborative but not distributive. More recently, however, a number of projects and approaches have been developed which not only use Free Software tools to create the work, but also realise a form of the Free Software ethos in how the projects are engaged with. The Openlab group in London have been developing small software systems for supporting improvised group performances which are themselves open to be reprogrammed during performance [50] . Social Versioning System (SVS) is a project which combines reprogrammable gaming and simulation systems with an integrated code repository. As with the Openlab tools, reprogramming is one of the key modes of engagement, but in linking this to a repository in which the changes and contributions of players can be examined, it aims to present coding itself as a form of dialogue, with the gaming projects critically exploring forms of social systematisation and urban development [51] . Plenum and Life-Coding are two performance projects which have adopted a similar approach [52] . All of these projects have, in different ways, adapted aspects of Free Software development tools and practises. Whilst less consciously designed as an artwork in itself, the pure:dyne project also demonstrates such an approach. Here, artists have collaborated in the development of a version of the GNU/Linux operating system specifically geared towards using and supporting Free Software based artistic tools and projects [53] . Pure:dyne is built on top of dyne:bolic, a version of GNU/Linux for artists and activists focusing on streaming and media tools [54] . Dyne:bolic was made partly possible through the Linux From Scratch project, which publishes guides for people wishing to create their own version of the GNU/Linux operating system [55] . This chain of development through successive projects is an excellent example of the distributive principle in action.

It would be wrong, however, to see these projects as a development from the earlier experiments with networked collaborative artworks. A much stronger analogy lies with the collective improvisation groups of the 1960s such as the Black Arts Group of St Louis (BAG) or the London-based Scratch Orchestra. In a move which in many ways pre-empts the hacklabs of today, the Free Jazz group BAG converted an abandoned warehouse into an inner-city 'training centre' jointly run by the group and local community. The space supported rehearsals and performances by BAG combined with various forms of public classes, workshops and discussions groups focused around issues affecting the local community. Here the collaborative musical practise of the jazz ensemble is structured upon, and reflective of, the wider distributive principle of the arts space in which they operate [56] . Another analogy is evident in the Scratch Orchestra who, like BAG, were an improvised music collective. Participation was open to anyone who wished to join regardless of musical background or experience. One of the key aspects of its practise were the development of 'scratch books' in which Orchestra members collated their own performance scores using whatever forms of notation they wished - from actual musical notation to abstract diagrams, written instructions and collaged found materials [57] . The scratch books were exchanged between different members who could use, alter and adapt from each other's work. Some versions of the scratch books were published under early forms of copyleft licences in which people were not only free to copy, but also encouraged to submit changes and additions that would be incorporated into later versions [58] . This was the embodiment of an idea that Cornelius Cardew, one of the founding members of the Scratch Orchestra, summarised in the statement: "the problems of notation will be solved by the masses." [59] Programs are notations just as music scores are, and we can clearly see how Cardew's sentiment applies as much to the practise of Free Software as it did to the music of the Scratch Orchestra.

Like the Free Software projects of today, these groups sought to distribute the knowledge and skills of production as an integral part of what they produced, but they also sought to explore and expose the social relations through which people worked. They deliberately challenged the notion of artistic authorship as the exclusive act of a single, socially disengaged, individual - one which still dominates the artworld today - as well as being critically aware of issues of power and hierarchy within collective practise. It is not just the distribution of knowledge, but also the distribution of access and power that these groups engaged with and faced up to. After several years of productive activity, the Scratch Orchestra fell apart due to internal tensions. Cardew put this down to a lack of sufficient self-criticism amongst its members, while others attributed it to one group seeking to exert its influence unfairly over the Orchestra as a whole [60] . Ultimately these are issues which any form of collaborative development needs to address and be critically aware of in some form, for, as the Scratch Orchestra discovered, working with others is not always an easy or inherently egalitarian process [61] . The success of any such project rests, not only in its ability to create something through collaboration, but in developing an awareness of what is at stake in collaboration itself.

Notes

1. see Jonathan Corbet, 2007, "Who wrote 2.6.20?", http://lwn.net/Articles/222773/ and David A. Wheeler, c.2006,"Counting Source Lines of Code", http://www.dwheeler.com/sloc/

2. there have been a number of studies on the structure of FLOSS development teams, see, for example: Rishab Aiyer Ghosh, 2003, "Clustering and dependencies in free/open source software development: Methodology and tools", http://www.firstmonday.org/issues/issue8_4/ghosh/index.html

3. Wikipedia is an online encyclopedia that anyone can contribute to: http://www.wikipedia.org

4. Richard Stallman, "The GNU Project", http://www.gnu.org/gnu/thegnuproject.html , also in the book Richard Stallman, 2002, Free Software, Free Society: Selected Essays of Richard M. Stallman, GNU Press; Boston.

5. for a more detailed account see Simon Yuill, 2007, "Free Software as Distributive Practise", forthcoming

6. this should not be confused with the idea of 'social media' which has become prominent in an approach to online commerce known as Web 2.0, Free Software foregrounds the social basis of production, whereas, Web 2.0 seeks to commodify social relations, see Dmytri Kleiner and Brian Wyrick, 2007, "InfoEnclosure 2.0", in MUTE, vol 2 issue 4, also online: http://www.metamute.org/en/InfoEnclosure-2.0

7. people can also build and learn apart, a common example being when one software project makes use of a library or codebase developed by a different group of people without any direct communication or collaboration between the people involved.

8. there are numerous online guides for programming with Free Software tools, such as Mark Burgess and Ron Hale Evans, 2002, "The GNU C Tutorial", http://www.crasseux.com/books/ctutorial/ , Mark Burgess, 2001, "The Unix Programming Environment", http://www.iu.hio.no/~mark/unix/unix_toc.html , Richard Stallman, 2007, "GNU Emacs manual", http://www.gnu.org/software/emacs/manual/emacs.html , and Eric Raymond, 2003, "The Art of Unix Programming", http://www.catb.org/~esr/writings/taoup/html/

9. http://www.gnu.org/software/mailman . For a discussion of the ideas behind mailman see: Barry Warsaw, 2000, "Mailman, the GNU Mailing List Manager", http://www.linuxjournal.com/article/3844

10. a good example is Nicolas Ducheneaut's OSS Browser, http://www2.parc.com/csl/members/nicolas/browser.html

11. for general information on IRC see http://www.irc.org and http://www.irchelp.org and Jarkko Oikarinen, undated, "IRC History", http://www.irc.org/history_docs/jarkko.html

12. ircII: http://www.eterna.com.au/ircii , irssi: http://irssi.org , BitchX: http://www.bitchx.org

13. this is described in Richard Stallman, "The GNU Project", ibid., and Eben Moglen, 1991, "Anarchism Triumphant: Free Software and the Death of Copyright", http://emoglen.law.columbia.edu/publications/anarchism.html

14. Dick Grune, Concurrent Versions System CVS. http://www.cs.vu.nl/~dick/CVS.html ACK was Stallman's original choice for a compiler in the GNU Project, when he approached Grune about using ACK however, Grune refused and, as a result, Stallman had to create his own compiler, which became GCC. ACK has since become obsolete whereas GCC is one of the most powerful and popular compilers currently in use today.

15. CVS: http://www.cvshome.org , SVN: http://subversion.tigris.org , Git: http://git.or.cz , darcs: http://darcs.net . For a detailed discussion of how Git works see: Jonathan Corbet, 2005, "The guts of git", http://lwn.net/Articles/131657/

16. David Roundy, undated, "Theory of patches", http://darcs.net/manual/node8.html#Patch

17. 'patches' are a common name for small changes to a code file, the repository is called a tree in Roundy's theory - all repositories have a tree-like structure.

18. the EMACS code editor was originally written by Richard Stallman, http://www.gnu.org/software/emacs

19. TkCVS and TkSVN: http://www.twobarleycorns.net/tkcvs.html

20. one example is Gitstat, which produces online visualisations of activity within Git repositories:

http://tree.celinuxforum.org/gitstat . Stats for the Linux kernel are available at: http://kernel.org

21. This idea is best represented in Donald Knuth's notion of 'literate programming', see: Donald E. Knuth, Literate Programming, Stanford, California: Center for the Study of Language and Information, 1992, CSLI Lecture Notes, no. 27.

22. Forth: http://thinking-forth.sourceforge.net , Python: http://www.python.org

23. PyDoc: http://docs.python.org/lib/module-pydoc.html , EpyDoc: http://epydoc.sourceforge.net , HappyDoc: http://happydoc.sourceforge.net

24. http://www.stack.nl/~dimitri/doxygen

25. http://www.graphviz.org

26. Christopher Alexander, et al., A Pattern Language: Towns, Buildings, Construction, New York: Oxford University Press, 1977, for a discussion of Alexander's ideas applied to software development see: Richard Gabriel, 1996, "Patterns of Software: tales from the Software Community", Oxford University Press; New York, PDF available online:
http://www.dreamsongs.com/NewFiles/PatternsOfSoftware.pdf

27. the classic text on software design patterns is: Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, 1995, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley. Design patterns are not specific to Free Software practise, and did not originate out of it, but have become popular amongst many programmers working in this way.

28. http://c2.com/ppr

29. see note 3 above.

30. For a comparison of different wiki systems see:
http://en.wikipedia.org/wiki/Comparison_of_wiki_software

31. Savannah: http://savannah.gnu.org , BerliOS: http://www.berlios.de

32. http://sourceforge.net

33. for a discussion of web platforms in relation to artistic practise, see: Olga Guriunova, 2007, "Swarm Forms: On Platforms and Creativity", in MUTE, vol 2 issue 4, also available online: http://www.metamute.org/en/Swarm-Forms-On-Platforms-and-Creativity

34. http://www.bugzilla.org

35. http://www.python.org/dev/peps

36. https://gna.org/projects/savane

37. http://trac.edgewall.org

38. there is a listing of LUGs organised by country available at: http://www.linux.org/groups , see also: http://www.gnu.org/gnu/gnu-user-groups.html

39. Dorkbots are worldwide, the main site from which all Dorkbot groups can be accessed is: http://www.dorkbot.org . Openlab groups currently exist in London, Glasgow, Amsterdam and Brussels, the two most active are in London: http://www.pawfal.org/openlab , and Glasgow: http://www.openlabglasgow.org

40. Indymedia is a worldwide movement to create independently produced news coverage. There are many local Indymedia groups across the world, the main site is at: http://www.indymedia.org

41. for background and discussion about the squatting movement see Anders Corr, 2000, No Trespassing!: Squatting, Rent Strikes and Land Struggles Worldwide, South End Press, and Advisory Service for Squatters, 2005, The Squatters Handbook, Freedom Press: London

42. Hackitectura: http://hackitectura.net , LOA: http://www1.autistici.org/loa/web/main.html , ASCII: http://www.scii.nl , C-Base: http://c-base.org . For a listing of different hacklabs internationally see: http://www.hacklabs.org . For a good overview of typical hacklab activities see: hydrarchist, 2002, Hacklabs - A Space of Construction and Deconstruction, http://info.interactivist.net/article.pl?sid=02/07/23/1218226&amp;mode=neste...

43. http://www.access-space.org   Access Space grew out of the Lowtech project which started earlier in the mid-1990s.

44. http://www.metareciclagem.org

45. http://autolabs.midiatatica.org , see also: Ricardo Rosas, 2002, "The Revenge of Lowtech: Autolabs, Telecentros and Tactical Media in Sao Paulo", in Sarai Reader 04: Crisis / Media, Sarai Media Lab; Bangalore, also available online: http://www.sarai.net/publications/readers/04-crisis-media/55ricardo.pdf

46. DebConf: http://www.debconf.org , Summer of Code: http://code.google.com/soc

47. Chaos Computer Club: http://www.ccc.de , TransHack: http://trans.hackmeeting.org , Piksel: http://www.piksel.no , MAKEART: http://makeart.goto10.org

48. two of the earliest such projects include Roy Ascott, 1983, The Pleating of the Text: A Planetary Fairy Tail, (see Frank Popper, 1993, Art of the Electronic Age, Thames and Hudson: London, p. 124) and SITO: http://www.sito.org , an online collaborative image bank started in 1993, for an overview of SITO's history see: http://en.wikipedia.org/wiki/SITO

49. Florian Cramer, 2000, "Free Software as Collaborative Text", http://plaintext.cc:70/essays/free_software_as_text/free_software_as_tex...

50. http://www.pawfal.org/openlab/index.php?page=LaptopDrummingCircle

51. http://www.spring-alpha.org , and http://www.spring-alpha.org/svs

52. Plenum was part of Node.London, 2005, Life-Coding was presented at Piksel in 2007.

53. http://puredyne.goto10.org

54. http://www.dyne:bolic.org

55. http://www.linuxfromscratch.org

56. Benjamin Looker, 2004, Point From Which Creation Begins: The Black Artists' Group of St. Louis, St. Louis: Missouri Historical Society Press. For BAG this was an approach that was shared with the wider Black Arts Movement: "Seeing their art as a communal creation, national leaders of the Black Arts Movement had rejected Romantic and post-Romantic notions of the individual artist working in isolation or estrangement from his social context. Instead, they stressed art's functional roles, urging that it be created in a communitarian and socially engaged stance.", Looker, p. 66

57. for an overview of the Scratch Orchestra see Cornelius Cardew, 1974, Stockhausen Serves Imperialism and Other Articles, Latimer New Dimensions: London, PDF available online: http://www.ubu.com/historical/cardew/cardew_stockhausen.pdf . Extracts from the scratch books were collated and published in Cornelius Cardew (editor), 1974, Scratch Music, MIT Press: Massachusetts, MA.

58. see for example Cornelius Cardew (editor), 1969, Nature Study Notes, Scratch Orchestra: London: "No rights are reserved in this book of rites. They may be reproduced and performed freely. Anyone wishing to send contributions for a second set should address them to the editor: C.Cardew, 112 Elm Grove Road, London SW13.", p. 2

59. Cardew, 1974, p. 88.

60. different perspectives on this are provided in Cardew, ibid, Stefan Szczelkun, 2002, The Scratch Orchestra, in "Exploding Cinema 1992 - 1999, culture and democracy", available online: http://www.stefan-szczelkun.org.uk/phd102.htm , and in Luke Fowler's film documentary about the

Scratch Orchestra, "Pilgrimage From Scattered Points" (2006).

61. For a more critical perspective on colloborative working see Ned Rossiter, 2006, Organized Networks: Media Theory, Creative Labour, New Institutions, NAi Publshers: Rotterdam, and Martin Hardie, The Factory Without Walls, http://openflows.org/%7Eauskadi/factorywoutwalls.pdf

Images

[1] Logo of BitchX, a free IRC client.

[2] Screenshot of Branch Browser in TkCVS.

[3] Bugzilla Lifecycle diagram. Included in the source package for Bugzilla and available under the Mozilla.org tri-licence (MPL, GPL, LGPL). For use in the Bugzilla article.

[4] Nerdcore Central, hackmeet, photo by Armin Medosch, from: www.thenextlayer.org/node/74

" >

span-->

‹ Working with others up Social networking ›

Social networking

By marloes
Published: 10/04/2007 - 09:19

Iman Moradi , October 2007

Before the rise of the worldwide web, the Internet was already a successful model for allowing communities to form and share information.   Newsgroups, mailing lists and countless forums and multi-user environments existed which allowed us to express our views and practice being social. It has been fittingly said that the Internet augmented traditional forms of communication and added to people’s social ties. (Irina Shklovski, 2004) [1]

Despite the opportunities and new modes of social communication that these augmentations allowed us, it's safe to say that what was on offer then was never as pervasive, inviting, interactive or facilitating as what we can enjoy today. As the web has developed, so too have the skills of countless specialist internet programmers and our insatiable appetite to communicate, consume and play with information in new and varied ways.

As a result, the web is now teeming with internet applications which are user friendly, rich in functionality and crucially provide features which allow us to form and maintain social ties in a manageable fashion. We can now exchange knowledge, benefit from aggregated views, share our personal media, access, manipulate, describe and generate items of data with unparalleled ease.

What is known in the industry as a perpetual beta, [2] refers primarily to how new, often experimental and cutting edge features are continuously integrated to such online services and software. Many of these features, usually offered freely, easily rival and surpass the potential benefits of locally installed paid for and free desktop software as they are developed quickly with a steady eye on direct user feedback and usage patterns.

However with the great potential that these tools have, there are some serious tradeoffs as well. The very mechanisms which allow us to enjoy social experiences and get things done quicker and more conveniently can potentially be put to other end uses as well which we may or may not find agreeable.

Of these concerns, there are the anticipated issues associated with implicitly entrusting our content with the service provider and third parties which ‘they’ trust, but more disconcertingly our mere interactions with any of these tools inevitably leave a rich information trail as well which can be stored and analysed without our knowledge or direct permission.

The broadly worded user license agreements or terms of service documents for these tools and services basically allow the companies which operate them great liberties to utilise any information we share with them, or any content we create, which may include tracking our usage even at times when we are not using their services! [3]

Whether the end purpose of any of these activities is to deliver a better more customised service, target us with advertising, sell our usage data to third parties, indemnify the providers from any responsibility when something goes wrong, or grant them rights to freely own and use our content as they please, it is all possible and indeed nonetheless questionable. [4]

This article aims to introduce a select few of the most well known tools for collaboration. It does so critically, highlighting their potential to enhance existing methods of communication and the way we work, as well as mentioning some of their drawbacks and pitfalls.

This article is aimed primarily at new users who want a starting point or introduction to an interesting and rapidly growing area of development.

Web 2.0 and Social Software

The Internet is awash with buzzwords and hype. You may read or hear some terminology which never existed until a couple of years ago. Some of this terminology is very specific, Blogs and Wiki’s are examples of such, and they are clearly defined things which have, needless to say, been around for a while. A Blog, short for weblog, or a Wiki both allow you to edit and update content which others can see, they are names for two generic types of product, with several variants and configurations and mutations of each available to use.

Blogs are like online diaries or journals, with each entry having a date and an archive of previous entries. Wiki's, such as the Wikimedia Foundation's suite of wikis and their online encyclopaedia "Wikipedia" referenced in this article, are non-linear flexible repositories which allow specific users or anyone to create pages, edit and interlink content while keeping a record of revisions and changes. Both are viewable within a web browser without any additional software being required. Open Source Blogging systems such as Wordpress [5] have also brought ease to blogging and many commercially hosted systems exist to allow users to maintain blogs without any technical know-how [6] .

These days however, other terms can be quite nebulous and can cover a number of different things. Web 2.0 (usually expressed two-point-zero, or two-point-o) is such a term. Web 2.0 is not the name of a new technology, but rather a way of referring to group of common technologies and methods which websites and associated services are developing and supporting to facilitate 'greater collaboration and sharing between users'. [7]

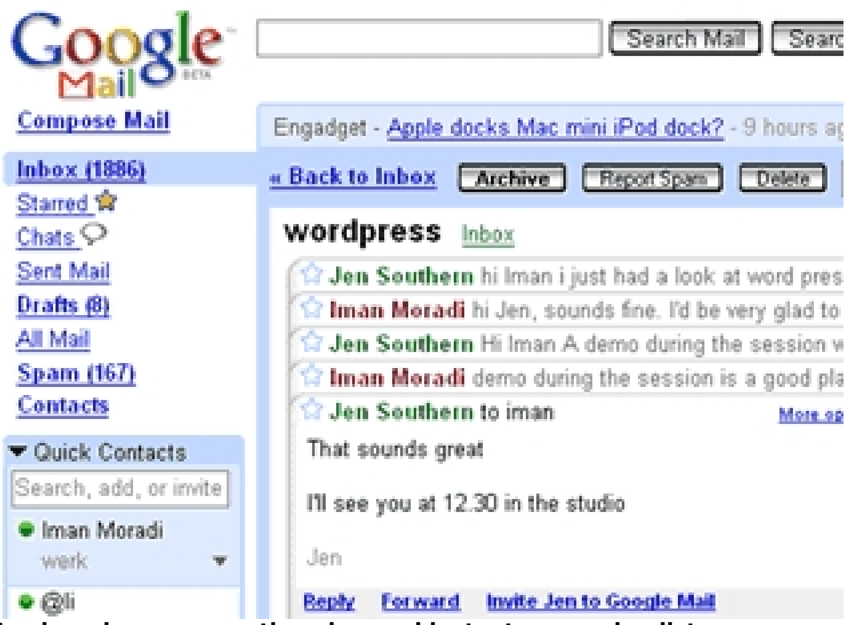
Social Software itself is the blanket term used to categorise tools which allow us to communicate and interact with other users, and although that in itself is nothing new, today this term is widely being used to describe web based software which adheres to some basic Web 2.0 principles.

The Wikipedia article on Social Software, gives a comprehensive list [8] of tools which are used for online communication. It has descriptions of instant messaging, text based chat technologies (IRC), Internet forums, Blogs, Wikis, as well as social bookmarking, virtual worlds and massively multiplayer games.

To better understand the area, it's important to know what these tools are doing and what they are enabling us, and the companies who offer these tools, to do differently.

Websites as applications

One of the first web based tools which led the way in terms of providing a very robust and capable service which behaved like a desktop application was Google's Gmail web based email service which has been around since the summer of 2004. Although it wasn't alone in doing so, it presented a shift in the way we view web based services and their sophistication.



Engadget - Apple docks Mac mini iPod dock? - 9 hours ag

wordpress

Jen Southern hi Iman i just had a look at word pres
Iman Moradi hi Jen, sounds fine. I'd be very glad to
Jen Southern Hi Iman A demo during the session w
Iman Moradi demo during the session is a good pla
Jen Southern to iman

That sounds great

I'll see you at 12.30 in the studio

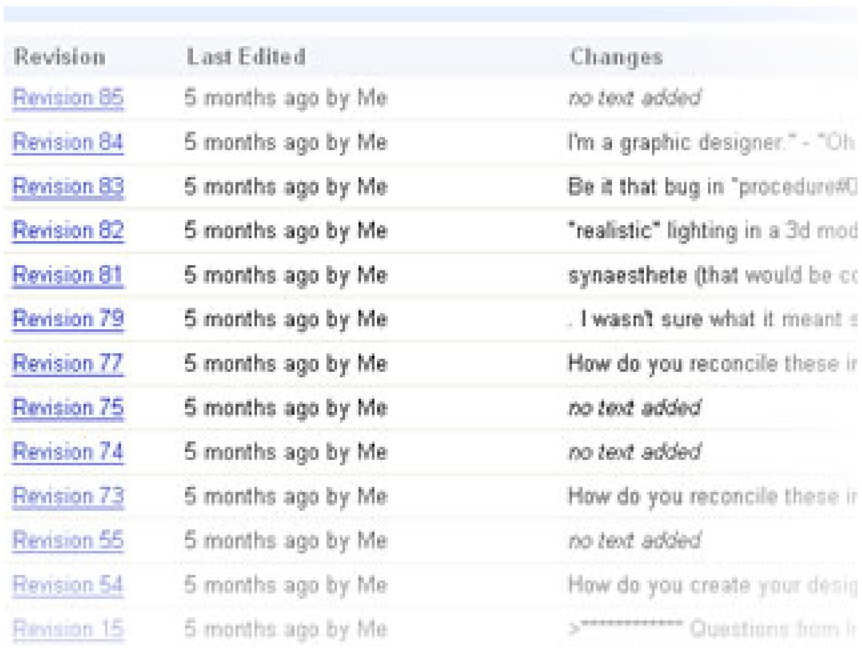
Jen

Feed reader, conversation view and instant messaging list.

With Gmail, you can simply search for emails instead of sorting them, and unlike other large web based email providers of a couple of years ago, the whole page doesn't refresh needlessly page by page in order to display your email. In fact in terms of functionality and user experience, when Gmail arrived, it was closer to a desktop based application than anything else on offer at the time. Other large web based email providers such as Yahoo mail and Microsoft Hotmail have since followed suit.

Currently, Gmail's main interface combines a feed reader (for showing you news items from your favourite sites) an instant messaging application (allowing you to converse with contacts in real-time), and it also has a very useful email clustering tool which groups related emails and their replies together.   Although these features are available in separate products and have existed for some time in other places, Gmail found a way of packaging them all together (by generating enough revenue from targeted advertising) and provided the service for free on the web. [9]

Google's Gmail now integrates with a suite of other tools such as Google Calendar and Google Docs and Spreadsheets, each other product allows you to share your data with other users with relative ease. For example you can share a calendar with other users, share documents and edit them collaboratively in one place to eliminate the need for excessive emailing and the chore of revision tracking. [10]



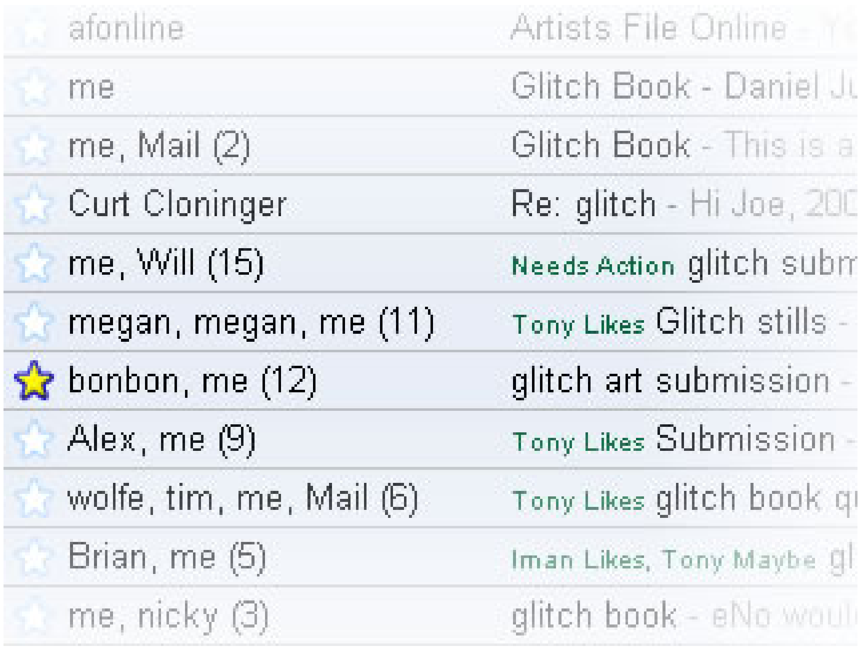
Revision	Last Edited	Changes
Revision 85	5 months ago by Me	*no text added*
Revision 84	5 months ago by Me	I'm a graphic designer." - "Oh
Revision 83	5 months ago by Me	Be it that bug in "procedure#(
Revision 82	5 months ago by Me	"realistic" lighting in a 3d mod
Revision 81	5 months ago by Me	synaesthete (that would be cc
Revision 79	5 months ago by Me	. I wasn't sure what it meant
Revision 77	5 months ago by Me	How do you reconcile these ir
Revision 75	5 months ago by Me	*no text added*
Revision 74	5 months ago by Me	*no text added*
Revision 73	5 months ago by Me	How do you reconcile these ir
Revision 55	5 months ago by Me	*no text added*
Revision 54	5 months ago by Me	How do you create your desig
Revision 15	5 months ago by Me	>************* Questions from I

Google docs revision tracking

Recently, myself and other collaborators decided to use a shared Gmail account extensively over the course of three years to track submissions for art and design book collaboration. Without it, it would have been very difficult to keep track of the 700 or so submissions we received from over 200 people. Additionally, we used Google Spreadsheets to comment on these submissions and work together on editing essays and interviews. Using social software greatly simplified the task at hand, in that we mainly worked on it during weekends away from regular work.

With all the functionality and interactivity which Gmail offers, there are some other issues associated with the use of Gmail which have made privacy advocates quite worried. They primarily relate to Google's policies regarding the use, analysis and retention of user data.

It's noteworthy to mention that although as editors of the book, we consented to have our sent and received emails scanned by Google's software to show targeted advertising (as the cost of having a free service) the people sending us email did not explicitly enter into any agreement to do so, to essentially have the contents of their emails to us scanned [11] . Neither did they agree to Google's vague data retention policies, which can keep emails (even deleted ones) on the system indefinitely. [12]



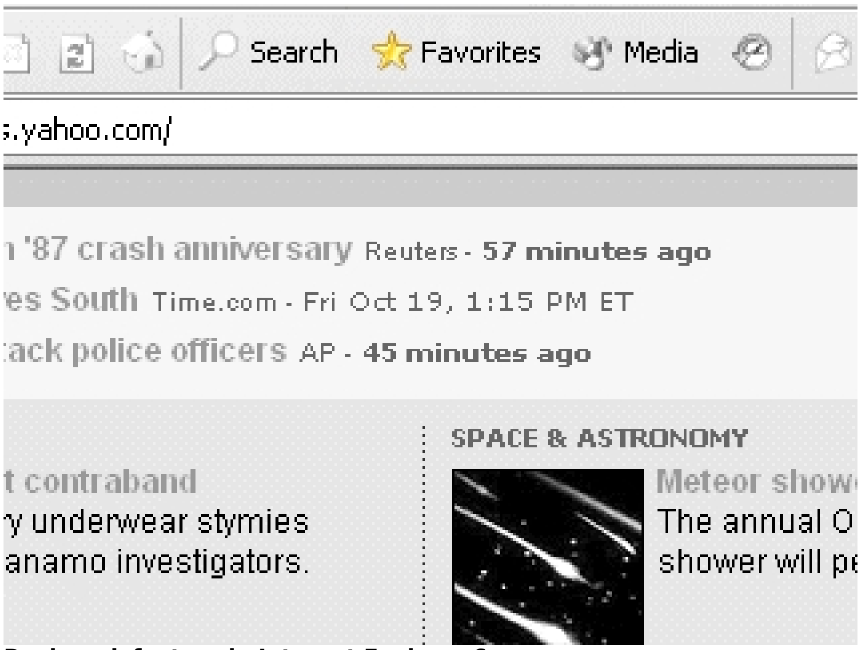
☆	afonline	Artists File Online - Y
☆	me	Glitch Book - Daniel Ju
☆	me, Mail (2)	Glitch Book - This is a
☆	Curt Cloninger	Re: glitch - Hi Joe, 200
☆	me, Will (15)	Needs Action glitch subm
☆	megan, megan, me (11)	Tony Likes Glitch stills -
★	bonbon, me (12)	glitch art submission -
☆	Alex, me (9)	Tony Likes Submission -
☆	wolfe, tim, me, Mail (6)	Tony Likes glitch book qu
☆	Brian, me (5)	Iman Likes, Tony Maybe gl
☆	me, nicky (3)	glitch book - eNo woul

Cropped portion of Gmail inbox interface, with correspondence clustering and labelling for messages.

Tagging / Folksonomies

Another great example to illustrate some of these key differences and familiarise you with common facets of many Web 2.0 applications is del.icio.us [13] . Delicious, which is now owned by Yahoo, is a social bookmarking tool.   It’s a website and service which can integrate with your internet browser using a number of easy to install plug-in’s. It basically allows you to bookmark web pages which you find interesting with two clicks. You are then able to access these links from any web browser, see what others are linking to and perhaps share your own links with specific users by forming networks.

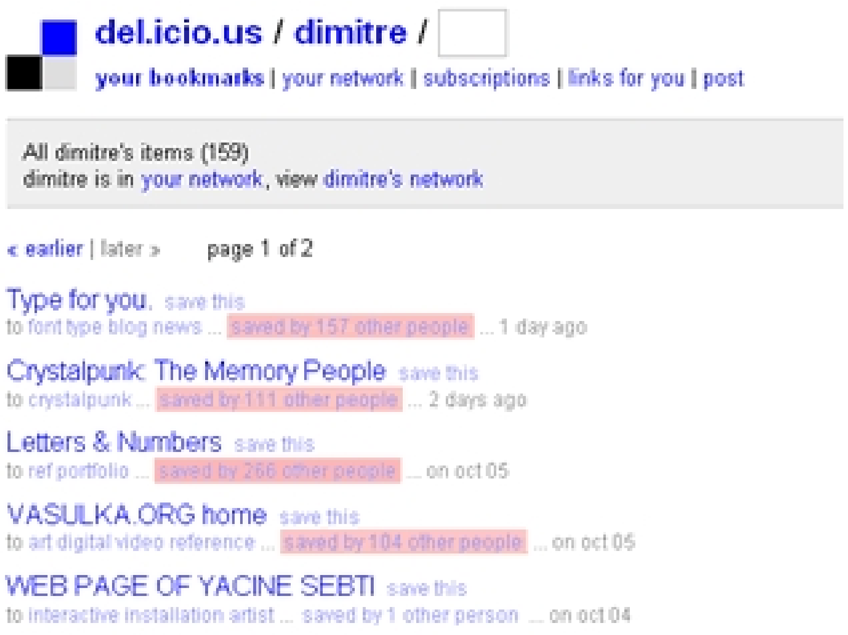
On surface there is nothing new to the storage of bookmarks aspect. For years, web browsers have had Favourites or Bookmarking functionality in-built, so the web pages which you frequently visit or find interesting can be retrieved later so long as you bookmark them or add them as a Favourite.



Bookmark feature in Internet Explorer 6.

