Mistelix

A DVD authoring tool for GNU/Linux desktop systems

5th January 2009

Student: Jordi Mas i Hernàndez

Tutor: Jordi Ceballos Villach

This work is licensed under

Cretieve Common's Attribution-Noncommercial-No Derivative Works 2.5 Spain

The complete license is available at

http://creativecommons.org/licenses/by-nc-nd/2.5/es/deed.en

The source code of the project at:

http://code.google.com/p/mistelix/source/browse/

This project is dedicated to the hundreds of free and open source hackers that invest their time in making all the infrastructure that made this project possible.

I want also to thank to the UOC their flexibility given to me to develop a project that is based on Linux (instead of Windows), that runs of top of Mono (instead of Microsoft .Net), that is written in English and that its outputs are open source.

Table of contents

1. Introduction

1.1 Introduction

This document describes Jordi Mas i Hernàndez final year project proposal as computer science master degree student at the Open University of Catalonia (UOC[1]).

The project is called Mistelix, an open source DVD authoring tool for GNU/Linux systems.

1.2 Objectives and scope

The objective of this project is to create a basic fully functional DVD authoring tool for GNU/Linux systems.

Mistelix project objectives in the **scope of the final year project are**:

- Build an application based on GTK that allows to create easily to any novice user DVDs in Linux systems, focusing on producing two types of DVD:
    - A DVD with slides of images (slideshows).
    - A DVD with videos.
- Provide the gluing to allow Mistelix to operate audio and video codecs and muxers for DVD authoring.
- Deliver a platform that the user can extend using plug-ins:
    - Transitions for slideshows (provide some effects as reference implementation).
    - Themes support
    - To connect to external sources to import content from other sources, like F-Spot[2] (a client side application) or web based systems like Flickr[3].
- Provide project management functions for creating the DVD projects that are stored locally.
- Integration with dvd-author and spumux for DVD file system production.
- Be localizable to any language (using Gettext).

The scope of the project within the UOC final project is limited by the time frame of the project (starting in early September 2008 and finishing in early January 2008). However there are plans beyond the scope of this project to keep working on Mistelix. These plans are described in the section **Further development and plans** section of this document.

1.3 Open source

Mistelix is developed in an open manner using Google Code hosting[4] and its software deliverables

1 http://www.uoc.edu
2 http://f-spot.org/
3 http://www.flickr.com/
4 http://code.google.com/hosting/

are published under the Mozilla Public License (MPL) [5] open source license. The MPL is the license used for the Mozilla Application Suite, Mozilla Firefox, Mozilla Thunderbird and other Mozilla free software.

Unlike other open source licenses, the code under the MPL may be combined with proprietary files in one program. This even allows to build closed source commercial software derivated from MPL licensed applications.

The MPL license has been certified as open source by the Open Solution Initiative[6] and as a free software license by the Free Software Foundation[7].

1.4 Project justification

This project meets the academia requirements as a final year project:

- It represents a complete software project, covering a full development cycle, including functional description, architecture, implementation or the deployment among users.
- The project outputs are open source and pubically available which provides a benefit for the society, other students and GNU/Linux users.
- The project uses a broad number of technologies:
    - Build system tools (automake)
    - GTK and Glade for user interface building
    - Systems level program (C library) and high level (using C#)
    - Video and audio encoding and muxing (Gstreamer and ffmpeg libraries)
    - Extensibility (using Mono.Addins) and translation friendly (Gettext)
- The project will continue after the formal ending of the final year project period and we will be further developed by his author and other open source hackers.

1.5 Project deliverables

This project has produced the following deliverables:

- A description of the architecture of the solution.
- A build system based on Autoconf that builds the application (218 lines of source code).
- Mistelix, the main application written in C#, Glade and GTK (6.058 lines of source code).
- libmistelix, a C library that glues Mistelix application with the underlying codec infrastructure to encode video and audio (431 lines of C source code).

5 http://en.wikipedia.org/wiki/Mozilla_Public_License
6 http://www.opensource.org/
7 http://www.fsf.org

- Mistelixvideosrc a Gstreamer plug-in that receives a list of images from the Mistelix application and puts them on the Gstreamer pipeline to allow them to be processed by the Gstreamer infrastructure (1285 lines of C source code)
- Two effect transition extensions (from slide to slide). Written in C#.
- Three default themes for creating DVDs.

1.6 Project risks

Mistelix is already free software[8]. However, some risks were identified when the project was started that could affect to its potential distribution have to be cleared before it can be widely distributed.

Risk: Software patents	
Description	A patent is the grant of a property right covering an invention. The right conferred by the patent gives its owner the right to exclude others from making, using, offering for sale, selling or importing the invention in the absence of a license. In some cases, standards and other technology platforms consist of many patents owned by many patent owners, the number of licenses required of users may be too costly and inefficient for users to negotiate. This is often referred to as a patent thicket. MPEG2 video format is heavily protected by patents, actually some sources have counted up 640 to patents. These patents cover different aspects of the video, audio encoding, decoding, transmission, recording and retrieval. The patent holders are companies like Alcatel-Lucent, Canon Inc., Columbia University, Fujitsu, LG Electronics, Matsushita, Mitsubishi, Philips. There is a full list[9] of patents covering the MPEG2 standard. To simplify the licensing process, a patent licensing authority for MPEG2 has established. The MPEG Licensing Authority[10] licenses the patents related to MPEG2 technologies. This licensing authority keeps a list of the large portfolio of patents[11] that they manage. Usually vendors implementing MPEG2 (set-top boxes, DVD players, television receivers/decoders, software, etc) software to encode and decode MPEG2 format pay around $2.5 (see licensing summary[12]) for every customer that uses MPEG2 related technologies. These $2.5 are patent royalties are paid to the MPEG Licensing

8 http://code.google.com/p/mistelix/
9 http://en.wikipedia.org/wiki/MPEG-2#Patent_holders
10 http://www.mpegla.com/
11 http://www.mpegla.com/m2/m2-att1.pdf
12 http://www.mpegla.com/m2/m2web_licenseterms.ppt

	Authority. **These patent royalties causes MPEG2 to be not compatible with the distribution of free software.** Most of the Linux distributions have currently no support for MPEG2. See as examples Fedora Linux distribution policy[13] or the OpenSuse Linux format restrictions page[14]. Companies like Fluendo for example sell codecs[15] for MPEG for end users or 7 euros, however they do not produce MPEG2 encoders that can be used for DVD authoring. In the section **Legal Strategy** of this document, I described the strategy defined to mitigate the patent risks.

Risk: DVD format restrictions	
Description	DVD Video format uses a very strict subset of the MPEG2 format. DVD authoring tools that do not stick to these restrictions generate DVDs that users not be able to reproduced in all DVD players. The restrictions affect areas like: • The video resolutions supported for encoding. • The video aspect rations supported. • The video and audio encoding mechanisms. Not all the MPEG2 encoded files produced can be used for recording DVDs. They need special characteristics that a libraries systems support.
Action	Make sure that the libraries selected produce MPEG2 that can play properly in most DVD players.

Risk: High dependency on third party modules	
Description	Mistelix builds up in a set of libraries and components that are largely used in the open source space, like Mono, GTK, Glade, Mono.Addins, autotools, Cairo. However, libraries like:

13 http://fedoraproject.org/wiki/Multimedia/DVD
14 http://en.opensuse.org/Restricted_Formats
15 https://shop.fluendo.com/

	• dvdauthor or spmux • ffmpeg • Gstreamer are somehow redundant and some of them have to be drop as the project progresses and we shoud consolidate different needs in a single library. Some of these libraries have different conditions for their distribution. It is important that they are compatible with Mistelix license and patent strategy.
Action	Try to minimize the number of libraries used before releasing the software.

1.7 Project plan

In this section I describe the tasks that have been performed to complete the project. All these tasks have been synchronized with the deliveries required by UOC in order to evaluate the progress of the project.

In the next table there is the **high level task plan and schedule for the project**.

