Towards a Permanently Temporary Software Art Factory (Notes for the Sustainability of Software Artifacts)

Javier Candeira

http://freesoftwareart.org

Javier Candeira is issuing a call to arms, leaping into the role of evangelist for the packaging and distribution of free software art wish great energy. He offers three primary goals for this project. First, to allow and promote code sharing between artists and therefore increase their productivity. Second, to facilitate software art distribution of easily installed packages. Finally, to aid conservation of software artworks through community maintenance of these packages.

Candeira's vision is compelling, one of a community helping each another freely, sharing their work openly and preserving their work for future generations. Further, his arguments are persuasive, instilling a sense of urgency, artwork being lost to bad licenses and ineffective distribution mechanisms.

In the style of a FAQ list, Candeira goes through many possible objections to Free Software Art, particularly those potentially held by software artists themselves. This forms a tightly argued, combative piece, using forthright language of evangelists from the wider free software world to head off many potential counter-arguments before they can be made.

So while Candeira's work leads towards a project with clear aims and direction, it leaves a great deal of room for dialogue of greater breadth and subtlety. Many software art mantras, such as «release early, release often» simply may not apply to software art. Further,

not all artists are so interested in wide distribution of their work, or in other people being able to pick through their workings, and gentler encouragement and debate may be required before they join a free software community. Indeed we might consider many culture clashes between existing free software communities and the free software artist community that Candeira encourages here.

But even with these doubts and more on our minds, we must look on with hope that something of great interest and worth can come of such a project. Reading between the lines we understand that Candeira is working towards a software art orientated sub-distribution of the Debian linux distribution. We wish him well.

Alex McLean

Abstract:

Free Software has already proved to be a viable method for developing operating systems and business applications such as Debian GNU/Linux and the OpenOffice.org suite. This paper explains why Free Software has a great deal to offer to the practitioners of Software Art, and why releasing Software Art works under Free licenses will help in their production, distribution and conservation. Some of those benefits will be derived from the licensing process, and some from the subsequent packaging by Free Software distributions. Finally, frequently asked questions about this project are answered in an accessible manner.

Table of Contents:

Free Software Art Manifesto

Software Art does not belong in a museum vault, but in a working processor, wherever that processor is located. When the artwork is removed from physical artifacts, the question of accessibility ceases to be about the audience's access to the work, but about the work's access to the audience's processor.

In order to have access to its audience, the work must first be able to be run on a wide range of hardware; second, it must be able to find its audience; and finally it has to endure the passage of time, if by «audience» we don't merely mean «audience at the time the work is created».

Therefore, Software Artists are advised to ahere to the Free Software ethos and methodology of software development, which fulfills the three objectives of enabling Production, Distribution and Conservation of software artworks.

Production

Most artists dealing with technology find that their first hurdle is acquisition of technical know-how, and that they often can't find assistance outside their social circle. The communal development methods of Free Software provide Software Art practitioners with a technically-gifted community of peers and mentors.

Software Artists Ship Code

Code reuse is very important in software development, and therefore in Software Art. Artists want to express their worldview, not to reinvent the wheel. The use of Free Software allows code creators to reuse code more effectively, and to build upon the work of oth-

ers not only technically, but also artistically, in a time-honoured tradition both of old masters and modern movements.

Production of Software Art is not only a technical and artistic process, but also a legal and economic one. Institutions funding software works request the assurance that they will be able to show and distribute the work they commission . As more and more institutions demand full freedoms for the work they paid for, software artists can help to fund their work by using Free Software both as their technical infrastructure and their licensing model.

Working with Free Software developers and the maintainers involved in its distributions will also help artists to navigate the sea of different Free Software licenses and to understand their implications.

Distribution

These are terms that have been long understood by practitioners of more demotic arts, like the novel and cinema: as soon as access to the means of production is acquired, and the cost of building the work is met, there are three aspects to a reproducible work's success: distribution, distribution, and... distribution.

Obscurity Is An Artist's Worst Enemy

Licensing software artworks under Free licenses allows for their inclusion in Free Software distributions, or «distros», By making the artworks part of their automatically installable and upgradeable repositories, distros allow any individual or institution to access the software programs for themselves or to present them to others.

The usage of free licenses is a legal guarantee of wider availability and diffusion: the technical possibility of diffusion and ease of installation is no guarantee if the works are limited by copyright law to particular geographical areas, social groups, or fields of endeavour.