But anyone who has used Favourites knows too well the problem of having many links to different sites, with non-descriptive names and the pain of not being able to find a particular site again. Also, you may end up changing computers without backing up your links or you wish to retrieve your links at someone else’s computer or at work and suddenly the non portable Favourites situation inside your browser seems quite limiting.

With Delicious, which is one of the largest social bookmarking sites, instead of storing your links on your computer, you save them onto a designated area on the Delicious website, and these links are associated with your username.



del.icio.us / dimitre /

your bookmarks | your network | subscriptions | links for you | post

All dimitre's items (159)
dimitre is in your network, view dimitre's network

« earlier | later » page 1 of 2

Type for you. save this
to font type blog news ... saved by 157 other people ... 1 day ago

Crystalpunk: The Memory People save this
to crystalpunk ... saved by 111 other people ... 2 days ago

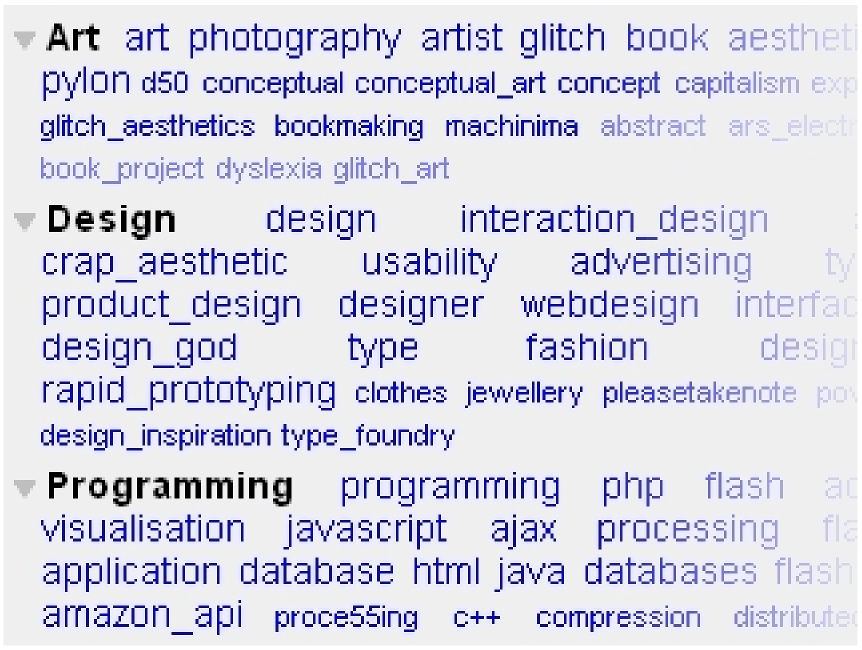
Letters & Numbers save this
to ref portfolio ... saved by 266 other people ... on oct 05

VASULKA.ORG home save this
to art digital video reference ... saved by 104 other people ... on oct 05

WEB PAGE OF YACINE SEBTI save this
to interactive installation artist ... saved by 1 other person ... on oct 04

Screenshot of a users delicious bookmarks feed.

As well as allowing other people to view your links and allowing you to view their links, Delicious offers an interesting way to search for and organise these links. You assign your own ‘tags’ or descriptive keywords to each link you save and in effect this assists in retrieving them later.



Art art photography artist glitch book aesthet
pylon d50 conceptual conceptual_art concept capitalism exp
glitch_aesthetics bookmaking machinima abstract ars_electr
book_project dyslexia glitch_art

Design design interaction_design
crap_aesthetic usability advertising ty
product_design designer webdesign interfac
design_god type fashion design
rapid_prototyping clothes jewellery pleasetakenote po
design_inspiration type_foundry

Programming programming php flash ac
visualisation javascript ajax processing fla
application database html java databases flash
amazon_api proce55ing c++ compression distribute

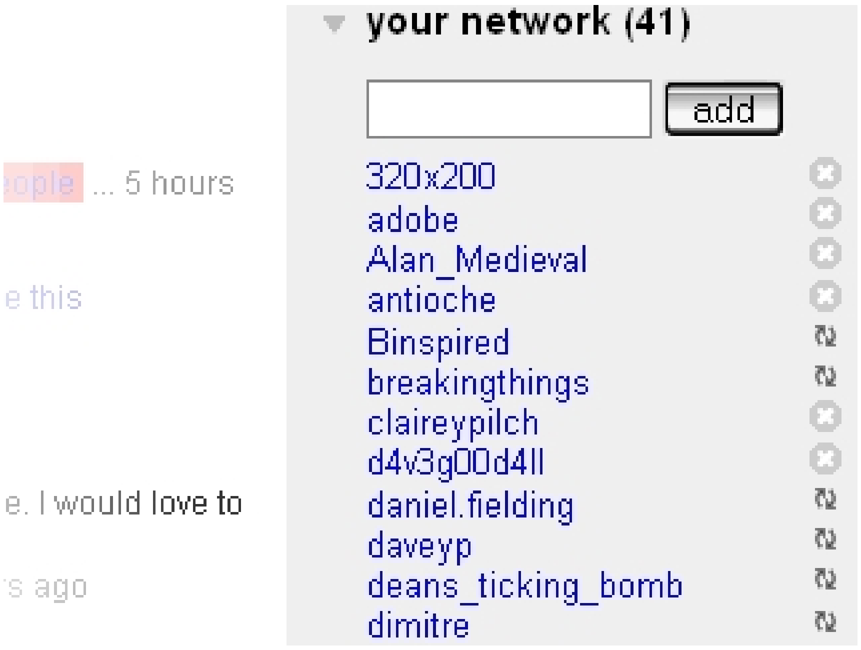
Crop of a clustered tag cloud. Showing frequency of tags visually.

Being able to tag items using a specialist vocabulary and the practice of doing so collaboratively, referred to as a Folksonomy [14] , clearly has its benefits over taxonomies and rigid information hierarchies [15] , and it has some drawbacks too.

Firstly, consider how a Dewey organised physical library can be so inherently confusing without knowing what the numeric shelf designations represent or without the benefit of having a library catalogue search tool or librarian at hand. Anyone can effectively become a subject specialist or categoriser within folksonomies, and therein lies a potential benefit and incentive, especially to categorise niche and newly emerging phenomenon.

Secondly, we could consider that freely allowing categorisation could also create Metanoise [16] which relates to inaccurate, erroneous, incomplete, or irrelevant data which is associated with any described item. [17]

Either way due to the large numbers using such services, any form of democratisation of the categorisation and description process can potentially be beneficial especially to minority users. (Scott Bateman 2006) [18]



320x200
adobe
Alan_Medieval
antioche
Binspired
breakingthings
claireypilch
d4v3g00d4ll
daniel.fielding
daveyp
deans_ticking_bomb
dimitre

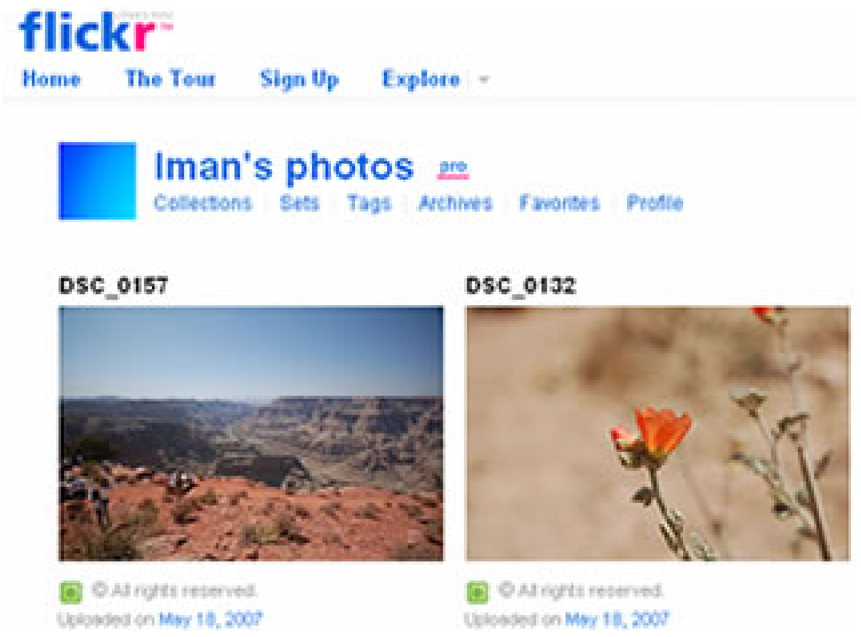
A user's personal network of other delicious users.

According to Thomas Van Der Wall (2005), who coined the phrase Folksonomy, “Who is doing the tagging is important to understand”. Using Delicious, as well as simply retrieving previously bookmarked content, we can actively share it with others too.   Delicious also allows you to form your own network and permits you to send any link to specific people in your network.

Some of the key features of Delicious are replicated across other systems. Collaborative content or knowledge filtering happens because we can see what others have shared and how they’ve described it. Searches for items unknown by name can instantly become quite fruitful if we know or can guess what others may categorise them as.

Additionally, when you bookmark an item you can discover who else has bookmarked it too; a quick visit to their bookmarks may uncover other hidden gems.

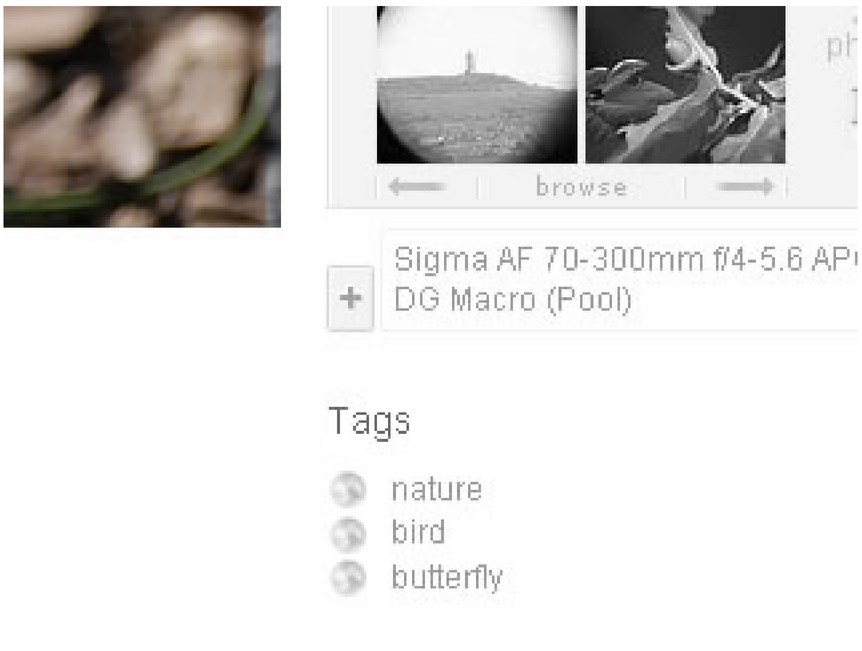
“Folksonomy tagging can provide connections across cultures and disciplines (a knowledge management consultant can find valuable information from an information architect because one object is tagged by both communities using their own differing terms of practice)” (Thomas Van Der Wall, 2005) [19]



Screenshot showing a crop of a personal Flickr page.

Flickr, which is a photo sharing website, allows the same affordances for personal photos and builds on the social network aspect quite extensively, while enriching photos with user provided tags.

According to Stewart Butterfield, one of Flickr's co-founders. "The job of tags isn't to organize all the world's information into tidy categories, It's to add value to the giant piles of data that are already out there." [20] Photo sharing websites have existed on the web but none have succeeded in adding as much value and use to our personal photos.



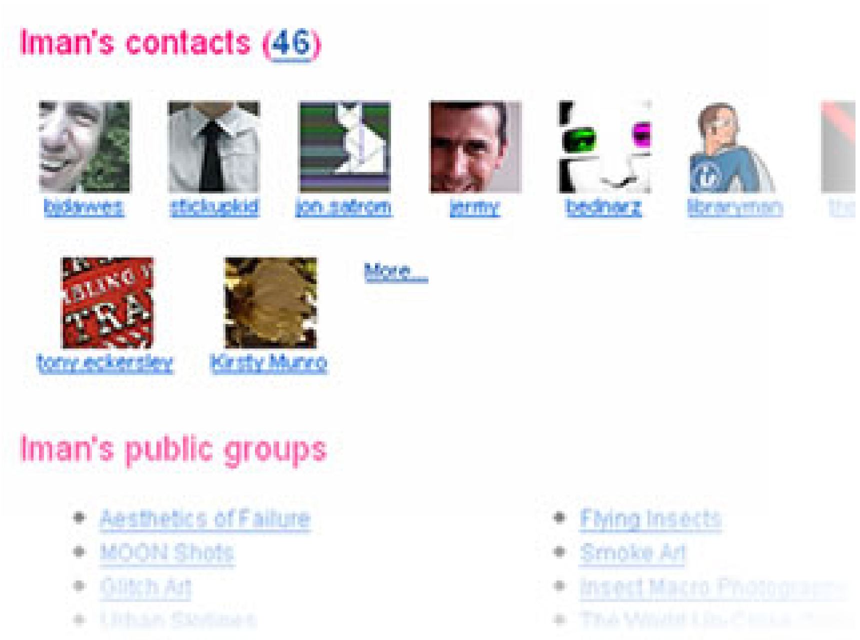
A crop of the photo detail page on Flickr showing three tags and a group (pool) which the photo belongs to.

While both these tools are successful and used by many, it's important to also state that there is a wealth of information which has yet to be fully utilised by the company which now owns both products, Yahoo!

In 2006, Yahoo’s former Chief Operating officer was quoted referring to the monetization of Flickr in due course. [21] “We have products like Flickr that are not yet really monetized right now.” and exactly how is something which will no doubt be informed and facilitated in part by “…user data about who they are and what they have done,…” although this wasn’t a wildly outrageous statement, it can be seen to go against the grain of the founding spirit of some of these tools, which weren’t initially designed, based and led by plans for monetary profit.

Social Networks

If you have a blog you are said to contribute to the blogosphere [22] which is the idea that blogs exist within a community. If you upload your photos to Flickr and add your friends to your contacts list you have formed a social network. The term friend can sometimes denote a looser designation in internet mediated communication and especially web 2.0 applications, than it does offline, and the term social network signifies the numerous types of connection between users of a website.


Iman's contacts (46)

bidawes stickupkid jon.satrom jermy bednarz libraryman

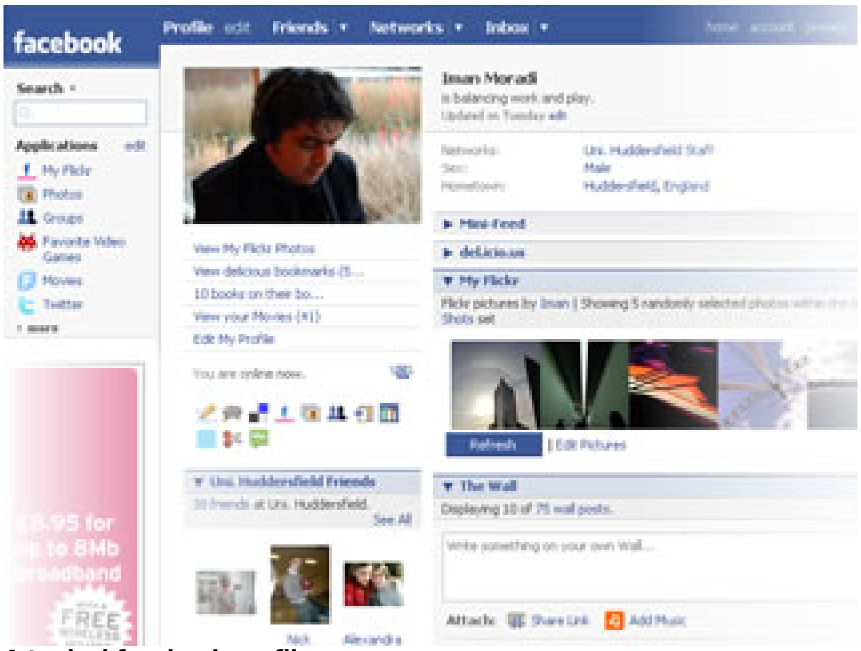
tony.eckersley Kirsty Munro More...

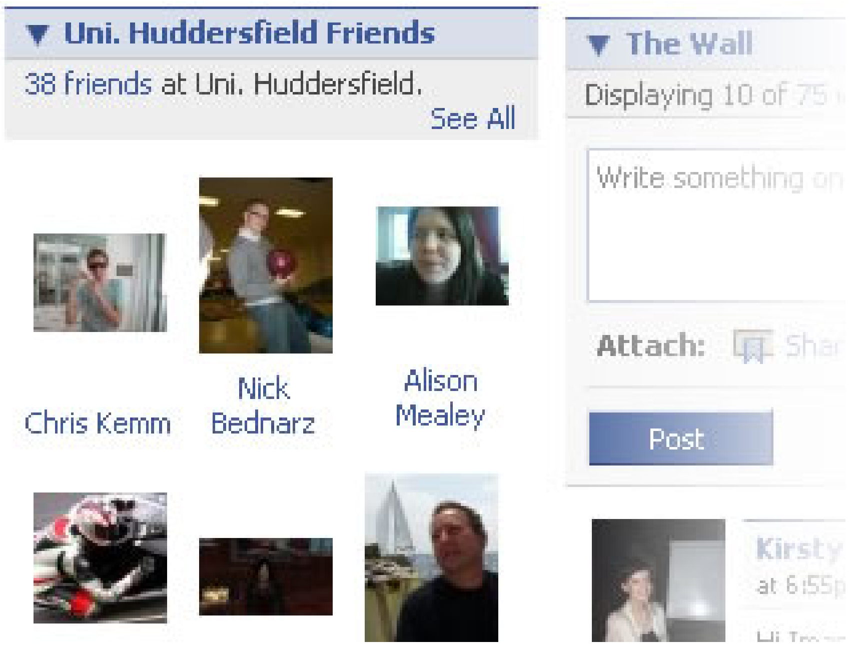
Iman's public groups

- Aesthetics of Failure
- MOON Shots
- Glitch Art
- Urban Skylines
- Flying Insects
- Smoke Art
- Insect Macro Photography

view of contacts page on Flickr.

The phenomenon of social networking sites [23] however, actually refers to websites which literally focus on showing, verifying and facilitating links between people sharing similar interests or members of particular groups.   Wikipedia maintains a growing list of these sites [24] . I’ve taken Facebook, currently the seventh most popularly visited site on the web as an example. [25]

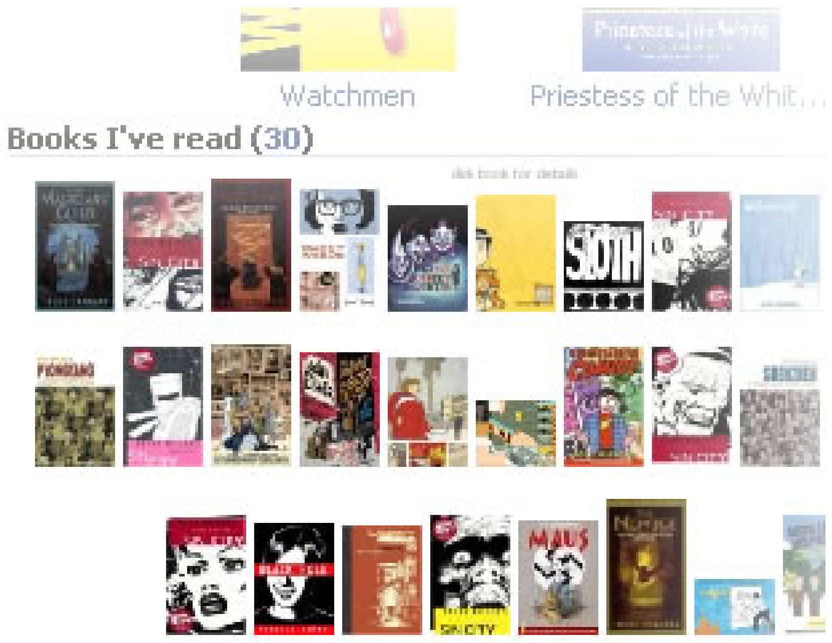

facebook
Profile edit Friends Networks Inbox
Search
Applications edit
My Flickr
Photos
Groups
Favorite Video Games
Movies
Twitter
Iman Moradi
is balancing work and play.
Networks: Uni. Huddersfield Staff
Sex: Male
Hometown: Huddersfield, England
Mini-Feed
del.icio.us
My Flickr
Refresh
Edit Pictures
Uni. Huddersfield Friends
The Wall
Displaying 10 of 75 wall posts.
Attach: Share Link Add Music
A typical facebook profile page.



Facebook Friends list.

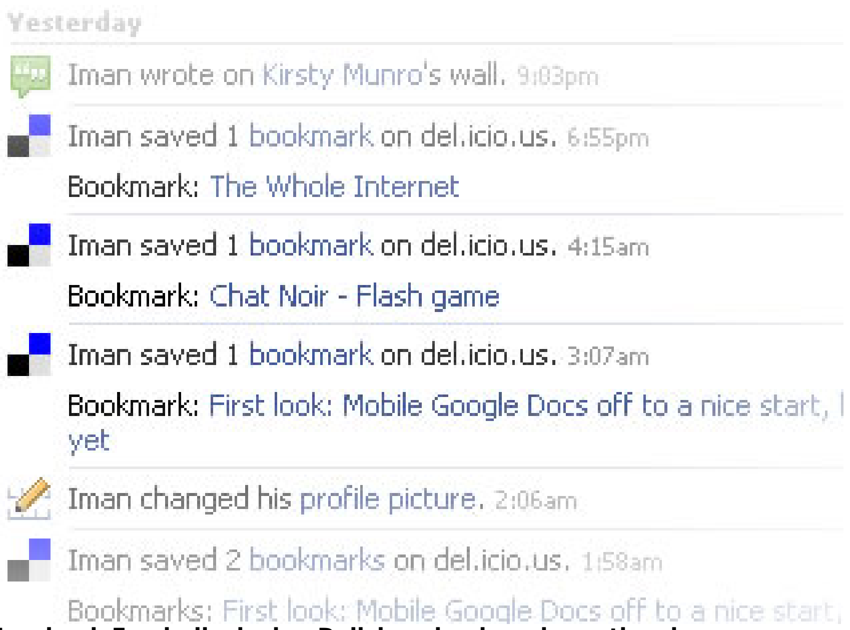
Facebook is a closed community which means you have to register or be invited before accessing many of the functions on the site, it allows you to keep a list of friends across various networks which you belong to. It incorporates messaging functionality, photo hosting, and status notifications as well as groups within its highly interactive profile pages.

It also allows developers to create applications which extend the core functionality of Facebook and utilise the power of existing networks to spread.



Facebook Visual Bookshelf Third Party Application.

Facebook also demonstrates the power for information to be replicated and viewed within different frames on different sites. When I bookmark something on delicious I’ve opted to have my recent bookmarks displayed within my Facebook profile as well.


Yesterday

Iman wrote on Kirsty Munro's wall. 9:03pm

Iman saved 1 bookmark on del.icio.us. 6:55pm

Bookmark: The Whole Internet

Iman saved 1 bookmark on del.icio.us. 4:15am

Bookmark: Chat Noir - Flash game

Iman saved 1 bookmark on del.icio.us. 3:07am

Bookmark: First look: Mobile Google Docs off to a nice start, I yet

Iman changed his profile picture. 2:06am

Iman saved 2 bookmarks on del.icio.us. 1:58am

Bookmarks: First look: Mobile Google Docs off to a nice start,

Facebook Feed, displaying Delicious bookmarks as they happen.

Likewise, I can have my Flickr photos displayed on there too. Some of this has been made possible using RSS or really simple syndication and other agreed protocols and methods for sites to inter-operate. In the case of RSS feeds, essentially my Flickr photos, my Delicious bookmarks or my Wordpress blog posts have their own automatically generated file which other sites, or feed readers can utilise to link to my freshly updated content. Facebook literally grabs the most recent information you allow it to (links and photos) and redisplays it within your profile page.

Although all of this aggregation is currently by choice, there are many aspects to the Facebook experience which have come under fire from critics as being far from the in users control or deliberately confusing. [26] Facebook doesn’t offer the same amount of granular control which Flickr offers for individual items of data on your profile being accessible or locked away. Joining a large network, by default, allows members of that network to view your page without restriction.

Privacy advocates and ordinary users who have lost their basic privacy due to the illusion of privacy which Facebook fosters are naturally outraged. There have been concerns over colleges in the US taking action against their students based on specific Facebook activity [27], and a notable case with MySpace [28] where a 27-year-old student teacher was allegedly denied teaching credentials as her photo labeled drunken pirate, depicting her holding a plastic cup (possibly containing alcohol), was deemed "unprofessional" and encouraged underage drinking.

A critical view of recent developments

We are experiencing a time of rapid change and development, where many possibilities exist in the use of the current offering of technologies and tools, to great benefit. Up to this point in this article, a number of these tools have been profiled while mentioning some of the potential implications and concerns over misuse of the data we are willingly sharing. There is very little debate that these tools are simplifying previously complex tasks and enabling us to do things quicker, with less hassle, but what else are they enabling? Are we discovering these features and the full extent of their reach only when it’s too late?

“information collected by many Web sites — browsing histories, search histories, wish lists, preferences, purchase histories — is amassed and manipulated in ways consumers never know about.” [29]

On Social networking sites, we can profile our musical interest, our reading lists, political affiliations and educational history, we can form groups with other people, and have discussions within those groups. Essentially new forms of expressing our interests and telling people what we are doing every minute of every day are being developed and popularly accepted. But given the clandestine nature of what goes on behind the scenes and those broadly worded terms of service agreements, it does seem strange that we are voluntarily submitting quite so much personal information, for other individuals and establishments to peruse with such ease.

With something as simple as advertising that is being displayed on Gmail pages, there was initially a reasonable amount of controversy which surrounded Google’s fabled content extraction technology. Essentially this feature delivers highly targeted, relevant advertising in relation to the contents of your email. Information concerning the geographic origins and local time of your email correspondent is allegedly amongst the things which content extraction uses and analyses. One could argue that these businesses are legitimately driven by profit, and therefore need to utilise ingenious methods to conduct business, but how far are we letting them go? Is our data and reputation more valuable to us than the price they place on it?

Literally speaking, “MySpace was sold in 2005 for a price that corresponded approximately to 35 US$ per user profile. In 2006 Facebook sources suggested a valuation for their network of 2 billion US$ which would translate to 286 US$ per user profile” (Giles Hogben, 2007), yet the damage that could be done by an invasion of privacy comes at much higher costs.

“The consequences of misusing Facebook can damage a student's chances for their career and hinder their social life.” (Brandon J. Mendelson 2005) [30]

Privacy International which are a human rights group, in a recent survey of several web based services and companies, deemed Google to be hostile to user privacy. They also identified breaches in several other services and their offerings. [31] Primarily, it's the lack of transparency and clarity regarding some of the issues raised in this article which are worrying, as they can set irreversible trends and lower the expectations of the levels of privacy by future users if we become complacent and accepting of the current methodologies for handling our privacy matters.

While the companies themselves aren’t necessarily out to get us, their attitudes and responsibilities towards protecting our data are questionable. Facebook's relationship with third party developers is an example of such. Third parties who develop applications for Facebook aren’t vetted by Facebook and their seemingly innocuous applications can access our user data. Or as some users have found, we are not in full control of our membership, [32] to the point that it’s very difficult to leave such sites without considerable effort.

As other users of these communities have found out, the slightest criticism of the service itself is sometimes met with very heavy handed resistance and censorship [33] . There have also been well documented cases of data retention and privacy concerns of companies such as Yahoo which have affected the lives of political activists in countries such as China [34] .

With all the affordances web mediated communication and specifically web 2.0 services offer us, and all of their benefits, it's good to bear in mind that there is after all no such thing as a free lunch, and to constantly assume that privacy is very fragile and never guaranteed in the hands of fickle multinational corporations.

Other Services not mentioned in this article

Wordpress.com allows anyone to create and maintain their own weblogs. http://wordpress.com/

Writeboard is a collaborative word processor which makes keeping note of revisions and collaborating with others on writing a document simple to manage. http://writeboard.com

Tadalist facilitates the creation of easy to use to-do lists. http://tadalist.com

Upcoming is a service which allows you to create, manage & share listings for events.
http://upcoming.org

Meebo is an in-browser instant messaging program, which supports multiple IM services, including Yahoo! Messenger, Windows Live Messenger, Google Talk, AIM, ICQ, and Jabber. Meebo also makes it easy to create instant rooms for mini conferences just by sharing a web address. http://meebo.com

Last.fm is an online radio and music based profiles site which allows you to profile your musical taste, and discover new music. http://last.fm

Librarything – allows you to catalogue & share your personal library. http://librarything.com

ELGG - As used by http://community.brighton.ac.uk is an open source social platform and content management system. http://elgg.org/

Bibsonomy, CiteULike, Connotea are further examples of social bookmarking tools.

http://www.bibsonomy.org , http://www.citeulike.org , http://www.connotea.org

Notable Project / Organisation

Attention Trust is an organisation aimed at Empowering people to exert greater control over their "attention data", leading efforts to drive developers and business leaders towards greater transparency and dialogue with users and lawmakers over privacy issues. http://attentiontrust.org/

Services mentioned in this article

Flickr allows you to upload and share your pictures online. Its one of the most successful web 2.0 services. http://flickr.com

Delicious is a fully featured social web-bookmark manager. http://del.icio.us

Google Documents, Spreadsheets and   Presentations.

Document editing with publishing and collaboration features. http://docs.google.com

Google calendar system with sharing/sms notification functionality. http://calendar.google.com

References

SCOTT BATEMAN, CHRISTOPHER BROOKS, GORD MCCALLA (2006)

Collaborative Tagging Approaches for Ontological Metadata in Adaptive E-Learning Systems

GILES HOGBEN (2007), Security Issues and Recommendations for Online Social Networks

European Network and Information Security Agency, ENISA

BRANDON J MENDELSON (2005) Misuse of Facebook. Blogcritics website, [accessed October 2007]

IRINA SHKLOVSKI, S. K., ROBERT KRAUT (2004) The Internet and Social Interaction: A Meta-analysis and Critique of Studies, 1995-2003

THOMAS, VAN DER WALL (2005) Explaining and Showing Broad and Narrow Folksonomies. [accessed October 2007]

Footnotes

[1] Katz & Aspden in The Internet and Social Interaction: A Meta-analysis and Critique of Studies.
http://www.cs.cmu.edu/~kraut/RKraut.site.files/articles/Shklovski04-Inte...

[2]   http://en.wikipedia.org/wiki/Perpetual_beta

[3] "We may use information about you that we collect from other sources, including but not limited to newspapers and Internet sources such as blogs, instant messaging services and other users of Facebook, to supplement your profile." Facebook Privacy Policy. http://www.facebook.com/policy.php

[4] Blog post outlining and highlighting the vaguely worded / veiled threats present in Terms of Service documents which the majority of ordinary users tend to ignore.
 http://scholarsandrogues.wordpress.com/2007/09/02/dont-be-evil-unless-yo...

[5] http://wordpress.org Open Source Blogging Software

[6] Wordpress.com, Tumblr, Typepad, Vox, Blogger, Edublogs are all examples of hosted blogging services.

[7] http://en.wikipedia.org/wiki/Web_2.0

[8] http://en.wikipedia.org/wiki/Social_software

[9] This possibility for linking search records, browsing history, emails, photos and instant messaging conversations is seen by many as being very invasive of ones privacy. Specially in the case of linking search queries to individual users via the unified Google account. Moreover its important to note, Gmail is offering these tools for free because it can make a profit from advertising revenue, as advertising revenues fall one has to be weary of other ways in which they could potentially make money by analysing and selling our data.

[10] Googles Promotional Video for Google Docs: http://www.youtube.com/watch?v=eRqUE6IHTEA

[11] http://www.epic.org/privacy/gmail/faq.html

[12] Letter written in 2004, by thirty-One Privacy and Civil Liberties Organizations to urge Google to suspend Gmail, over concerns that they are lowering privacy expectations.
http://www.privacyrights.org/ar/GmailLetter.htm

[13] http://del.icio.us

[14] http://en.wikipedia.org/wiki/Folksonomy

[15] http://www.vanderwal.net/random/entrysel.php?blog=1635

[16] http://en.wikipedia.org/wiki/Meta_noise

[17] In Metacrap, Cory Doctorow refers to seven insurmountable obstacles with regards to user categorisation or objects. http://www.well.com/~doctorow/metacrap.htm

[18] Online PDF version of “Collaborative Tagging Approaches for Ontological Metadata in Adaptive E-Learning Systems”
http://www.win.tue.nl/SW-EL/2006/camera-ready/02-bateman_brooks_mccalla_...

[19] Thomas Vander Wall’s, “Explaining and Showing Broad and Narrow Folksonomies”
http://www.personalinfocloud.com/2005/02/explaining_and_.html

[20] Daniel Terdiman, “Folksonomies Tap People Power.”
http://www.wired.com/science/discoveries/news/2005/02/66456

[21] Yahoo! Q3 2006 Earnings Call Transcript.
http://seekingalpha.com/article/18639-yahoo-q3-2006-earnings-call-transc...