		Task name	Duration	Start	End	Predecessor
1						
2		**Analysis**	**22 days**	**Mon 01/09/08**	**Tue 30/09/08**	
3		Project proposal	15 days	Mon 01/09/08	Fri 19/09/08	
4		Project at Google Forge	1 day	Mon 19/09/08	Fri 19/09/08	
5		Project plan	5 days	Mon 22/09/08	Sat 27/09/08	4
6		PAC1 completed	0 days	Tue 30/09/08	Tue 30/09/08	5,3
7		**Prototypes**	**19 days**	**Wed 01/10/09**	**Mon 27/10/08**	
8		Functional specifications	5 days	Wed 01/10/08	Tue 07/10/08	6
9		Architecture description	5 days	Wed 08/10/08	Tue 14/10/08	8
10		User interface mock-ups	5 days	Wed 15/10/08	Tue 21/10/08	9
11		PAC2 completed	0 days	Mon 27/10/08	Mon 27/10/08	10
12		**Development**	**60 days**	**Mon 22/09/08**	**Fri 12/12/08**	
13		Build and dependency system	60 days	Mon 22/09/08	Fri 12/12/08	
14		Mistelix	60 days	Mon 22/09/08	Fri 12/12/08	
15		Lib Mistelix	60 days	Mon 22/09/08	Fri 12/12/08	
16		gstreamer plug-in	60 days	Mon 22/09/08	Fri 12/12/08	
17		PAC3 completed	0 days	Fri 12/12/08	Fri 12/12/08	13,14,15,16
18		**Report and presentation**	**35 days**	**Mon 15/12/08**	**Fri 30/01/08**	
19		Build final report	10 days	Mon 15/12/08	Fri 26/12/08	
20		Prepare video presentation	5 days	Mon 15/12/08	Fri 19/12/08	
21		Final delivery	1 days	Wed 07/01/08	Wed 07/01/08	19,2
22		Class group discussion	5 days	Mon 27/01/08	Fri 30/01/08	

During the t**he analysis phase** started on the 1st of September the following tasks were executed:

Task name	Description
Project proposal	After evaluating many different ideas for the final year project I decided to do a DVD authoring tool for GNU/Linux systems. During this phase I crafted a proposal for my tutor where I exposed the idea of this project, the scope and main milestones. The project was accepted at the end of September 2008.
Project at Google Forge	I registered Mistelix project in the Google Forge collaborative web

	platform. This made possible to use Subversion source control system since the beginning of the project.
Project plan	A project plan is created taking into account deliverables required by UOC in order to evaluate the progress of the project and making sure that the scope of the project could be accomplish during the given time frame.

During the **prototype phase** started on the 1st of October the following tasks were executed:

Task name	Description
Functional specifications	I focused on doing the functional specifications for the application by analyzing the features offered by some products in other platforms. Afterwards, I wrote some use cases that capture what I defined as the minimum set of requirements that the application must accomplish.
Architecture description	Once the requirements were defined, I focused on defining the architecture. This task included analyzing the different libraries and external tools available and selecting the best ones for the project.
User interface mock-ups	I crafted the mock-ups for the user interface, that is also one of the key aspects of the project since it targets end-users. The mock-ups take advantage of the capabilities of user interface libraries used. It is strongly based on a single document and drag and drop approach,

During the **development phase** started on the 22nd of November the following tasks were executed:

Task name	Description
Build and dependency system	This includes configuring the automake and autoconf systems for the project and building the configure.in[16] file for Mistelix. In this file all the dependencies for the application are defined, such as the Mono environment, the different libraries (gtk-sharp, mono-addins, etc) and the external tools required.
Mistelix	This the development of the main application done using C#. This includes four main areas: • Core: main application classes • Backends: classes that interface with external tools

16 http://code.google.com/p/mistelix/source/browse/trunk/configure.in

	• Datamodel: classes that define the application data model • Dialog: user interface dialog classes • Widgets: application user interface widgets
Libmistelix	libmistelix is a C library that glues the Mistelix application with the underlying codec infrastructure provided by GStreamer. The objectives of libmistelix are to make ease to replace the multimedia backends in the future if patents problems arise or if we decide to use another multimedia backend different to GStreamer
Gstreamer plug-in	To make the communication possible with Gstreamer, Mistelix provides a GStreamer plug-in written in C called mistelixvideosrc. The objective of this plug-in is to receive a list of images from the Mistelix applications and put them on the Gstreamer pipeline to allow them to be processed by the Gstreamer infrastructure (e.g converting it to video).

During the **report and presentation** started on the 15th of December the following tasks were executed:

Task name	Description
Build final report	Write this near 60 page document that is a complete report for the project that covers all the phases of the project from its inception to its delivery.
Prepare video presentation	Record the video presentation that explains with slideshows the main objectives, milestones and conclusions of the project with also a recorded demo of the application.
Final delivery	To deliver the final outputs of the project to UOC for the final evaluation.
Class group discussion	Participate in the class discussion to solve any potential doubt about this project.

1.8 Installation

Mistelix requires a GNU/Linux system, like Ubuntu[17], Fedora[18], OpenSuse[19] or other modern GNU/Linux distribution, with the following packages installed:

- Mono 1.1.7 or higher
- GTK sharp 2.8 or higher
- gnome-sharp 2.8 or higher
- glade-sharp 2.8 or higher
- gstreamer-0.10 version 10.3 or higher
- gstreamer-base-0.10 10.3 or higher
- gstreamer-plugins-base-0.10 10.3 or higher
- mono-addins 0.3 or higher
- mono-addins-setup 0.3 or higher
- mono-addins-gui 0.3 or higher
- dvdauthor 0.6 or higher

In RPM based systems like Fedora o Suse you can use the command line rpm[20] to install packages and in Ubuntu and Debian systems you can use apt-get[21].

Additionally, the gst-ffmpeg[22] GStreamer plug-in has to be built from sources. This due to two issues:

1) Most of the standard gst-ffmpeg packages distributed with Linux have the MPEG2 codecs (required by Mistelix) disabled to avoid patents claims in countries were patents are enforceable.

See for example how in the standard *gst-ffmpeg* Debian package[23] they disable H26, mpeg2video and MPEG4 codecs in the following script fragment:

```
# strip/clean the code from potentially dangerous patented code
for codec in 'h26.*' mpeg2video mpeg4 'msmpeg4.*'; do
   F=libavcodec/allcodecs.c
```

17 http://www.ubuntu.com/
18 http://fedoraproject.org/
19 http://www.opensuse.org
20 http://linux.die.net/man/8/rpm
21 http://linux.die.net/man/8/apt-get
22 http://gstreamer.freedesktop.org/modules/gst-ffmpeg.html
23 http://archive.ubuntu.com/ubuntu/pool/main/f/ffmpeg-debian/ffmpeg-debian_0.svn20080206-12ubuntu3.diff.gz

```
   sed -i "/REGISTER_ENCODER.*\\<$codec\\>/d" $F
   sed -i "s/REGISTER_ENCDEC\\(.*\\<$codec\\>\\)/REGISTER_DECODER\\1/" $F
   F=libavcodec/*.c
   sed -i "/AVCodec *${codec}_encoder *=/,/^[[:space:]]*}/d" $F
done
```

2) The *gst-ffmpeg* plugin does not expose all the muxers available in the ffmpeg backend needed by Mistelix. More precisely, the *ffmux_dvd muxer*.

Before building gst-ffmpeg from sources the following patch should be applied:

```
diff -u -r1.172 gstffmpegcodecmap.c
--- ext/ffmpeg/gstffmpegcodecmap.c7 Nov 2008 11:43:42 -0000   1.172
+++ ext/ffmpeg/gstffmpegcodecmap.c         3 Dec 2008 10:00:09 -0000
@@ -2117,7 +2117,7 @@

     *video_codec_list = mp4_video_list;
     *audio_codec_list = mp4_audio_list;
-  } else if (!strcmp (format_name, "mpeg")) {
+  } else if (!strcmp (format_name, "mpeg") || !strcmp (format_name, "dvd")) {
     static enum CodecID mpeg_video_list[] = { CODEC_ID_MPEG1VIDEO,
       CODEC_ID_MPEG2VIDEO,
       CODEC_ID_H264,
```

This patch basically enables the *ffmux_dvd muxer* in gst-ffmpeg needed by Mistelix.

Once this patch has been applied, you have to build *gst-ffmpeg* from sources you follow these steps:

1) Install following additional packages:

- cvs
- gcc
- libgstreamer0.10-dev

2) Download the latest ffmpeg source code using CVS source control system:

```
cvs -d:pserver:anoncvs@anoncvs.freedesktop.org:/cvs/gstreamer co gst-ffmpeg
```

3) Apply the patch to activate the *ffmux_dvd muxer*.

4) Build and install the plug-in from sources using:

```
./autogen.sh && make install
```

once it is compile you can run it by just typing 'mistelix' and the application will start.

2. Technologies

In this section the technologies involving this project are described briefly. The description includes DVD related technologies and software libraries and components used to build the application.

2.1 The DVD standard

DVD is the most popular format for delivering multimedia content: films, company presentations, weddings and others.

The DVD standard is set by an association of 220[24] hardware and software manufacturers called DVD Forum[25] that was founded in 1997.

The DVD Forum has two major proposes:

- To establish a single format for each DVD application product, including revisions, improvements and enhancements for the benefit of consumers and users.
- To promote broad acceptance of DVD products on a worldwide basis, including the entertainment, consumer electronics and IT industries as well as the general public.

DVD-Video is a standard for storing video content on DVD media. These DVD can be played in DVD players at home connected to a TV set, video games consoles, personal computers or other devices.

DVD-Video discs support features like menus, selectable subtitles, multiple camera angles, and multiple audio tracks.

To produce DVD-Video titles with these features you need to use a DVD authoring tool (like Mistelix).

The DVD-Video standard **does not provide support for slideshows**. Authoring tools have to produce a video from a set of images by their own, since the standard does not support showing static images or the concept of transitions.

2.2 DVD disc physical standards

DVD disc is a very popular optical disc storage media format defined by the DVD Forum. Its main uses are video and data storage. Single layer DVDs can store up 4.7 gigabytes and double layer DVD can store up 8.5GB.

There are many variations of the DVD disc format, the following ones are the most popular ones:

- DVD-ROM has data that can only be read and not written.
- DVD-R and DVD+R can record data only once and then function as a DVD-ROM.