The characteristics of availability of source code and its modifiability by third parties also help with distribution to other platforms: Free Software is often «translated» or «ported» to run on different hardware than the one originally used by the artist.

Conservation

An actively maintained Free Software distro's repository is a living artifact, not a dead pile of code that once ran. Libraries are updated, and programs recompiled to adhere to new standards and run on new platforms, all without vendor troubles or time limitations.

Software Is A Process, Not A Product

The Free Software maintenance process solves the issue of conservation of Software Art works. Licensing software artworks under Free licenses allows for their inclusion in Free Software distros, thus making their conservation part of that process.

Software Art works require a physical substrate to survive, and that substrate, the computer, quickly becomes obsolete. A software artwork included in a Free Software distribution survives its original hardware and circumstances.

The fact that libraries are free to use without restriction also saves the work from vendor obsolescence as the work is not tied to proprietary code that ceases to be available if the vendor disappears or stops renewing their software licenses. Even in the case that one particular distribution stops being updated, all the code is still available to be picked up by another distribution.

Finally, the use of free licenses is also a guarantee of wider temporal availability and diffusion: works are no longer limited to the period of one exhibition by the owners of some of the assets, as free software is essentially available to everyone, forever. Free Software artworks do not have to wait until they lapse into the Public Domain for an enterprising curator to revive them .

Free Software Art

Individual artists like David Griffith (Fluxus), projects like the Open Art Network (The Great Game(boy)) and even loose networks of hacker-activists (Carnivore) are now freeing their code in order to fulfill the promise of sharedness and open access for the practice of Software Art. Some of their works are starting to seep into Free Software distributions, again thanks to individual efforts. These individual efforts need to progress into more coordinated ones, a «rough consensus and running code» meta-project of artists freeing their code and Free Software distribution maintainers packaging it for ease of access and conservation.

The Free Software Art movement does not yet exist as such, but its seeds are already in both communities, especially in the handful of individuals who inhabit the intersection of both cultures.

Free Software Art: it can either mean «Software Art whose code is Free», or an imperative call to «liberate Software Art» from obsolescence, obscurity and oblivion. Let's do it. Let's Free Software Art.

Frequently Asked Questions about Free Software Art. Yes, people really ask this stuff!

How do I make a living if my code can be copied by anyone?

The same way you make it now. If you think you can make more money by closely guarding your code so nobody can use it without permission, by all means dig your own hole of obscurity and irrelevance. It is a bloody lottery, and you may well be the next Toshio Iwai. But this is the real world, where there is only one Toshio Iwai, and most of us will never be published by Nintendo. The question boils down to what are your other choices.

This is the real world where 99,9% of software artists make a living by teaching, doing gigs at festivals, getting commissions

from museums and institutions if they are lucky, working for other artists (roboticists, old-school installation artists wanting to update their craft), writing code for commercial software companies, consulting for other type of businessess and waiting tables. In this world, getting your code to the higher number of people out there is the best way to make yourself a name and live on the ancillary benefits of public recognition. So if you want to make a living as an artist you would do well by promoting the distribution of your work by any means, including making your code Free for anyone to copy it, study it, use it in their own work.

But won't everybody else then be able to make money off my code?

Yes. Anybody will be able to teach with it, perform with it live, curate exhibitions in which it will be shown. And they will get money for that. And you won't get any of their money.

But it's unfair!

Well, put it this way: what would you rather have, 100% of a paltry earning, or 1% of a take more than 100 times bigger? The diffusion afforded by Free Software allows more people's work to achieve more relevance, and to collect a smaller share of a higher amount of returns from their work.

Let me give you an example using Processing, the Free software development tool for artists: Casey Reas and Ben Fry, the project's originators, get the sweetest gigs teaching Processing seminars. They are the ones who go to Ars Electronica and collect a Golden Nica, and also derive other benefits from their centrality to a Free Software project that is so important to the Software Art scene. Casey is writing the Processing book, and if there were two books most people would buy his. This is as fair as I can think. If what you are saying is that they should teach all the Processing seminars in the world and write all the books about Processing for all publishers, I don't think this would be fair for others, and it wouldn't be fair for them either.

But then other people will retread my work by running my code, and my brilliance will become trite and cliched!

That is as fair a question as you could have posed, and you are right. But it is also true that if you are succesful people will emulate you, copy you, and reduce you to cliche anyway. Seen this way, having your code out there might even raise the standard for emulators (people who emulate other people, not code that emulates other hardware platforms).