[22] http://en.wikipedia.org/wiki/Blogosphere

[23] http://en.wikipedia.org/wiki/Social_network_service

[24] http://en.wikipedia.org/wiki/List_of_social_networking_websites

[25] Alexa ranking November 2007.

[26] Danah Boyd, Confused by Facebook
http://www.zephoria.org/thoughts/archives/2007/09/06/confused_by_fac.htm...

[27] Misues of Facebook may cost you. http://blogcritics.org/archives/2005/12/11/213844.php

[28] http://www.bloggingstocks.com/2007/04/30/myspaces-drunken-pirate-sues-co...

http://www.thesmokinggun.com/archive/years/2007/0426072pirate1.html

[29] Online Ads vs. Privacy. NY Times
http://www.nytimes.com/2007/05/12/technology/12online.html?ei=5090

[30] See 27.

[31] http://www.privacyinternational.org/article.shtml?cmd[347]=x-347-553961

[32] http://siyavash2005.googlepages.com/facebook

[33] http://www.theregister.co.uk/2007/09/14/facebook_group_deleted

[34] http://www.nytimes.com/2005/09/08/technology/08yahoo.html

Images

All images courtesy of the author.

" >

span-->

‹ Collaborative development: theory, method and tools up Publishing your work ›

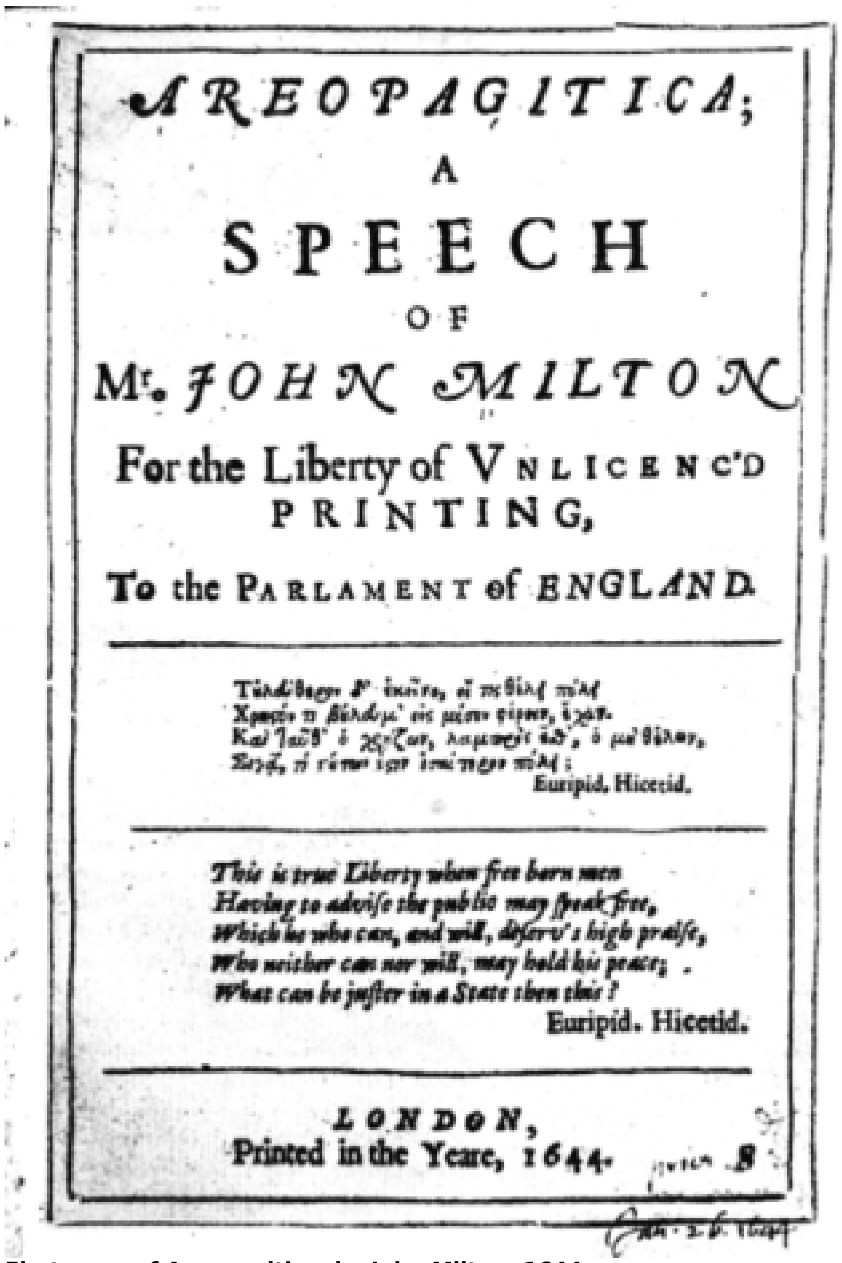
Publishing your work

By admin
Published: 10/04/2007 - 08:30

Kathryn Lambert , October 2007

Self publishing is not a new phenomenon. It is the publishing of all media by the author/artists of those works rather than by established, third party publishers. Although self publishing has been around since the beginning of publishing [5] , it has seen a huge increase in activity with the advancement of publishing technology and the World Wide Web.

The roots of self-publishing lie in social protest and freedom of speech (1993, Kritzinger). Self-publishing, known in the Middle Ages as unlicensed printing, was incredibly risky bussiness, sometimes even punishable by death. What should and should not be made public in printing was strictly controlled by Church and state. A striking historical example of self-publishing is John Miltons Areogapitica (1644) speech on unlicensed printing, published at the height of the English civil war [6] .



AREOPAGITICA;
A
SPEECH
OF
Mr. JOHN MILTON
For the Liberty of VNLICENC'D
PRINTING,
To the PARLAMENT of ENGLAND.

Τοὐλεύθερον δ' ἐκεῖνο, εἴ τις θέλει πόλει
Χρηστόν τι βούλευμ' εἰς μέσον φέρειν, ἔχων.
Καὶ ταῦθ' ὁ χρῄζων, λαμπρὸς ἔσθ', ὁ μὴ θέλων,
Σιγᾷ, τί τούτων ἔστιν ἰσαίτερον πόλει;
Euripid. Hicetid.

This is true Liberty when free born men
Having to adviſe the public may ſpeak free,
Which he who can, and will, deſerv's high praiſe,
Who neither can nor will, may hold his peace;
What can be juſter in a State then this?
Euripid. Hicetid.

LONDON,
Printed in the Yeare, 1644.

price 8
[illegible]

First page of Areopagitica, by John Milton, 1644

Self-publishing is a powerful and essential tool for artists working in the 21st century that can offer a wealth of creative and professional possibilities.

Self publishing is possible both on and off line using desktop, print and electronic publishing. There are a number of reasons that an artist may choose to self publish, ranging from getting your work seen and heard by as broad and new an audience as possible, to harnessing the creative potential inherent in publishing itself.

There are huge opportunities for artists to use self publishing to attract exhibition, purchasing and publishing interest from galleries, festivals, organisations, publishers and collectors. A website for example, can simply act as a calling card for interested people to find out more about your work and get in touch. Or alternatively, can be used to make a strong public statement.

Essentially, electronic forms of self publishing follow radically new models of distribution and consumption by eliminating third party publishers or gatekeepers. Therefore, it can offer artists the ability to act with greater entrepreneurialism and independence by limiting restrictions set out by publishers and galleries and will not only help eliminating potential issues of censorship but will also allow you to directly target specialist audiences and markets for your work. However, this mass marketplace creates new difficulties in achieving an impact with your work as self publishing is now so easy that it is difficult to reach end users and achieve a dent in the marketplace or 'hits' to your work.

One of the most powerful possibilities of electronic publishing is the interactive capability of the internet, that will help you build your regional, national and international networks through a wide range of ever increasing social networking sites and other online collaborative possibilities such as the tools mentioned in the chapter “Working with others”. These ever increasing tools provide platforms for receiving both peer review and audience feedback for your work which has been so difficult to achieve in more traditional forms of publishing, presentation and distribution.

There are also possibilities for you, the author or artist, to retain greater control over how your work is experienced. Although there are some direct challenges for artists to ensure an end user has the experience of their work that was originally envisioned, self publishing does eliminate the need to work within the constraints of a publisher's or gallery's editorial or curatorial control.

In addition to the issue of control, many artists or authors may choose to self publish as it is far cheaper than traditional publishing methods and you can also scale production to an audience or consumers demand. For example, Print On Demand publishing (POD), a digital printing technology employed by publishers to print a book after receipt of order limits wastage of print and makes small scale book publishing affordable.

There are many artists working online who are interested in directly challenging and harnessing the creative potential of self publishing and exploiting this new model of engagement in a more direct conversation with audiences. For example in the Autumn of 2006 artist duo Human Beans established 'What's Cooking Grandma' [1] , a series of films documenting grandmas cooking their favourite recipes that played with notions of brand, product placement and market need.

However, in the age of digital media, although a whole new set of possibilities have flooded the domain of self publishing, this has also brought about an increased set of challenges for artists to face.   For example, limitations of control in terms of ensuring appropriate and quality end user experience and the variety of implications of intellectual property, copyright and plagiarism that need to be considered in this domain.

Limitations of streaming and sharing content can also prove problematic when dealing with varying access to bandwith and choosing your optimum publishing formats. Choosing a format is difficult because of the lack of standards in some parts of this domain, or the refusal of certain software companies to comply with those standards, as for example standards regarding CSS [4] .   When you choose FLOSS publishing formats to share your work there is a great need to educate   your audience about the differing formats one can use and also the value and impact of using them. The dominance of certain non-free formats is limiting the freedom of both artists and audience. This can only be changed by growing numbers of people choosing free and open formats to publish their work. For example, when working with embedded video, using Flash is the only way to reach a large audience at this moment. Free solutions are on the way though. Flash video from ffmpeg [2] and Gnash [3] the GNU Flash Player could provide free alternatives in the very near future.

Another key area of online publishing that needs new consideration is tackling interpretation online through text, spoken word, moving image and a range of rich possibilities that can be explored. Artists are well placed to maximise the potential of new forms of electronic interpretation in a variety of interesting ways. There are also challenges of monitoring impact that artists will need to explore further to monitor the success of their publishing endeavours.

Although there are challenges to navigate, the opportunities often far outweigh the difficulties and can have real impact on your practice, profile and professional development.

Notes

[1]   http://www.humanbeans.net/whatscookinggrandma

[2] Ffmpeg (Fast forward mpeg) is a collection of software libraries that can record, convert and stream digital audio and video in numerous formats. http://ffmpeg.mplayerhq.hu

[3] http://www.gnu.org/software/gnash

[4] Cascading Style Sheets (CSS) is a simple language for adding style (e.g. fonts, colors, spacing) to Web documents written in HTML or XHTML

[5] Kritzinger, A., 1993. Self-publishing: An honourable history, a new dynamism and a bright future. Logos: Journal of the World Book Community, 4/1, 21-25

[6] Areogapitica: A speech for the Liberty of Unlicensed Printing to the Parliament of England, is a text by John Milton, published November 23, 1644. Milton published his text in the form of a pamphlet, defying the same publication censorship he argued against.

Images

[1] First page of Areopagitica, by John Milton, 1644. This image is in the public domain.

" >

span-->

- Licensing: copyright, legal issues, authorship, media work licensing, Creative Commons

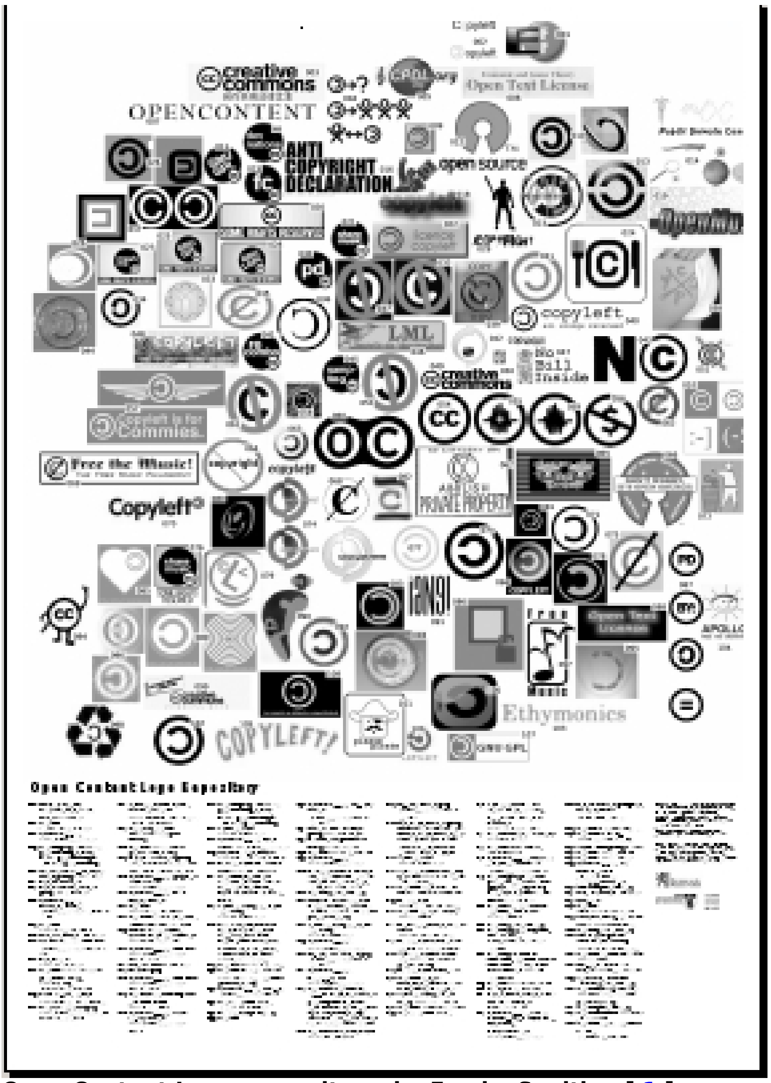
‹ Social networking up Licensing: copyright, legal issues, authorship, media work licensing, Creative Commons ›

Licensing: copyright, legal issues, authorship, media work licensing, Creative Commons

By marloes
Published: 09/18/2007 - 13:32

Nicolas Malev&eacute; , *November 2007*

In this article, we will cover a few questions and principles of open content licensing. We will discuss why to use a license and how it helps to give a stable legal background to start a collaboration. As choosing a license means accepting a certain amount of legal formalism, we will see the conditions required to be entitled to use an open license. Using the comparison of the Free Art License and the Creative Commons, we will try to give an accurate picture of the differences that co-exist in the world of open licensing, and approach what distinguishes *free* from *open* licenses. We will end by envisioning briefly the case of domain specific licenses and with a more practical note on how to apply a license to a work.


Open Content Logos repository, by Femke Snelting [1]

Sharing your work

Let's assume you want to share your work. There may be many reasons to do that. You may want to have fun with others and see unexpected derivatives of your work. You may want to add to a video piece that you found on a website. You may have a band and you want to let your fans download your songs. You may be an academic and you are interested to see your writings circulate in different schools. There are as many reasons as there are people creating artworks and cultural artefacts. This variety is reflected in the ways people work together with each other. This article will discuss some of these ways, those which are characterised by the use of a open license.

Many collaborations happen informally. Your friends take one of your work and incorporate it in one of their works. You insert in your webpage a picture you found somewhere on the net. Someone else does the same with yours. Everything goes smoothly until someone suddenly disagrees. The audio file you sampled from your friend becomes a hit and he summons you to take it down from your webpage because he signed a contract. Someone didn't like the way you used his image on your blog and asks for removal. Whatever happens, if there is no formal agrement, the rule that will apply is the rule of copyright.

As the video maker, Peter Westenberg, sums it up in Affinity Video[2]:

"In this context it is worthwhile taking notice that ‘collaboration’ has a connotation of ‘working with the enemy', Florian Schneider writes in his text on the subject: “It means working together with an agency with which one is not immediately connected.” In order to make sure these people I don’t know, or don’t like, can actually copy, change and redistribute my work, I need to permit them to do so. If I do not, the copyright laws will be fully effective , and these laws might be too restrictive for what we want to achieve. I choose and attach an open license to my work, such as Creative Commons, Free Art License in the case of publishing media files, or the GPL General Public License in case of publishing software.

Counter to copyright, Free Licenses do not protect my exclusive rights of exploiting the object. They defend the rights of the object to be freely used and exploited by anybody, that is: including me. A license also helps me to protect me from what I want: which is applying trust, friendlyness, generosity and all other warm qualifications of personal relationships as if they were reliable protocols for exchange."

If not mentioned otherwise, your work is considered copyrighted, at least in the USA and most European countries. To open it for reuse or distribution, you need to make your intentions clear. To give warranty that you will not change your mind, to the people that will build on top of your work or re-distribute it, the easiest way to proceed is to chose a license. Using a work which is released under an open license gives you the insurance that the conditions under which it is released will not change. This license will list all the conditions under which you, the author, grant the rights to copy, distribute or modify your work as well as the duration of such an agreement. This license will list all the conditions under which you, the subsequent author, can use this work in your creations and will have consequences on the way you will release your own creation.

Open licenses help to clarify many things. This is why they must go along with the creative process. The choice of the license should be made early in the process of creation. If you plan to make a collaborative work or to base your work on someone elses, it is preferable to think about the conditions under which you will release your production. Choosing a license can help clarify the roles of the collaborators in a project, discuss each other expectations about its outcome and define the relationships of this work towards other works.

Finally, a license can be a very effective tool to counter the predatory behaviour of people using free material but unwilling to share the resulting work. Most of the open licenses include a clause that force people to share the subsequent work they produce under the same conditions as the material they used to create it. A license is a wonderful tool to keep available to the public the cultural material that belongs to it.

Going legal

Choosing a license means that you will formalise the way you want to share, collaborate. This means that you will have to enter (as briefly as possible) in the strange logic of lawyers.

Are you a 'valid' author?

This question sounds absurd. Why would some authors be more valid than others?

To be able to legally open your work to others, you need to be the 'valid' author of your work which means that you can't incorporate copyrighted material in your work. You can't use an open license to launder copyrighted material that you have borrowed elsewhere. If your creation has been made as part of your job, check if by contract your copyright doesn't belong to your employer, or if you haven't transfered your rights to anyone.[3] When it comes to open licensing, it is crucial to understand that you operate under the conditions of copyright, but you reformulate them. Open licenses propose a different use of copyright: they use the author's right to authorise, rather than to forbid.

Licenses

To authorize

The author's right is a right to authorise and prohibit. This authorisation can be negotiated subject to remuneration. An author can, for example, authorise, against financial compensation, the reproduction of its work, its audio-visual adaptation, etc. This practice is so widespread that many people amalgamate author's right and right of remuneration. The open licenses stress the possibilities for an author to authorise certain uses of its work free.

An open license:

- offers more rights to the user (the public) than the traditional use of the author's right. For example, the author can grant the free access to a work, can grant the right to redistribute it, to create derivative works.

- clarifies and defines the use which can be made of a work and under which conditions.

For example, the author can grant the free redistribution of his work, but only in a non-commercial use. Authorisation is given automatically for the uses which it stipulates. It is not necessary any more to ask the permission the author since its work is accompanied by an open licence.

Open and Free

Until now in this text we only talked about *open* licenses. It is time now to make an important difference. Open means a 'less restrictive' license than default copyright. It englobes licenses which grant quite different types of permissions. Some licenses, however, are described as *free*. The use of the term *free* as in free software[4] or free license has a very specific meaning. A software that is free, aka copyleft, must give the user the following rights:

1. the freedom to use and study the work,
2. the freedom to copy and share the work with others,
3. the freedom to modify the work,
4. and the freedom to distribute modified and therefore derivative works.

And the license has to ensure that the author of a derived work can only distribute such works under the same or equivalent license.

Typically, if a developper wants to make his/her software free (instead of open), he or she will choose the Gnu GPL [5], a license that grants and secures these rights. A free license for creative content will therefore grant the same rights to other creators.

1 the freedom to use the work and enjoy the benefits of using it

2 the freedom to study the work and to apply knowledge acquired from it

3 the freedom to make and redistribute copies, in whole or in part, of the information or expression

4 the freedom to make changes and improvements, and to distribute derivative works

As stated on the freedomdefined.org website [6]: "These freedoms should be available to anyone, anywhere, anytime. They should not be restricted by the context in which the work is used. Creativity is the act of using an existing resource in a way that had not been envisioned before."

This distinction between free and open is not only a political one but also has important practical consequences. To approach these differences we will compare two important types of licenses that are used by many artists and creators, the Free Art License and the Creative Commons licenses.

The Free Art License

The Free Art License (FAL) [7] was drawn up in 2000 by Copyleft Attitude, a French group made of artists and legal experts. The goal was to transfer the General Public License to the artistic field. In the GPL, Copyleft Attitude was looking for a tool of cultural transformation and an effective means to help disseminate a piece of work. The world of art was perceived as being entirely dominated by a mercantile logic, monopolies and the political impositions deriving from closed circles. Copyleft Attitude tried to seek out a reconciliation with an artistic practice which was not centred on the author as an individual, which encouraged participation over consumption, and which broke the mechanism of singularity that formed the basis of the processes of exclusion in the art world, by providing ways of encouraging dissemination, multiplication, etc. From there on, the FAL faithfully transposes the GPL: authors are invited to create free materials on which other authors are in turn invited to work, to create an artistic origin from which a genealogy can be opened up.

The FAL shares with the GPL the project of re-examining the existing terms of the relations between individuals and access to creation and artworks. The FAL does include elements of great interest from an egalitarian point of view between the creators who use them. The position of the different authors in the chain of works does not consist of a hierarchy between the first author and a subsequent one. Rather, the licence defines the subsequent works as original works "resulting from modification of a copy of the original work or from modification of a copy of a consequent work", and throughout the text of the license they are mentioned regularly. This concern has left its mark on various of the group's practices and, of course, on the license logo -- of which there are as many different versions as there are interested users.

Creative Commons

Set up in 2001 by an essentially academic group (legal experts, scientists, employers and a director of documentaries) and backed by one foundation and several universities, the Creative Commons (CCs)[8] acknowledged that their inspiration came from the GPL. However, they are more influenced by the pragmatic potential (how to solve a problem) of the GPL than by its potential to transform. In effect, the CCs present themselves as the "guarantors of balance, of the middle ground and of moderation". Unlike the GPL, which is a specific mechanism for effecting a modification in the system of creation/dissemination of software, the CCs have been set up to smoothen it out, make it more flexible, more moderate, although not entirely different. The main aim is to save the cost of a legal transaction when drawing up a contract, and to restore the friendly image of the Internet - which has been turned into a battlefield with the growing number of lawsuits against Internauts - in order to restore confidence among possible investors.

What the CCs propose is a palette of licenses that offer the possibility of granting users certain rights. These rights may be more limited than those awarded by the GPL and the FAL. Users of the CCs can choose between authorising or prohibiting modification of their work, commercial use of their work and a possible obligation to re-distribute the subsequent work under the same conditions. In the CCs, two distinctions are re-introduced which were not contained in the GPL: the possibility of prohibiting modification of a work and the difference between commercial and non-commercial use. The CCs give the author a predominant position. He or she can decide whether to authorise the subsequent use of the work, and is defined as the original author. When this decision is taken, the authors can request that their names not be associated with a derived work whose contents they do not approve of. If the GPL excludes the commercial/non-commercial distinction (the user is given the freedom to sell the software), it is because the possibility of trading with the resulting code will help accelerate its propagation. The greater the propagation, the greater the dissemination achieved by the free software and the greater the number of monopolies that will be abolished. The business made from a piece of free software is simply considered as another means of propagation.

The GPL (and the FAL) is a license that is monolithic. All the programmers that use the GPL grant the same rights and respect the same rules. Even if they do not know each other, the programmers who use the GPL form a community. This is not the case for the Creative Commons licenses. They are conceived as modular tools for renegotiating individual contracts, based on individual relations. Naturally, we can use the CCs to create a license close to the FAL/GPL; accepting the transformations and commercial use, on condition that the author is mentionned and that these conditions are applied to subsequent works. But this is just one of the possibilities on offer. As tools, these licenses logically anticipate the varieties of conflicts which might arise with the use of the work as a commercial reappropriation or the deformation/de-naturalisation of a text or a film. The CCs don't decide for you what kind of freedom you want to give to the people who would like to access, play with, modify or redistribute your work. They give you the tools to specify these conditions yourself in a form that is legally recognised.

Domain specific licenses

As we said in the introduction, there are many ways one might like to share their work. Many important licenses have been dedicated to specific areas of artistic/cultural production.[9] You can find licenses dedicated to music, to documentation, etc. As these licenses are specific to a domain, they may contain very precise constraints that are absent to general-purpose licenses. Therefore they require careful attention because they may include particular clauses.

The case of the GNU Free Documentation License.

The GNU Free Documentation License (GFDL) [10] is used by one of the most important projects of free culture: wikipedia[11]. Originally, the GFDL has been conceived to:

"make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software."

With the rise of Wikipedia, one can easily imagine making a series of publications based on this rich pool of articles. It is however, very important to read the license carefully since it requires from the publisher a series of extra references and constraints if more than 100 copies are to be printed. Extra precautions are also required if one agregates (mix GFDL licensed texts with others) or modifies the original text.

How to apply a license to your work?

Every license has a website that will give the correct information to help you add the license information regarding your work. Most of them will give you a small piece of text to include in place of the copyright notice, that refers to the full text of the license for in-depth information. Others provide a tool to generate the license info and the links in the format you need (HTML if you need to include in your blog, text format, logos, etc). Additionally, they will create for you a version that will be easily readable for search engines in a standard format called RDF[12]. We will call all these informations related to the license: license metadata[13]. Many platforms on the web that help publish media (Flickr, blip.tv etc) will offer interfaces to select a license and generate the links to their official documents. Your favourite image or sound editing software may assist you in creating the license metadata. And last but not least, search engines may help your work to be found according to the criteria you have chosen in your license.

Doing it by hand

In all cases, you can include the license information manually. First check the official website of your chosen license. You will find there a small piece of legal text that you will have to include next to your creation. As well as tips on how to explain to other people what this legal information means to them.

For example, if you publish an audio-CD under the Free Art License, you will include this information on the cover [10]:

[- A few lines to indicate the name of the work and to give an idea of what it is.]

[- A few lines to describe, if necessary, the modified work of art and give the name of the author/artist.]

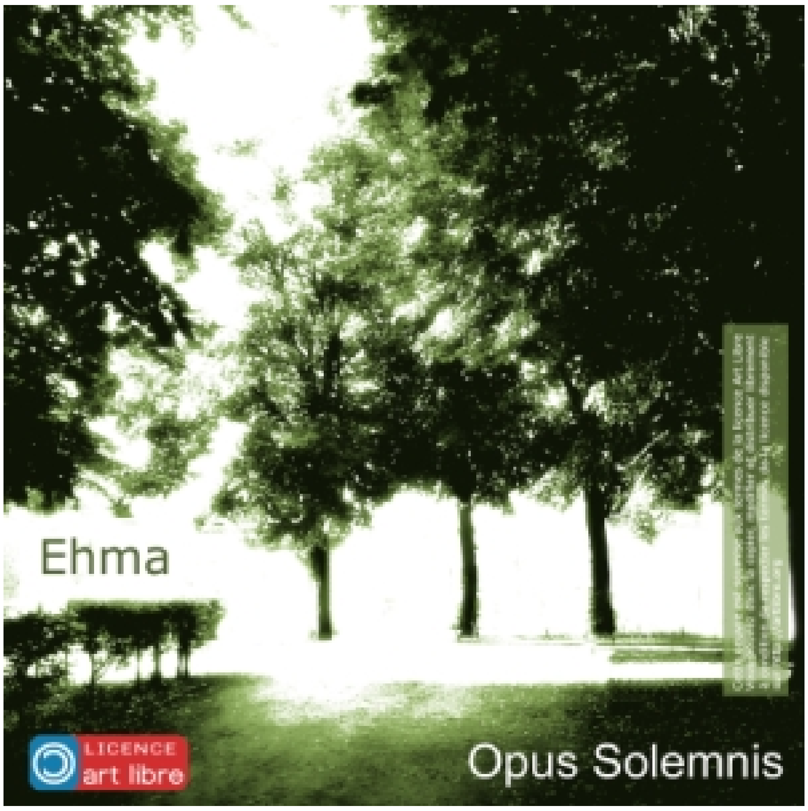
Copyright © [the date] [name of the author or artist] (if appropriate, specify the names of the previous authors or artists)

Copyleft: this work of art is free, you can redistribute it and/or modify it according to terms of the Free Art license.

You will find a specimen of this license on the site Copyleft Attitude http://artlibre.org as well as on other sites.

It is usually a good idea to explain what this license authorises in a few lines in your own style:

You can copy the music on this CD, make new creations with it and even sell them as you want, but you need to give credit to the author and to share the creations you made with this work under the same conditions.

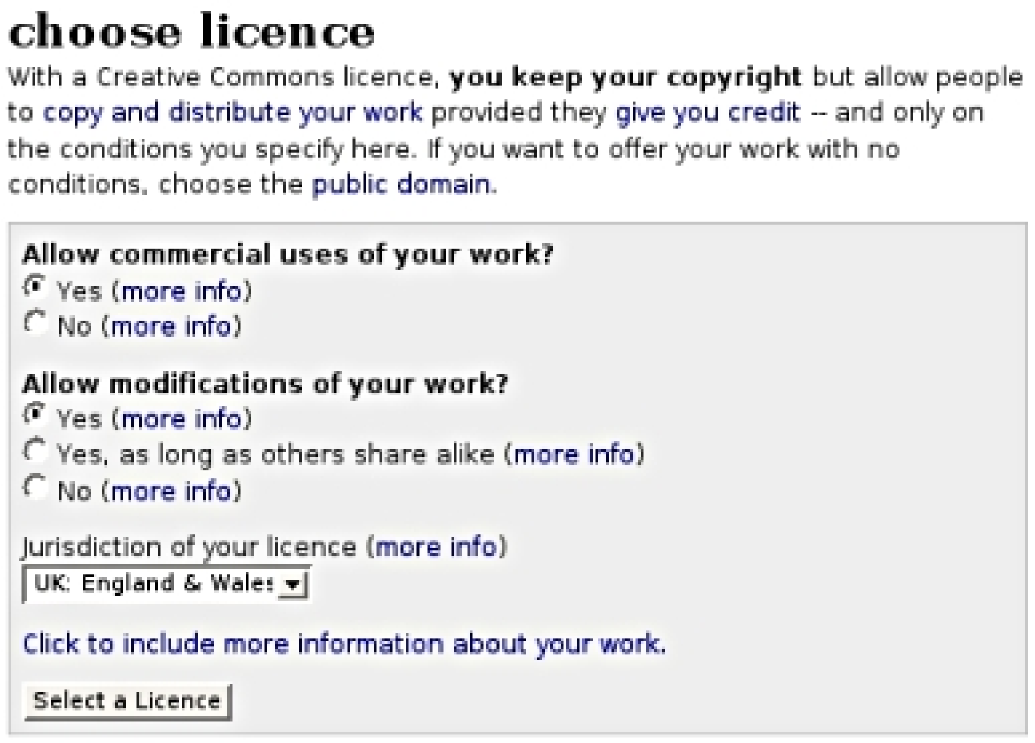


CD cover of the artist Ehma

Above is an example of a CD cover of the artist Ehma [15]. He mentions (bottom left of the image) the type of license he has chosen and more discretely (at the right-hand side), he gives a small description of what you can do with the music and where to find the complete text of the license.

Using Creative Commons license generator.

If you say: "my work is released under a Creative Commons license" it doesn't tell much about what one can or can't do with the work. It is only when one knows if commercial use is granted or if modification is allowed that the license starts to make sense. To make it easier to understand at first sight, Creative Commons produced a series of logo to be put next to the work. Different formats for the same licenses can also be easily produced from the website and links provided to different references documents (the human readable version, the lawyer-redable version and the machine readable version, the one you will use for search engines, ie).   The generation of all these documents and their related links is done via a web interface.


choose licence

With a Creative Commons licence, **you keep your copyright** but allow people to copy and distribute your work provided they give you credit -- and only on the conditions you specify here. If you want to offer your work with no conditions, choose the public domain.

Allow commercial uses of your work?
[✓ Yes] (more info)
No (more info)

Allow modifications of your work?
[✓ Yes] (more info)
Yes, as long as others share alike (more info)
No (more info)

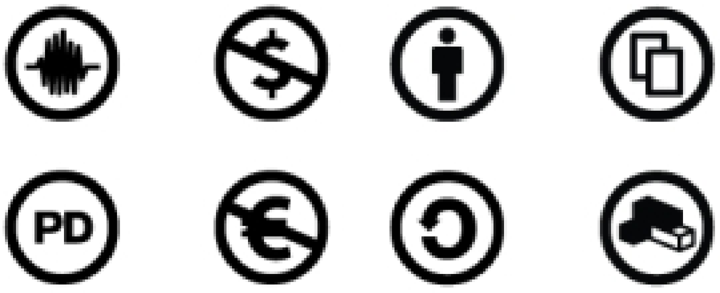
Jurisdiction of your licence (more info)
UK: England & Wales

Click to include more information about your work.