24 http://www.dvdforum.org/about-memberlist.htm
25 http://www.dvdforum.org

- DVD-RW, DVD+RW and DVD-RAM can both record and erase data multiple times.

Modern DVD players support all these physical variations.

2.3 DVD Video

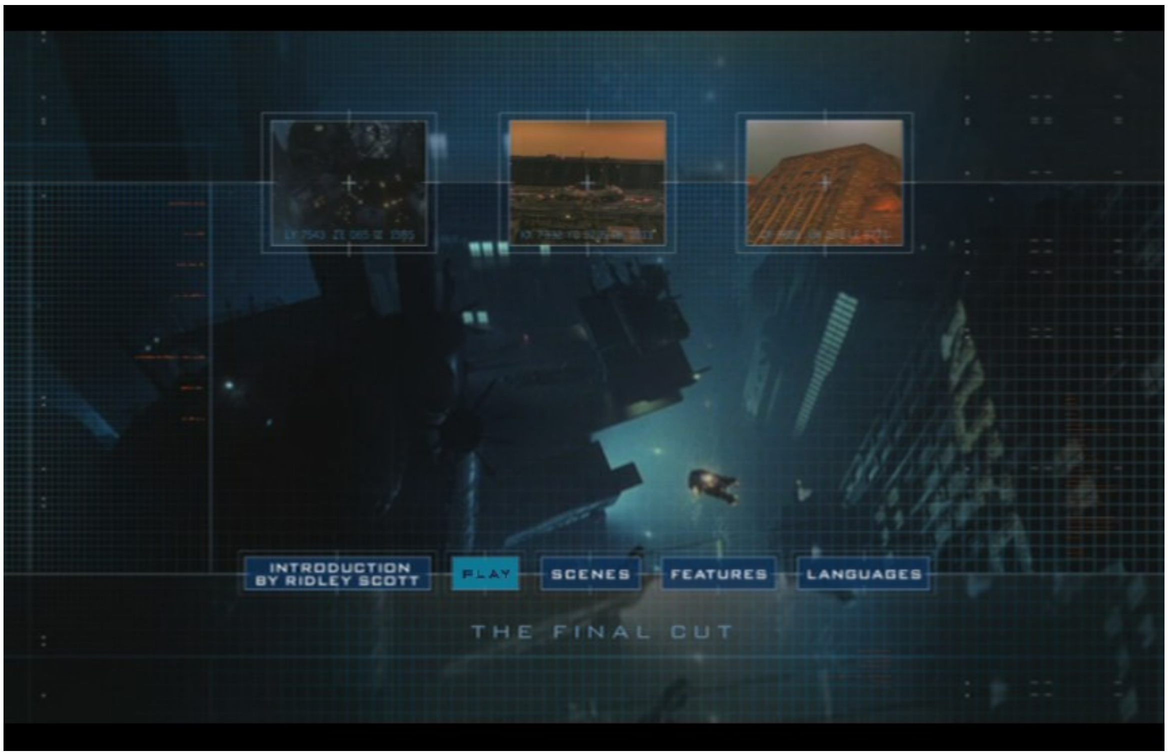
In this section the features and technical characteristics of the DVD-Video format defined by the DVD forum are described.

2.3.1 Features

A DVD physical disc can be structured using several logic formats. The most common logic standard is DVD-Video, defined by the DVD Forum, that is the most popular format used by the movie industry to distribute films.

A DVD-Video supports the following features:

- **Menus**. Each disc has a main menu from which the user can navigate the content using his DVD player remove control. It is possible to have submenus attached to every menu. The menus are the most important part of a DVD because they are responsible of the user navigation.
- **Subtitles**. DVD Video may also include up to 32 subtitle or subpicture tracks in various languages, including those made especially for the deaf and hearing impaired. They are stored as bitmap images with transparent background and are shown over the video during playback.
- **Chapters.** Chapters are like bookmarks in a video that allow the user to skip or jump to specific points on the DVD.
- **Content protection**. Since the DVD is widely use by the movie industry several copy protection systems have been put in place. The most basic protection is the region code that restricts the area of the world in which a DVD can be played. There are 8 regions defined in the world, DVD titles bought in one region (for example United Estates) cannot be reproduced in other regions (for example Europe). The are other protection systems like Content Scramble System and derivates to protect the DVD from begin copied.
- **Deferent camera angles**. It allows to view a video with different camera angles. The films have to be reordered with multiple cameras to capture all the angles. When the user is playing the DVD, he can switch from his remote control to different angles.



Main menu of the DVD Blade Runner

In the previous screen capture you can see the DVD Blade Runner main menu. The user can navigate the different contents (Play, Scenes, and so on) using the remote control.

2.3.2 Technical characteristics

DVD-Video builds on top of already existant standards. To store its content uses the following standards:

- MPEG-2 to encode video as defined in the ISO/IEC International Standard (ISO 13818) for digital television. MPEG-2 was conceived as an encoding and compression standard for broadcast television based on interlaced scanning of images.
- Dolby Digital (AC-3) formats to store audio. Dolby® Digital is a versatile audio encoding/decoding technology. Dolby Digital technology can transmit mono, stereo (two-channel), or up to 5.1-channel surround sound.

Many different video resolutions and formats are supported within the DVD-Video standard, but the most common are:

- PAL (Phase Alternating Line, is a color encoding system) used mainly in Europe and stored at a resolution of 720x576 pixels. This standard was developed by Telefunken during 1967 and since when has been widely adopted in Europe.
- NTSC (National Television System Committee) used mainly in United States and stored at a resolution of 720x480 pixels. NTSC is also the name of the U.S. standardization body that defined the format.

DVD authoring is the process of creating a DVD video that can be played on a DVD player. DVD authoring software must conform to the specifications set by the DVD Forum for the DVD Video standard.

2.4 Platform

In this section the software that provides the basic infrastructure used to build Mistelix is described.

2.4.1 Mono

Mono[26] is an open source implementation of the Microsoft .NET Development Framework led by Novell.

Mono consists of three groups of components:

1. Core components
2. Mono/Linux/GNOME development stack
3. Microsoft compatibility stack

The **core components** include the C# compiler, the virtual machine, and the base class libraries. These components are based on the Ecma-334 and Ecma-335 standards, allowing Mono to provide a standards compliant, free and open source CLI virtual machine.

The **Mono/Linux/GNOME development stack** provide tools for application development while leveraging existing GNOME and free and open Source libraries. These include: GTK for GUI development, Mozilla libraries for working with the Gecko rendering engine, Unix integration libraries, database connectivity libraries, a security stack, and the XML schema language RelaxNG. GTK allows Mono applications to integrate into the Gnome desktop as native applications.

The **Microsoft compatibility stack** provides a pathway for porting Windows .NET applications to Linux. This group of components include ADO.NET, ASP.NET, and Windows.Forms, among others.

2.4.2 Mono.Addins

Mono.Addins is a generic framework for creating extensible applications, and for creating libraries which extend those applications.

The main features of Mono.Addins are:

- Supports descriptions of add-ins using custom attributes (for simple and common extensions) or using an xml manifest (for more complex extensibility needs).
- Support for add-in hierarchies, where add-ins may depend on other add-ins.
- Lazy loading of add-ins.
- Provides an API for accessing to add-in descriptions, which will allow building development and documentation tools for handling add-ins.
- Dynamic activation / deactivation of add-ins at run time.

26 http://www.go-mono.com

- Allows sharing add-in registries between applications, and defining arbitrary add-in locations.
- Allows implementing extensible libraries.
- Supports add-in localization.
- In addition to the basic add-in engine, it provides a Setup library to be used by applications which want to offer basic add-in management features to users, such as enabling/disabling add-ins, or installing add-ins from on-line repositories.

2.4.3 Autoconf and automake

Automake[27] is a programming tool that produces portable makefiles for use by the make program, used in compiling software. It is made by the Free Software Foundation as one of GNU programs, and is part of the GNU build system. It is written in the Perl programming language and must be used with GNU autoconf.

Automake aims to allow the programmer to write a makefile in a higher-level language, rather than having to write the whole makefile manually. In simple cases, it suffices to give:

- A line that declares the name of the program to build.
- A list of source files.
- A list of command-line options to be passed to the compiler (for example, in which directories header files will be found).
- A list of command-line options to be passed to the linker (which libraries the program needs and in what directories they are to be found).

From this information, Automake generates a makefile that allows the user to compile the program, clean (remove the files resulting from the compilation), install the program in standard directories; uninstall the program from where it was installed, create a source distribution archive (commonly called a tarball) and test that this archive is self-sufficient.

Autoconf[28] is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of UNIX-like systems. The configuration scripts produced by Autoconf are independent of it when they are run.

Together with Automake, Autoconf forms the GNU build system. It comprises several tools like Autoheader, etc.

2.4.4 MonoDevelop IDE

MonoDevelop[29] is an open source integrated development environment (IDE) for the Linux platform, primarily targeted for the development of software that uses both the Mono and Microsoft .NET framework.

MonoDevelop integrates features similar to that of Eclipse and Microsoft's Visual Studio such as

27 http://www.gnu.org/software/automake/
28 http://www.gnu.org/software/autoconf/
29 http://www.mono-develop.com/

Intellisense, source control integration, and an integrated GUI and Web designer. MonoDevelop integrates a GTK GUI designer.

