The Current State of Linux Video Production Software and its Developer Community

Richard Spindler
Schönau 75
6323, Bad Häring, Austria
richard.spindler@gmail.com

Abstract

According to Linux and Open Source/Free Software thought leaders, the current situation in Linux video editing and production software still leaves a lot to be desired. Here we give an accurate overview of the software that is already being developed for Linux video editing, or that is currently being worked on. We will introduce the developers or/and the development communities that are behind these projects, and how collaboration between projects works. We will mention interesting events and places where Linux video developers meet, and what groups might be interested in our work, or ought to be involved into specific aspects of our work. We will outline where projects are still lacking to support specific use cases, and why they matter. Furthermore, we will discuss open problems, as well as opportunities for participation and collaboration, and I will make some predictions about the future, based on the work that was already done, and that is currently being tackled.

Keywords

Linux, Video, Community, Developers.

1 Introduction

The author of this paper has been working on a Linux/Open Source/Free Software video editing application called “Open Movie Editor” for the last few years. Originally the project was started, because the author felt that the software available for working creatively with video under Linux was lacking some aspects that he considered important for his work, and the kind of work that he wanted to support.

At the time it was started, there were two potential applications available that could have been used instead of starting a new project. Those two options were Cinelerra[1] and Kino[2] which were already quite mature and widely used at that time. However for the needs of the author, Kino was too simple to support the kind of workflow that he anticipated, and Cinelerra was too complex to teach quickly to his collaborators. They were not the only choices, as new editing applications emerged, likely because other Free Software enthusiasts were finding themselves in a similar dilemma. Those however, were also very much at a starting point of development, and far from being mature enough for general use.

Over those years, the situation improved quite rapidly, with different projects and communities working on applications to fill a wide variety of user preferences. Also with the continued progress, the awareness in the broader Free Software community for the need of user friendly video editing software was rising tremendously, giving those projects and communities the exposure they need to flourish. For example the lead of the Ubuntu Linux distribution Mark Shuttleworth mentioned the lack of Linux video tools, and it is also an entry in the Free Software Foundations Priority List[3].

Actually for video processing software in Linux and Open Source there are quite active and very successful projects, but those are mainly concerned with encoding, processing and playback issues, and the authoring and creative work associated

[1] http://heroinewarrior.com/cinelerra.php

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

[3] http://www.fsf.org/campaigns/priority.html

with non-linear video editing is far beyond their agendas. However, the availability of software libraries, like ffmpeg[4], gstreamer[5], libquicktime[6] and many many more are a great and vital starting point, for creating a non-linear video editing application, and all of the available efforts are based on those projects, and ultimately those foundational libraries enable the high-level projects to make the great progress that they are doing.

And very much like the diversity of the available support libraries, the active projects that are pushing the agenda of bringing video editing to the Linux using masses, are quite diverse. The projects, while similar in their goals, are using quite different means to achieve their objectives. While this may appear as unnecessary fragmentation of efforts, for which projects are occasionally criticized in public forums, it is necessary and positive. This of course is a personal opinion of the author, which is very much based on his experiences working on a video editor over the last few years. Writing a video editor is not that difficult, and the low communication overhead that a very small team or individual developer has, can achieve much higher efficiency and development speed. Especially for young projects, where the roles of team members and developers are still unclear and the optimal division of labor is not yet worked out.

However, even when working on projects that appear to be competing, it can be quite beneficial to share knowledge and experiences between project members, and this is what the author started to work on in the last year, and this is also what he is planning to continue and intensify in the future.

2 Foundations

In the Open Source and Free Software community, there has always been quite a vivid interest in a lot of problems that are closely related to issues that are also relevant in video editing. Starting with the Gimp, and ImageMagick for image processing, and video compression technology that became widely interesting with the advent of the MPlayer[7] project, and its associated communities, and also animation where Blender is likely the most widely known example. But there are also forays into territory that is more exotic, citing FreeFrame[8] as an example for an Open Source video effect plugin standard, that is enjoying great enthusiasm and adoption in the VJing community. There is EffectTV, which provides wacky realtime video effects that cover a wide range of creative applications, and there are modular approaches that aim to provide a standard infrastructure and a platform for the development of video based applications as well as a possibility to easily add video to any kind of application that could make good use of it.