Select a Licence

The license generator of Creative Commons website

For example, let's imagine that you want to publish your photos on your website under a license that allows others to copy, modify, but restricts commercial use and obliges them to share their subsequent work under the same conditions. Going on the creativecommons website, in the "publish your work" section [16], a page will offer you the many options you need to specify the conditions under which you want to publish your work. Once your options are selected, a snippet of html code is provided. You just need to include it in your website next to your photos.

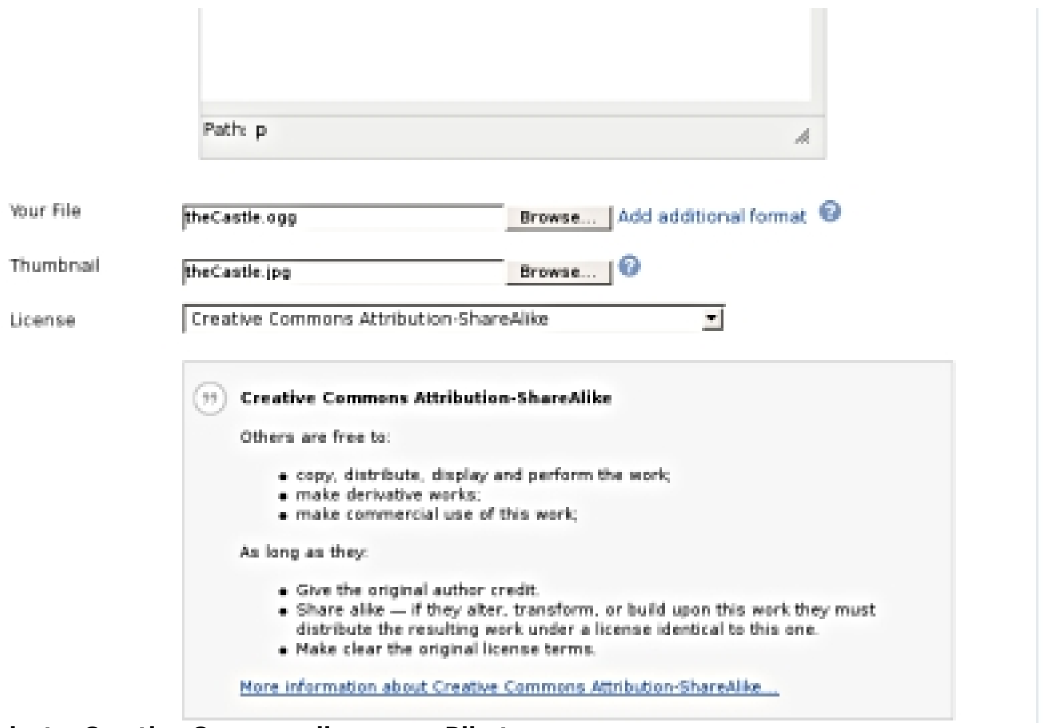


a few examples of Creative Commons logos

Creative Commons put a lot of effort in finding solutions to make it easy to generate license metadata but also to give intuitive symbols explaining what you can do with the artwork. Each condition of use is represented by a logo. A combination of these logos acts as a visual summary of your choices regarding your work.

Using a sharing platform (be cautious!)

If you are a user of a sharing platform like Flickr.com, blip.tv or others, you will be asked to specify the conditions under which you want your work to be distributed. Each of these websites have an easy interface to let you select the appropriate license and automatically generate the reference to the official license text. Beware: if you use one of these services, read their terms of use carefully. Usually they will ask you, before using their service, to agree on a specific set of conditions regarding your copyrights and the use they, as a company (or their affiliates), can make of the works you publish on their platform. These conditions may vary strongly. You have to understand that when you use these platforms, you have to make two licenses. One between you and the platform, that will be dictated by its terms of use and validated when you click on the button "I agree". And a second one between you and the users/visitors of this platform, that can be a license of your choice, selected when you upload your file on the platform. If you use this platform to redistribute the work of someone else (as many open licenses allow you to do) or to publish derivative works (as many open licenses allow you to do), be careful that the terms of use (the contract between you and the platform) are not in contradiction with the license of the work you want to redistribute or publish.


Path: p

Your File: theCastle.ogg

Thumbnail: theCastle.jpg

License: Creative Commons Attribution-ShareAlike

Creative Commons Attribution-ShareAlike

Others are free to:

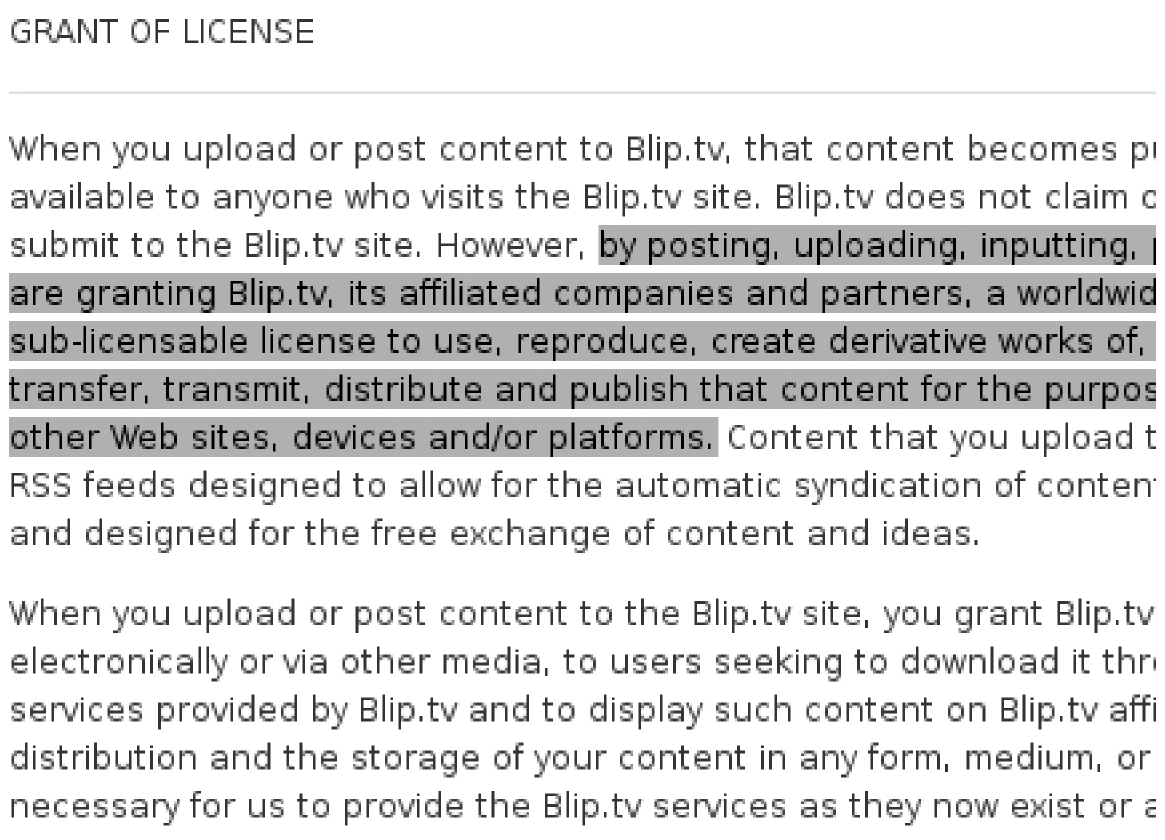
- copy, distribute, display and perform the work;
- make derivative works;
- make commercial use of this work;

As long as they:

- Give the original author credit.
- Share alike — if they alter, transform, or build upon this work they must distribute the resulting work under a license identical to this one.
- Make clear the original license terms.

More information about Creative Commons Attribution-ShareAlike ...

Select a Creative Commons license on Blip.tv



GRANT OF LICENSE

When you upload or post content to Blip.tv, that content becomes pu
available to anyone who visits the Blip.tv site. Blip.tv does not claim o
submit to the Blip.tv site. However, by posting, uploading, inputting, p
are granting Blip.tv, its affiliated companies and partners, a worldwid
sub-licensable license to use, reproduce, create derivative works of,
transfer, transmit, distribute and publish that content for the purpos
other Web sites, devices and/or platforms. Content that you upload t
RSS feeds designed to allow for the automatic syndication of conten
and designed for the free exchange of content and ideas.

When you upload or post content to the Blip.tv site, you grant Blip.tv
electronically or via other media, to users seeking to download it thr
services provided by Blip.tv and to display such content on Blip.tv affi
distribution and the storage of your content in any form, medium, or
necessary for us to provide the Blip.tv services as they now exist or a

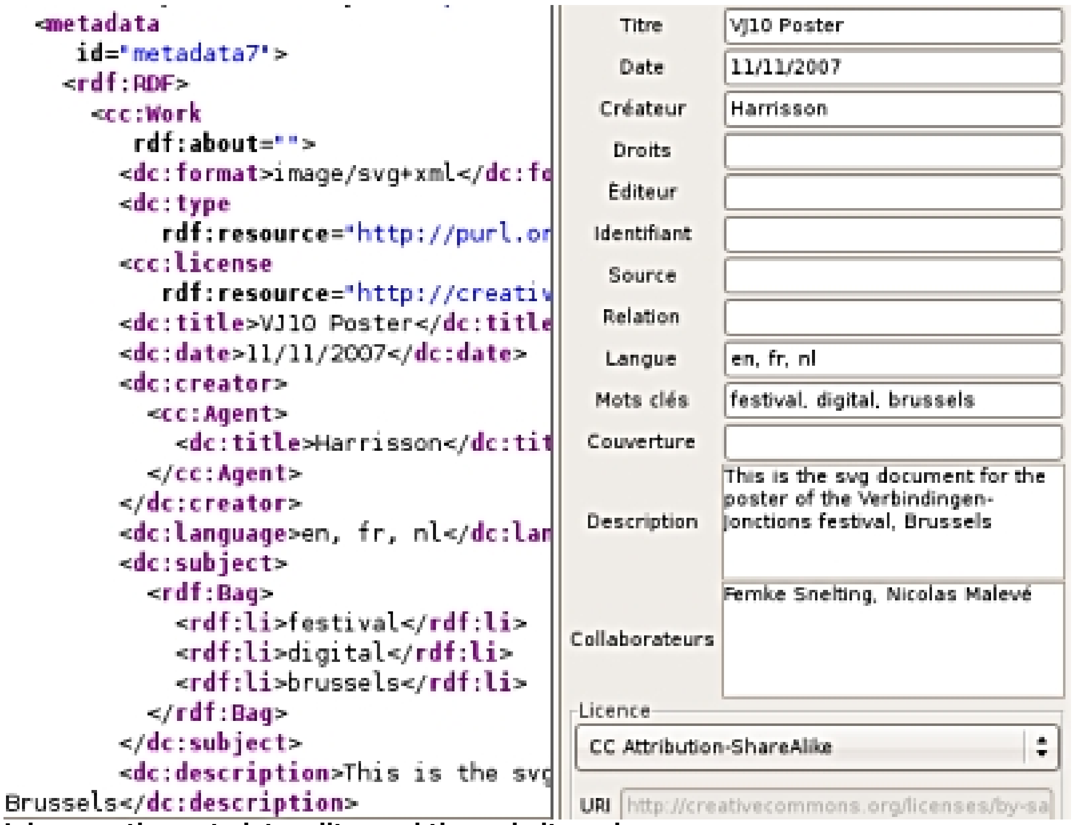
Sample from Terms of service document from Blip.tv

In the example above, a user subscribed to the blip.tv [17] service may choose between full copyright, public domain and several variations of the Creative Commons. But to access such a service, the user has already accepted the terms of use of blip.tv that specify that s/he accepts that blip.tv may reproduce, create derivative works and redistribute her/his videos on blip.tv and affiliated websites. At this point, it is really important to pay attention to potential contradictions.

For instance, if you want to post a video from someone else who accepts that his/her video to be redistributed but not transformed, you can't post it to blip.tv: his/her licensing choice and blip.tv's terms of use are in contradiction. You can't accept for him/her that blip.tv makes derivative work from his/her video. Another problem comes from the non-commercial clause of the Creative Commons licenses. If you want to publish a video based on the work of someone else's footage and if this footage is released under a non-commercial CC license, be extra-careful about what you accept regarding advertising in the terms of service. Usually the platforms reserve the right of placing advertisements in the pages where they show your content and because Creative Commons doesn't define exactly is meant by "non-commercial" [18], you may enter a grey zone. Be even more careful if you chose a formula in which the platform proposes you to share the revenues on advertisements placed in your video or next to it.

Metadata from your favourite editor

Many tools that help you create content may also help you insert license metadata. For writers, Creative Commons propose add-ons that allows license information to be embedded in OpenOffice.org Writer, Impress and Calc documents. For weblogs as Wordpress, a plugin called Creative-Commons-Configurator [19] provides the blog owner with the ability to set a Creative Commons License for a WordPress [20] blog and control the inclusion or display of the license information into the blog pages or the syndication feeds.



```
<metadata
   id="metadata7">
  <rdf:RDF>
    <cc:Work
       rdf:about="">
      <dc:format>image/svg+xml</dc:fo
      <dc:type
         rdf:resource="http://purl.or
      <cc:license
         rdf:resource="http://creativ
      <dc:title>VJ10 Poster</dc:title
      <dc:date>11/11/2007</dc:date>
      <dc:creator>
        <cc:Agent>
          <dc:title>Harrisson</dc:tit
        </cc:Agent>
      </dc:creator>
      <dc:language>en, fr, nl</dc:lan
      <dc:subject>
        <rdf:Bag>
          <rdf:li>festival</rdf:li>
          <rdf:li>digital</rdf:li>
          <rdf:li>brussels</rdf:li>
        </rdf:Bag>
      </dc:subject>
      <dc:description>This is the svg
Brussels</dc:description>
```

Titre	VJ10 Poster
Date	11/11/2007
Créateur	Harrisson
Droits	
Éditeur	
Identifiant	
Source	
Relation	
Langue	en, fr, nl
Mots clés	festival, digital, brussels
Couverture	
Description	This is the svg document for the poster of the Verbindingen-Jonctions festival, Brussels
Collaborateurs	Femke Snelting, Nicolas Malevé

Licence

CC Attribution-ShareAlike

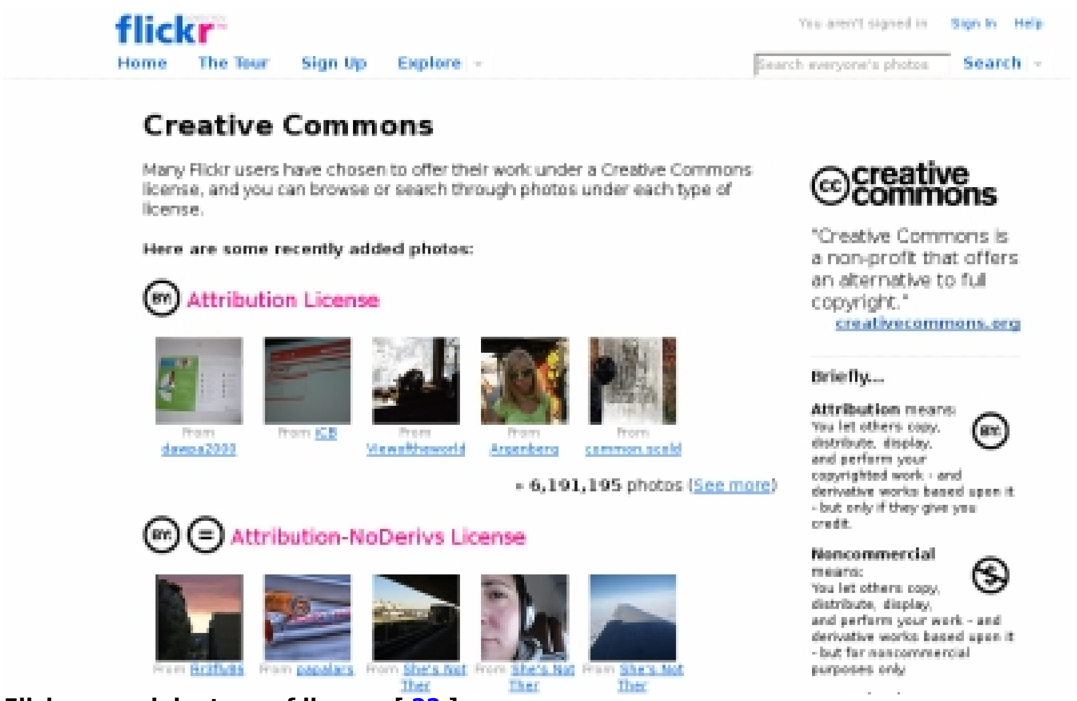
URI http://creativecommons.org/licenses/by-sa

Inkscape, the metadata editor and the code it produces.

The screenshot above shows the example of the Inkscape [21] vector graphic editor. In the menu preferences, if you select the tab "metadata", you will be presented a series of fields to describe your image and a menu that will allow you to select among many licenses (not just Creative Commons). The result is a standard metadata embedded in the code of the image and ready to be parsed by search engines if you post it on the web.

Be found

One of the benefits of using the tools proposed by Creative Commons is that it will help the search engines to "understand" the kind of uses you grant to the internauts.



Creative Commons

Many Flickr users have chosen to offer their work under a Creative Commons license, and you can browse or search through photos under each type of license.

Here are some recently added photos:

Attribution License

From dawna2003 · From ICB · From Viewoftheworld · From Argenberg · From common.scald

» 6,191,195 photos (See more)

Attribution-NoDerivs License

From Kriftyks · From papalars · From She's Not Ther · From She's Not Ther · From She's Not Ther

"Creative Commons is a non-profit that offers an alternative to full copyright."
creativecommons.org

Briefly...

Attribution means:
You let others copy, distribute, display, and perform your copyrighted work - and derivative works based upon it - but only if they give you credit.

Noncommercial means:
You let others copy, distribute, display, and perform your work - and derivative works based upon it - but for noncommercial purposes only.

Flickr, search by type of license [22].

Therefore, Yahoo, Google, Flickr and other search engines or services allow their users to make queries with specific filters: *ie, Search only pages that are free to use or share, even commercially.* They will then list only results that match these particular permissions.

This text has only scratched the surface of open content licensing. We hope that we have clarified a few important principles and given useful starting points. Nothing, however, will replace your own experience.

Notes

[1] http://www.constantvzw.com/downloads/posterOCL.pdf

[2] http://osvideo.constantvzw.org/?p=97

[3] Creative Commons made a list of things to think about before chosing a license: http://wiki.creativecommons.org/Before_Licensing

If you are a member of a collecting society, good chances are that you will not be allowed to use free licenses since you transfered your rights management to it. Check with them if any compromise can be found.

[4] http://www.gnu.org/philosophy/free-sw.html

[5] Freshmeat that "maintains the Web's largest index of Unix and cross-platform software, themes and related "eye-candy"" provides statistics of the referenced projects: more than 63 % of these projects are released under the Gnu GPL. http://freshmeat.net/stats/

[6] http://www.freedomdefined.org/Definition

[7] http://artlibre.org/licence/lal/en/

[8] http://www.creativecommons.org

[9] For more development on domain specific licenses, check the fourth chapter of the Open Licenses Guide, by Lawrence Liang published on the website of the Piet Zwart Institute: http://pzwart.wdka.hro.nl/mdr/research/lliang/open_content_guide/05-chap...

[10] http://www.gnu.org/licenses/fdl.html[11] http://www.wikipedia.org

[12] Rdf, Resource Description Framework, general method of modeling information, through a variety of syntax formats. See: http://en.wikipedia.org/wiki/Resource_Description_Framework

[13] Metadata simply means data about data. See: http://en.wikipedia.org/wiki/Metadata#Image_metadata

[14] http://artlibre.org/licence/lal/en/ see bottom of the page, FAQ.

[15] Ehma, http://www.jamendo.com/en/artist/ehma

[16] http://a3-testing.creativecommons.org/license/

[17] http://blip.tv/

[18] There are a lot of questions concerning the use of the non-commercial clause in the Creative Commons licenses. The CC website anounces that:" *In early 2008 we will be re-engaging that discussion and will be undertaking a serious study of the NonCommercial term which will result in changes to our licenses and/or explanations around them.*"
http://wiki.creativecommons.org/NonCommercial_Guidelines

[19] http://www.g-loaded.eu/2006/01/14/creative-commons-configurator-wordpres...

[20] http://www.wordpress.org

[21] http://www.inkscape.org

[22] http://www.flickr.com/creativecommons/

" >

span-->

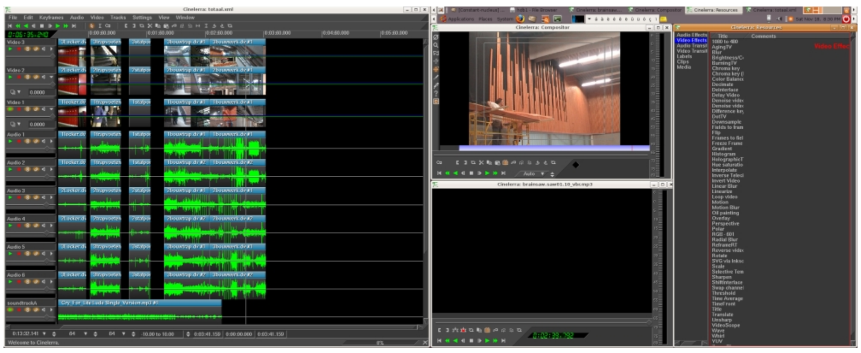
‹ Publishing your work up Working with digital video ›

Working with digital video

By admin
Published: 10/04/2007 - 08:35

Peter Westenberg , October 2007

Working with digital video is part of many artistic disciplines. Besides single screen narratives, video productions can range from animation, multiple screen installation to interactive work. Still, many aspects of digital video can be traced back to the history of film. The interface of a timeline editing software such as Cinelerra [1] shows a multitrack timeline, a viewing monitor, a bin for clips; echoing the setup of a flatbed table for editing celluloid.



A dual head set up Cinelerra work station

The physical materiality of film and video are fundamentaly different: celluloid versus pixels, chemicals versus algorhytms, but the relationship between film and video has mutually matured. As outlined by Matt Hanson [1b] , video expands cinematographic traditions in new directions, filmmakers can benefit from digitisation by reclaiming the central position of creativity in the film process, as pointed out by Samira Makhmalbaf. [1c]

An 'Old Delft Cinemonta' 16mm editing table in use at the Filmwerkplaats in Rotterdam

Digital video also roots in artistic practices of the sixties and seventies. [1a] Artists started using video to capture temporary performances (Joan Jonas [2] , Vito Acconci [3]), they integrated video monitors in installations (Nam June Paik [4]), experimented with filters and mixing in video paintings (Peter Campus [5]). Compared to film cameras, video cameras had a strong feature: it became possible connect a monitor and view directly what the camera recorded. Today, artists can use softwares such as Lives [5] , Jahshaka [6] , Zone Minder [7] or Pure Data [8] and Linux distributions aimed at audio and visual creation such as Dyne:bolic [9] Apodio [10] and Ubuntu Studio [11] to further explore the possibilities of real time video, multiple camera input and live interaction.

When integrating programming into the work process, videomakers can use the computer's calculating characteristics to its fullest potential. Scripts, written in programming languages such as Perl [12] or Python [13] , combine functionalities of different software-programmes into automated processes. They can be used to generate (abstract) imagery, to carry out repetitive actions, to place images into sequences. Combine for example the functionalities of 'Swiss army knive' ImageMagick [14] and the commandline options of Mplayer [15] with other small scripts to create a set of tools which enables a unique, made to measure, visual vocabulary.

During the past few decades, portability, affordability and improving technical quality popularised the usage of video. Many generations of standards saw the light, from Betamax [16] , Video 2000 [17] to VHS [18] . Stand alone solutions such as the now obsolete 'Casablanca' videosystem [19] , and later Imovie and MovieMaker fulfilled the home cinematographers desire for easy editing solutions. Kdenlive [20] and Kino [21] fit in this tradition of user friendly tools allowing for quick drag and drop cuts and pastes. With PVR initiatives such as MythTV [22] , television is entering the free desktop.

Since the late eighties journalists and activists armed themselves with lightweight material. With protagonists such as Paper Tiger TV [23] , video activism matured rapidly. The birth of the internet provided suitable platforms for alternative media (Indymedia [24] , Democracy now [25]) and the contemporary civil journalist publishes on videoblogs and video sharing websites. (Blib.tv [26])

Sampling of sounds and images has become common practice for artists and the web and P2P networks offer great opportunities for exchanging mediafiles.

Video platforms such as Miro [27] bring together video rss feeds from various hosts and websites. Media archives such as Archive.org [28] provide makers with good quality content which can be reworked and published under a permissive license (Free arts License [29] , Creative Commons [30]). The advancing development of online editors (Movie Masher [31]) allows for the remixing of video without having to download heavy files.

Many videomakers work with material which is being produced by others; starting a project with clips of various different digital formats is common practice. Encoding softwares such as FFmpeg [32] , Mencoder [33] and Transcode [34] enable video makers to transfer digital formats into others. Codecs [35] are small softwares to Code / Decode, Compress / Decompress video. Many codecs are not free, and thus difficult to manage in a free software environment. Free codecs (Ogg vorbis, Theora [36]) on the other hand can be read on all platforms using popular mediaplayers such as VLC [37] . Streaming content can be embedded in webpages using the Cortado Java applet [38] , guaranteeing playback in browsers such as Firefox without having to install additional plug ins. Choosing free video does not mean that others cannot see your work, on the contrary.

When you start working with free software as a videomaker, it is likely that you need to invest some time and energy in understanding certain basics of the video production process. Sometimes this might mean you have to look for alternative workflows, to dive 'under the hood' of a digital tool, or reconfigure an existing solution to suit your needs. Investigating the tools you use as a video maker is an important part of the job, it can help gain insights, it can be an inspiration to explore new ways of working and imagemaking.

Notes

[1] Cinelerra : http://cinelerra.org

[1b]End of Celluloid : http://www.endofcelluloid.com

[1c] At the occassion of the inclusion of her film Blackboards in the festival of Cannes, Samira Makhmalbaf gave this talk on 10 May, 2000 : The Digital Revolution and The Future of Cinema : http://www.library.cornell.edu/colldev/mideast/makhms.htm

[1a]History of Video art : http://en.wikipedia.org/wiki/Video_art#History_of_video_art

[2] Wikipedia on Joan Jonas : http://en.wikipedia.org/wiki/Joan_Jonas

[3] Wikipedia on Vito Acconci : http://en.wikipedia.org/wiki/Vito_Acconci

Work from Acconci on Ubuweb : http://www.ubu.com/film/acconci.html

[4] Wikipedia on Nam June Paik : http://en.wikipedia.org/wiki/Nam_June_Paik

[5] Wikipedia on Peter Campus : http://en.wikipedia.org/wiki/Peter_Campus

[5] LiVES : http://lives.sf.net

[6] Jahshaka : http://www.jahshaka.org

[7] Zone Minder : http://www.zoneminder.com

[8] PureData : http://puredata.info

[9] Dynebolic : http://dyne.org

[10] Apodio : http://www.apo33.org/apodio/doku.php

[11] Ubuntu Studio : http://ubuntustudio.org

[12] Perl : http://www.perl.org

[13] Python : http://www.python.org

[14] ImageMagick : http://www.imagemagick.org/script/index.php

[15] MPlayer : http://www.mplayerhq.hu/design7/news.html

[16] Betamax : http://en.wikipedia.org/wiki/Betamax , http://www.sony.net/Fun/SH/1-13/h5.html

[17] Video 2000 : http://en.wikipedia.org/wiki/Video_2000

[18] VHS: http://en.wikipedia.org/wiki/VHS

[19] Casablanca : http://www.macrovideo.co.za/video_editing_systems.html

[20] Kdenlive : http://www.kdenlive.org

[21] Kino : http://www.kinodv.org

[22] MythTV : http://www.mythtv.org

[23] Paper Tiger Television : http://papertiger.org

[24]Indy Media : http://www.indymedia.org/en/index.shtml

[25] Democracy Now : http://www.democracynow.org

[26] Blib.tv : http://blip.tv

[27] Miro : http://getmiro.com

[28] Archive.org : http://www.archive.org

[29] Free Arts License : http://artlibre.org/licence/lal/en

[30] Creative Commons : http://creativecommons.org

[31] Movie Masher : http://www.moviemasher.com

[32] FFMPEG : http://sourceforge.net/projects/ffmpeg

[33] Mplayer : http://www.mplayerhq.hu/design7/news.html

[34] Transcode : http://www.transcoding.org/cgi-bin/transcode

[35] Codecs : http://en.wikipedia.org/wiki/Codec

[36] Ogg + Theora : http://www.theora.org

[37] VLC : http://www.videolan.org/vlc

[37] Cortado : http://www.flumotion.net/cortado

Images

[1] A screenshot of a dual head set up Cinelerra work station. Info on this topic and related subjects can be found on Open Source Video. http://osvideo.constantvzw.org

[2] This image shows an 'Old Delft Cinemonta' 16mm editing table in use at the Filmwerkplaats in Rotterdam, one of the few places in Europe where artists can work with celluloid. The WORM.filmwerkplaats is an open studio in which you are allowed to work independently. Filmmakers and artists are free to use our equipment to be able to shoot, edit and complete an entire film at minimum cost. We can also act as a producer for film projects and assist in the production and completion of the films themselves.

Image by Esther Urlus (Attribution-Non-Commercial-Share Alike 2.0 Generic)

http://creativecommons.org/licenses/by-nc-sa/2.0/deed.en_GB

" >

span-->

- Video editing with open source tools

‹ Licensing: copyright, legal issues, authorship, media work licensing, Creative Commons up Video editing with open source tools ›

Video editing with open source tools

By jennie
Published: 05/11/2009 - 12:30

Valentina Messeri & Eleonora Oreggia , May 2009

Video editing is the process of cutting and pasting video sequences, in order to create a fluid, often narrative, combination of images and sound that develops in time following a certain structure or story board. While analogue video editing requires both a substantial investment in equipment and an aesthetic project, for example a detailed edit list and a well structured script, digital video editing on the other hand has become much more accessible. It is an easy and experimental way to create combinations of any type of footage. Huge databases of video images are readily available and the costs for setting up a home made video editing station are progressively decreasing. Software applications are more reliable, stable and easy to find, machines are faster, video cameras cheaper and hard drives larger. Digital video editing, a non-linear or non destructive method (in comparison to the physical cutting of the negative in film), can also be viewed as the audio/video equivalent of word-processing.

The video-image, unstable sister of the film-image, has become the low-key protagonist of late 20th Century audio-visual imagery. Since the beginning of the 21st century video images are proliferating on the web. Video is nowadays a common medium of expression, and has even become a substitute for socialisation. For example, the widespread use of webcams, and the integration of those in VoIP (Voice over IP) systems, modified the phone call into an audiovisual experience (teleconferencing). The possibility to upload and share fragments of any kind of video on socialisation platforms such as YouTube also creates the possibility to use video as an instrument to network with people, using it to present oneself, and to express personal opinions. Video, in this sense, becomes the mediator between internal, intimate space and external, public context, materialising relations and the interaction between Self and Other.

Digital Video Editing

Before starting any video editing project, it is very important to make sure that the computer has enough RAM (Random-Access Memory) [1] , sufficient hard disc space [2] , and possibly that an external FireWire drive is at hand. USB and USB2 are too slow for this purpose and frames could drop during capturing (this means some frames could be skipped while importing the video, leading to an incomplete file). It is a good habit to clean your computer before beginning a long term video project, in order to avoid congestion.

If you are running an open source operating system like Linux, it is very important that everything is well installed and configured, that the software is recognising and using all the hardware and its capabilities (for video editing FireWire, external or extended monitor and in some cases 3D acceleration are important). For real time video editing, the X environment (the graphical interface) has to be properly configured [3] . A number of video friendly Linux distributions, such as Ubuntu Studio [4] and pure:dyne [5] can make installation and configuration really easy. When your computer is ready to go, you need to choose a software to do the editing with.

What a video editing application is supposed to offer is four basic functions: import, edit, export, and the possibility to manipulate audio and video.

Importing and capturing media

A video sequence is made of sound, video, still images and graphics. To use digital media (sound, video, etc.) inside a video editing software it is necessary to import all the material required into the project. When the material already consists of digital files, the import process simply creates a symbolic link between a folder inside the source view of the program, and the physical location on the hard-drive. To capture from any external device, such as a video camera, a mini-disk or any other type of player, a digital file on the disk will be created as new. The best quality for importing from a digital camera is obtained by using FireWire. To use it you need to have a FireWire port on your computer and a FireWire output on the video-camera. From photo-cameras it is normal to import using USB. S-video and composite ports provide a lower video quality and can be used if the computer has an analogue port. Audio is normally captured from the audio input.