The main features of MonoDevelop are:

- Code Completion. MonoDevelop's intelligent code completion attempts to complete type, method and field names as you're typing. The IDE will automatically get the class information from your source code files and from the libraries referenced in your project.
- Class Management. MonoDevelop has a class viewer which allows you to list the classes in your project, their methods, and properties. Your namespaces are also kept track of to keep the classes separated.
- Built-in Help. The .NET documentation and the Gtk# documentation are built into MonoDevelop for easy access.
- Project Support. MonoDevelop comes with built in projects that help get you started with your console, Gnome# or Gtk# application.
- Add-ins. MonoDevelop has a powerful add-in engine, which together with a modular API and a complete set of extension points, provides a seamless platform upon which to build your own development tools. MonoDevelop also provides an Add-in Manager you can use to install add-ins from on-line repositories.

2.5 User interface libraries

In this section the main user interface libraries used to build Mistelix are described. These are open source libraries very widely used.

2.5.1 GTK

GTK[30] is a cross-platform widget toolkit for creating graphical user interfaces. It is one of the most popular toolkits for Linux platforms but also works on Microsoft Windows and other systems.

GTK provides several features:

- An object system written in C, complete with inheritance, type checking, and a signal/callback infrastructure (event driven system). The type and object systems are not GUI-specific.
- A GtkWidget object written using the object system, which defines the interface GTK's graphical components implement.
- A large collection of useful GtkWidget subclasses (widgets); this collection includes labels, edit controls, trees, comboboxes, and other popular controls.

GTK is used by many popular applications, like the GNOME desktop, Abiword, Inkscape, Evolution, GIMP and many other open source applications.

30 http://www.gtk.org

2.5.2 Cairo

Cairo[31] is a software library used to provide a vector graphics-based, device-independent API for software developers. It is designed to provide primitives for 2-dimensional drawing across a number of different backends. Cairo is designed to use hardware acceleration when available.

The Cairo API provides operations similar to the drawing operators of PostScript and PDF. Operations in Cairo including stroking and filling cubic Bézier splines, transforming and compositing translucent images, and antialiased text rendering. All drawing operations can be transformed by any affine transformation (scale, rotation, shear, etc.)

2.5.3 Glade

Glade[32] Interface Designer is a graphical user interface builder for GTK, with additional components for GNOME. Glade is programming language–independent, and does not produce code for events, but rather an XML file that is then used with an appropriate binding (such as gtkada for use with the Ada programming language).

2.5.4 Gettext

Gettext[33] is the GNU internationalization (i18n) library. It is commonly used for writing multilingual programs.

Gettext provides the necessary tools to separate easily in external files the parts of the application that can be translated. Gettext is currently the most popular options for writing localizable application in the open source space.

2.6 Multimedia libraries

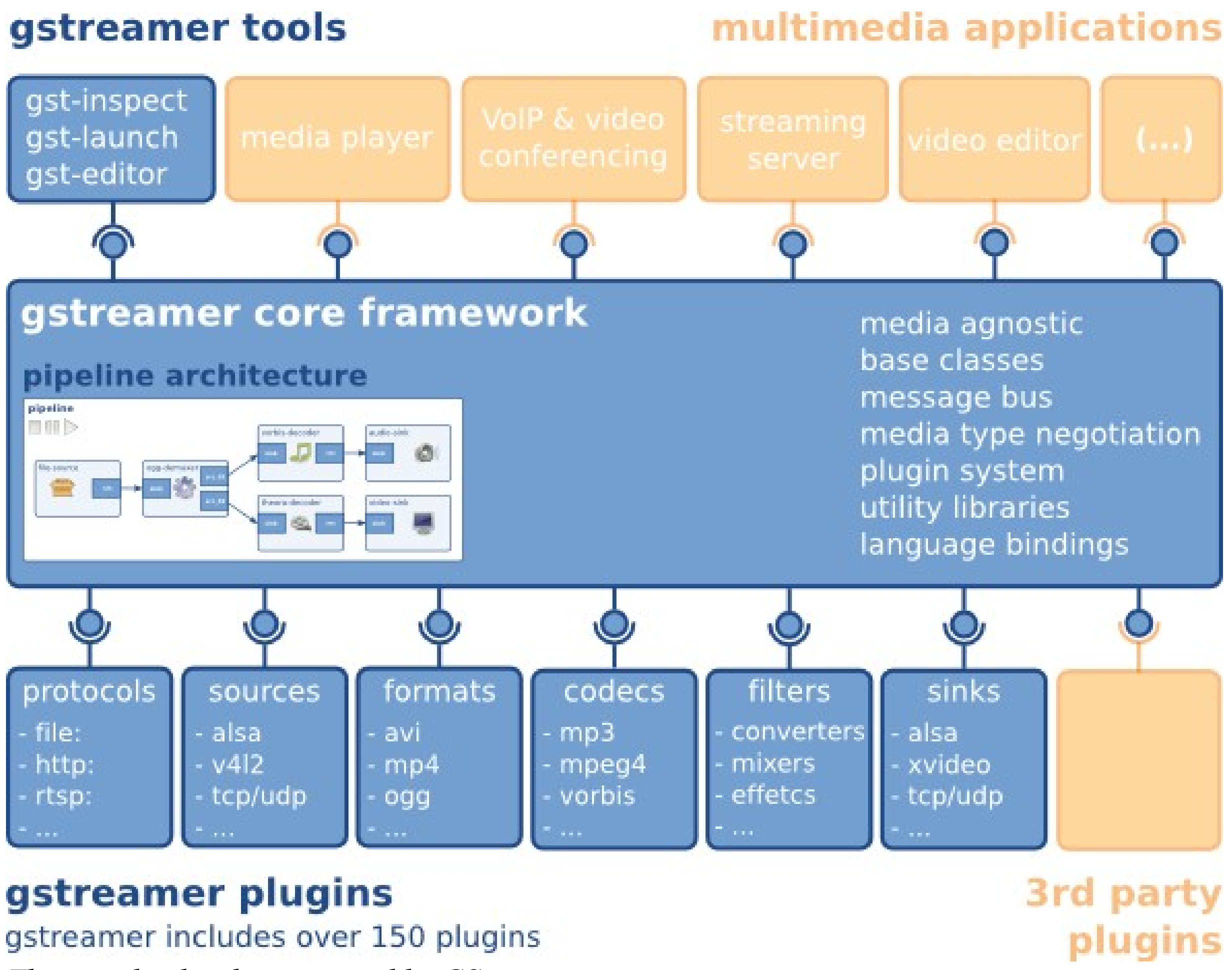
In this section the main multimedia libraries used to build Mistelix are described. These are open source libraries very widely used.

2.6.1 GStreamer

GStreamer is a pipeline based multimedia framework. GStreamer processes media by connecting a number of processing elements into a pipeline. Each element is provided by a plug-in. Elements can be grouped into bins, which can be further aggregated, thus forming a hierarchical graph.

Elements communicate by means of pads. A source pad on one element can be connected to a sink pad on another. When the pipeline is in the playing state, data buffers flow from the source pad to the sink pad. Pads negotiate the kind of data that will be sent using capabilities.

31 http://www.cairographics.org/
32 http://glade.gnome.org/
33 http://www.gnu.org/software/gettext/



This graphic has been created by GStreamer project

2.6.2 ffmpeg

ffmpeg[34] library can record, convert and stream digital audio and video in numerous formats including MPEG-2. It includes libavcodec, an audio/video codec library used by several other projects, and libavformat, an audio/video container mux and demux library.

GStreamer FFmpeg plug-in[35] allows to access fmpeg library code. It contains most popular decoders as well as very fast colorspace conversion elements.

2.7 External tools

In this section the external tools used by Mistelix are described. These are standard open source tools that help to produce DVDs.

34 http://www.ffmpeg.org/
35 http://gstreamer.freedesktop.org/modules/gst-ffmpeg.html

2.7.1 Dvdauthor

dvdauthor is a tool that assembles multiple MPEG program streams into a suitable DVD file system that can be later recorded.

Dvauthor allows to define an XML file with the DVD-Video structure of a DVD (defining menus, chapters and so on) and produces the DVD-Video file system structure that can be then recorded in a physical DVD.

2.7.2 Spumux

Spumux generates and multiplexes subtitles into an existing MPEG2. It can be used to generate menus and subtitles.

Spumux allows to define an XML with subtitles or menu actions and multiplexes then in already existing video stream.

3. System Analysis

In this section a system analysis for the problem is performed. A comprehensive list of features and use cases is provided that helped to define the functional scope that the application covers.

3.1 Uses cases

Use cases are used in software engineering to capture the functional requirements of a system. They describe the interaction between a user and the system itself. A set of well-defined use cases help to define a good set of functional requirements. For Mistelix, I have defined some user cases, that help me to craft the functional specifications.