But I don't want anyone to tweak my code a bit and claim it's theirs!

The 19th Century called; it wants its novelists back. If you wanted a real answer, well, most Free Software licenses do not authorise anyone to say your code is theirs, as copyright notices must normally be maintained. Also, we live a world where hex editors exist and anyone can illegaly modify any binary code and say it's theirs anyway without a Free License; bootleggers do it all the time with oldschool videogame ROMS. You are complaining about a problem that you might already have if you had shipped any code, and that this project doesn't do anything to worsen. Jeez!

But people might think my work is bad because someone modified it but it still carries my name!

Free Software licenses can and do include clauses stating non-endorsement by you of what other people do. There are conflicting opinions on whether licenses with clauses requiring that the changed binary has a different name are completely free, but code under such licenses are included in Free Software distributions such as Debian. And you can always publicly ridicule anyone who makes an ass of themselves by spoiling your precious code.

Doesn't packaging modify the original artwork?

It does and it doesn't. A Debian package contains your original source code in its original pristine form, and all the changes made by the packager are stored in separate files called patches. This way you, your users and art historians of the future can have the best of both worlds: integrity of the artwork and full compatibility of the binary.

But my artwork depends on a very specific and unique piece of hardware!

If your artwork is only true to itself if it runs on a unique and particular piece of hardware that cannot be reproduced, then it is an art installation, not software art according to our definition, it could never be packaged and distributed anyway, and you are reading the wrong FAQ.

And if that specific and unique piece of hardware is an old computer that most people can't have access to, maybe we can reproduce it under emulation, and have your Software Art work run on that emulator.

A legal-technical intervention on the Political Economy of Software Art

Doesn't emulation modify the original artwork?

That is a bit of a conundrum for which I have two answers; and both of them are «no»:

a) An emulator is something that stands in for hardware. The program can't tell a good emulator from the original hardware, and neither can the audience/user/operator. We all like vintage machines, but emulation is good for those who can't afford them, and what goes for videogames goes also for software art pieces.

b) Software art pieces are like theater plays. Consider this metaphor: software is run the way theater plays are played. If the code is Shakespeare's Hamlet, it can be played on any hardware: Gibson+Zefirelli+cinematography, funny_guys+IRC, paper+your_brain, Ethan_Hawke+Bill_Murray+awesomeness, TheRoyalShakespeareCompany+a_theatre.

What if a package is abandoned and ceases to be updated?

An abandoned package can leave the work in a much better state than the original code, and never in a worse one. It contains the original code plus metadata about its conservation history in the form of changelogs and patches. Patches amount to decisions by a conservator, and some of those patches will incorporate

changes done for policy reasons, but some of them will be bugfixes in the original code or adaptations for later technology.

So even if your work is packaged today, maintained for fifty years and then abandoned, it will be in a much better state when the 2105 arrives and your school experiences a centenary revival.

What if Debian(or Fedora/BSD/Whatever) ceases to exist?

Allow me first to say that Debian or BSD can die, yes. But I can't envisage any scenario in which Debian or BSD can die without Free Software being made illegal and a worldwide Martial Law being established by hostile aliens from some dorky kid's imagination.

Seriously, Debian, the *BSD and RedHat/Fedora will exist, in some form or another, for as long as Free Software exists. They might change names, splinter into derivatives or merge into the One True Distribution, but the goals of this project can be upheld as long as there is still one Free Software Distribution.

What if Free Software is just a fad?

Then Google and Amazon and Yahoo! will go away and disappear forever, and so will IMDB and most of the world DNS and HTTP servers, and most of the companies rendering FX for Hollywood films and... . If Free Software turns out to be a fad, the flying pigs covering the sun will give you enough food for thought that you will forget about your Software Art.

What if Software Art is just a fad?

I don't think software art will ever go away either. The label might, but generative art, experimental videogames and self-made performing tools are here to stay, among other sub-genres of what is now called Software Art.

I might be wrong too, and Sofware Art could well be, as the gentleman scholar who asked me this question feared, destined to wane as quickly as it has waxed. In that case its conservation is more needed than everything, and the distro-packaged Free Software Art works will be the best maintained collection of artifacts from this particular period in the history of art and tech.

The 19th Century called; it wants its novelists back

Isn't Runme.org already doing this?