Editing

Video editing applications are normally organised in the following way: one screen, usually on the left side, displays the source material; the display on the right side shows the sequence as it is structured on the timeline. The timeline can be expanded in different layers that allow superimposition and the combination of different clips, images and sound. On the timeline the source material is mixed, merged and combined to create the new edited sequence.

Manipulating video

Every editing software offers a set of effects and transitions. The traditional ones are inherited from analog video editing, others belong to the digital world. Transitions are used to attach one clip to the next without any visual incongruence or noise that can break the cinematic illusion or distract the eyes. Experienced video editors tend to use a very small amount of those, because in fact, a masterly cut does not need any transition, and the sequence is fluid by itself. Effects can be used to modify colours, lights, image size and basic motion characteristics, or to create psychedelic and unnatural compositions. Although the newbie video editor might experiment with effects, a clean and well organised video piece does not need a lot of manipulation. Individual style and aesthetics can obviously contradict these guidelines. Once set the desired transitions and effects, the process of 'Rendering' will construct a model of the sequence which can be played and displayed. Rendering helps to make small corrections and to calibrate the sequence on the timeline before exporting the video clip.

Exporting (Formats and Codecs)

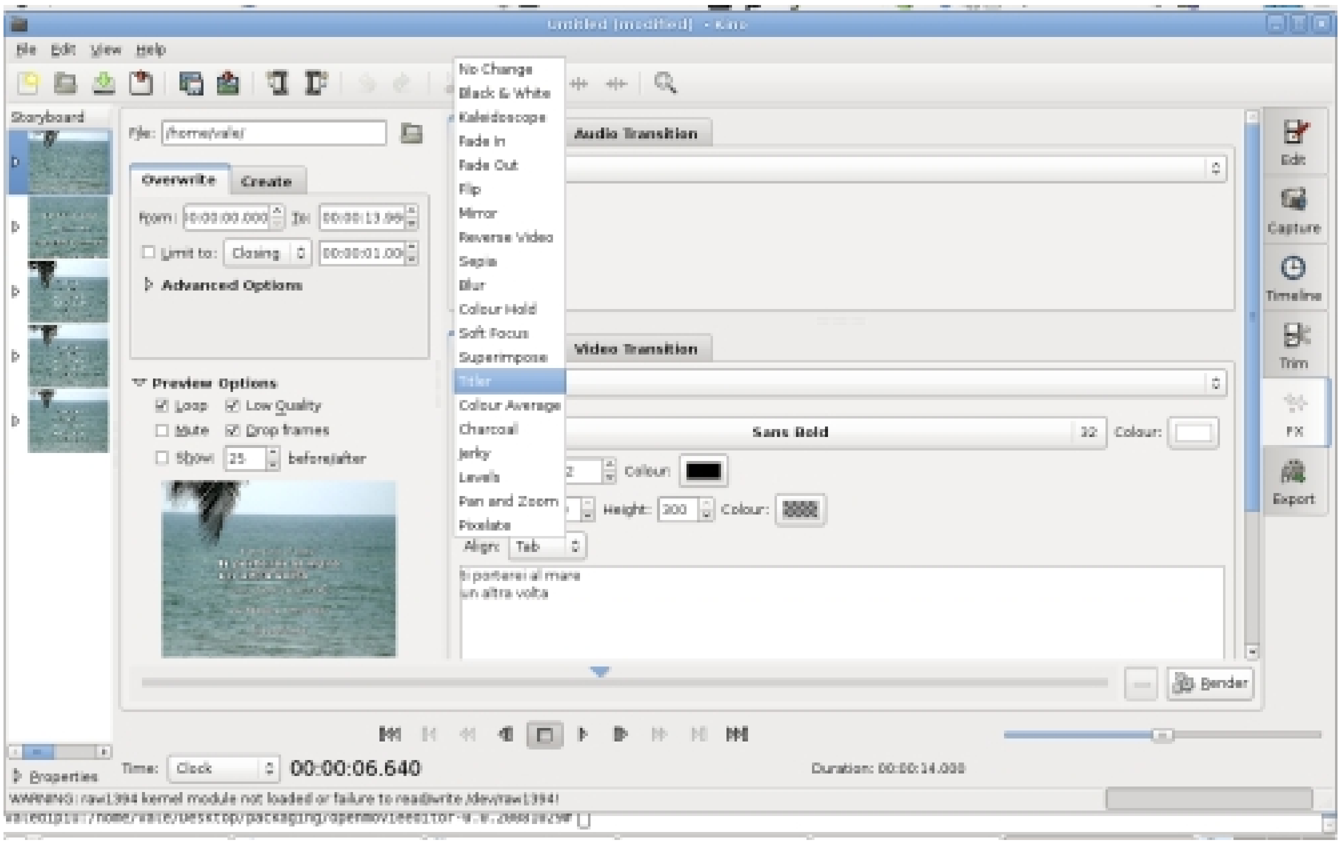
Understanding digital video means getting accustomed to formats and codecs. Whereas in analogue systems video can be defined as an electronic signal stored on a certain carrier, digital video is a numeric description of this electronic signal. A codec is an algorithm used to optimise, in terms of file size, the description of this data. If a codec is a method used to store video information in a compressed form, a video format can be imagined as a box which contains the data compressed. For example, if a video file has .mov as its extension, its format is QuickTime, while if a file presents a .ogg extension, the container is Ogg, the free, open standard container format maintained by the Xiph.Org Foundation, which can contain an audio file compressed with a Vorbis codec, or a media file whose video codec is generally Theora. Mp4 can be confusing because it is both an extension (thus a format: .mp4) and a codec which can be used to compress video to be contained in any other format. The combination of format and codec structures the compatibility of a file with different video players ; for example the free player VideoLAN [6] and MPlayer [7] can play almost any type of video file, proprietary players are often more restrictive. The selection of a certain codec rather than another, is more significant and consequential than one might think:using free open source codecs means, amongst other things, not binding your content to a commercial company.

Once a video sequence or a project is ready, or requires to be saved and stored, it is time to export the data as a separate video file independent from any project or application. Different encoding settings and formats are required for web, DVD or archive use. For example a video made to be visible online in live streaming must be heavily compressed, to make the file small enough to be transferable over an internet connection. But when storing media for archiving, quality is more important than the file size. Even though most video editing software has internal possibilities to compress the video sequence in a certain variety of formats and codecs, it is best to create a high quality master video for archiving, and re-compressing it to different settings using other applications specifically developed for this purpose, such as MEncoder, which will be discussed below.

FLOSS showcase

The simplest and most effective tool to capture from digital input is, in GNU/Linux systems, dvgrab, which is part of the Kino [8] project and has no GUI (Graphical User Interface) [9] .

Kino, a very simple video editing tool, can import video files or capture [10] from a camera; it can perform simple editing tasks, basic effects and useful features like for example titling. Kino can export to many formats, using FFmpeg and MEncoder engines and is really well documented by its main developer, Dan Dennedy.



File: /home/vale/
From: 00:00:00.000 To: 00:00:13.960
Limit to: Closing 00:00:01.000
Show: 25 before/after

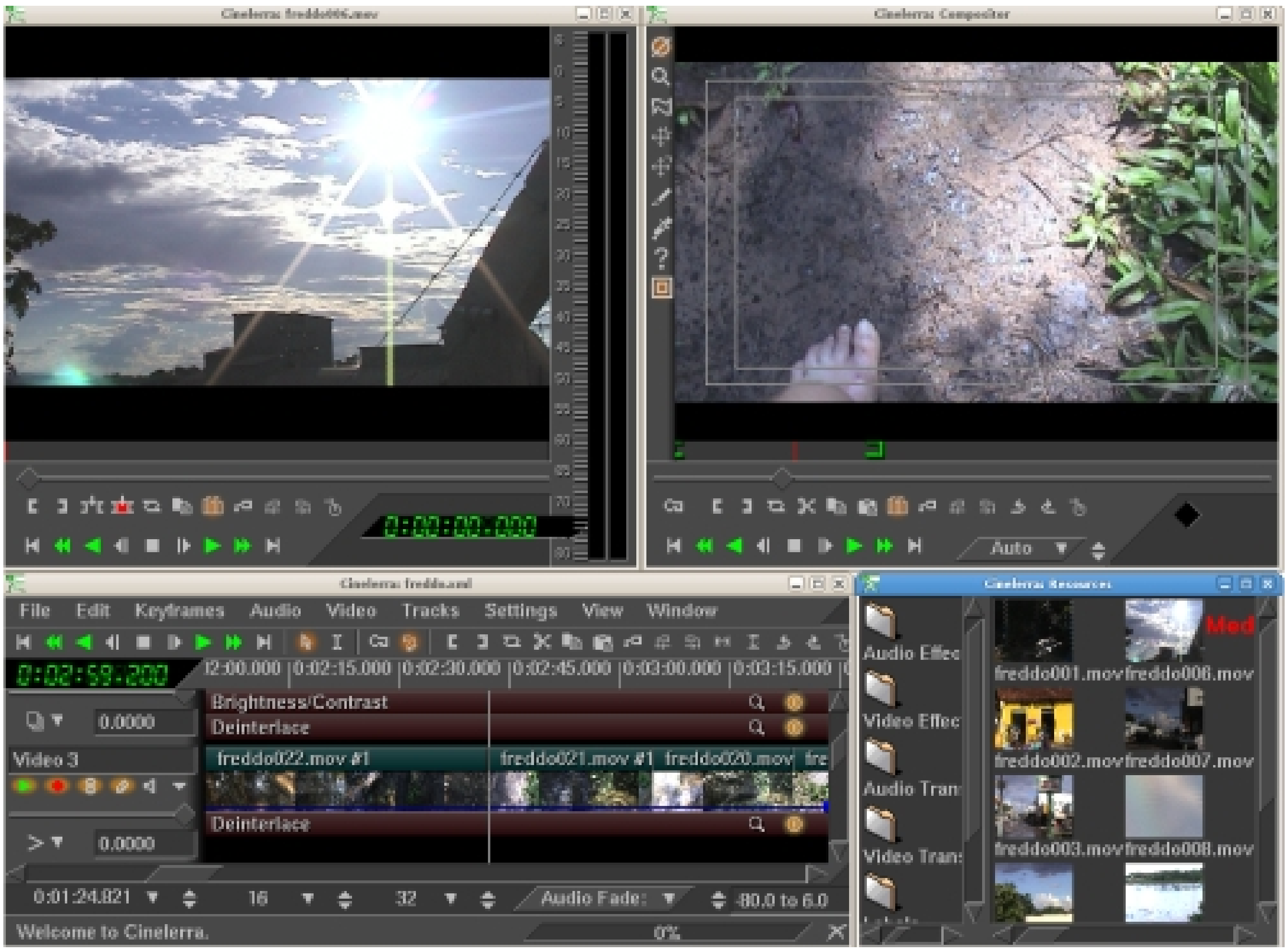
Ti porterei al mare
un altra volta

WARNING: raw1394 kernel module not loaded or failure to read/write /dev/raw1394

Screenshot of Kino

A more advanced video editing application is Cinelerra, which can be used to build bigger projects that require an extensive level of audio and video compositing. Cinelerra does not support capturing, and it is necessary to use Dvgrab or Kino to capture video clips that will be then imported in the project as data files. Cinelerra is probably the best open source video editor currently existing: totally free and independent, it is supported by a huge community of users. This project is forked in two: heroine Warrior [11] on one side, and, on the other, the community version [12] .

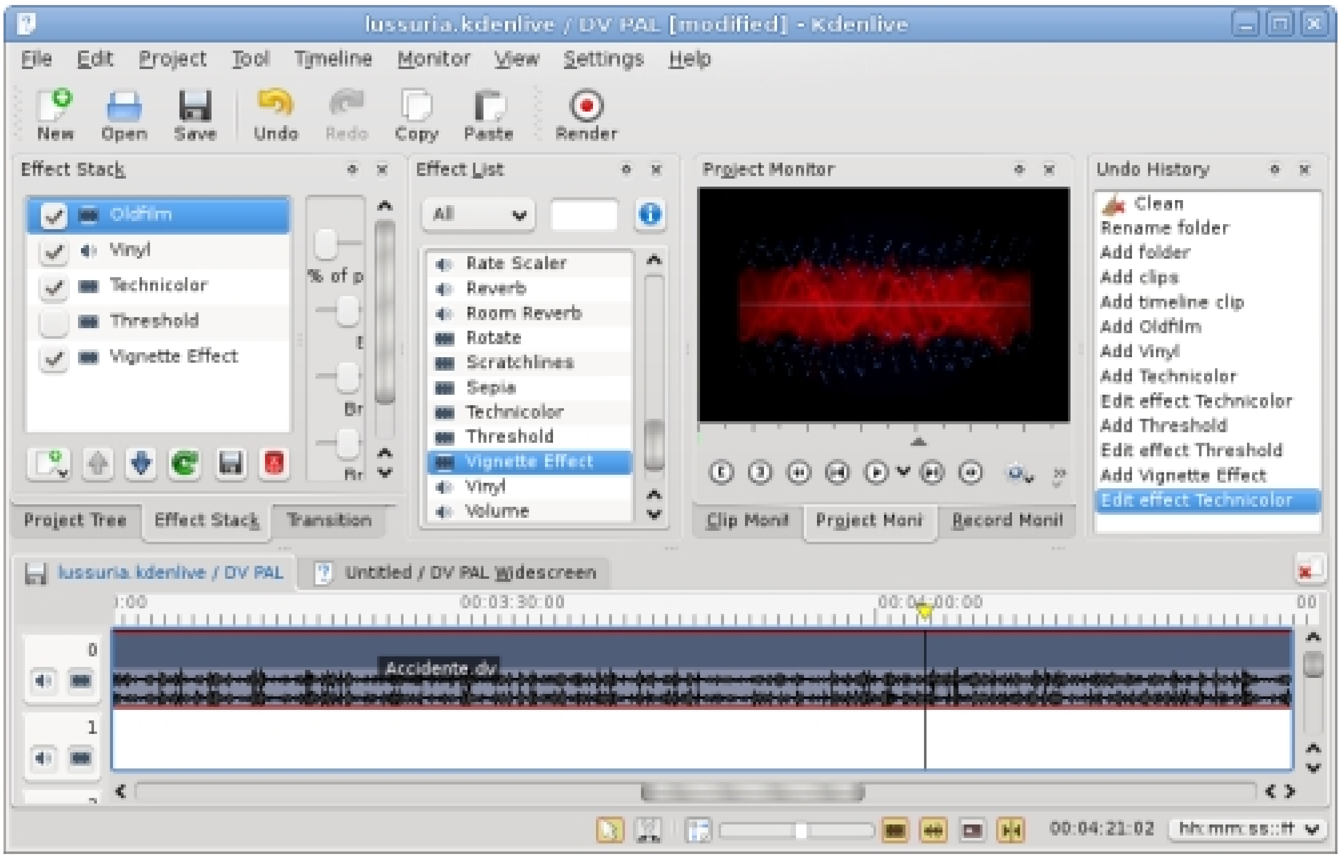
Cinelerra supports DV (Digital Video) and HDV (High Definition Video), and can import footage in many format. It is resolution and frame independent, thus it can work with video of any size and speed. A great feature is the possibility to import .VOB (Video Object File) directly from DVD. Cinelerra is timeline based and multi-tracking. The compositing window allows masking, panning, and various basic operations. This application is perfectly able to work with key-framing and a considerable amount of realtime video effects (some of them, as for example TimeFront, are really special and unusual). Export and encoding are possible thanks to a wide range of render options, from high quality resolution like mpeg2 (DVD) to low resolution web formats like Ogg [13] or Mpeg4. Batch rendering as well as XML (Extensible Markup Language) and EDL (Edit Decision List) import and export are all implemented. EDL is a textual representation of a film or video edit. It contains an ordered list of reel and time-code data (in and out points of the video clips) forming the project and the final edited sequence.



Screenshot of Cinelerra

Although nowadays Cinelerra is quite stable, it is still possible to load the recovered backup data after a crash, selecting from the menu File>LoadBackup.

Even if not as sophisticated and complex as Cinelerra, Kdenlive [14] can do a wide range of amazing operations. Its interface will be familiar to all those who used any video editor on Windows or Mac: at the top the clips and imported video are shown, then the effects list, video preview and project history; along the bottom is the multitrack timeline. Kdenlive will allow you to import material from DV camera, and any video file that FFmpeg can read. You can create a project that combines FLV (Flash Video), H.264, 3G and VOB (DVD MPEG2) video files with raw material [15] . You can import slideshows and add titles and colour filters to the final project. Kdenlive features some nice effects. To use the audio effects make sure you've got the SWH plugins package installed, then you can have a vinyl crackle, if you wish... just right-click the track in the timeline.



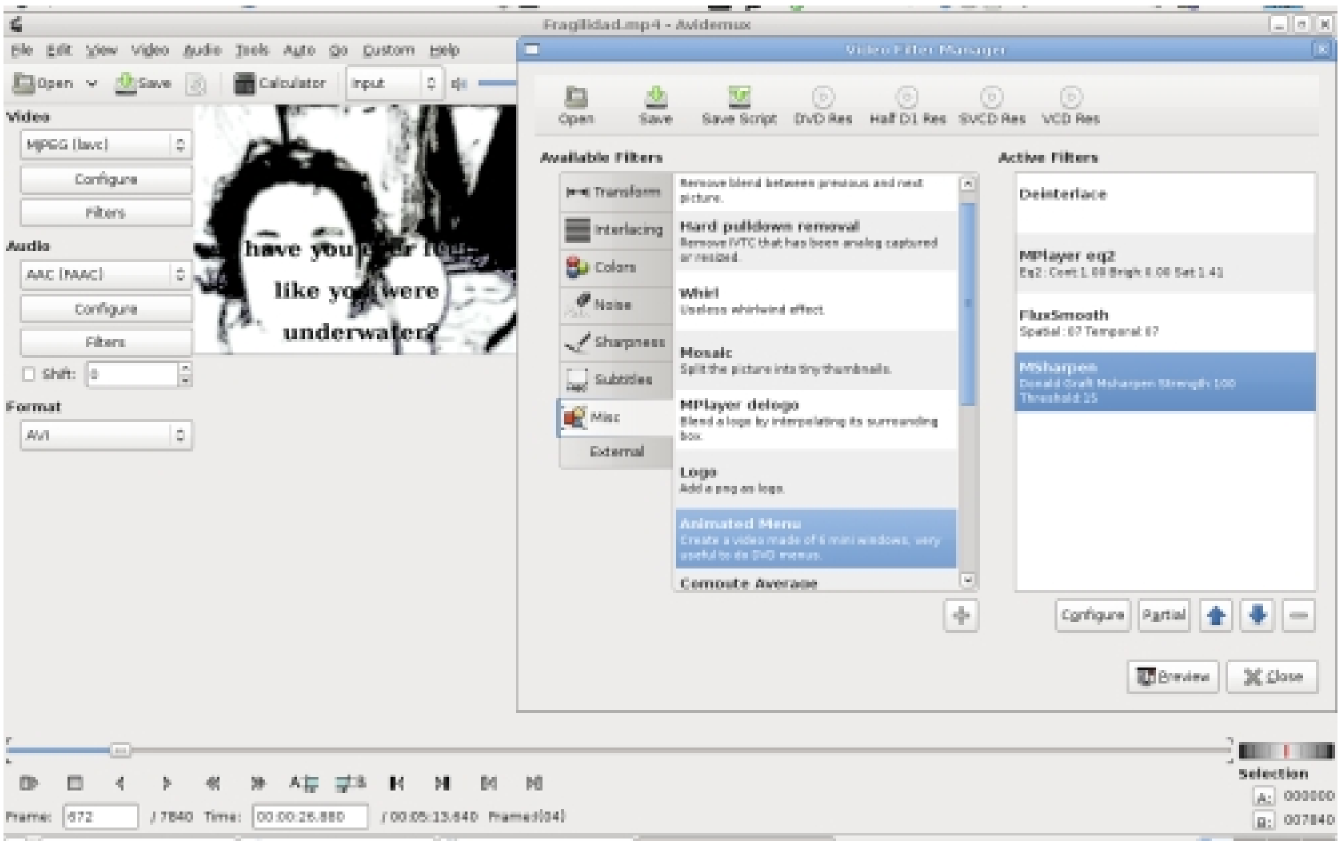
Screenshot of Kdenlive

You can export to any format supported by FFmpeg, that is the encoding engine. You will find a wide range of pre-configured formats for exporting, and you can select video size, quality and different audio formats. You can also export to tape, recording on your DV camera using the FireWire output. In addition, if you've got dvdauthor installed (see section below dedicated to this software), you can export to DVD, creating simple menus.

Kdenlive version 0.7 supports HDV, and features superimposing titles or text on video. At this stage (this article was written early 2009) it is still a bit unstable and complicated to install. If you are a Ubuntu user, there are great instructions on the Ubuntu website about this [16] .

Two other simple video editing applications are Avidemux and Open Movi Editor.

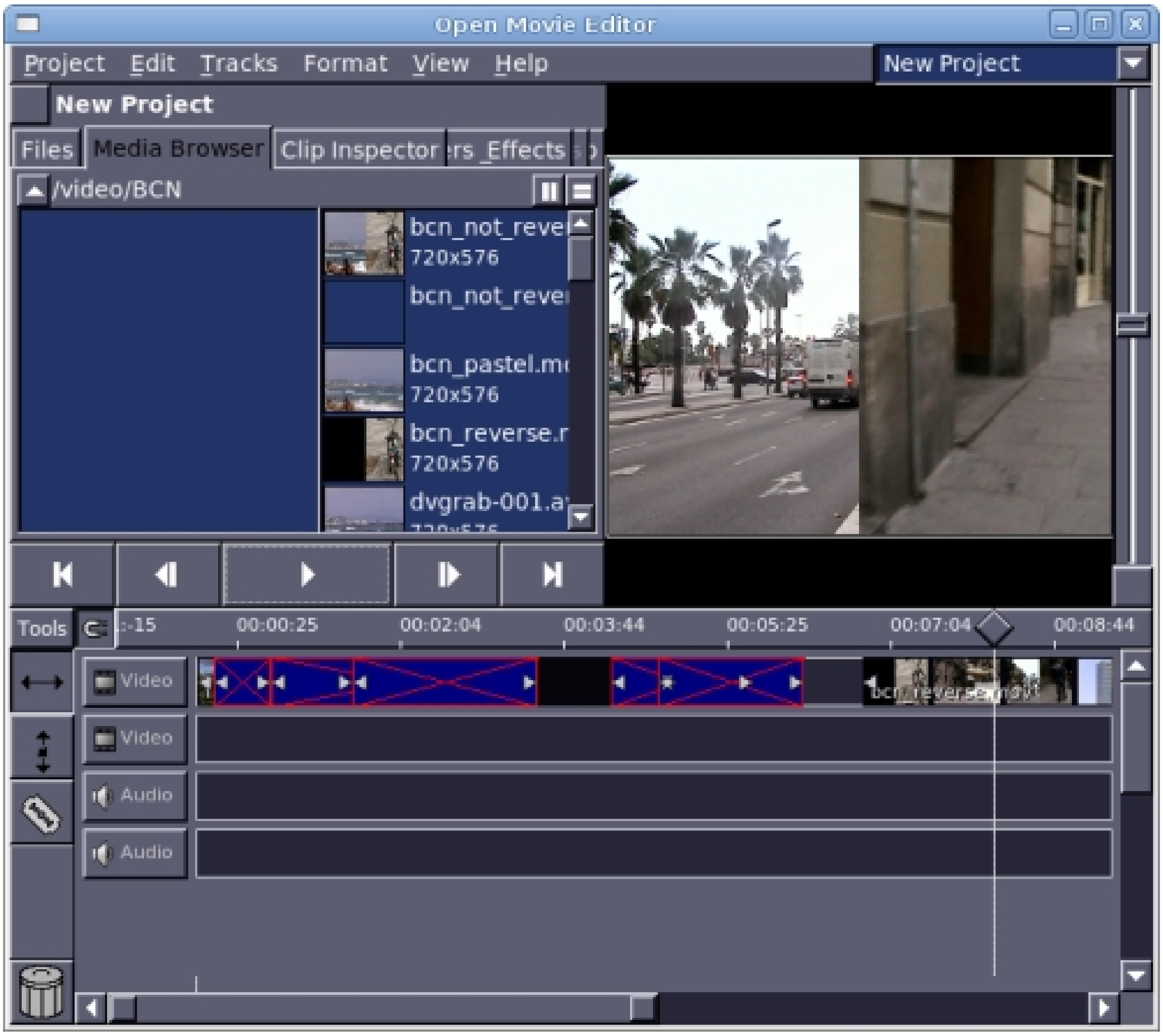
Avidemux [17] is a small non-linear editor, with encoding and filtering capabilities. Special features are the possibility to embed subtitles in a video stream or split the video output in 6 channels to create an animated DVD menu. A nice tool, called Calculator, can calculate compression options by a given size.


Fragilidad.mp4 - Avidemux
File Edit View Video Audio Tools Auto Go Custom Help
Open Save Calculator Input
Video
MJPEG (lavc)
Configure
Filters
Audio
AAC (FAAC)
Configure
Filters
Shift: 0
Format
AVI
have you ever felt like you were underwater?
Video Filter Manager
Open Save Save Script DVD Res Half D1 Res SVCD Res VCD Res
Available Filters
Active Filters
Transform
Interlacing
Colors
Noise
Sharpness
Subtitles
Misc
External
Remove blend between previous and next picture.
Hard pulldown removal
Remove IVTC that has been analog captured or resized.
Whirl
Useless whirlwind effect.
Mosaic
Split the picture into tiny thumbnails.
MPlayer delogo
Blend a logo by interpolating its surrounding box.
Logo
Add a png as logo.
Animated Menu
Create a video made of 6 mini windows, very useful to do DVD menus.
Compute Average
Deinterlace
MPlayer eq2
Eq2: Cont 1.00 Brigh 0.00 Sat 1.41
FluxSmooth
Spatial: 07 Temporal: 07
MSharpen
Donald Graft Msharpen Strength: 100 Threshold: 15
Configure Partial
Preview Close
Selection
A: 000000
B: 007840
Frame: 672 / 7840 Time: 00:00:26.880 / 00:05:13.640 Frame:I(04)

Screenshot of Avidemux

Avidemux can't capture from FireWire port, but can use Ffv1rec [18] to capture from analogue input and import .VOB from any DVD project. It is multi-platform and available in most Linux distributions, but it is best is to build Avidemux from the latest snapshot [19] .

Open Movie Editor [20] is an interesting and new project with a good developing potential. Despite its simplicity, it has different tracks for audio and video and offers the possibility to select a source directory to be displayed on the top left of the main window. This way adding titles or graphics becomes as easy as drag and drop. Filtering is really simple, and exporting is just as easy: go to the "Project" menu, name your file, select the a/v codecs you prefer, and... there you go. More encoding presets will be available in the future. As with other applications, it is worth the effort to compile it from the source code. A simple way to solve basic problems when compiling is to paste any error you get in Google: reading about other people's experience can help solve many little problems.



Screenshot of Open Movie Editor

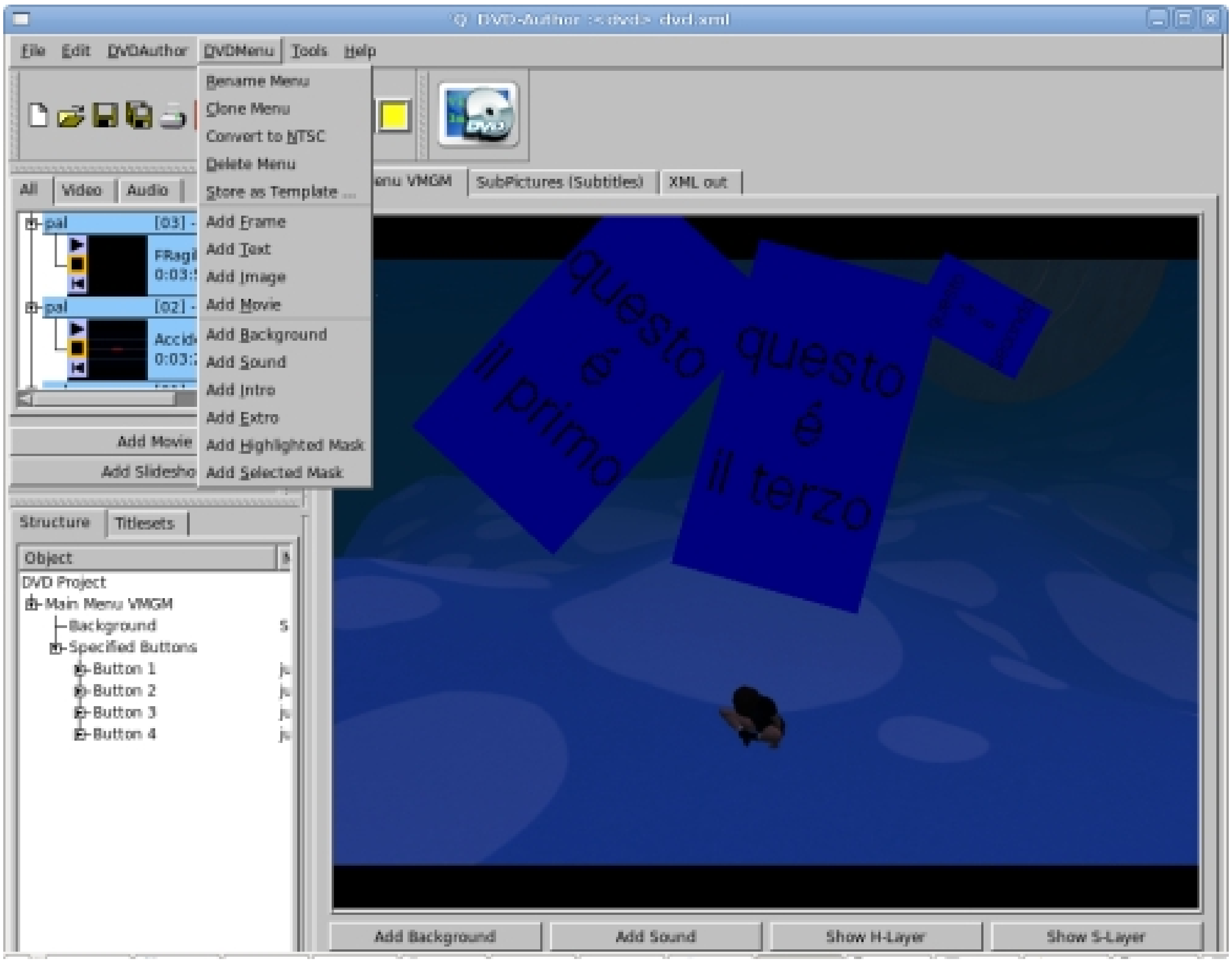
Applying one or more video filters or effect onto a video clip is easy, but before you start you need to install freiOR (effects plugins) [21] .

MEncoder [22] , a simple movie encoder designed to encode MPlayer-playable [7] movies to other MPlayer-playable formats, can be used to compress audio/video files using command line instructions. MEncoder and MPlayer are real jewels of the open source panorama, and there is no equivalent in the commercial world. The only player that can be compared to MPlayer is videoLAN [6] , which is a multi platform open source software.

FFmpeg [23] and ffmpeg2theora [24] are other command line tools composed of a collection of FLOSS libraries. They can be used to compress, for example, a lossless quality, high resolution audio/video file (suitable for DVD authoring) [25] .

DVD authoring

DVD is a physical support for storing video in good quality, although data is still quite compressed. The main advantage is the possibility to play it from any home player, without the need of a computer. Authoring a DVD means combining video and audio sequences with titles, menus and other features (software interface). The usual FLOSS tool able to achieve such work is dvdauthor [26] . 'Q' dvdauthor [27] is intended to be more of a frontend, in fact it collects several tools needed to achieve the most complete DVD authoring environment possible. It imports only .VOB format files, although an encoder engine will be released soon (statement accurate at time of writing, early 2009).