3.1.1 Creating slideshow photo DVD

A user has a set of pictures from his holidays and wants to create a DVD to give away to family members and friends:

Case description

- The user has a set of pictures in his hard drive from three different locations
  - The pictures are in JPEG format
  - They may be in high resolution
- The user wants to create a slideshow for every one of three locations
- The user wants to add a song as a background music
- The user expects the application to provide a suitable theme for his needs
- The user should be able to reorder the images
  - To change their position within the editing view
  - To sort them by date on disc
  - To sort them by filename
- The user should be able to delete images already selected
- The user should be to specify the time and transition for every image
- The user wants to burn the final project in a DVD

3.1.2 Modifying an already existing slideshow

A user has author a slideshow and wants to modify it. This is an important case to document since the slideshows are converted to video. The user should be able to modify the source set of images and their parameters at any time.

Case description

- The user should be able to open an already created slideshow
- The user should be able to reorder the images
- The user should be able to delete images already selected
- The user should be able to add new images
- The user should be to change the time and transition for every image

3.1.3 Creating a DVD with recorded videos

A user has a set of videos from a business event and wants to create a DVD to give it to the attenders.

Case description

- The user has a group of videos in his hard drive
  - These videos are in MPEG2 format
  - They may be not in the right resolution for DVD encoding
- The user wants to create a slideshows for every one of three locations
- The user wants to add a song as a background music for everyone of the locations
- The user expects the application to provide a suitable theme for his needs
- The user wants to burn them the project in a DVD

3.1.4 Creating a DVD with slideshows and recorded videos

A user has a set of videos and slideshows from a wedding and wants to create a DVD to give it to friends and family.

Case description

The sideshows:

- The user has a set of pictures in his hard drive from one single location
  - The pictures are in JPEG format
  - They may be in high resolution
- The user wants to create a slideshow for every one of three locations
- The user wants to add a song as a background music
- The user expects the application to provide a suitable theme for his needs
- The user wants to burn the final project in a DVD

The videos:

- The user has a group of videos in his hard drive

    - These videos are in MPEG2 format
    - They may be not in the right resolution for DVD encoding
- The user wants to create a slideshows for every one of three locations
- The user wants to add a song as a background music for everyone of the locations
- The user expects the application to provide a suitable theme for his needs
- The user wants to burn them the project in a DVD

3.1.5 A user wants to translate the application

It is very common in open source that the users want to participate in the development on their favorite applications. One of the areas that is more easy to get user involve is translation, because it does not require technical skills and can usually be done without the need of using development tools (like compilers or an IDE).

Case description

- The user should be able to translate the application using external files
- The user expects these files to be compatible with regular translations tools like translation memories and glossaries to reuse previous translations
- The user should be able to test the translation without the need of recompiling the application or dealing with development tools
- The user expects to do the translation using open source tools

3.1.6 A user wants to extend the application.

As we commented before, it is very common in open source that the users want to participate in the development on their favorite applications.

However, the skills of some developers are limited. It is always convenient to provide mechanisms to extend the application without the need to understand the full design of the system or having to recompile the application.

Extensibility is key in open source to be able to build communities of developers around the applications. Communities that can extend the application original capabilities.

Case description

- Provide a mechanism to extend the application in predefined anchor points without the need of recompiling it
- To be able to install and test the extensions in easy manner
- That the extensions are binary compatible between different platforms
- To be able to provide a central repository that can serve extensions to users remotely using a client / server approach.

3.2 Functional requirements

This list describes the functional requirements that the application should fulfill:

- Project Management
  - Create new projects
    - Indicate output format
    - Default output location
  - Load / Save projects
- Slideshow creation
  - Select images and compose a slideshow
  - Select effects between images
  - Set the slide expose time
  - Sort and reorder the images within the slideshow
- Video elements
  - Able to import videos
- Theme support
  - Able to support multiple themes
- Generate a final DVD from the project

3.3 Non-functional requirements

This list describes the non-functional requirements that the application should fulfill:

- Provide a easy and intuitive user interface suitable for novice users
  - Based on drag and drop operations
  - The application should not block the user interface
  - It should hide the complexity of the components and external tools
- It should provide a large collection of themes based on open licenses
- Extensibility
  - The application set of themes can be extended by third parties
  - It should be possible to the default transitions using plug-ins
  - The user interface should be completely localizable to any language
- The application should support major digital photography, video and audio formats for importing content
  - JPEG, PNG and TIFF for image importing

    - MPEG2 for video importing
    - AC and MP3 for audio importing
- Multiplatform. There are many versions of Linux and Unix systems and they run in different architectures. The system should work in the most popular combinations.

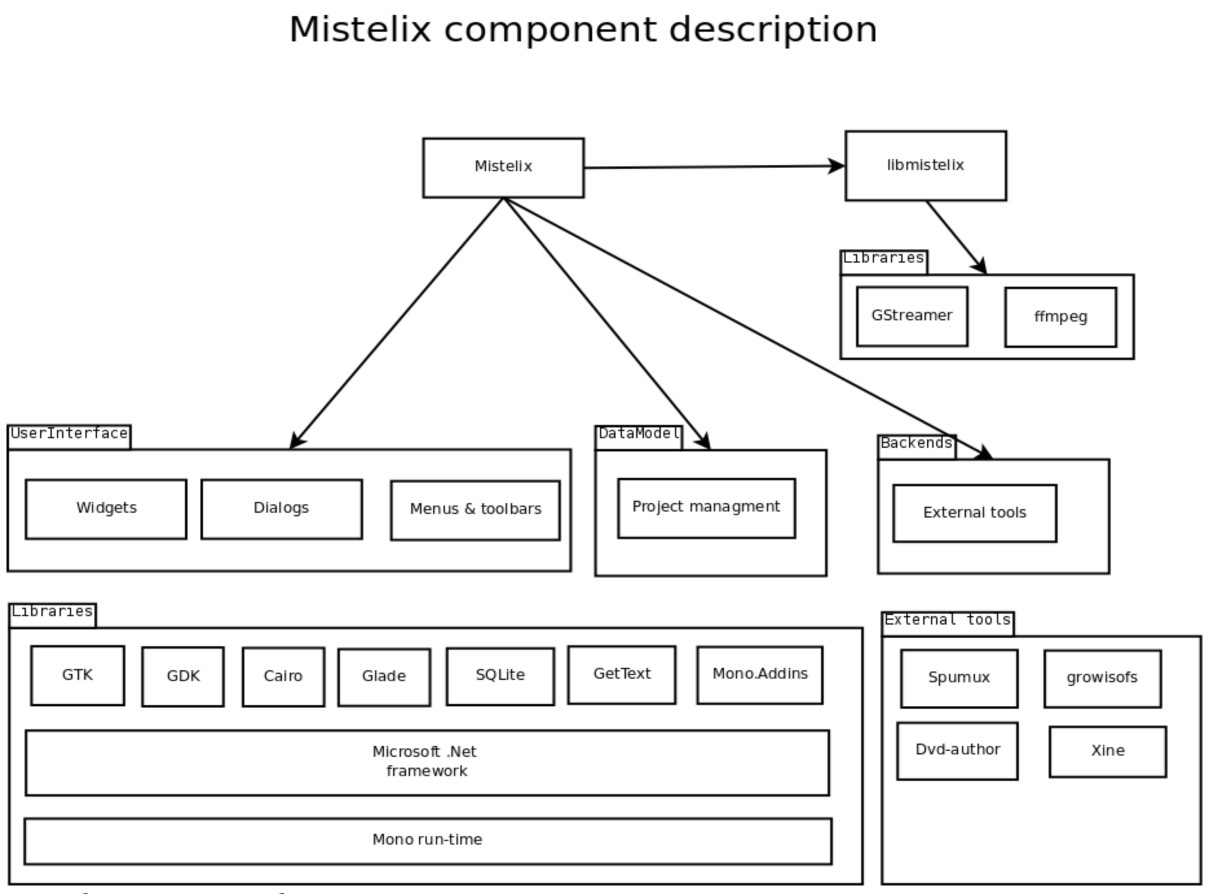
4. Application architecture

In this section the Mistelix architecture is described. An overview of the different modules of the application is given, the libraries and external tools used and how all these components work together to structure the application.

4.1 Components description

This is high level description of all the components that Mistelix uses, including:

- Namespaces (Dialogs, DataModel, etc)
- Libraries (GTK, Cairo etc)
- Externals tools (dvdautor, Spumux, Xine, etc)



Mistelix component description

4.1.1 Library usage description

Mistelix builds on top of many open source libraries. The table below describes the main libraries used and which services they provide to the application.

A comprehensive description of these libraries is provided in the previous**Technologies** section of this document.

Library	***Usage***
GTK	Used to provide the user interface for the application.
Cairo	Used by Mistelix to do advanced drawing in surfaces (like buttons).
GDK	Used to load images from different formats (JPEG; GIF) and scale them.
Gettext	Mistelix uses Gettext to enable the translation of Mistelix in other languages using PO files.
Glade	Mistelix uses an Glade XML[36] file to define it user interface.
Mono Addins	Used to extend the slideshow transition effects.
Mono	This is the .Net framework implementation.
GStreamer	Uses Gstreamer from libmistelix to connect to the different multimedia backends, like ffmpeg.
ffmpeg	Uses ffmpeg to encode MPEG2 files.

4.1.2 External tools usage description

Mistelix builds on top of some open source external tools. The table below describes the main tools used and which services they provide to the application.