They are and they aren't. They invited me to give this talk, and write this paper, and paid my trip, my hotel and a fee. I am grateful of that, and that is a way for them to support the project. But then they aren't doing it themselves, because although the aims of a Free-Software-Art initiative overlap with their charter, the overlap is incomplete.

By their own commitment to the form, runme.org select and curate software art under all types of licenses, including non-distributed (just exhibited and performed) and undistributable works. Some of those are undistributable because they are not under free licenses, and some are because... they aren't even pieces of software as such. The promotion of Free Software among software artists is not their main priority: it is ours. Free-Software-Art is about software distributed under Free and Open Source licenses. It is about putting software artworks in Free Software distributions, and that is work that has to be done from inside the distributions themselves. Runme.org could serve as a bridge between artists and distros, but the packaging still has to be done.

Isn't Jaromil/$name already doing this?

Jaromil is already doing the first part of this: by releasing his own work under free licenses, he is making sure that his work is accepted into distributions such as Debian and Fedora, which will make his work more long-lasting, will help other creators learn and work from his code, and serves also as a great example.

The second part of the process, the systematic packaging of Software Art works inside distros, is something that is just too big for one person. A distribution is not only a CD you can install on your computer, or an online-accessible repository of all their programs.

It is also the process of co-ordinated communication and teamwork amongst the thousands of maintainers and original program developers, a process that can survive any given individual's personal effort.

Aren't institutional archives already doing this?

No, they aren't. And they aren't at so many levels that this answer could easily be an article in its own right.

It is true that many institutions are starting archiving efforts, but merely archiving a software artwork is not the same as conserving it. An archived Software Art work is not running on a processor, it is merely stored in some mass storage media. It is resting, if not pining for the fjords. Geographical, technical, legal and economical hurdles preclude most people from access to those works, perfectly archived as they may be.

Even well-funded organisations face legal and economical challenges when trying to archive and make available the works they have the rights to. Access to these works for the world at large will still have to wait until they lapse into the Public Domain. We are mortals, and copyright terms nowadays are, by design, much longer than the average human's life. Archiving is not enough!.

What will this accomplish for Art and Culture?

Culture is that which is shared. Free Software encourages the spreading of ideas, the sharing of code and know-how, and the building on the base of other people's accomplishments.

Also remember that the practice of art is not just something artists do. Art historians and archivists also play a part, and Free Software Art allows them to keep the works alive in a usable state.

Right, Free Software is good for society as a whole, but what will it accomplish for artists?

Artists will accomplish more visibility for themselves by choosing Free licenses for their work. Free distributions will be able to package their work, and will do so either due to personal interest of individual developers, or with funding from museums and institutions.

Unless a software artists want to package their work themselves, packaging and funding is up to individual distro maintainers and curators. Artists can talk others into packaging their work into Free Software distributions as much as they can talk others into exhibiting their work: that hasn't changed much.

Won't this perpetuate the existing models if funded projects can get their projects packaged by paying?

Not really. Under the current model, funded artists get into museums and galleries, and their work gets out to people through their channels and the festival circuit. Unknown artists remain unknown unless they gain access to the insitutional scene and circuit. If artists release their work under Free licenses, it can get into Free Software distributions on equal footing with institutional-funded work. Free Software enhances the opportunities of experiencing all the Software Art in the world to a global audience, and the opportunities of having their work experienced by the world to all software artists.

Put it another way: although making Software Art Free won't reverse all inequalities, a world with a Free Software Art scene is a more level playing field for new, aspiring and unnafiliated artists than one in which there is no such scene.

Archiving is not enough!

Won't this get a lot of bad art/code into the archives or Free Software distro repositories?

Probably. But bad art abounds anyway. The focus is in getting all art conserved so historians and scholars of the future will be able to understand software art. The purpose of a Free-Software-Art project is not to make a canon, or to select «good» art and have that preserved. The purpose is to allow all Software Art to get produced, distributed and conservated, and let Art History sort the good from the bad.

Your examples are not the ones I would use! I know more canonical Software Art!

Earlier drafts of this paper used the word «*canonical*» to refer to my examples (see Terms and Conventions). By «*canonical*» I never intended to mean «belonging to an artistic-historical Canon». I meant «standing for all that share the same characteristics». The main characteristic of the software packages listed, apart from their Software-Art-ness, is that they are under a Free license, and they do not depend on non-free code.