Screenshot of 'Q' dvdauthor

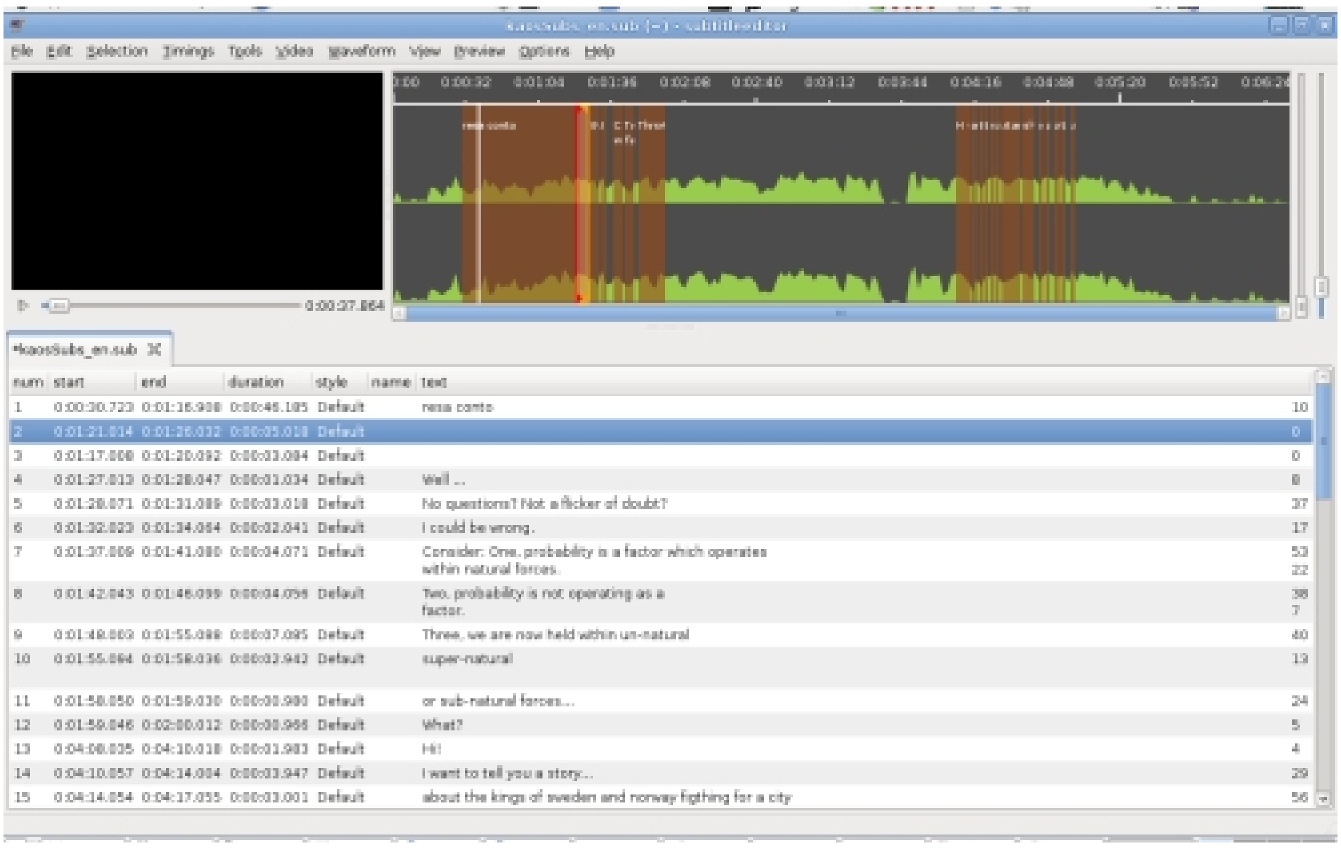
While rendering, 'Q' dvdauthor displays a funny and useful terminal window which reports about errors and other possible complications of creating a menu. This application is continuously growing into a stable and reliable application. It also permits scripting for graphic animations and other advanced menu features.

Subtitling

The open source panorama offers various simple possibilities for subtitling. The following tools work perfectly, and can perform almost all the facilities other common subtitlers can do. MPlayer has to be properly configured and off course it is always a good idea to check all the general options (in Preferences) before starting. For example, have a look at 'video output': if xv doesn't work, try X11 [28] .

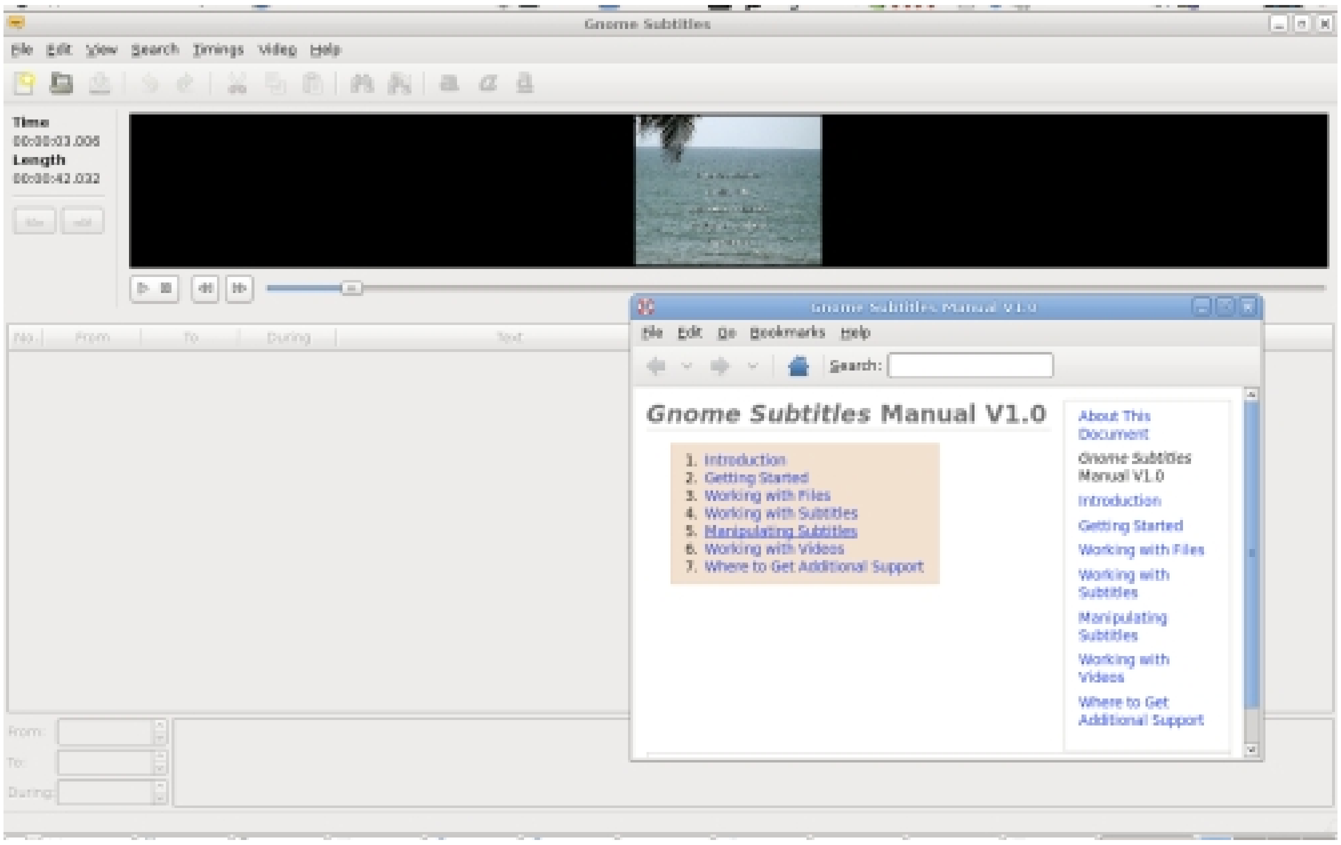
Subtitle Editor [29] is really easy to use and presents the following features: in the main window an integrated video player can play video preview (using MPlayer). It has timing and can generate and display waveforms. The subtitling is previewed over the video. It is possible to edit the script header (authors, translators, timers, video information, etc.), check the spelling, convert frame-rate and scale. It supports the following formats: SSA, ASSA, SubRip, MicroDVD, MPsub, Adobe DVD and plain text. The internal format is Advanced SubStation Alpha [30] .

KSubtile [31] and Gnome Subtitles [32] are the subtitlers for KDE and GNOME (the two main desktop environment systems for GNU/Linux) so choose them according to what window manager you use. They work in a very similar way and are very easy to install in any Linux distribution.


kaossubs_en.sub (~) - subtitleeditor
File Edit Selection Timings Tools Video Waveform View Preview Options Help
0:00:37.864
*kaossubs_en.sub

num	start	end	duration	style	name	text	
1	0:00:30.723	0:01:16.908	0:00:46.185	Default		resa conto	10
2	0:01:21.014	0:01:26.032	0:00:05.018	Default			0
3	0:01:17.008	0:01:20.092	0:00:03.084	Default			0
4	0:01:27.013	0:01:28.047	0:00:01.034	Default		Well ...	8
5	0:01:28.071	0:01:31.089	0:00:03.018	Default		No questions? Not a flicker of doubt?	37
6	0:01:32.023	0:01:34.064	0:00:02.041	Default		I could be wrong.	17
7	0:01:37.009	0:01:41.080	0:00:04.071	Default		Consider: One, probability is a factor which operates within natural forces.	53 22
8	0:01:42.043	0:01:46.099	0:00:04.056	Default		Two, probability is not operating as a factor.	38 7
9	0:01:48.003	0:01:55.098	0:00:07.095	Default		Three, we are now held within un-natural	40
10	0:01:55.094	0:01:58.036	0:00:02.942	Default		super-natural	13
11	0:01:58.050	0:01:59.030	0:00:00.980	Default		or sub-natural forces...	24
12	0:01:59.046	0:02:00.012	0:00:00.966	Default		What?	5
13	0:04:08.035	0:04:10.018	0:00:01.983	Default		Hi!	4
14	0:04:10.057	0:04:14.004	0:00:03.947	Default		I want to tell you a story...	29
15	0:04:14.054	0:04:17.055	0:00:03.001	Default		about the kings of sweden and norway figthing for a city	56

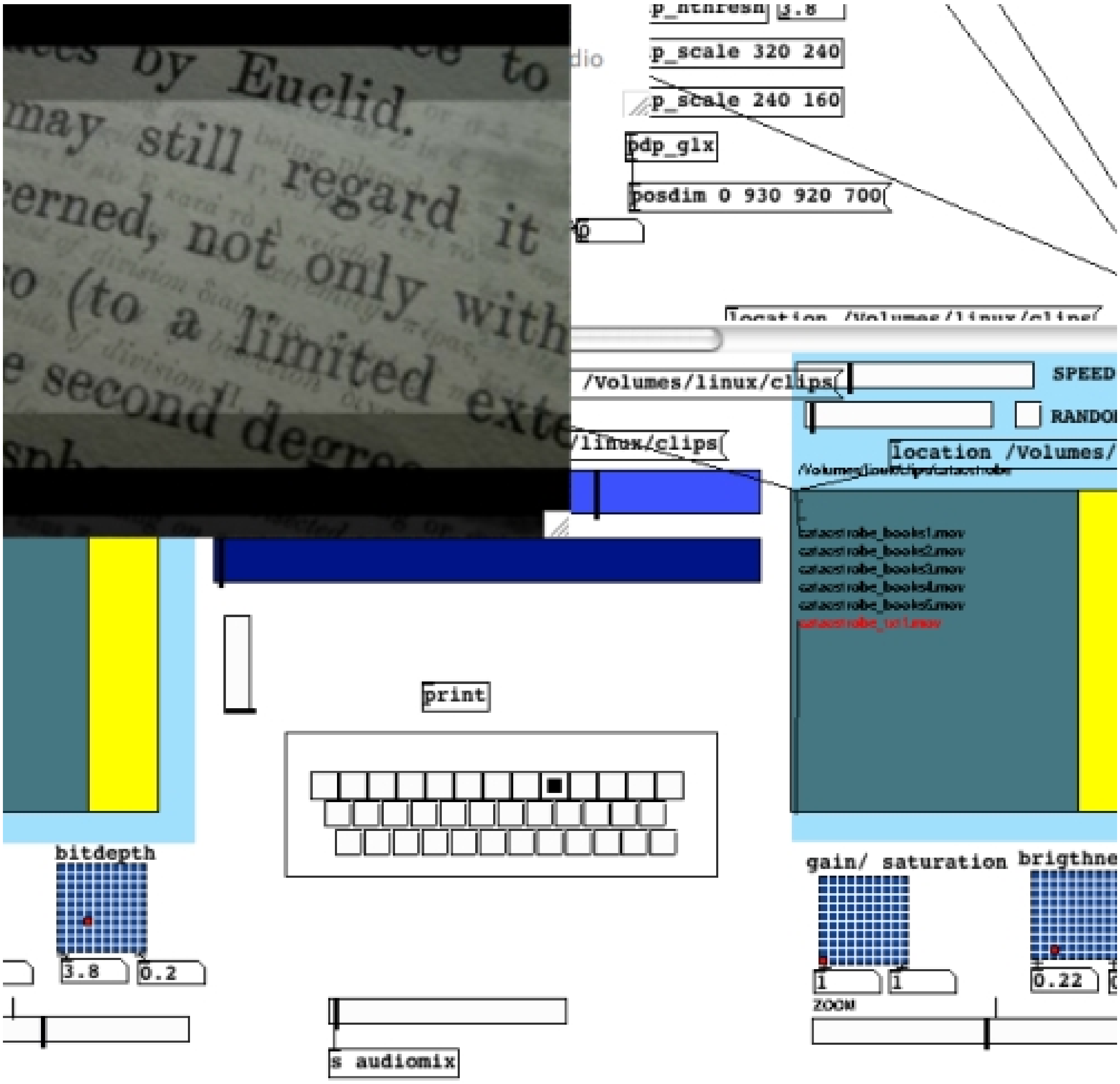
Screenshot of Ksubtile



Screenshot of Gnome Subtitles

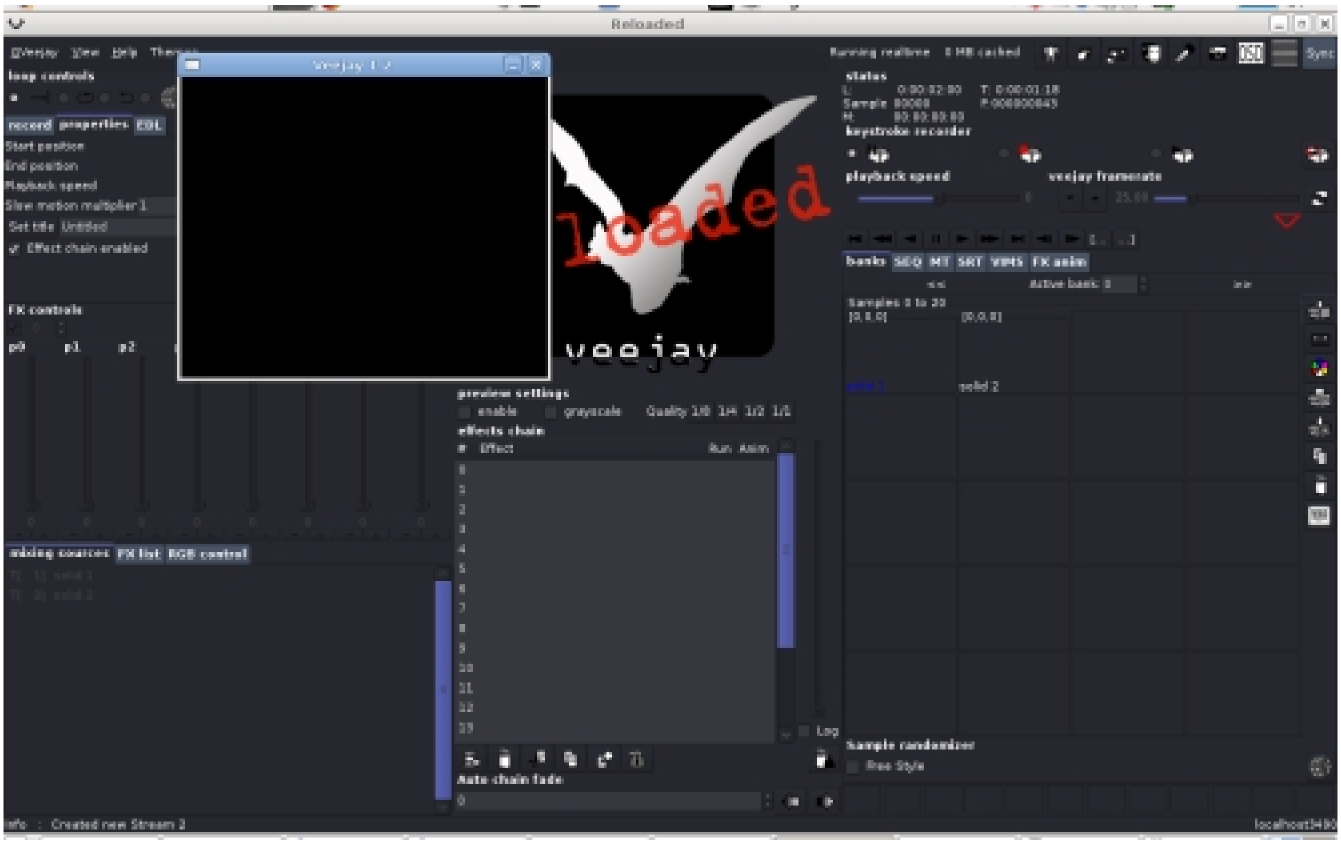
Perspective on Real-time and Procedural video

Pure Data [33] , and other systems, provide the best environment for live video making and interactivity. Pure Data [34] is an infinite source of applications for real-time audio,video and interactivity. Pd-extended [35] offers an easy to install system integrating most of the up-to-date libraries and extensions.



Screenshot of Pure Data

Veejay [36] is a visual instrument and real-time video sampler allowing you to play video 'like you would play a piano'. It features on-the-fly video sampling, a wide range of possibilities for connecting to other applications and devices and a veejaypuredata object [37] . It has support for (multiple) live camera, and streaming capabilities. A very promising application.



Screenshot of Veejay

Installing is possible only by building from source, but there is a package for the dynebolic project[38], and Ubuntu Hardy [39] .

Conclusion

A video editing software is a very big and complex application, and its development is not fully complete within the Floss panorama. Nowadays it is still a challenge to make video using only FLOSS, but development is progressing and each user is supported by a great community of other users and by the developers. Because Free Software has many forms, from big and complete packages such as Cinelerra to smaller applications like dvgrab, which can often be combined and used in all kinds of different configurations, FLOSS can propose a workflow which is not organised from above, but reticular, "bottom-up", and highly customisable to the individual needs of the artist.

A user/creator approaching the FLOSS video editing world for the first time is better off using one of the video friendly distributions, such as pure:dyne and Ubuntu Studio, bypassing manual installation processes, quickly getting the hands on the digital project. Once inside this brave new world, great fun will start: working (and wandering around) inside a system whose structures are open implies a number of important advantages, like, for example, the possibility to get actively involved in the projects one uses, and being able to suggest future directions of development. In fact the margin between users and creator is, in FLOSS projects, soft and blurred, and nobody is a customer.

While complete and complex video editing applications, like for example Cinelerra, become more and more usable and reliable, a great number of small video applications, often very functional and well performing, as well as innovative and experimental, are available within the FLOSS panorama. These smaller applications for creative and experimental video making allow the user/artist to create a different workflow, segment the working process in smaller steps and glue together several different applications (piping [40]), rather than structuring an entire project inside one single software. In this way both the audiovisual material and the working method can be radically manipulated.

Notes

[1] This is the operative memory of the computer, used on the fly by the system. When working with video you need a substantial amount of RAM. The exact amount depends on the hardware within your computer.

[2] Consider, by example, that 30 minutes of raw DV (the usual capturing setting) are approximately 7 Gigabytes. For exporting, on the other side, 1.49 minutes of Raw uncompressed material is 2.13 Gigabytes.

[3] For example, when using compiz or beryl effects and transitions, the video plays back a bit flaky. It is possible to fix this problem by changing window manager, for instance by choosing a non-composite one.

[4] http://ubuntustudio.org/

[5] http://puredyne.goto10.org/

[6] http://videolan.org

[7] MPlayer is a free and open source media player. The program is available for all major operating systems, including Linux and other Unix-like systems. MPlayer supports a wide variety of media formats. A companion program, MEncoder, can take an input stream or file and transcode it into several different output formats.

[8] http://www.kinodv.org/

[9] For the poetical implication of the command line method see $(echo echo) echo $(echo): Command Line Poetics in this handbook

[10] Capturing from video camera or audio device means digitising analogue material from FireWire, analogue video or audio port. When the footage required is already a digital file, it can be easily imported inside a project or application.

[11] http://www.heroinewarrior.com/

[12] http://cinelerra.org/

[13] Normally the format Ogg contains audio compressed with Vorbis codec, and video compressed with Theora codec. For Theora Video compression see: http://theora.org/Theora is a free and open video compression format from the Xiph.org Foundation. Like all our multimedia technology it can be used to distribute film and video online and on disc without the licensing and royalty fees or vendor lock-in associated with other formats. To know more about Vorbis please visit: http://www.vorbis.com/

[14] http://www.kdenlive.org/

[15] For a detailed and critical overview on codecs see this research (now outdated) produced by the NIMK in Amsterdam: http://xname.cc/text/video-streaming-on-wan.pdf

Use Wikipedia to gather information on FLV and other codecs not included here.

[16] https://help.ubuntu.com/community/KdenliveSVN

[17] http://www.avidemux.org/admWiki/index.php?title=Main_Page

[18] http://www.avidemux.org/admWiki/index.php?title=Ffv1rec

[19] http://www.avidemux.org/admWiki/index.php?title=Compiling_Avidemux#Choos...

[20] http://www.openmovieeditor.org/

[21] http://www.piksel.org/frei0r

[22] http://www.mplayerhq.hu/design7/news.html

[23] http://www.ffmpeg.org/

[24] http://v2v.cc/~j/ffmpeg2theora

[25] A variety of encoding settings can be found in the research mentioned above:
http://xname.cc/text/video-streaming-on-wan.pdf

Some more examples here:

```
ffmpeg -i Test.dv -target dvd -aspect 16:9 -sameq my_dvd_video.mpg
```

Here is an example of how to encode an audiovisual file in Ogg Theora (open source codec for good quality on the web). This gives best results encoding video from PhotoJPEG :

```
ffmpeg2theora test.mov --audiobitrate 128 --samplerate 44 -a 8 --videobitrate 600 --videoquality 8 -o test.ogg
```

[26] http://dvdauthor.sourceforge.net/

[27] http://qdvdauthor.sourceforge.net/

[28] The X Window System (commonly X or X11) is a computer software system and network protocol that provides a graphical user interface (GUI) for networked computers. It implements the X display protocol and provides windowing on raster graphics (bitmap) computer displays and manages keyboard and pointing device control functions.

[29] http://subtitleeditor.sourceforge.net/

[30] http://en.wikipedia.org/wiki/SubStation_Alpha

[31] http://ksubtile.sourceforge.net/

[32] http://gnome-subtitles.sourceforge.net/

[33] See article in this handbook - Pure Dataflow - Diving into Pd

[34] http://puredata.info/

[35] http://puredata.info/downloads

[36] http://www.veejayhq.net

[37] http://www.veejayhq.net/2007/05/sendvims-for-pure-data-a-veejay-client/a...

[38] http://sourceforge.net/project/showfiles.php?group_id=47564&amp;package_id=2...

[39] http://www.veejayhq.net/download-2/

[40] In Unix-like computer operating systems, a pipeline is an original software pipeline: a set of processes chained by their standard streams, so that the output of each process (stdout) feeds directly as input (stdin) of the next one.

Images

All images courtesy of the authors

" >

span-->

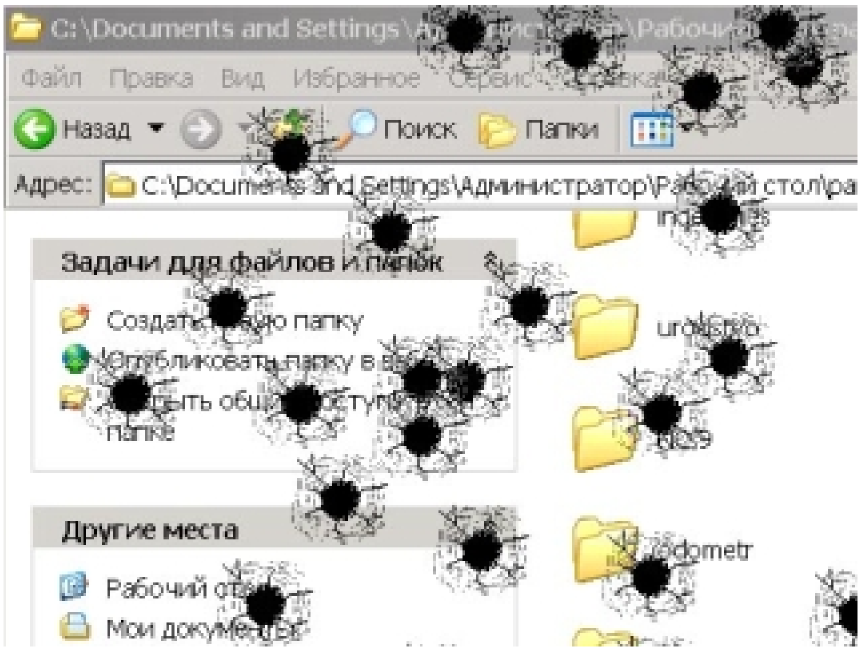
‹ Working with digital video up Software art ›

Software art

By admin
Published: 10/04/2007 - 08:31

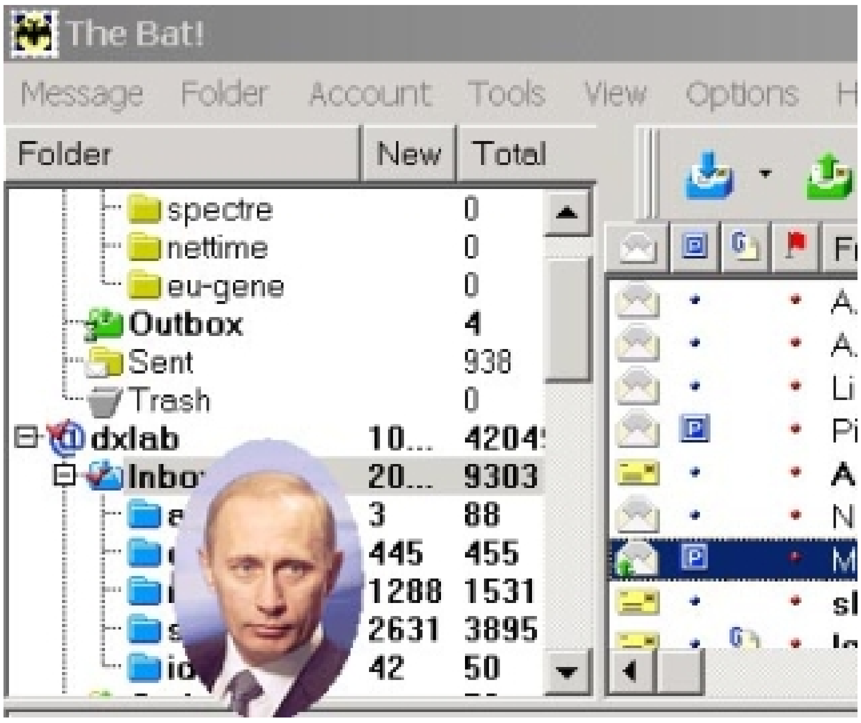
Olga Goriunova , September 2007

The term ‘software art’ acquired a status of an umbrella term for a set of practices approaching software as a cultural construct. Questioning software culturally means not taking for granted, but focusing on, recognising and problematising its distinct aesthetics, poetics and politics captured and performed in its production, dissemination, usage and presence, contexts which software defines and is defined by, histories and cultures built around it, roles it plays and its economies, and various other dimensions. Software, deprived of its alleged ‘transparency’, turns out to be a powerful mechanism, a multifaceted mediator structuring human experience, perception, communication, work and leisure, a layer occupying central positions in the production of digital cultures, politics and economies.



gun.exe by Anonymous

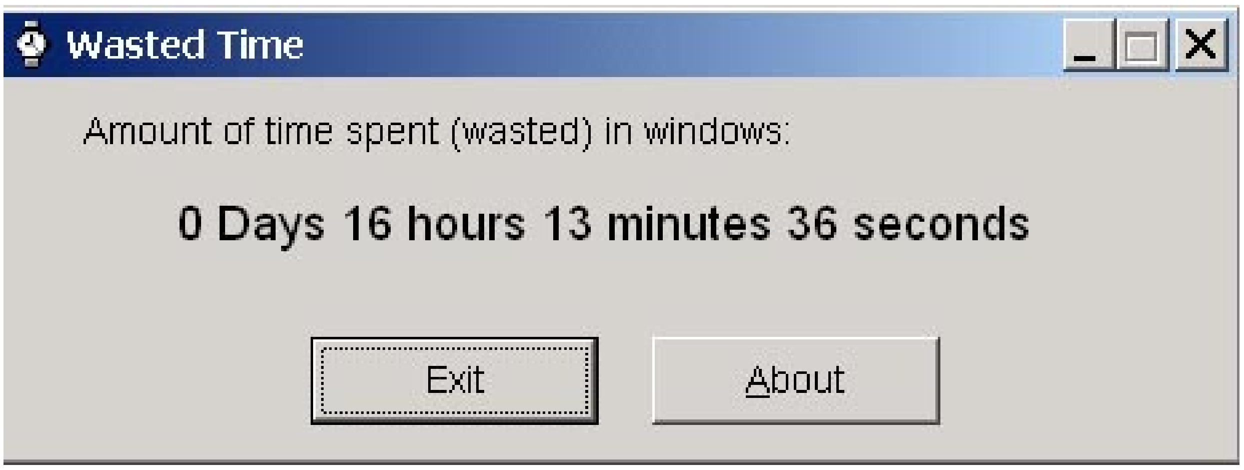
The history of framing ‘software art’ allowed for inclusion of a wide variety of approaches, whether focused on code and its poetics, or on the execution of code and art generated ‘automatically’, or on the visual-political aesthetics of interfaces and ‘languages’ of software, or on the traditions of ‘folklore’ programmers’ and users’ cultures creation, use and misuse of software, or on software as a ‘tool’ of social and political critique and as an artistic medium, even perceived within the most traditionalist frameworks, and many more. ‘Genres’ of software art may include text editors that challenge the culture of writing built into word processing software and playfully engage with the new roles of words, all indexed in the depths of search engine databases; browsers, critically engaging with the genre of browser software as a culturally biased construct that wants to both trigger and control users’ perceptions, ideas and desires; games, rebelling against chauvinist, capitalist, military logic encoded into commercial games and questioning the seeming “normality” of game worlds; low-tech projects based on creative usage of limited, obsolete algorithms; bots as pieces of software that, like normal bots, crawl the web in gathering information or perform other tasks, but, unlike them, reveal hidden links, perform imaginative roles and work against normative functions; artistic tools working towards providing new software models for artistic expression and rich experience, sometimes made in spite of mainstream interface models; clever hacks, deconstructions or misuse of software; projects providing environments for social interactions, including non-normative, or looking at the development of software as a social praxis, and many others, complex enough or too elegantly simple to be locked into the framework of a single ‘genre’ [1] .



Portrait of the President (putin.exe) by Vladislav Tselisshev

Software art as a term appeared around the late 1990's from net art discourse [2] . Interest in the code of Internet web-pages and the functioning of networks, characteristic of net art, developed towards problematising software running Internet servers and personal computers, towards different kinds of software behind any image, process, instrument or channel [3] .

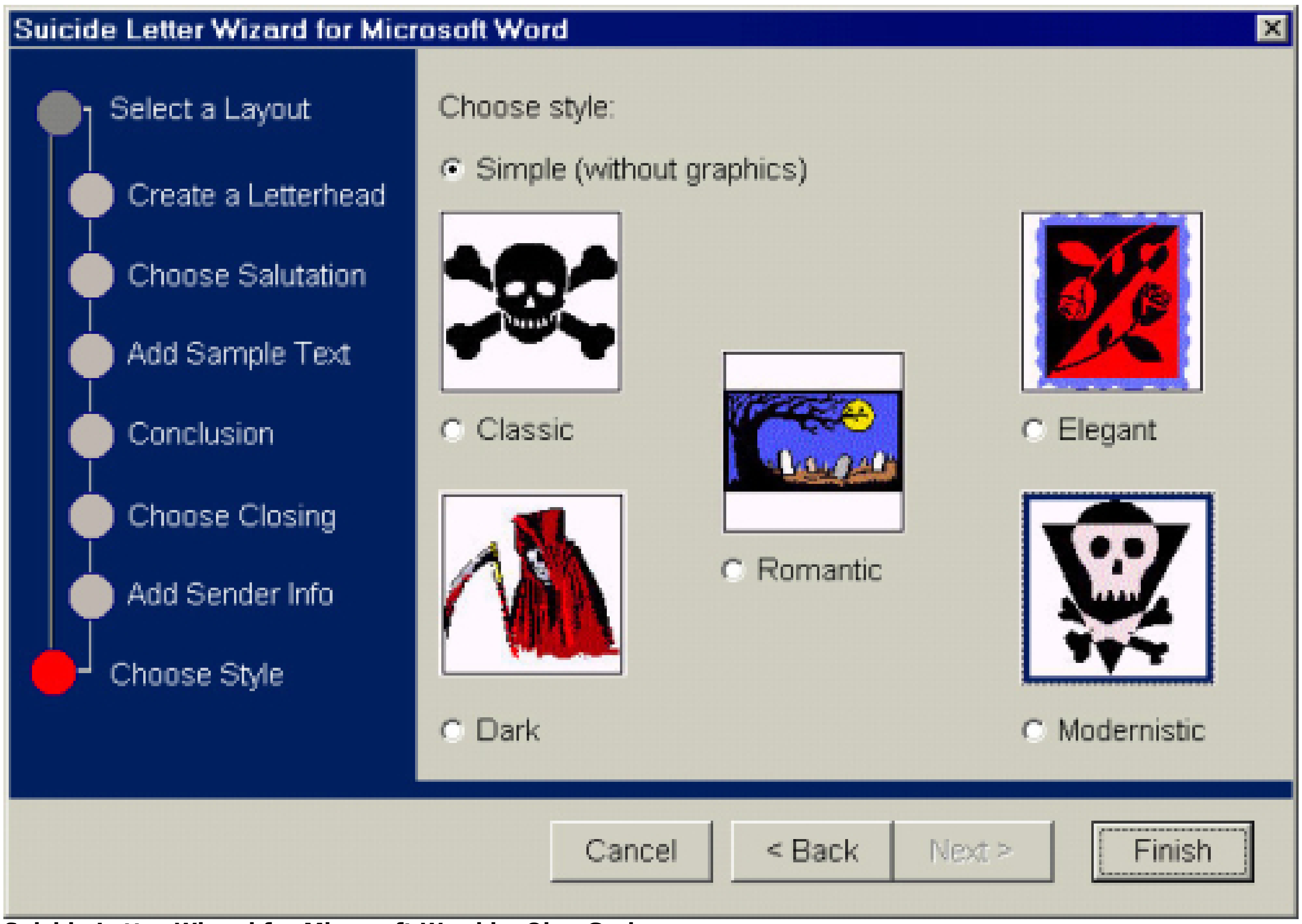
Since software can be defined as a set of formal instructions that can be executed by a computer, or as code and its execution at the same time [4] , a history of precursors of software art, including permutational poetry, Dada and Fluxus experiments with formal variations and, especially, conceptual art based on execution of instructions, can be retrospectively built[5] .