36 http://code.google.com/p/mistelix/source/browse/trunk/src/mistelix.glade

A comprehensive description of these external libraries is provided in the previous **Technologies** section of this document.

Library	***Usage***
spumux	Used to generate DVD buttons into the MPEG2 stream.
dvdauthor	Used to generate the final DVD file system.
growisofs	Used to convert the DVD into an ISO image that can later be recorded.
Xine	Used to preview the final generated DVD.

4.2 libmistelix

4.2.1 General overview

libmistelix[37] is a C library, part of the Mistelix project, that glues the Mistelix application with the underlying codec infrastructure provided by GStreamer.

The objectives of libmistelix are to make ease to replace the multimedia backends in the future if patents problems arise or if we decide to use another multimedia backend different to Gstreamer. To achieve this, libmistelix provides a very simple API that encapsulates the functionality provided by the different codecs and muxers.

As an example, this is a part of the API that allows to create slideshows from a set of images:

int **mistelix_slideshow_createstream** (const gchar* filename, unsigned int weight, unsigned int height, unsigned int framesse);

void **mistelix_slideshow_add_image** (unsigned char* bytes, unsigned int len);

void **mistelix_slideshow_add_imagefixed** (unsigned char* bytes, unsigned int len, unsigned int frames);

void **mistelix_slideshow_close** ();

libmisteix makes the appropriated calls to GStreamer backend to accomplish the different tasks that have been requested to libmistelix.

4.2.2 Mistelixvideo source GStreamer pluggin

To make the communication possible with Gstreamer, **Mistelix provides a GStreamer pluggin** written in C called *mistelixvideosrc*[38]. The objective of this plug-in is to receive a list of images from the Mistelix application and put them on the Gstreamer pipeline to allow them to be processed by the Gstreamer infrastructure (e.g converting it to a video).

37 http://code.google.com/p/mistelix/source/browse/trunk/libmistelix
38 http://code.google.com/p/mistelix/source/browse/trunk/#trunk/gstreamer

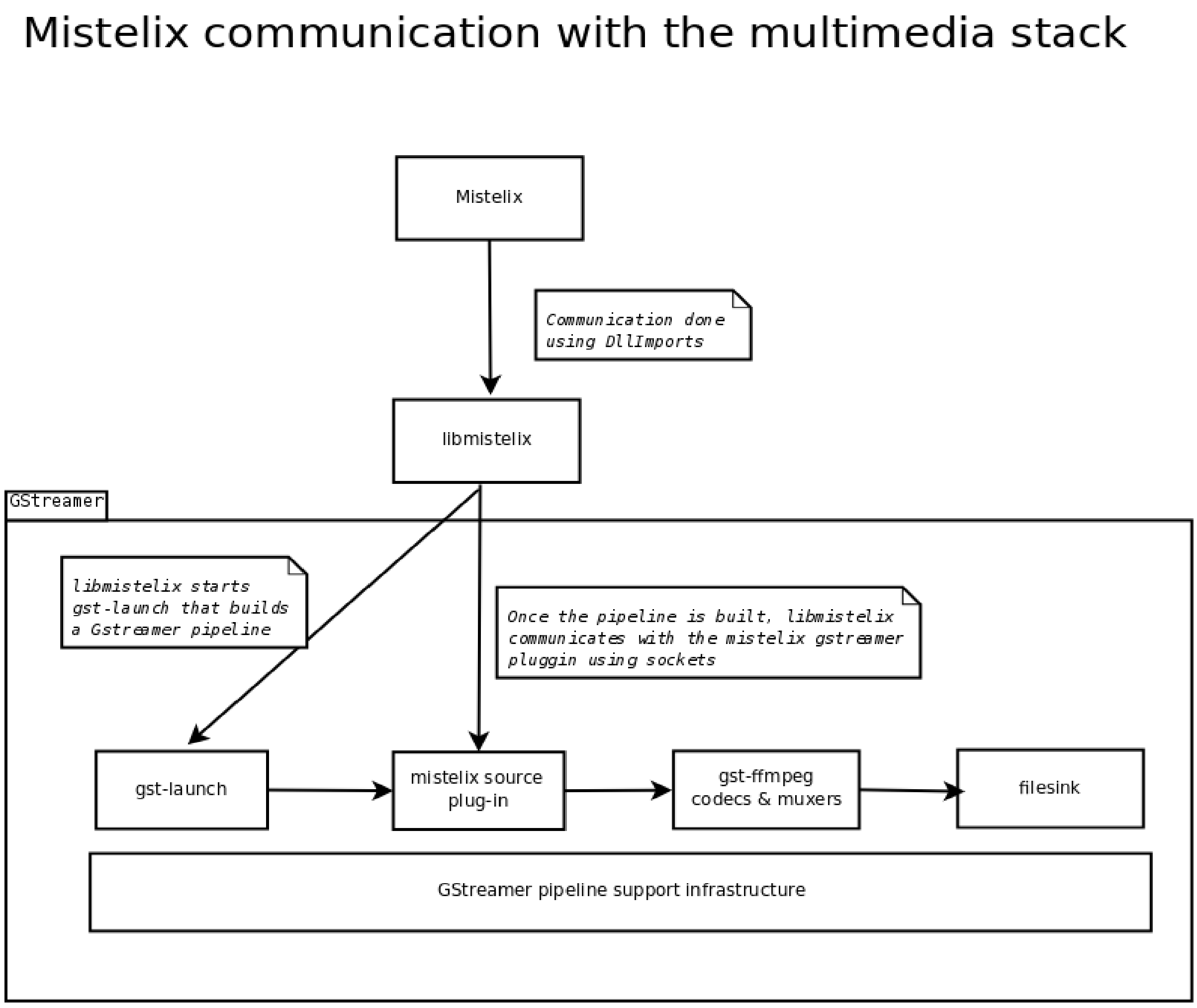
Mistelix GStreamer source plug-in has a listening TCP/IP deamon that allows it to be operated as a source pad from Mistelix.

4.2.3 Communication between Mistelix and the multimedia stack

To show how the communication between Mistelix and the multimedia stack works, lets assume that we are going to produce a slideshow from a set of pictures. The communication process will be the following:

- Mistelix calls the C library libmistelix. through the class *LibMistelix*[39] using DllImport[40].
- It calls the C function **mistelix_slideshow_createstream.** This function starts a GStreamer pipeline calling *gst_parse_launch gstreamer* C API call creating a pipiline and setting up the following elements:
  - *mistelixvideosrc* (our pluggin) as the source.
  - *ffenc_mpeg2video* as the encoder. This is the ffmpeg MPEG2 encoder.
  - *filesink* as the sink that outputs the final MPEG-2 into disc.
- Once the GStreamer pipeline is created *mistelixvideosrc* listens to a TCP/IP socket. From Mistelix, a connection is open to the local socket and a set of images is sent.
- The images are processed by the *ffenc_mpeg2video* encoder and output into disc by filesink.

39 http://code.google.com/p/mistelix/source/browse/trunk/src/MistelixLib.cs
40 http://msdn.microsoft.com/en-us/library/system.runtime.interopservices.dllimportattribute.aspx


Mistelix communication with the multimedia stack
Mistelix
Communication done using DllImports
libmistelix
GStreamer
libmistelix starts gst-launch that builds a Gstreamer pipeline
Once the pipeline is built, libmistelix communicates with the mistelix gstreamer pluggin using sockets
gst-launch
mistelix source plug-in
gst-ffmpeg codecs & muxers
filesink
GStreamer pipeline support infrastructure

Mistelix communication with the multimedia stack

4.3 Background process scheduler

Modern workstations nowadays have more than a single processor. Mistelix takes an advantage from multiprocessing by performing on the background operations that can take a long time to execute. For example:

- Converting videos that are going to be part of the DVD to MPEG2
- Converting sequences of images (slideshows) into MPEG2
- Creating DVD images

There is a class *BackgroundTaskCollection*[41] in Mistelix that implements a priority queue data structure. Every process that requires large processing is send to the queue for processing. The application executes them in the background and honors their priority.

41 http://code.google.com/p/mistelix/source/browse/trunk/src/datamodel/BackgroundTaskCollection.cs

On top of this background processing queue, all the user interface elements that perform operations that require time, like creating thumbnails, are executed in a separated thread to avoid blocking the user interface using the *BackgroundWorker*[42] class. See for example *ProjectElementView*[43] Mistelix's gadget.

4.4 Themes

Mistelix bundles with a collection of themes based on open licenses that can be a starting point for authoring the user's DVD.

The themes can be extended using a external XML file that registers all the themes. There is a *Theme* class that encapsulates all the functionality around Themes.

4.5 Extending the application

In this section the different ways of extending the application without the need of modifying its source code directly are described.

4.5.1 Introduction

Mistelix leverages on Mono.Addins framework to provide extensibility capabilities for the application using extensions.

These extensions are external Microsoft .Net assembly files that Mistelix recognizes at runtime and uses in defined points of the application. The extensions are developed as stand-alone components, there is no need to recompile the application or even has access to the source code.