Many works of art could be released under a Free license but still not be freely distributable due to their dependence on non-Free code. Some of them (hacks on commercial games, works where the absence of source is part of the artistic statement) will never be distributable, and we can live with that. This does not mean we don't like them as artworks, just that we can't include them in our packaging effort.

But Pure Data and Processing are not Software Art, they are tools!
Agreed, but artists working with Free code need Free tools and freely-distributable runtimes too. Free Software Art is not a purely artistic project, but rather a legal-technical intervention on the Political Economy of Software Art. Thus Pure Data and Processing are included as «source code that will inevitably be part of Free Software Art works», although they are not works of Software Art per se.

Why don't you do this project on Windows? I code on Windows! Linux is scary/difficult/not my cup of tea!
This project is currently at the proposal stage, and it is being proposed on Free Software distros because they are the ones that allow people to do things like distribute livecds that contain whole Software Art collections in them, or deploy unexpensive machines in schools without paying expensive per-machine licenses for the Operating System.

As to you and your code, you don't have to do anything you don't want to. This is the beauty of Free Software: if you free your code

for whichever platform you work on, you will allow Linux people to do the scary/difficult/not your cup of tea thing and port it to Linux. Or the Mac, or whatever. It really is a win/win situation.

Terms and conventions used in this paper:

For the purpose of this paper,

A Distribution is both a codebase and the people who develop and maintain it. A distribution project compiles a kernel (Linux, BSD, Hurd, OpenSolaris) plus tools and userland applications into an Operating System. This can be delivered in physical form (in CD form) or online, through special servers called repositories. Debian, Red Hat, and FreeBSD are all Free Software distributions according to this definition.

Free Software is that which can be freely used, copied, modified, and distributed in unchanged or modified form. Free here means «libre», not «gratis»: hence Free Software can be sold and still be Free as in free speech, although maybe not free as in free beer. The requisite of modificability of Free Software requires for its source code to be available. This is why some people prefer to call it by the name Open Source Software.

Free Software and Open Source are, outside very extreme edge cases, functional synonyms, as there are virtually no Open Source programs that are not Free according to either Debian or the Free Software Foundation, and there are hardly any Free Software programs that are not Open Source. I use «Free Software» or «Software Libre» throughout, but if you want to use «Open Source», it is mostly ok, as long as you mean Open Source to carry the connotations of «free to modify and distribute» as well as «has available source code».

Software Art is whatever you want to make of it, but for the purpose of this paper and for Software Art packaging efforts, «software art»means «code that compiles into aesthetic objects, into artistic

performance tools, or into tools specifically designed and promoted for artists creating the previous two categories». Free-Software-Art is a software packaging effort, so it deals with software which can be distributed and run on generic hardware.

What is and is not Software Art is fuzzy, as many categories are not binary, but rather points in a continuum. The following examples stake out some territory of Software Art by using examples whose licenses are free and depend only on free code:

Category	*Example*
Standalone applications	Electric Sheep (distributed generative screensaver)
Client-server applications	Carnivore (online surveillance tool)
Performance instruments	Fluxus (scheme-based GPL visuals livecoding tool)
Art games and game mods	Rrootage, Transcend, Fijuu
Coding Platforms *	Processing, Pure Data (code that compiles into aesthetic objects)
Social Web Services	Everything2

Packaging means preparing a composite of a program's compiled binary (or its data) plus the appropiate metadata in a format that can be automatically installed by an operating system. As our project is about, we will talk about packaging for Free Software distributions such as GNU/Linux, the BSD family, OpenVMS or your own homecrafted one. Red hat .rpm files are packages, and so are Debian .deb files. BSD packages are called ports, but for the purpose of this talk they are also packages, and the process of making a raw source code tarball into a port will be called «packaging».

* Not Software Art per se; included as coding platforms specifically designed for and aimed at Software Artists

1 Julian Oliver aka Delire, the factotum of Art Gaming website Selectparks.net,has this to say about Free Software as an enabler for software artists: «*artists wanting to sell work to museums and/or have work shown in museums/galleries have hit a legal 'glass ceiling' due to the issue of IP*». http://games.slashdot.org/comments.pl?sid=158904&cid=13310740

2 Or try to fruitlessly in the absence of source code.

3 At Ars Electronica 2005, during the Digital Archives Conference, the representative from Medienkunstnetz.de Rudolf Frieling talked about their project having taken «*3 years, which is not long if you take into account the copyright issues*», and Matt Locke, from the BBC archive, said that their project had taken «*2 years because of negotiations with rightholders*».