Amount of time spent (wasted) in windows:

0 Days 16 hours 13 minutes 36 seconds

Wasted time by Anonymous

Software art's relation to net art and art history does not exhaust its 'roots' and scope of approaches in a historical context. One of the forces of software art lies in its capitalization on the grass-roots traditions of programmers' and users' cultures of software creation and usage, some 'digital folklore', and in its references to more specific traditions, such as demo (a multimedia presentation computed in real time and used by programmers to compete on the level of the best programming graphic and music skills, originating from the hackers' scene of the early 1980's, when short demos were added to the opening visuals of cracked video games) and some other powerful and old traditions, such as generative art, rooted in both artistic and programmers' cultures.



Choose style: [✓ Simple (without graphics)]

Suicide Letter Wizard for Microsoft Word by Olga Goriunova

Since the 1950's (when the first high-level programming languages were created and when the first mainframe computers were bought by a few large universities) smart and funny uses of technology, entertaining games, code poetry contests, fake viruses and other forms have been building up cultures of 'digital folklore': '... there is the field of gimmicks, nerdy tricks and playing with the given formulas, the eyes which follow the mouse pointer. ... there is the useless software production, insider gags, Easter eggs, hidden features based on mathematical jokes, the Escher - trompe-l'oeil effects of playing with perception...' [6] Digital folklore exists in relation to a group of programmers, designers or informed users worldwide; it embeds, and is embedded within, codes of behaviour, of relations to employers and customers, to users, colleagues and oneself; it is correlated with the most important institutes, repeated actions, situations, models and is often based on non-formal means of production, dissemination and consumption. The variety of software cultures rooted in 'geek' practices, traditions of 'bedroom' cultures, cultures of software production and patterns of usage and adjustment build up sources of inheritance and inspiration for software art along with activist and artist engagements with software, each with histories of their own.

Software art benefited from currents of interest of more academic nature in the cultural and socio-political contexts of technology and, precisely, software [7] . The current form and 'state' of software art is due to a particular history of institutional development [8] . There also could be named a number of projects, practices and discourses conceptually and practically close to what is termed 'software art', which chose or happened to distance themselves from the usage of this term and from its institutional incarnations, for their potentially ‘limiting’ character in linking it to specific art fields, histories, and agents[9] . Moreover, departing from the term ‘software art’ might allow for starting up new stories and making visible some undervalued and overlooked practices and approaches.

Notes

[1] The above-mentioned ‘genres’ are taken from Runme.org software art repository’s ‘taxonomy’ that includes more categories than listed and presents one of the attempts to be open to a wide variety of approaches by building contradictory and, to a degree, self-evolving classificatory ‘map’ of software art.

[2] One of the first written mentionings of the term ‘software art’ can be found in Introduction to net.art (1994-1999) by Alexei Shulgin and Natalie Bookchin; http://easylife.org/netart/

[3] Deconstructions of HTML by Jodi, for instance, turned into the modification of old computer video game Wolfenstein 3D - SOD (1999) and other games. Projects such as alternative web browsers that were considered part of the net art movement started to be written back in 1997 with Web Stalker by I/O/D, continued by Web Shredder by Mark Napier in 1998, and Netomat by Maciej Wisniewski in 1999, and often got labeled as ‘online software art’.

[4] Cramer, Florian. Concept, Notation, Software, Art. 2002; http://userpage.fu-berlin.de/~cantsin/homepage/writings/software_art/con...

[5] Ibid. See also: Cramer, Florian and Gabriel, Ulrike. Software Art. 2001; http://www.netzliteratur.net/cramer/software_art_-_transmediale.html and Albert, Saul. Artware.1999. Mute, Issue 14, London. http://twenteenthcentury.com/saul/artware.htm

[6] Schultz, Pit. Interviewed by Alexei Shulgin. Computer Age is Coming into Age. 2003. http://www.m-cult.org/read_me/text/pit_interview.htm

[7] It is difficult and lengthy to enumerate all the interesting publications on the issue. An excellent example is Software Studies: a Lexicon. Ed. by Matthew Fuller, 2008, Cambridge, The MIT Press.

[8] Transmediale, Berlin-based media art festival introduced an ‘artistic software’ category in 2001 (ran until 2004), Readme software art festivals (2002-2005) and Runme.org software art repository (running from 2003) added on to the construction of the current ‘layout’ of ‘software art’ if it is taken as a scene defined institutionally. The list of events is continued by exhibitions, such as I love you

[rev.eng] (Museum of Applied Arts Frankfurt, 2002; (http://www.digitalcraft.org/iloveyou/), Generator (Liverpool Biennale, 2002; http://www.generative.net/generator/), CODeDOC (Whitney Museum, NY, 2002; http://artport.whitney.org/commissions/codedoc/index.shtml), and a number of symposiums, conferences and discussions.

[9]   Matthew Fuller, for instance, suggested ‘critical software’, ‘social software’ and ‘speculative software’ as generic titles for practices software art attempted to draw in; see: Fuller, Matthew. Behind the Blip, Software as Culture. (some routes into “software criticism”, more ways out). 2002. In: Fuller, Matthew. Behind the Blip. Essays on the Culture of Software. 2003, New-York, Autonomedia.

Images

[1] gun.exe by anonymous

[2] Portrait of the President (putin.exe) by Vladislav Tselisshev, image courtesy of the author

[3] Wasted time by anonymous

[4] Suicide Letter Wizard for Microsoft Word by Olga Goriunova, image courtesy of the author

" >

span-->

- $(echo echo) echo $(echo): Command Line Poetics
- Game art: theory, communities, resources

‹ Video editing with open source tools up $(echo echo) echo $(echo): Command Line Poetics ›

$(echo echo) echo $(echo): Command Line Poetics

By marloes
Published: 09/18/2007 - 13:39

Florian Cramer, 2003 / 2007

Design

Most arguments in favor of command line versus graphical user interface (GUI) computing are flawed by system administrator Platonism. A command like "cp test.txt /mnt/disk" is, however, not a single bit closer to a hypothetic "truth" of the machine than dragging an icon of the file.txt with a mouse pointer to the drive symbol of a mounted disk. Even if it were closer to the "truth", what would be gained from it?

The command line is, by itself, just as much an user interface abstracted from the operating system kernel as the GUI. While the "desktop" look and feel of the GUI emulates real life objects of an analog office environment, the Unix, BSD, Linux/GNU and Mac OS X command line emulates teletype machines that served as the user terminals to the first Unix computers in the early 1970s. This legacy lives on in the terminology of the "virtual terminal" and the device file /dev/tty (for "teletype") on Unix-compatible operating systems. Both graphical and command line computing are therefore media; mediating layers in the cybernetic feedback loop between humans and machines, and proofs of McLuhan's truism that the contents of a new medium is always an old medium.

Both user interfaces were designed with different objectives: In the case of the TTY command line, minimization of typing effort and paper waste, in the case of the GUI, use of - ideally - self-explanatory analogies. Minimization of typing and paper waste meant to avoid redundancy, keeping command syntax and feedback as terse and efficient as possible. This is why "cp" is not spelled "copy", "/usr/bin/" not "/Unix Special Resources/Binaries", why the successful completion of the copy command is answered with just a blank line, and why the command can be repeated just by pressing the arrow up and return keys, or retyping "/mnt/disk" can be avoided by just typing "!$".

The GUI conversely reinvents the paradigm of universal pictorial sign languages, first envisioned in Renaissance educational utopias from Tommaso Campanella's City of the Sun to Jan Amos

Comenius illustrated school book "Orbis Pictus". Their design goals were similar: "usability", self-explanatory operation across different human languages and cultures, if necessary at the expense of complexity or efficiency. In the file copy operation, the action of dragging is, strictly seen, redundant. Signifying nothing more than the transfer from a to b, it accomplishes exactly the same as the space in between the words - or, in technical terms: arguments - "test.txt" and "/mnt/disk", but requiring a much more complicated tactile operation than pushing the space key. This complication is intended as the operation simulates the familiar operation of dragging a real life object to another place. But still, the analogy is not fully intuitive: in real life, dragging an object doesn't copy it. And with the evolution of GUIs from Xerox Parc via the first Macintosh to more contemporary paradigms of task bars, desktop switchers, browser integration, one can no longer put computer-illiterate people in front of a GUI and tell them to think of it as a real-life desk. Never mind the accuracy of such analogies, GUI usage is as much a constructed and trained cultural technique as is typing commands.

Consequentely, platonic truth categories cannot be avoided altogether. While the command line interface is a simulation, too - namely that of a telegraphic conversation - its alphanumeric expressions translate more smoothly into the computer's numeric operation, and vice versa. Written language can be more easily used to use computers for what they were constructed for, to automate formal tasks: the operation "cp *.txt /mnt/disk" which copies not only one, but all text files from the source directory to a mounted disk can only be replicated in a GUI by manually finding, selecting and copying all text files, or by using a search or scripting function as a bolted-on tool. The extension of the commmand to"for file in *; do cp $file $file.bak; done" cannot be replicated in a GUI unless this function has been hard-coded into it before. On the command line, "usage" seamlessly extends into "programming".

In a larger perspective, this means that GUI applications typically are direct simulations of an analog tool: word processing emulates typewriters, Photoshop a dark room, DTP software a lay-out table, video editors a video studio etc. The software remains hard-wired to a traditional work flow. The equivalent command line tools - for example: sed, grep, awk, sort, wc for word processing, ImageMagick for image manipulation, groff, TeX or XML for typesetting, ffmpeg or MLT for video processing - rewire the traditional work process much like "cp *.txt" rewires the concept of copying a document. The designer Michael Murtaugh for example employs command line tools to automatically extract images from a collection of video files in order to generate galleries or composites, a concept that simply exceeds the paradigm of a graphical video editor with its predefined concept of what video editing is.

The implications of this reach much farther than it might first seem. The command line user interface provides functions, not applications; methods, not solutions, or: nothing but a bunch of plug-ins to be promiscuously plugged into each other. The application can be built, and the solution invented, by users themselves. It is not a shrink-wrapped, or - borrowing from Roland Barthes - a "readerly", but a "writerly" interface. According to Barthes' distinction of realist versus experimental literature, the readerly text presents itself as linear and smoothly composed, "like a cupboard where meanings are shelved, stacked, safeguarded".[1] Reflecting in contrast the "plurality of entrances, the opening of networks, the infinity of languages", [2] the writerly text aims to make "make the reader no longer a consumer, but a producer of the text". [3] In addition to Umberto Eco's characterization of the command line as iconoclastically "protestant" and the GUI as idolatrously "catholic", the GUI might be called the Tolstoj or Toni Morrison, the command line the Gertrude Stein, Finnegans Wake or L.A.N.G.U.A.G.E poetry of computer user interfaces; alternatively, a Lego paradigm of a self-defined versus the Playmobil paradigm of the ready-made toy.

Ironically enough, the Lego paradigm had been Alan Kay's original design objective for the graphical user interface at Xerox PARC in the 1970s. Based on the programming language Smalltalk, and leveraging object oriented-programming, the GUI should allow users to plug together their own applications from existing modules. In its popular forms on Mac OS, Windows and KDE/Gnome/XFCE, GUIs never delivered on this promise, but reinforced the division of users and developers. Even the fringe exceptions of Kay's own system - living on as the "Squeak" project - and Miller Puckette's graphical multimedia programming environments "MAX" and "Pure Data" show the limitation of GUIs to also work as graphical programming interfaces, since they both continue to require textual programmation on the core syntax level. In programmer's terms, the GUI enforces a separation of UI (user interface) and API (application programming interface), whereas on the command line, the UI *is* the API. Alan Kay concedes that "it would not be surprising if the visual system were less able in this area [of programmation] than the mechanism that solve noun phrases for natural language. Although it is not fair to say that `iconic languages can't work' just because no one has been able to design a good one, it is likely that the above explanation is close to truth". [4]

Mutant

CORE CORE bash bash CORE bash

There are %d possibilities. Do you really
wish to see them all? (y or n)

SECONDS
SECONDS

grep hurt mm grep terr mm grep these mm grep eyes grep eyes mm grep hands
mm grep terr mm > zz grep hurt mm >> zz grep nobody mm >> zz grep
important mm >> zz grep terror mm > z grep hurt mm >> zz grep these mm >>
zz grep sexy mm >> zz grep eyes mm >> zz grep terror mm > zz grep hurt mm
>> zz grep these mm >> zz grep sexy mm >> zz grep eyes mm >> zz grep sexy
mm >> zz grep hurt mm >> zz grep eyes mm grep hurt mm grep hands mm grep
terr mm > zz grep these mm >> zz grep nobody mm >> zz prof!

```
if [ "x`tput kbs`" != "x" ]; then # We can't do this with "dumb" terminal
        stty erase `tput kbs`
```

DYNAMIC LINKER BUG!!!**Codework by Alan Sondheim, posted to the mailing list "arc.hive" on July 21, 2002**

In a terminal, commands and data become interchangeable. In "echo date", "date" is the text, or data, to be output by the "echo" command. But if the output is sent back to the command line processor (a.k.a. shell) - "echo date - sh" - "date" is executed as a command of it own. That means: Command lines can be constructed that wrangle input data, text, into new commands to be executed. Unlike in GUIs, there is recursion in user interfaces: commands can process themselves. Photoshop, on the other hand, can photoshop its own graphical dialogues, but not actually run those mutations afterwards. As the programmer and system administrator Thomas Scoville puts it in his 1998 paper "The Elements Of Style: UNIX As Literature", "UNIX system utilities are a sort of Lego construction set for word-smiths. Pipes and filters connect one utility to the next, text flows invisibly between. Working with a shell, awk/lex derivatives, or the utility set is literally a word dance." [6]

In net.art, jodi's "OSS" comes closest to a hypothetic GUI that eats itself through photoshopping its own dialogues. The Unix/Linux/GNU command line environment is just that: A giant word/text processor in which every single function - searching, replacing, counting words, sorting lines - has been outsourced into a small computer program of its own, each represented by a one word command; words that can process words both as data [E-Mail, text documents, web pages, configuration files, software manuals, program source code, for example] and themselves. And more culture-shockingly for people not used to it: with SSH or Telnet, every command line is "network transparent", i.e. can be executed locally as well as remotely. "echo date - ssh user@somewhere.org" builds the command on the local machine, runs it on the remote host somewhere.org, but spits the output back onto the local terminal. Not only do commands and data mutate into each other, but commands and data on local machines intermingle with those on remote ones. The fact that the ARPA- and later Internet had been designed for distributed computing becomes tangible on the microscopic level of the space between single words, in a much more radical way than in such monolithic paradigms as "uploading" or "web applications".

With its hybridization of local and remote code and data, the command line is an electronic poet's, codeworker's and ASCII net.artist's wet dream come true. Among the poetic "constraints" invented by the OULIPO group, the purely syntactical ones can be easily reproduced on the command line. "POE", a computer program designed in the early 1990s by the Austrian experimental poets Franz Josef Czernin and Ferdinand Schmatz to aide poets in linguistic analysis and construction, ended up being an unintended Unix text tool clone for DOS. In 1997, American underground poet ficus strangulensis called upon for the creation of a "text synthesizer" which the Unix command line factually is. "Netwurker" mez breeze consequently names as a major cultural influences of her net-poetical "mezangelle" work "#unix [shelled + otherwise]", next to "#LaTeX [+ LaTeX2e]", "#perl", "#python" and "#the concept of ARGS [still unrealised in terms of potentiality]". [7] Conversely, obfuscated C programmers, Perl poets and hackers like jaromil have mutated their program codes into experimental net poetry.

The mutations and recursions on the command line are neither coincidental, nor security leaks, but a feature which system administrators rely on every day. As Richard Stallman, founder of the GNU project and initial developer of the GNU command line programs, puts it, "it is sort of paradoxical that you can successfully define something in terms of itself, that the definition is actually meaningful. [...] The fact that [...] you can define something in terms of itself and have it be well defined, that's a crucial part of computer programming". [8]

When, as Thomas Scoville observes, instruction vocabulary and syntax like that of Unix becomes "second nature", [9] it also becomes conversational language, and syntax turns into semantics not via any artificial intelligence, but in purely pop cultural ways, much like the mutant typewriters in David Cronenberg's film adaption of "Naked Lunch". These, literally: buggy, typewriters are perhaps the most powerful icon of the writerly text. While Free Software is by no means hard-wired to terminals - the Unix userland had been non-free software first -, it is nevertheless this writerly quality, and break-down of user/consumer dichotomies, which makes Free/Open Source Software and the command line intimate bedfellows.

[This text is deliberately reusing and mutating passages from my 2003 essay "Exe.cut[up]able Statements", published in the catalogue of ars electronica 2003.]

References

[Bar75]
Roland Barthes. *S/Z*. Hill and Wang, New York, 1975.

[Ben97]
David Bennahum. Interview with Richard Stallman, founder of the free software foundation. *MEME*, (2.04), 5 1997. http://www.ljudmila.org/nettime/zkp4/21.htm .

[Kay90]
Alan Kay. User Interface: A Personal View. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design*, pages 191-207. Addison Wesley, Reading, Massachusetts, 1990.

[Sco98]
Thomas Scoville. The Elements of Style: Unix as Literature, 1998. http://web.meganet.net/yeti/PCarticle.html .

Footnotes:

[1] [Bar75], p. 200

[2] [Bar75], p. 5

[3] [Bar75], p. 4

[4] [Kay90], p. 203

[5] Codework by Alan Sondheim, posted to the mailing list "arc.hive" on July 21, 2002

[6] Thomas Scoville, The Elements of Style: Unix As Literature, [Sco98]

[7] Yet unpublished as of this writing, forthcoming on the site http://www.cont3xt.net .

[8] [Ben97]

[9] [Sco98], ibid.

" >

span-->

‹ Software art up Game art: theory, communities, resources ›

Game art: theory, communities, resources

By marloes
Published: 09/18/2007 - 13:30

Tom Betts , September 2007

Like all digital media, video-games can be designed, produced, deconstructed and re-appropriated within the context of art. Even though the history of video-games is relatively short, it is already rich with examples of artistic experimentation and innovation. Unlike film or video, games still represent a fairly immature medium, slowly evolving to locate itself in mainstream culture. The majority of games often present simplistic or crude visions of interactivity, narrative and aesthetics, but the medium offers unique potential for the creation of exciting new forms of art. Like any digital medium the evolution of art/games is closely tied to the development of software, hardware and the socio-cultural forms that grow around this technology.

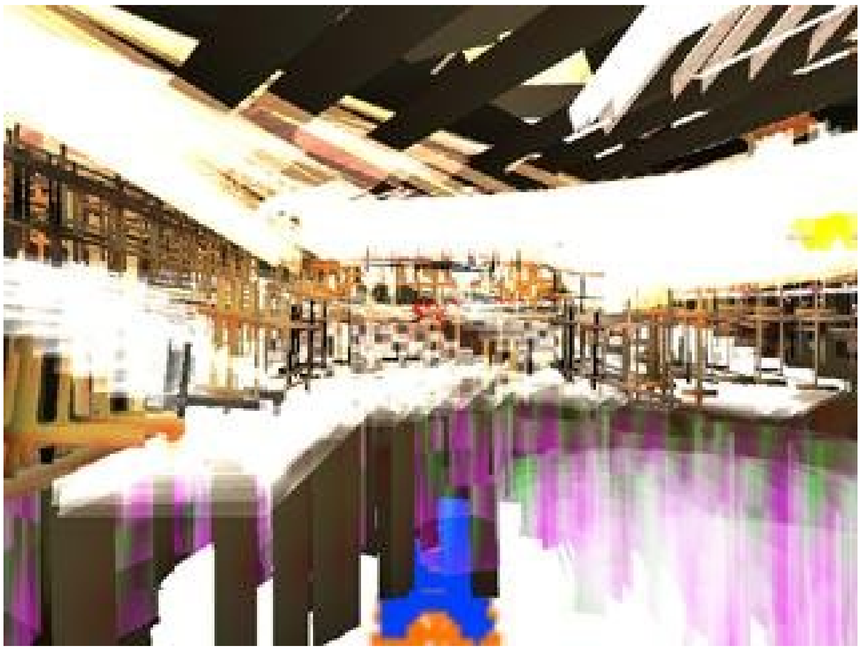
In broad terms there are two general approaches by which artists interface with gaming culture and technology. They can either modify existing games/games systems to produce work ranging from small variations to total conversions. Or they can code their own work from scratch. Of course there are blurred margins between these two practices, in some cases a modification becomes totally independent, or a new code engine may borrow from an existing system.

Modification

Modifying existing game systems is usually easier than programming your own, especially now that game modification is an encouraged part of many product life cycles. Games are frequently produced with modification tools included and developers strive to promote a community of amateur 'add on' designers and 'mapmakers'.

Early game modification grew out of the culture of hacking. In the 8bit era [41] many magazines would publish 'pokes', cheat codes that directly altered the memory locations of a running game to increase the players chances of winning (extra lives etc). It became common to add graffiti-like 'tags' to loading screens or even alter the graphics within the game itself (as is still obvious in the 'warez' community today [42]). Although often requiring arcane machine code knowledge a few intrepid coders re-wrote levels from popular games and made other conversions and modifications to commercial software. While most of these interventions were little more than novelties, some successful conversions highlighted the advantages of an open ended game engine [45] where modification was possible if not encouraged. Soon developers and publishers realised that a stream of modifications could greatly extend the shelf life of a product and in many cases act as free beta testing for subsequent iterations of a series. The most prominent early expression of this was with the Doom/Quake series of games where modifications and tools became widespread.

At this stage it became easy to identify three different types of game modification: Resource Hacking, Code editing and Map/Texture making. Resource hacking is the method closest to original hacking , in this case the user modifies aspects of the program without the use of any publisher-provided tools or code. The second method involves the developers policy to release source code for the game (either for 'add ons' or total conversions. However to make use of such code releases often requires substantial programming skills. ID software (the developers of quake/doom) are one of the few companies that release source code to the public. The final method focuses on using official (and unofficial) tools to manipulate the game content and game play. The third area of practice involves much less technical prowess than the previous two and is increasing in popularity with both mod-makers and developers.



qqq by nullpointer

Due to the increasing range of software tools to aid modification and the support for modding by the industry, a large number of artistic projects have followed this path. Notable works include: JODI's minimalist modification of Wolfenstein (SOD) [1] , The psychadelic Acid Arena project [2] , Langlands and Bell's Turner prize nominated The House of Osama bin Laden [3] , Retroyou's abstraction of the open source X-Plane engine Nostalg [4] , nullpointer's network/performance mod of quake3 QQQ [5]

, the NPR quake [6] modification from quake1 source code, UK artist Alison Mealey's UnrealArt [17] , and many more.

Jake by Alison Mealey

Some artists approach the idea of modification in a more social context, disrupting game systems with more interventionist approaches. These modifications are often more time based and performative. The Velvet Strike [7] project introduced the idea of an anti-war graffiti protest into CounterStrike, Joseph Delappe [8] has performed readings of war poetry within multiplayer death match environments, and both RSG [9] and Lars Zumbansen [10] have investigated various 'prepared controller' [43] type setups.

This form of performance (within the game) is closely aligned with the growing machinima movement. In machinima productions game software is used to set, script and record in-game movies. Although prominently used as a cheap but hip film making process (see Red vs Blue [11]), machinima can be a powerful tool to present artistic ideas to a passive audience (see This Spartan Life [12] - Mckenzie Wark)

Games modification can also incorporate hardware hacking. Eddo Sterns Fort Paladin [13] has a PC pressing its own keys in order to control a confused Americas Army recruit, the painstation project and the Tekken Torture [15] events punish the player with physical injury when they lose a game, and Cory Arcangel[16] has produced a series of re-written Super NES [40] cartridges where he has radically altered Super Mario World [44] .

Extensive modification can sometimes take the designers/artists so far from the original game engine that the project becomes a Total Conversion (TC). At this point people often consider the work as an independent game-object rather than a direct modification. Artists collective C-Level used heavily modified game engines and content to produce Waco Resurrection [18] , a fully playable game, allowing the player to enter a hyper real re-enactment of the Texas Waco siege.

Outside modification / Developing Independently

Although modification provides a relatively easy path in terms of learning curve and software, it can be limiting for certain projects. If artists/programmers require more power or control they can either use existing software engines as a basis for development or they can code their own from scratch. Due to the higher level of skills/resources needed to do this (mainstream games developers often have multi-million budgets and 100+ staff), large projects based on original code are less frequent than modifications. However, rather than pursuing high end technological ambitions, independent developers tend to have more success working with simpler technology.

Working with introductory level systems like java applets or flash movies has allowed artists to produce relatively big impact projects. Flow [19] , is an abstract game-environment written by Jenova Chen, SodaPlay [20] is a sandbox physics game by Soda, Samarost [21] is a twisted fairytale adventure, The Endless Forest [22] is a whimsical multiplayer space and N [23] is a minimalist platform game by Metanet. All of these projects have received significant praise despite(and because of) their opposition to mainstream gaming. In some cases these independent projects have broken through to mainstream consoles (flow [19] , braid [24] , everyday shooter [25])

Spring Alpha by Simon Yuill

Artists with access to larger teams and resources can produce larger projects that move closer to using the same production model as the mainstream. These works can spread over longer timescales and involve complex iterations. The spring_alpha [26] project, directed by Simon Yuill is a project using gaming to explore forms of social organisation. Based on the drawings of artist Chad McCail and the unusual dystopia/utopia we see in the Sims, Spring Alpha has evolved into a number of side-projects and FLOSS developments.

Within mainstream publishing a small minority of titles are often upheld as representing the 'art' side of the industry. Games like Rez, Ico and Okami are frequently cited for their atmospheric environments and abstract aesthetics, Metal Gear Solid 2 is upheld as a postmodern remix and sandbox gaming project, promoted by the Sims, that has led to the current growth and popularity of content/community systems such as Second Life.

Resources

Although still a niche area of practice there are a few notable resources that are well worth investigating in order to learn more about game-art. Selectparks [27] is a collaborative blog and archive for discussing and promoting game art activities. It has a substantial collection of articles and news items and also hosts several artists projects. Based in Australia, the Selectparks team often have some influence on FreePlay [28] a Melbourne based games conference/festival that concentrates on independent and creative development of games culture. Aside from these organisations several influential exhibitions are worth mentioning. Cracking the Maze [29] was probably the first example of a solely game themed art show. Games [30] , an exhibition in Dortmund was a particularly large collection of game based artwork. More recently the Gameworld [31] exhibition at Laboral in Spain attempted to mix mainstream innovation with artistic statements and gamer culture.

Perhaps more than any other forms of digital art, games based art has the benefit, and sometimes barrier, of a strong existing gamer community. This is the place you are most likely to find up to date help, criticism and examples on all aspects of game design and production. Developer/Player blogs and forums often host heated debates and criticism, such discussion is vital for a medium that is slowly adopting more mature forms and content. The Escapist [33] is a weekly e-zine that covers the more thought provoking aspects of gaming culture. Play this Thing [34] is a games review blog that focuses on fringe gaming genres or controversial titles. Gamasutra [46] is a site devoted to both industry articles and critical commentary. Experimentalgameplay.com [32] is a site dedicated to prototyping innovative game ideas in short time frames. From a more production based approach sites like NeHe [36] and GameDev [37] provide links to information on programming in C or 3D, whereas Processing [38] and GotoandPlay() [39] introduce novice programmers to working with java and flash. As with all programming, observing and deconstructing other projects is a great way to train your own skills, luckily most programmers are more than happy to compare notes and offer advice via the forums attached to the above sites.

One crucial piece of advice is to start simple and have achievable goals. Producing or modifying games can be a difficult and time consuming task. It's wise not to be overambitious in your designs and consider the scale of existing art projects that have used similar technology. It is also important to consider how your final work can be distributed or presented as different technologies will require different hardware. Aside from these issues games provide a vast range of expressive forms for creating art and are one of the cheapest ways to access cutting edge technology and distribution.

Notes

[1] http://sod.jodi.org

[2] http://acidarenaweb.free.fr

[3] http://www.langlandsandbell.demon.co.uk/obl01.html

[4] http://nostalg.org/nostalg_pw.html

[5] http://www.nullpointer.co.uk/qqq

[6] http://www.cs.wisc.edu/graphics/Gallery/NPRQuake

[7] http://www.opensorcery.net/velvet-strike

[8] http://www.delappe.net

[9] http://r-s-g.org

[10] http://www.hartware-projekte.de/programm/inhalt/games_file_e/work_e.htm#...

[11] http://rvb.roosterteeth.com/home.php

[12] http://www.thisspartanlife.com/episodes.php?id=4

[13] http://www.eddostern.com

[14] http://www.painstation.de

[15] http://www.c-level.cc/tekken1.html

[16] http://www.beigerecords.com/cory/tags/artwork

[17] http://www.unrealart.co.uk

[18] http://waco.c-level.cc/

[19] http://intihuatani.usc.edu/cloud/flowing

[20] http://sodaplay.com

[21] http://www.amanita-design.net/samorost-1

[22] http://www.tale-of-tales.com/TheEndlessForest

[23] http://www.harveycartel.org/metanet/downloads.html

[24] http://braid-game.com/news

[25] http://www.everydayshooter.com

[26] http://www.spring-alpha.org

[27] http://www.selectparks.net

[28] http://www.nextwave.org.au

[29] http://switch.sjsu.edu/CrackingtheMaze

[30] http://www.hartware-projekte.de/programm/inhalt/games_e.htm

[31] http://www.laboralcentrodearte.org/portal.do?TR=C&amp;IDR=105

[32] http://www.experimentalgameplay.com

[33] http://www.escapistmagazine.com

[34] http://playthisthing.com

[35] http://www.gamasutra.com

[36] http://nehe.gamedev.net

[37] http://www.gamedev.net

[38] http://www.processing.org

[39] http://www.gotoandplay.it

[40] Super Nintendo Entertainment System or Super NES is a 16-bit video game console that was released by Nintendo in North America, Europe, Australasia, and Brazil between 1990 and 1993.

[41] 8 bit era refers to the third generation of video game consoles. Although the previous generation of consoles had also used 8-bit processors, it was in this time that home game systems were first labeled by their "bits". This came into fashion as 16-bit systems were marketed to differentiate between the generations of consoles.

[42] The warez community is a community of groups releasing illegal copies of software, also referred to as 'the Scene'

[43] Prepared controllers are video game input devices such as game pads and joysticks that have been modified, by for instance locking certain buttons or rewiring them.

[44] Super Mario World is a platform game developed and published by Nintendo.

[45] A game engine forms the core of a computer video game. It provides all underlying technologies, including the rendering engine for 2D or 3D graphics, a physics engine or collision detection, sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, and a scene graph.

[46] Gamasutra http://www.gamasutra.com

Images

[1] qqq by nullpointer, image courtesy of the artist

[2] Jake by Alison Mealey, image courtesy of the artist

[3] Spring Alpha by Simon Yuill

" >

span-->

‹ $(echo echo) echo $(echo): Command Line Poetics up Developing your own hardware ›

Developing your own hardware

By admin
Published: 09/26/2007 - 13:37

Kristina Andersen , September 2007

Making your own technology can mean voiding the warranty of your iPod by "modifying to alter functionality or capability without the written permission of Apple" [8] , to straight up hacking and bending commercial products and on to building circuits from scratch. These techniques are all methods to graduate from user to owner in our everyday relationship with technology. If you cannot open it, you don't own it [9] and if you work with circuits as an artist and maker it becomes especially important to exercise ownership on the technological vehicles of the work.