The author has been following most of those foundational projects and their communities as close as he could afford, because he is convinced that code reuse will be the key to getting Linux video editing to artists and users as fast as possible.

One of the key projects in the area of foundational libraries became *ffmpeg* over the last years. Currently it is likely one of the most valuable toolboxes for the aspiring video developer. The project aims to develop the most complete set of video compression decoders and encoders, and promotes being self contained to achieve the most efficient means of internal code reuse. While there always were specific development libraries that focused on a specific encoding or decoding technology, or a family of codecs, during the development of ffmpeg it occurred, that many video codecs that appear to use different technologies are actually quite similar internally, and source code and routines for basic operations and inner loops could be reused over a wide range of codecs. And this fact remaned pretty much a mantra for the development of ffmpeg.

Even if a user is not aware that he is using ffmpeg, it is likely that if he has any kind of video related Open Source software installed, it is based on ffmpeg. Video players, media frameworks, tools and applications, there is simple nothing that can do without. Even the more high-level frameworks for creating video applications, like Gstreamer, Gmerlin and MLT wrap ffmpeg.

[4] http://ffmpeg.mplayerhq.hu/

[5] http://www.gstreamer.net/

[6] http://libquicktime.sourceforge.net/

[7] http://www.mplayerhq.hu/

[8] http://freeframe.sourceforge.net/

The selection of a high-level video framework is actually one of the distinguishing features that characterizes the different non-linear video editing applications. PiTiVi[9] for example choose to use Gstreamer, a framework that is built upon the Gobject system in C, and was originally intended to become an universal open media framework with pluggable modules, similar to what is available on non-free software platforms. While the framework was never exercised to the level that PiTiVi needs, it is currently being extended and fixed for the sometimes more difficult interactions that are required for non-linear editing. Before, Gstreamer was used in the realm of media playback and streaming, and accurate cuts and compositions were never needed. With the introduction of PiTiVi, that has changed, and as a matter of fact, this work also benefits the framework itself.

Quite the opposite is the case with MLT, which is the foundation of Kdenlive[10] a video editor that is developed within the KDE framework. MLT itself originated from an open source project that was commissioned by a broadcast enterprise from India, where it powered playout servers for TV programs. MLT already supported complex time-lines and composites, before it was adopted for Kdenlive, so it was quite a good match.

Gmerlin on the other hand uses quite a different metaphor, as it is more of a toolbox with very different possibilities for adoption, and as far as the author knows, some of its design was inspired by *libquicktime*, which in turn originated from Cinelerra. Gmerlin provides a low level library that contains basic data structures for uncompressed video and audio frames, functions and converters to translate between different formats and framesizes, etc. However, these low level functions can be used independently from the more high level components, like the Gmerlin avdecoder, which provides access to compressed video files and streams. Also Gmerlin provides access to a wide variety of video filters, often wrappers around third party effects that are made available under a common API. Of course, Gstreamer and MLT do so too, so all the frameworks mostly differentiate in terms of usage style, and not so much in functionality, even though sometimes the focus is quite different, with Gmerlin being more and more pushed towards its use for video production applications.

3 Applications

Cinelerra, the oldest and most complete application that comes close to the ideal of an universal non-linear editing software, is still under active development today. And while it requires some effort to get started with, due to its complexity, it is actually known for leaving little to be desired, its productive feature set is impressive, and very complete. Its developer, or development team however favors more of a “lone wolf” style of working, so there is little public discussion about the direction and vision of it, much to the dismay of interested users and potential contributing developers. However, there is a community around Cinelerra, that consist of active users, and several developers, that keep their own branch of Cinelerra, with patches, bugfixes, releases and repositories. Also Cinelerra-CV[11] as this third party community calls itself provides support and documentation around the Software. Additionally, rising from the Cinelerra community, a project was formed, that is relatively new, which tries to recreate the functionality from scratch, using a development process that is very much focused on best practices and community involvement. This project is called *Lumiera*[12].

Right now the state of Lumiera is that it is not yet usable, however, it is making significant progress, both in source code, user interface, and documentation. As the author is an outsider to the project he cannot make any accurate estimates about its future progress, but he is definitely looking forward to its first release.