4.5.2 What can be extended

In the file mistelix.addin.xml[44] we define the points of the application that can be extend.

```
…

<ExtensionPoint path="/Mistelix/SlideTransitions">
     <ExtensionNode name="SlideTransitions"
      objectType="Mistelix.Transitions.ITransition" />
</ExtensionPoint>
</Addin>
```

This allows to external extensions to add extra functionality to the application by implementing the interface ITransition, which defines the transitions between two slides.

42 http://msdn.microsoft.com/en-us/library/system.componentmodel.backgroundworker.aspx
43 http://code.google.com/p/mistelix/source/browse/trunk/src/widgets/ProjectElementView.cs
44 http://code.google.com/p/mistelix/source/browse/#svn/trunk/src

4.5.3 Anatomy of an extension

An extension should be a self-contained code. It should provide at least two files:

- The source code of the extension
- A manifest file (XML file) that describes the extension

It may also contain graphics or additional resources required by the plug-in.

To compile the plug-in is as easy as:

```
gmcs -t:library Opaque.cs -resource:Opaque.addin.xml -rgbrainy.exe
```

Once the extension is compiled, the assembly containing the extension has to be copied either in $PREFIX/lib/mistelix/extensions (e.g. /usr/lib/mistelix/extensions) to make it available system-wide, or in ~/.gnome2/mistelix/addins if the user has no enough rights to install it system-wide.

4.6 Translations

Mistelix uses the Gettext library to enable the translation of Mistelix into other languages using PO (portable object) files. These are text files that are very simple to edit and allow to localize Misteli'x user interface.

For example, the PO file for the Catalan translation[45] looks like:

```
#: ../src/mistelix.glade.h:1
msgid "<b>Output Directory</b>"
msgstr "<b>Directori de sortida</b>"
```

Every translation is stored in a separated PO file that contains an entry for the original string in English and the translated string.

Users can add translations very easily by editing the LINGUAS[46] file and adding a PO file corresponding to a new language.

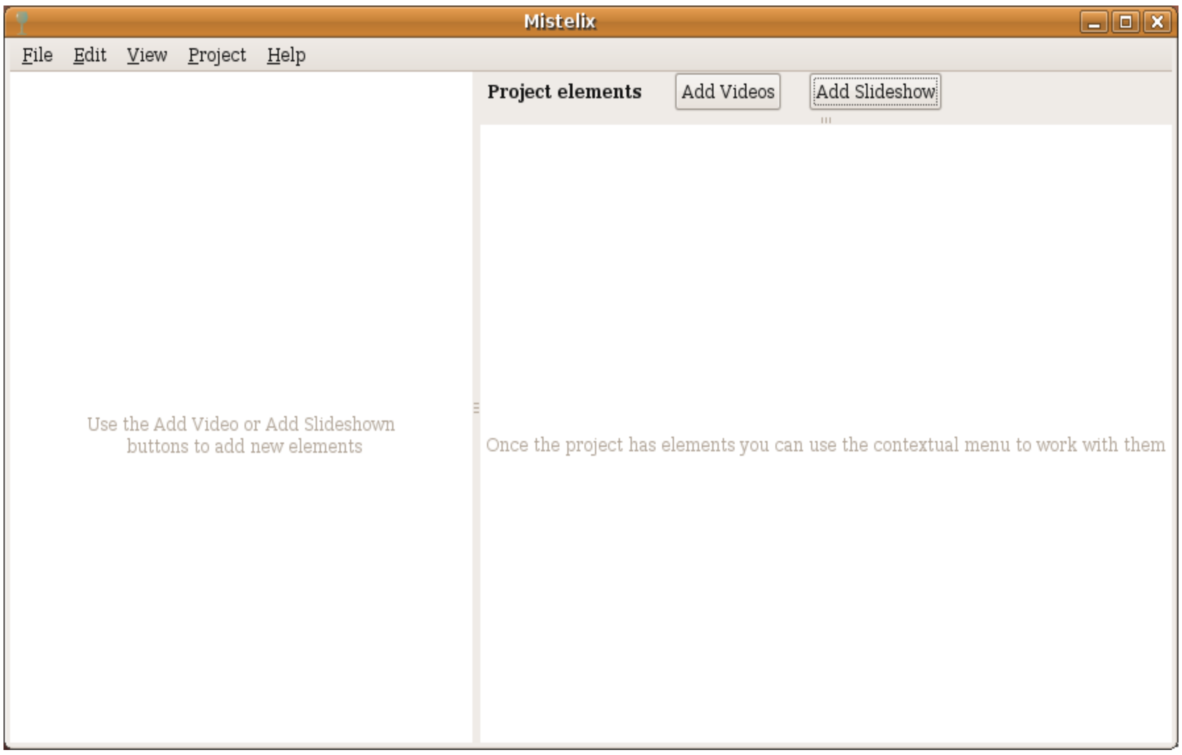
45 http://code.google.com/p/mistelix/source/browse/trunk/po/ca.po
46 http://code.google.com/p/mistelix/source/browse/trunk/po/LINGUAS

5. User interface design

In this section the user interface mockups of the application are described.

5.1 User interface description

Mistelix uses a standard window with a menu as a main user's working area. This window is shown when the application is loaded and is the starting point of all the user actions.


Mistelix
File Edit View Project Help
Project elements
Add Videos
Add Slideshow
Use the Add Video or Add Slideshown buttons to add new elements
Once the project has elements you can use the contextual menu to work with them

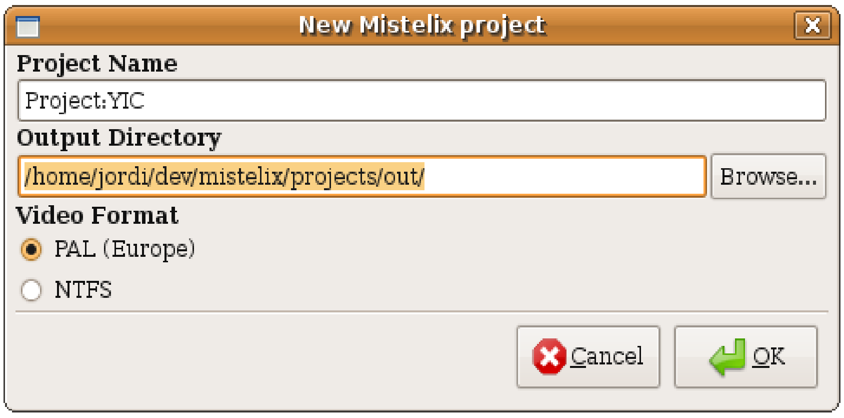
Mistelix's main user interface screen

The main window has three working areas:

- The menu bar at the bottom that gives access to the main Mistelix functions.
- The right pane that is the project element selection area, where the project elements are shown.
- The left pane that is the authoring area, where the user drops the elements that compose the main DVD menu.

5.2 Creating a new project

The new project dialog is used to start a new DVD authoring project and it available from the main menu (File → New Project).


New Mistelix project

Project Name

Project:YIC

Output Directory

/home/jordi/dev/mistelix/projects/out/

Video Format

(●) PAL (Europe)

( ) NTFS

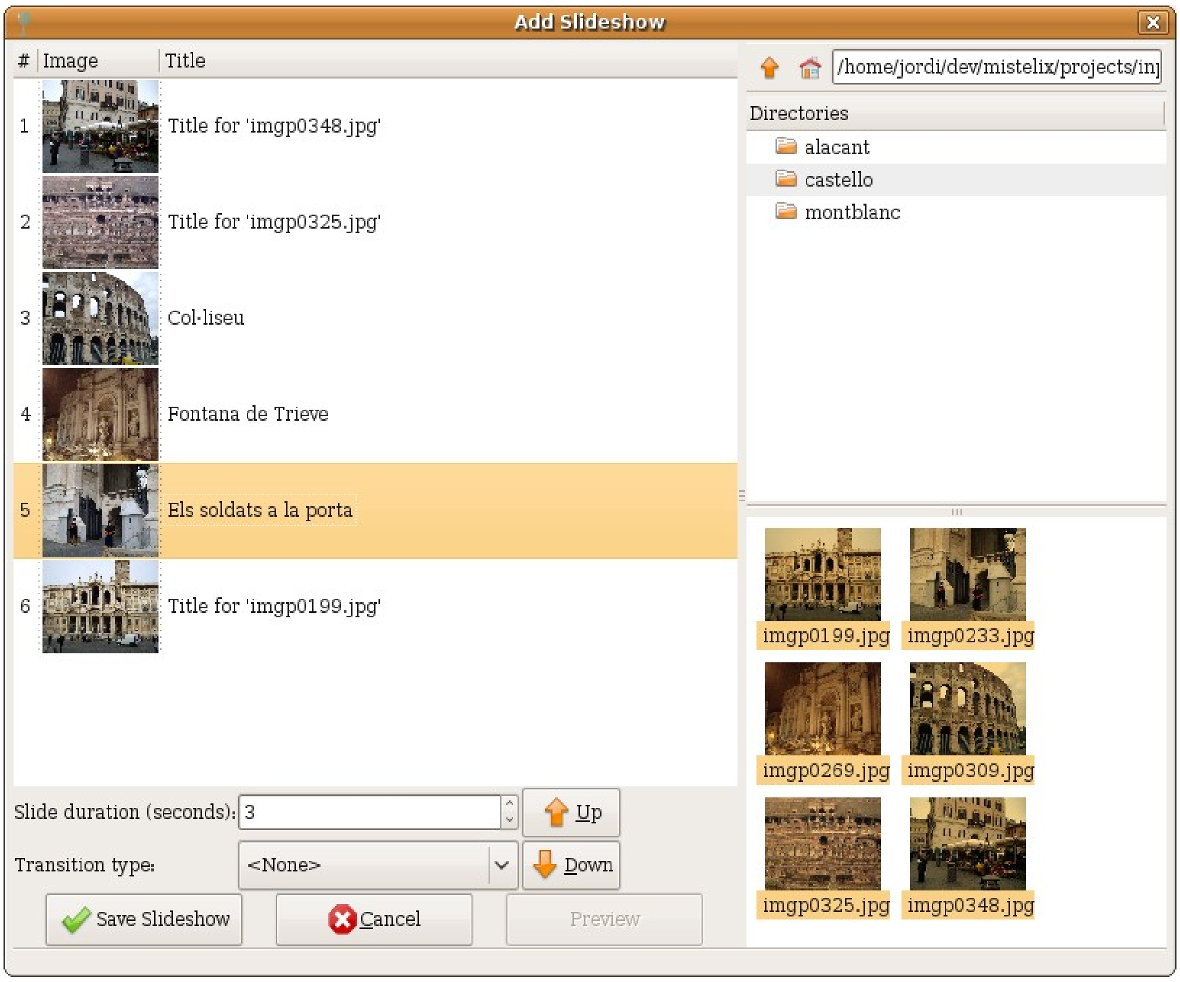
Mistelix's new project creation dialog box