B087 Lie-detector, Kemo-Electronic GmbH

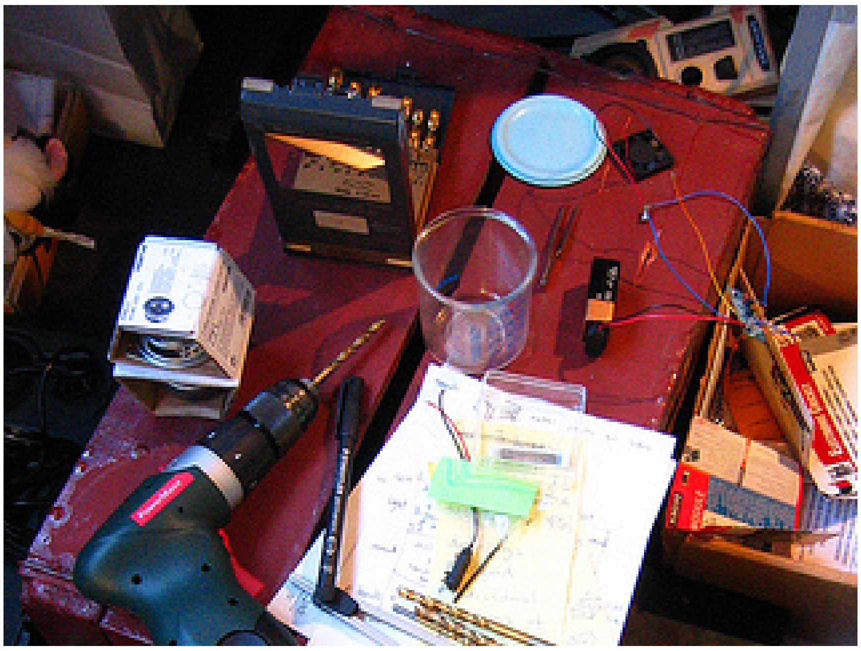
Since the 50's it has been possible to buy elaborate DIY electronic kits with farfetched uses and over-Xeroxed line drawings - made for father and son bonding sessions. When you first begin to solder and build circuits these are the kits you start out with. You order them from giant illegible catalogues and when they arrive they look as if they have been composed entirely out of reclaimed components from the soviet space programme. Which is why you keep ordering them: The beauty of the matte square green LED that sometimes comes with the KEMO lie detector kit [1] . Where else would you find that?

It is worth noting that almost nobody teaches their daughters to solder. The stupidity of the dog barking circuit kits, paired with a generally patriarchal electronics culture, has so far made electronics a largely non-female pursuit even as other areas like fashion and fabric crafts moves further towards the customised and the self-made. In fact the process of crafting fabrics is remarkably similar to the building of circuits. Like other forms of crafts, making your own technology can be seen as a sculptural process of allowing objects to be formed in your hands and as such it is an almost primal process of discovery and recognition. As an object slowly takes form on the table in front of us, we can begin to develop intuitions about its capabilities and behaviour. Maybe in an extrapolation of Hayes' 'naive physics' [6] we could say that it is through the actual making that we begin to comprehend the objects we are building.

Details of the artists' studio

Making and crafting can also be said to be a form of play as long as one keeps in mind that playing is not necessarily light-hearted or fun; but rather a way of approaching a particular design space and a set of rules and limitations [3] . We are making our home made technological objects to the best of our ability and the limitations we face are our own. Claiming this kind of independence is a way to retain some level of control over the work. This does not mean we do not need help and collaborators. It just means that when it fails, we stand a chance of asking better questions and questioning the advice we receive. In the crafters manifesto Ulla-Maaria Mutanen says that the things we make ourselves have magic powers and hidden meanings that other people can't see [11] , and this is true for the hand made circuit as well.

Things are changing in the world of making things. In the wake of the emergence of Interaction Design as a discipline, basic stamps and PIC boards are suddenly on every curriculum and there are enormous amounts of "make your own" tutorials on the web. Instructables, a website where users share step-by-step project instructions, has its origin at the MIT Media Lab as a platform for students to share projects [7] , Hack a Day publishes a new hack each day harvested from enthusiasts on the web [5] , popular websites like BoingBoing [2] and expensive paper publications like Make magazine from O'Reilly [10] have brought the hobbyist and part time makers out into some sort of mainstream.



Details of the artists' studio

For the artist and the maker DIY is an ongoing process of discovery and failure. You are trying to use your clumsy and limited abilities to create that which did not exist before: unusual behaviours, objects that exist in response to flimsy thoughts and desires. Along the way through the making you encounter the limitations of your hands and your patience, but you find something else as well: the accidental object, the technology that exist merely as a quirk, the successful misconstruction. These are all glimpses of the material speaking back to you, the maker. They are the gifts from working with your hands and making your own. Nick Collins says it in the 16th rule of hardware hacking: "If it sound good and doesn't smoke, don't worry if you don't understand it" [4] . This is perhaps why we do it. By using our crude and clumsy hands to make aspects of computational machinery we are re-inserting ourselves into a process that we are otherwise excluded from. We make technological objects in an attempt to glean understanding, to see if our naïve intuitions still functions in a world of current and solder.

Maybe we can say, we are making technology in order to understand it, and understanding technology in order to make our own.

Notes

[1] B087 Lie-detector, Kemo-Electronic GmbH, viewed 4 October 2007, http://www.kemo-electronic.de/en/bausaetze/b087/index.htm

[2] Boingboing, 2007, Happy Mutants LLC, viewed 4 October 2007, http://www.boingboing.net

[3] Caillois, R.1961, Man, Play, Games, University of Illinois Press. Illinois.

[4] Collins, N. 2006. `Laying of Hands in Handmade Electronic Music' in The Art of Hardware Hacking, Routledge, New York.

[5] Hack A Day, 2006, Hack A Day Beta, viewed 4 October 2007, http://www.hackaday.com

[6] Hayes, P. 1978, The naive physics manifesto Systems in the Micro-Electronic Age, Edinburgh University Press, Edinburgh.

[7] Instructables, 2007, Instructables, viewed 4 October 2007, http://www.instructables.com

[8] iPod and iSight Warranty, 2007, Apple's Limited Warranty document for iSight and iPod, Apple

Inc., viewed 4 October 2007, http://images.apple.com/legal/warranty/docs/ipodisight.pdf

[9] Jalopy, M. 2005. Owner's Manifesto, Make04, viewed 4 October 2007, http://makezine.com/04/ownyourown

[10] Make Magazine, 2007, O'Reilly Media, Inc. viewed 4 October 2007, http://www.makezine.com

[11] Mutanen, U. 2005, Crafter Manifesto, Make04, viewed 4 October 2007, http://www.makezine.com/04/manifesto

Images

[1] B087 Lie-detector, Kemo-Electronic GmbH, viewed 4 October 2007, http://www.kemo-electronic.de/en/bausaetze/b087/index.htm

[2] Studio details. Photographed by Suno

" >

span-->

- Hardware hacking: open hardware and stand alone objects

‹ Game art: theory, communities, resources up Hardware hacking: open hardware and stand alone objects ›

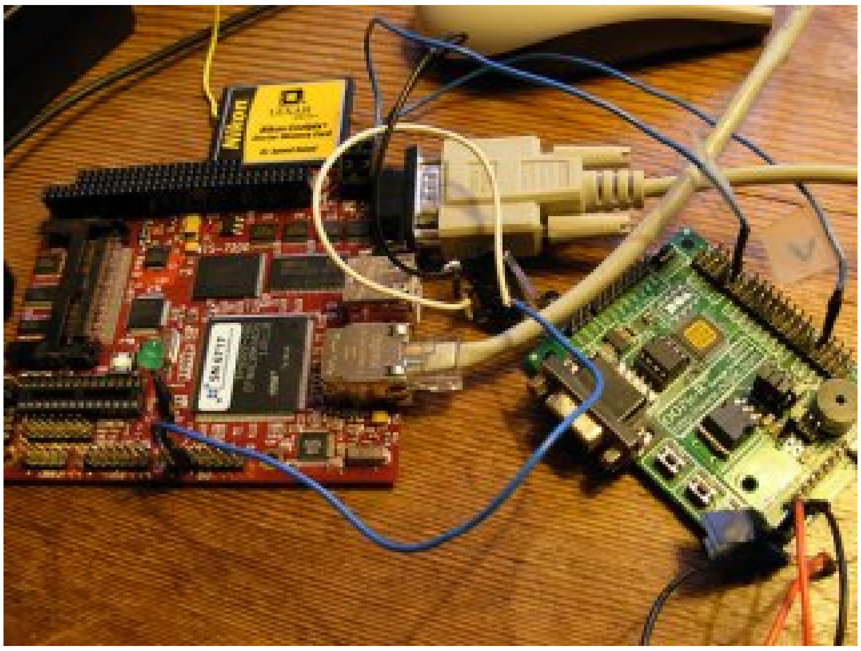
Hardware hacking: open hardware and stand alone objects

By marloes
Published: 09/18/2007 - 13:20

Tom Schouten , September 2007

That's a broad title. Building your own hardware can mean a lot of different things. To narrow the scope a bit, this article talks about embedded Single Board Computers (SBCs) and microcontrollers from birds eye view. An embedded SBC is anything that is complex enough to comfortably host a standard operating system, while a microcontroller is too small for that.

Physically an SBC consist of a single printed circuit board (PCB) with some RAM and Flash ROM memory chips, and usually a single highly integrated chip with CPU and peripheral devices. Often the smaller microcontrollers have RAM and Flash integrated together with CPU and peripherals on chip. For larger memory sizes, it is still (2007) more efficient to put memory in separate chips. To give you an idea, the minimal specifications of an SBC is something like a 32-bit CPU with about 8MB of RAM and about 4MB of permanent storage, usually in the form of flash memory.



TS-7200 ARM Single Board Computer

Anything simpler than an SBC we call a microcontroller. A microcontroller is typically a System on Chip (SoC). A SoC is a complete computer system in a single chip package, containing processor, RAM, permanent storage (Flash), and I/O peripherals. A microcontroller can run completely stand-alone, which makes them very interesting from a DIY perspective.

This article aims to give an overview of available tools without going into too much detail. The projects mentioned are well documented and most have and active community. Hopefully this will tickle your interest, so you can go on exploring for yourself.

In order to interact with the physical world, you need sensors and actuators. A sensor is a measurement device, and an actuator is a control device. Most embedded systems convert some kind of physical information (i.e. a button press, or a sudden acceleration) into a physical action (i.e. the switching of a light or the speed of a motor). What happens inbetween is handled by the computing system, and can be quite elaborate. Sensors and actuators are also called transducers: devices that convert one form of energy into another. A sensor is a low-energy transducer which converts any form of energy into electrical energy, while an actuator is a high-energy transducer which transforms electrical energy into any other form of energy. Getting these things to work is in the domain of analog electronics or popularly called physical computing.

It is assumed that you have some basic knowledge on the electronics and physical computing side, or know how to obtain this knowledge (see introduction). This article deals mostly with embedded software and tools: these days, doing hardware means also doing software. Some nostalgic and crazy hacks set aside, most problems are better, easier, cheaper and faster solved in software. Open up any electronic device; inside you'll find some kind of microprocessor.

A PIC 18F8720 microcontroller

Digital electronics systems can get quite complex, but it is not necessarily so that a such a highly complex system is difficult to develop for, or that a system of low complexity is easy. The main reason of course is that complexity can be hidden by modular design. If you can (re)use a lot of hardware and software designed (and made available!) by others, the perceived complexity of a system can be fairly low, even if it is extremely complex on the inside.

For example, a state of the art Single Board Computer (SBC) tends to be a very complex machine. However, due to the availability of a huge amount of reusable software tools, programming languages and code libraries, making it do something interesting and original does not have to be a complete nightmare. On the other hand, using a very simple and small microcontrollers can be quite difficult when you need to write some arcane low level code to squeeze out that last bit of performance. The middle road has a lot of wiggle room; for most projects that are not too sensitive to hardware cost, one can opt for a platform that makes the development more straightforward.

Also, let's assume it is not feasible to design your own embedded SBC circuit due to complexity and cost, and assume that such a system will have to be bought as a building block. On the other hand, we can assume it is feasible to design your own microcontroller circuits, and will briefly mention a collection free software tools that can be used to this end.

This article does not talk about sensor and actuator boards. Such a device is intended to be connected to a SBC or PC and used merely as an I/O extension, and cannot be programmed beyond simple configuration. If you're looking for something simple and want to avoid microcontroller programming, such an I/O board might be an ideal approach. A popular one seems to be the Muio [10] . It is a modular system and has extensions to hook up sensors to control Pure Data, MAX/MSP, SuperCollider and Processing running on a PC.

A note about vendors. The embedded electronics market is extremely diverse. For SBCs this is not such a big deal since the software can be made largely independent of platform. For smaller microcontrollers however, it can become a problem because there is usually not enough memory space to have a layer of software that abstracts away hardware differences. Details of the target bleed through to the software, making the software more low-level.

Instead of endorsing any specific vendor, the article points out how to look for vendors providing products that can be combined with free software tools, and indicates which vendors are most popular from the perspective of free software tools. Note that a lot of vendors provide their own tools, often free of charge. These tools are in general not free in the "libre" sense, and as such not the subject of this article. They are interesting to check out nonetheless. If you allow yourself to be locked to a single vendor, using the tools bundled with delivered hardware might be the most pragmatic solution.

Embedded SBC

We defined an embedded Single Board Computer (SBC) as a system that is powerful enough to comfortably run an operating system. The operating systems of interest here are those based on the Linux kernel. The Linux kernel is a widely used open source operating system kernel. An operating system kernel is the lowest level program in an operating system which handles the communication to the hardware through device drivers, and manages the communication between other programs and hardware, and programs themselves. As a result, programs that run on top of the kernel can talk to the hardware in a generic way, without having to know the particularities of the devices used. A kernel also allows programs on the same machine, or programs on different machines to communicate with each other. An example of this is the TCP/IP network protocol, which is a set of conventions used by machines to communicate through a packet switched computer network.

The Linux kernel is a fairly standard software platform that is largely independent of the underlying hardware platform. The most popular processor architectures that fall in this class are ARM9, MIPS and PowerSBC. Think wireless network router, network storage device, cell phone, PDA and handheld gaming consoles.

There is a tremendous amount of free software available to run on Linux. If you are in a position to use such a device, "building your own hardware" might be nothing more than finding the right software to use, compiling it for your embedded SBC and gluing together the parts. Most SBC vendors provide their own GNU Linux based tool chain. There are a lot of websites about embedded linux; it is a booming business. One that deals specificly with devices can be found here [9] .

Note that from a DIY or educational perspective it might even be more economical to gut a finished consumer product and put your own software on it! These products are mass produced, and thus cheap and ubiquitous. These properties usually cause the emergence of a community working on alternative software, enabling people to repurpose these locked down devices. The hard work of opening up a particular mass-produced device might already be done.

A good example of this is the OpenWrt [11] project. This is a complete operating system for embedded devices, originating on the Linksys WRT54G wireless router platform, but now available for a large amount of different hardware configurations, some of which are readily obtainable, general purpose embedded boards. The OpenWrt project is based on the Buildroot [3] suite and the ipkg binary package management system. Buildroot is a system that makes it easy generate a cross-compilation toolchain and root filesystem for your target Linux system using the uClibc C library.

```
[illegible]:~$ ssh root@[illegible]
sh: /usr/X11R6/bin/[illegible]: not found

BusyBox v1.00 (2005.09.14-15:55+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

 W I R E L E S S   F R E E D O M
WHITE RUSSIAN (RC3) ---------------------------------
 * 2 oz Vodka    Mix the Vodka and Kahlua together
 * 1 oz Kahlua   over ice, then float the cream or
 * 1/2oz cream   milk on the top.
 ---------------------------------------------------
root@fred:~# 
```

OpenWRT on a Linksys WRT54G

Now what does this all mean? A tool chain is a collection of programs that can transform software in source form into machine code your embedded SBC can understand. The source code language for Linux and the GNU system is mostly C. The component that performs the main part of this task is called a compiler, which in the case of Linux is the GNU GCC compiler collection [6] . For embedded development, the tool chain usually does not run on the embedded SBC, but on a different machine we call the host. In this case the compiler is called a cross-compiler.

A root file system is a directory tree that contains the collection of binary programs and data files that comprise the rest of the embedded operating system complementing the kernel. It also contains your application(s). Essential parts are the init process, which is the first program that runs after the Linux kernel is loaded, and a command shell, which can be used to interact with an embedded system once it is booted, by typing in commands at a console keyboard. Most embedded systems allow for the connection of an RS232 serial console to avoid the need of separate keyboard and monitor connections.

A shell is also used to write scripts: a sequence of commands executed without human intervention. The init process usually runs a collection of shell scripts that configure the hardware for its intended use. In embedded systems, an often used shell is Busybox [17] . This shell is optimized to consume little space; it includes stripped down versions of standard unix commands.

An embedded Linux system which contains the a binary package management system like ipkg can download and install programs and the libraries and programs they depend on while the system is live. This can be very convenient, giving the embedded system much of the feel of a Desktop GNU/Linux distribution, only smaller in size. The difference between OpenWrt and a raw image you create yourself with Buildroot, is that the binary package management system makes it possible to obtain OpenWrt in pre-compiled form. The system is split into a core system which you install as a whole, and a lot of optional packages managed with the ipkg system.

Systems like OpenWrt can make the learning curve for embedded development rather smooth. The system components are standard and most of them are very well documented, with a lot of community support available. Add to that the ability to start from a running system without having to do any real work except to figure out how to upload the first binary image, and you get an ideal framework for experimentation.

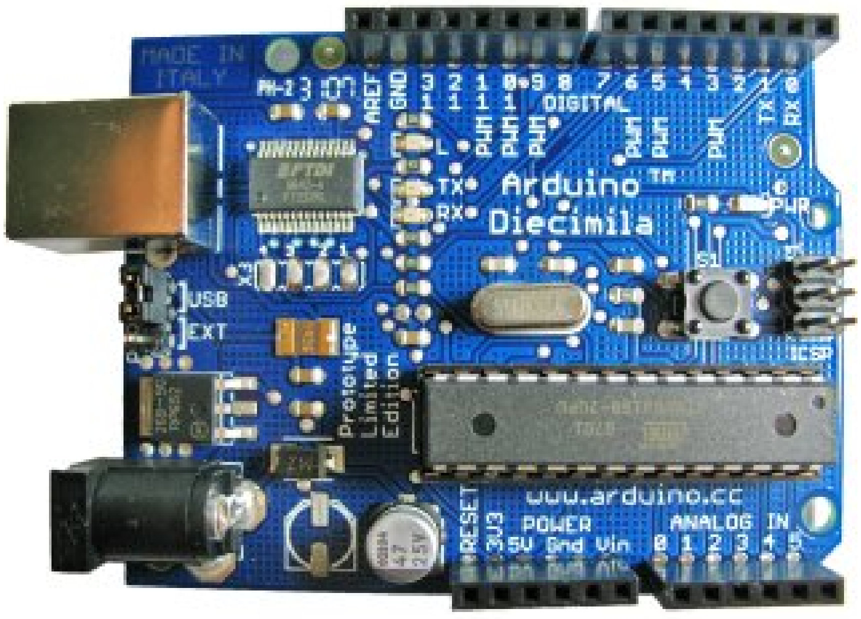
Microcontroller

Because of it's generality and still fairly high level of complexity, an embedded SBC solution might not be the best approach for simple sensor or actuator applications. While using an Embedded Linux SBC might make some complex tasks simpler, it can also make some simple tasks more complex. Some simple things are easier to do using a microcontroller.

It is mostly the absence of an external memory bus that makes it a lot simpler and cheaper. The embedded SBCs we talked about often consist of a highly integrated chip with external RAM and Flash memory, which creates the need for a lot of extra chip pins to be used as data and address busses for memory access. A SoC does not need a memory bus, so pins can be used for other things, or simply omitted. For example, there are microcontrollers with just 6 pins. A disadvantage of a SoC is that it is usually a lot smaller and offers less performance than a single board computer.

The main advantage of using microcontrollers is that they are simple from both an electronic circuit and a programming perspective. It is good to know that designing and building a circuit, and writing BASIC, C or assembly code is a task that can be done by a single individual in a matter of days or weeks, once the necessary skills are acquired.

If you do not want to design your own circuit, you can choose from a huge collection of simple controller boards. By far the most popular one seems to be the Arduino [15], which can be seen as a "sensor board on steroids" if used in its native environment. Arduino is targeted at artists who are looking for a low threshold entry in the world of physical computing, without giving up too much device performance. The Arduino language is based on C/C++, and is well documented. Underlying the Arduino project is the Atmel AVR 8-bit microcontroller, and the GNU GCC compiler. The Arduino design is open hardware, which means you can build the board yourself if you want to. The tools for Aruino are mostly free software, and run on OSX, Linux and Windows.

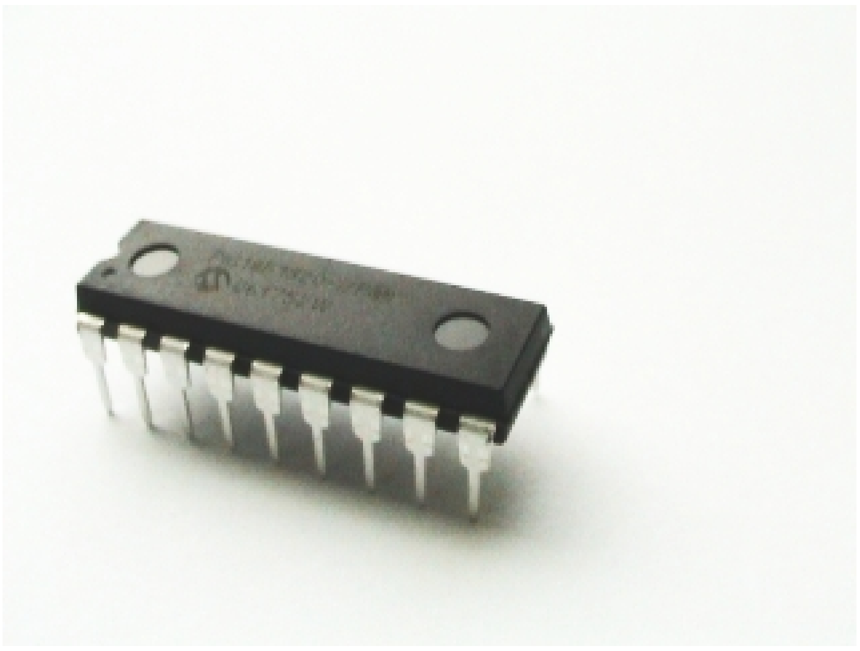


The arduino microcontroller board

Another off-the-shelf solution is the very popular Basic Stamp [22] . A Basic Stamp microcontroller is a single-board computer that has the Parallax PBASIC language interpreter in its permanent memory. Basic stamp software is provided by Parallax for the Windows, MAC and Linux platforms. Due to its popularity a lot of free embedded PBASIC software to run on the Basic Stamp is available. There is even an open source tool that lets you develop code for the Basic Stamp on the Linux platform [2] . Using the Basic Stamp is fairly straightforward. In addition a lot of community support is available, and like the Arduino, there are lots of places where you participate in a workshop for a hands-on introduction. The downsides of this platform are that it is fairly expensive and that it has limited performance compared to directly programmed chips.

The rest of the article will deal with the available open source tools to build your own custom hardware based on raw microcontroller chips. The first problem you encounter after deciding to use a microcontroller is to choose a chip and possibly board architecture. In the embedded SBC world this choice is less important due to the availability of the standard Linux platform which runs on almost anything. In the microcontroller world, there is an equally huge amount of different companies and architectures, but platform choices are more important due to the absence of a standard operating system layer.

For DIY projects, most people seem to go for the Atmel AVR [16] , or the Microchip PIC [21] . Which one is best? Unfortunately, the world is not 1-dimensional, and this includes the world of microcontrollers. One thing which seems to be a reality is that if you're not primarily cost-driven, you'll make the platform choice only once, and probably because of some arbitrary reason, insignificant in the long term. It takes some "getting to know", and can be hard to say goodbye to an old friend. Also note that per chip vendor, there is usually a very large set of architecture variants.



Another PIC 18F microcontroller

To be fair I have to switch to first person narrative here. Acknowledging my own bias, I'll tell you up front I chose the Microchip PIC18 architecture. What is more important than my choice, is the way I got to it. One often hears "just pick one", and there might be a lot of truth in that. However, I will do my best to give some arguments which might lead to an informed choice, because there are differences that can get important later on.

- Can you get direct support? Pick a platform that is used by an accessible expert in your direct surrounding.

- Programming languages, Tools and Libraries. This might be the most difficult one to know up front, but fortunately the easiest to change. Pick a platform with a lot of choice and diversity in this domain.

- Is there a large community surrounding the platform and its tools and libraries? Is there a forum where you can post questions? Do people seem helpful? Can you find indications of people getting stuck without anybody caring?

- Cost of hardware, including availability of free chip samples.

Free chip samples might seem an odd gift from the gods at first, but realize this is a form of vendor lock-in. Microchip has a very liberal sample policy. A good reason to choose for the Atmel AVR microcontroller is the existence of the Arduino board; it seems to have taken the world by storm. It provides an easier entry point to the world of microcontrollers, and uses a chip which is popular on its own. Once you are familiar with using the pre-fabbed board, the road to making your own microcontroller circuits becomes less steep. For more information, google for "pic vs avr". The first link that turns up for me is ladyada's comparison page [20] .

What do you need beside the actual hardware? You'll need to pick a programming language. Which language to choose is a topic of great debate, not only in the embedded software world. Some things you need to know about languages are that they tend force you think in a certain way. And in general, there is no silver bullet; the more power a language gives, the more time it usually takes to get to know it well, but the more interesting things you can do. Keep in mind that if you get stuck or your programs get too complicated, switching language might be an option. A language can be a dead end street: once you've exhausted its capabilities, it might pay off to back out and take a different road.

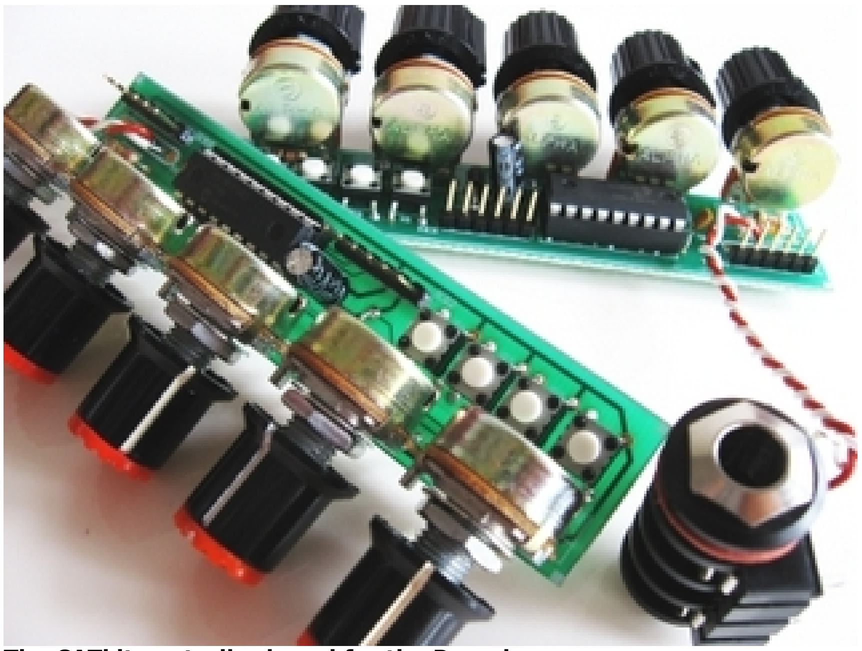
Traditionally, microcontrollers are programmed in assembly language. The reason for this is that they are often very resource constrained; the smallest versions have only a couple of bytes of ram, and putting the chip to good use can require some tricks specific to the platform. The advantage of using assembly language is that you are not limited in what you can do, but this is also a disadvantage since you have to take care of a lot of details yourself.

If you ever learn an assembly language just out of interest, a microcontroller might be the right choice due to its simplicity. Both AVR and PIC have very simple instruction sets, both being Reduced Instruction Set Computers (RISCs), although the PIC instruction set is a bit older and quirkier. Both chips have open source assembler tools [1] [7] . For PIC there is a very nice integrated development environment called PikLab [12] , similar to Microchip's MPLAB, which integrates with several compiler and assembler toolchains (like gputils, sdcc, c18) and with the GPSim simulator. It supports the most common programmers (serial, parallel, ICD2, Pickit2, PicStart+), the ICD2 debugger, and several bootloaders (Tiny, Pickit2, and Picdem).

Following the tradition, the next popular step up from assembly language is the C programming language. C is a bit more highlevel than assembly, in that it largely independent of the processor architecture. Remember that C is the language in which GNU/Linux is written. C is sometimes called portable assembler. The GNU/GCC compiler is available for AVR, and is used as the default compiler for the Arduino project. Also available is SDCC [14] , which supports Intel 8051, Maxim 80DS390, Zilog Z80 and the Motorola 68HC08 based microcontrollers. Work is in progress on supporting the Microchip PIC16 and PIC18 series.

In the DIY world, the BASIC language has always been fairly popular. Explicitly intended as a beginner language, it is a bit more limited and less expressive than C or assembly, but perceived as easier to learn and use. Due to the limitations and specific mindset it imposes, it is somewhat frowned upon in the professional world. Depending on your needs it might be the better choice nonetheless. There are several commercial solutions available. For PIC there is an open source compiler called Great Cow BASIC [5] .

Acting on the idea that popularity is not always a good indicator, I provide some pointers to more obscure language choices. For PIC there is a language called Jal [8] , loosely based on Pascal, which seems to have quite some followers. Then there are a couple of Forth dialects, the author's personal favorite. For PIC there is PicForth [23] , FlashForth [4] and (shameless plug) still in development, Purrr [13] .


The CATkit controller board for the Purrr language

Then finally, once you have selected a tool chain and a board or chip, you need a device called a programmer to dump the code generated by your tool chain onto the microcontroller's Flash memory. There are several ways of doing this. Both PIC and AVR provide their own in-circuit programmers. These devices can be used to program a chip while part of (soldered to) a circuit. In addition to this, there are a huge amount of DIY circuits you can build. Pick one you know somebody else uses!

To conclude the microcontroller section, I quickly mention board design. Commercial CAD packages are expensive for the simple reason that they are fairly complicated and highly specialized programs. In DIY circles, the most popular tool seems to be Eagle [18] . This program is free for noncommercial use and small board designs. For anything else you need to buy a licence. An alternative is GEDA [19] , an open source collection of design tools. Its main components are gschem, a schematics capture program, and PCB, a printed circuit board layout program. This tool set allows you to create CAD file, often called Gerber or RS-274X files. These files contain descriptions of the different board layers like copper, silk, solder resist and drill holes. To have a board built, you simply send the Gerber files to a PCB production or prototype house after you make sure your board respects their design rules.

Conclusion

Building your own hardware using open software and hardware tools can be done with a moderate amount of effort. By dividing the world into embedded SBCs and microcontrollers, I have showed how one might go about collecting the necessary tools. Especially for the embedded SBC approach there is a lot of free software available for reuse, and the GNU GCC tool chain is fairly standard. The microcontroller world is a bit more ad-hoc, not being as independent from hardware platform.

The trick in many cases is to take the best of both worlds; use a central general purpose embedded SBC connected to a couple of microcontroller based special purpose circuits, which you either build on perf board, or for which you design a circuit and have it fabricated.

Hope this helps. Good luck building!

Notes

[1] AVR assembler for Linux and others.

[2] Basic Stamp tools for Linux.

[3] Buildroot, a tool for building embedded Linux systems.

[4] FlashForth.

[5] Great Cow BASIC.

[6] The GNU Compiler Collection.

[7] GNU PIC Utilities.

[8] Jal, (not?) Just Another Language.

[9] Linux Devices.

[10] Muio, a modular system for sensing and controlling the Real World.

[11] OpenWrt a Linux distribution for embedded devices .

[12] PikLab.

[13] Purrr.

[14] Small Device C Compiler.

[15] Arduino, an open source electronics prototyping platform.

[16] Atmel AVR.

[17] Busybox.

[18] Eagle, a schematic capture and printed circuit board design package.

[19] Geda, a GPL suite of Electronic Design Automation tools.

[20] PIC vs. AVR.

[21] Microchip PIC.

[22] Basic Stamp by Parallax, a single-board computer for the PBASIC language.

[23] PicForth

Images

[1] TS-7200 ARM Single Board Computer

[2] A PIC 18F8720 microcontroller

[3] OpenWRT on a Linksys WRT54G. Photo by Tom Schouten

[4] The arduino microcontroller board

[5] Another PIC 18F microcontroller. Photo by Marloes de Valk

[6] The CATkit controller board for the Purrr language. Photo by Aymeric Mansoux

" >

span-->

‹ Developing your own hardware up