Kino started out as a tool purely focused on editing with the then widely used DV format, and pretty much stayed within that objective. Today its development has mostly halted apart from bug fixes and minor improvements. Its stability and maturity however is likely unmatched by any of the runner ups. On the other hand, the user interface metaphors used are slightly clunky, and while a new user might need some work to get

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

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

[11] http://cvs.cinelerra.org/

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

used to the concepts, the limits of the approach are quickly reached as well. Simple multi-layer or track based editing is not possible, and a quite rigid workflow is imposed.

Those were the oldtimers, now on to the newcomers, that are apart from Lumiera, *Kdenlive*, *PiTiVi* and *Open Movie Editor*. There are some more, that will be mentioned later for the sake of completeness, but those are the ones that are the most actively developed, while others are either appearing abandoned or aimed at uses that differ from the idea of a basic non-linear video editor.

Kdenlive is the application that is based on MLT, and with a former Kino developer working on MLT, it has quite a solid foundation, and is making significant progress. It is already in a usable state, and users claim to be able to achieve decent results. Little is known about the goals, visions and lifes of its developers, at least as far as for publicly available information from blogs or mailinglists. The direction of Kdenlive is going more for the advanced user, with features that are known from many similar tools in the proprietary space of video applications.

PiTiVi is the video editor project that works with Gstreamer, therefore it gathers quite some mindshare from the development community around Gstreamer, but also the Gnome project appears to be interested and emotionally involved into its development and success. While aside from Kino all the other applications aim to cater to the more ambitious user, PiTiVi seems to focus its energy towards the casual user, and goes for a simplified user interface. There are a number of small development consulting companies that are part of the Gstreamer community, and one or several lead PiTiVi developers were hired to work on the application. Also some of its developers keep public blogs, where they occasionally provide interesting insights into the progress of the project.

The Open Movie Editor is the authors project, so he[13] has more initmate insight into its state of development than he has for the other applications. In some cases Open Movie Editor is the odd one out, because some development decisions are not widely appreciated, for example choosing a niche user-interface toolkit, rather than going for one of the big two, Qt and gtk+. But never the less, the project is quite ambitious, and considering that there is only one developer working on it, the progrss is quite satisfactorily. However, it has to be pointed out that the author works closely together with Burkhard Plaum, who maintains the Gmerlin project, on which significant parts of Open Movie Editor are based. The responsibilities are strictly separated though, so there is only one person working on Open Movie Editor. Regarding its projected audience, Open Movie Editor tries to go for a hybrid approach, with a very simple, easy to use interface, while providing more advanced features through plugins and an extensible node editor.

[13]Richard Spindler

4 Communications

Every one of the afore mentioned projects, with the exception of Cinelerra upstream, provides mailinglists, and/or forums to assist its users, or aspiring contributing developers with getting up to speed with the project. However, between the projects, even though they are all working on closely related fields, very little communication happens, and therefore the wheel is reinvented quite regularly. In general this is not a problem, and differing requirements, philosophies and abstractions make it necessary to deviate a bit sometimes. For issues however, that can be tackled collectively, the author aimed to create a channel, like a backdoor to the minds of the different projects. This idea was manifested as a mailinglist called libopenvideo. After the creation of the mailinglist members of all the non-linear video editing projects were approached, and invited to join the list. It was explaining to them how this idea expected to explore opportunities for collaboration. It did work quite well, and some enlightening discussions ensured that significant progress for all involved projects happend, at least according to the authors impression.

Of course, there is much more potential for increasing the amount of collaboration, and eventually this will bring another boost in Linux video editor development in the next year. The progress that was seen this year was quite impressive already.

Also, it remains open to establish contact to groups and individuals that are outside of the

immediate context of video editor development, but whose results and work is essential to the success and completeness of the work of the non-linear editor developers. Those groups range from developers of common Linux audio APIs, to webcam and video4linux developers, OpenGL/Mesa/DRI groups, input devices such as dedicated video consoles and jog-wheels, developers of image processing/computer vision algorithms, but also downstream packagers from Linux distributions. Codec developers could be involved as well, especially in regards to the development of intermediate video codecs for production workflows.

5 Issues

These ideas about collaboration and related groups, leads to the last topic for this paper. A selection of challenges that the author thinks the Linux video editor developers will face during this year.

The first would be the refactoring of rendering and compositing architectures to use programmable shaders in OpenGL 3D Hardware. While it is possible and works well to do any kind of image processing and compositing in software on the CPU, pushing some work to the GPU[14], can increase the throughput considerable, especially in the case of high definition video. The development overhead on the other hand increases, due to the more complicated interaction with the GPU. It is necessary to upload shader programs that are written in a C variant, and video textures, and then use the OpenGL API to interact with the data on the GPU, which is quite a different experience for the developer who is accustomed to doing work on the CPU. However, the OpenGL API is very much designed for doing 3D visualization work, while the needs for video image processing are much simpler, a convenient subset of helper functions or an abstraction layer might come in helpful to tackle this issue. Some work is already done on that front though, for example there are already OpenGL based video elements in the collection of Gstreamer plugins, and also the FreeFrame filter API defines OpenGL based video filters, and offers a number of example plugins. And there are already open source video applications that use OpenGL for image processing and color space conversion, see DJV[15] and yuvtools[16]. Of course there are many more issues and opportunities connected to this field, and there are also proprietary APIs and projects that could be interesting for inspiration.

Another issue, that is doing quite well, is the compatibility with video codecs that are widely used in camcorder and video camera hardware. However, during the last year interesting work was done by the ffmpeg, Gmerlin and Gstreamer projects, so this issue is largely solved, although the work in that field continues. Also for using video files in certain codecs “natively” in an application, it is necessary to support frame accurate seeking, which can be a problem with codecs as MPEG, which divide the stream into GOPs, groups of pictures, however there is lots of progress with handling this in the afore mentioned projects.

An open problem is still interaction with capture hardware, even though there is plenty of Linux based software and drivers, the space is very fragmented. There is an API that wraps different video4linux variants, and provides some additional means of interacting with firewire cameras, there is Gmerlin that provides some kind of unified API for certain webcams, there is of course Gstreamer, that does some work in that field too, and there is Kino and dvgrab that can interact with firewire based DV camcorders, and there are the video4linux APIs for interaction with TV-cards. However, there is no real simple, easy to use, unified, low level capture API, so implementing capturing for a wide range of devices in one application requires still significant effort, and testing. Also screen capturing uses again quite different mechanisms, the xvidcap[17] application does this quite well though for example.

As for capture hardware, the author believes that there is quite some knowledge about that in the community of embedded Linux based set-top-box developers, as well as media center developers. These communities are believed to be quite large, certainly larger than the community of non-linear

[14] Graphics Processing Unit

[15] http://djv.sourceforge.net/

[16] http://yuvtools.sourceforge.net/

[17] http://xvidcap.sourceforge.net/

video editor developers, so this is another possibility to go for interesting collaborations.

The opposite problem, monitoring and playout of video streams, is another issues, that lacks any kind of standard solution. As far as the author knows the only application that is capable of doing this is again Kino, which implements monitoring with DV over firewire. Other than that, it likely requires expensive dedicated hardware to do monitoring, so it is not widely used. Eventually with the advent of HDMI and DisplayPort, the need for dedicated monitoring hardware will diminish, and this problem will solve itself. However, the TV-out port that many graphics cards feature is not really suitable for doing serious video monitoring, because it most often scales and processes the image, as the hardware and drivers are optimized to display a desktop on the TV, rather than providing an accurate video monitor.

The authoring of media for distribution, for example DVDs, is another issue that is still problematic, even though plenty of software exists. The issue though is that hardly any kind of tools are available to test a finished project for conformance to DVD and MPEG specifications. So any kind of quality control requires tedious manual testing. Also tests for compression quality and comparisons of codecs and codec settings are largely not automated. Any kind of serious production therefore requires much more intimate knowledge of standards, pitfalls, tools and process, than actually necessary, if the right tools were available. There is tremendous potential for improvements.

For video production workflows, it is often necessary to have a high quality, high bandwidth codec, for intermediate renderings between worksteps. Codecs that provide higher resolution colorspaces than currently associated with common MPEG codecs for example. A topic that is largely neglected by popular open source/free software codecs, which are more engineered towards final distribution of media. There are some completely uncompressed codecs in libquicktime though, and there are the YUV4MPEG pipes, but those solutions are not appropriate for all use cases, as uncompressed video uses much more disk space than eventually necessary, and pipes are only useful for immediate processing. However, with the advent of the Dirac[18] codec family, there is some potential that this issue will be resolved this year. Dirac features different profiles, also high quality ones that aim at minimizing multi generational loss of quality. Currently Dirac seems to be still slow, being largely unoptimized, but it appears as if the Schroedinger implementation of the dirac codecs is making progress in this area.

For compositing and special effects, it is often necessary to provide some kind of motion tracking, and as far as I know, the computer vision community is the most interesting group to collaborate with in this respect. I even think that the OpenCV library has many interesting algorithms, that just wait for being ported into Linux video editors. Also face detection as found in OpenCV, can be useful for anonymization of people in documentaries and reports, for example.

6 Summary

This paper gave a hopefully complete overview about the most interesting developments in open source non-linear video editing under Linux. It highlighted achivements in this field, and gave an outlook into the future, and interesting problems to work on. It gave a rough idea about how inter-project communications happens, and pointed out potential for improving synergies.

[18] http://diracvideo.org/

7 Appendix

7.1 List of events

The following list was compiled to give an overview about interesting events and meeting, that could be helpful to establish contacts to groups and developers that are concerned with similar and related problems that we have in Linux video editing. Also there could be potential common interests and opportunities for collaboration.

- **Piksel**
  http://www.piksel.no/
  A festival for electronic arts and open source software.
- **Libre Graphics Meeting**
  http://www.libregraphicsmeeting.org/
  A developer meeting for open source graphics software, image editing, vector graphics, animation, etc.
- **Foundations of Open Media Software, FOMS**
  http://www.foms-workshop.org/
  A developer meeting for all kinds of media, audio and video related open source software, very wide range of interested parties.
- **Linux Plumbers Conf**
  http://linuxplumbersconf.org/
  A meeting for issues concerning low-level interoperability with the Linux kernel, interfaces to hardware and drivers, etc.
- **Linux Audio Conference**
  http://lac.linuxaudio.org/
  A conference about open source audio software, with a focus on production, recording, composition, creative arts, etc.

7.2 List of video editing applications

- **Cinelerra**
  Status: *under active development*
  Homepage: *http://heroinewarrior.com/cinelerra.php*
  Community Branch: *http://cinelerra.org/*
- **Lumiera**
  Status: *under active development*
  Homepage: *http://lumiera.org/*
- **Kdenlive**
  Status: *under active development*
  Homepage: *http://kdenlive.org/*
- **PiTiVi**
  Status: *under active development*
  Homepage: *http://www.pitivi.org/*
- **Open Movie Editor**
  Status: *under active development*
  Homepage: *http://www.openmovieeditor.org/*
- **LiVES**
  Status: *under active development*
  Homepage: *http://lives.sourceforge.net/*
  Comment: *focus on realtime video mixing, VJing*
- **Vivia**
  Status: *likely abandoned?*
  Homepage: *http://vivia-video.org/*
- **Scilab Aurora**
  Status: *likely abandoned*
  Homepage: *http://scilab-aurora.sourceforge.net/English/index.html*
- **Avidemux**
  Status: *under active development*
  Homepage: *http://avidemux.sourceforge.net/*
  Commant: *focus on transcoding and processing, only simple editing*
- **Kino**
  Status: *in maintenance mode*
  Homepage: *http://www.kinodv.org/*
  Comment: *only simple editing, very mature*
- **Blender**
  Status: *under active development*
  Homepage: *http://www.blender.org/*
  Comment: *an animation all in one toolbox, has video editing and advanced compositing built in.*
- **Diva**
  Status: *abandoned*

Homepage: ~~*http://www.diva-project.org/*~~
(not valid anymore)

- **Pihlaja**
  Status: *on hold*
  Homepage: *http://pihlaja.sourceforge.net/*
- **Saya-VE**
  Status: *stumbling?*
  Homepage:
  http://sayavideoeditor.sourceforge.net/

7.3 List of Media Frameworks

- **Gstreamer**
  http://www.gstreamer.net/
- **Gmerlin**
  http://gmerlin.sourceforge.net/
- **FreeFrame**
  http://freeframe.sourceforge.net/
- **Frei0r**
  http://www.piksel.org/frei0r
- **MathMap**
  http://www.complang.tuwien.ac.at/schani/mathmap/
- **EffecTV**
  http://effectv.sourceforge.net/