The user can specify the project name, output directory, where the DVD file system will be generated and the video format.

5.3 Adding a slideshow

Users can add new slideshows using the *Add Slideshow* button from the main screen (see the *user interface description* section).

A slideshow is a key part of the user interface because allows the user to select a set of pictures a build a slideshow using a specific set of parameters.



/home/jordi/dev/mistelix/projects/in

#	Image	Title
1		Title for 'imgp0348.jpg'
2		Title for 'imgp0325.jpg'
3		Col·liseu
4		Fontana de Trieve
5		Els soldats a la porta
6		Title for 'imgp0199.jpg'

Slide duration (seconds): 3

Transition type: <None>

Mistelix's add slideshow dialog box

The dialog box is divided in the following areas:

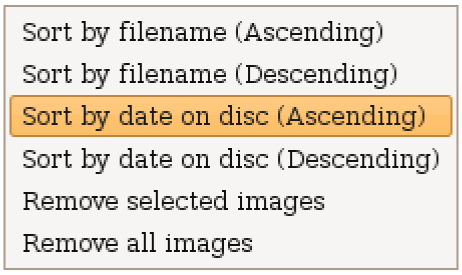
- At the right-top, there is a directory tree that allow users to navigate their hard-drive directory structure. When the directory is changed, the set of images available is loaded in the view below.
- At the top-left, there is a treeview[47] (with *headers #, image* titles) that show the images that the user has drop from the left view and that are going to be part of the slideshow.
- At the left-bottom, you have a set of controls that allow you to select how the transition behaves:
  - *Slide duration* determines the amount of seconds that the slide is shown
  - *Transition type* selects the transition effect between slides
  - The *Up* & *Down* buttons allow to change the order of the images within the slideshow

47 In GTK, treeviews look very similar to listviews in other graphical systems, for example Microsoft Window.

- At the right-bottom, you can browse the images that you can drag in the top-left view to become part of your slide.

Once the user has selected an image or a group of images (in the screen capture, the long light orange selected row) can use the *Slide Duration* and *Transition type* controls to set the time for exposition of the slide and the transition type used.

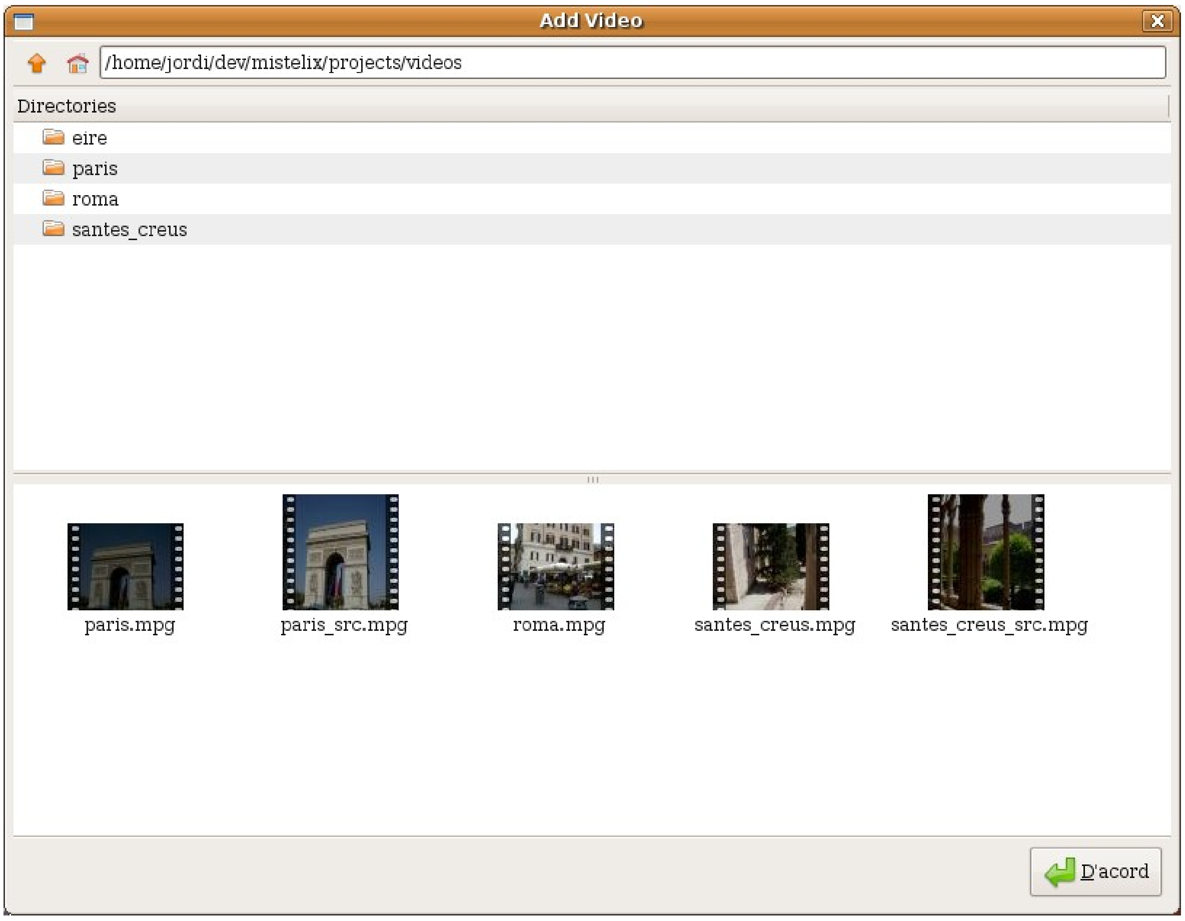
Additionally, the left view has a right menu that offers some contextual operations with the images selected.


Sort by filename (Ascending)
Sort by filename (Descending)
Sort by date on disc (Ascending)
Sort by date on disc (Descending)
Remove selected images
Remove all images

Mistelix's Add slideshow dialog contextual menu

5.4 Adding videos

The *Add Videos* dialog allows the user to select a set of videos and add them as project elements.



Mistelix's add video dialog box

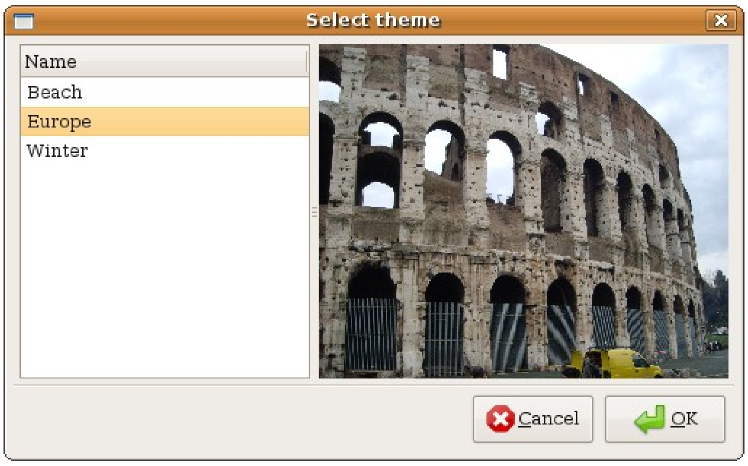
The dialog box is divided in the following areas:

- At top, there is a directory tree that allow users to navigate their hard-drive directory structure. When the directory is changed, the set of videos available is loaded in the view below.
- At the bottom, the user can browse the videos and select them to become project elements.

The user should press the OK button once has selected the videos to add.

5.5 Selecting a theme

Mistelix allows the user to select a theme to base their DVD main menu. This option is available from Mistelix's main menu (Edit → Select Theme).

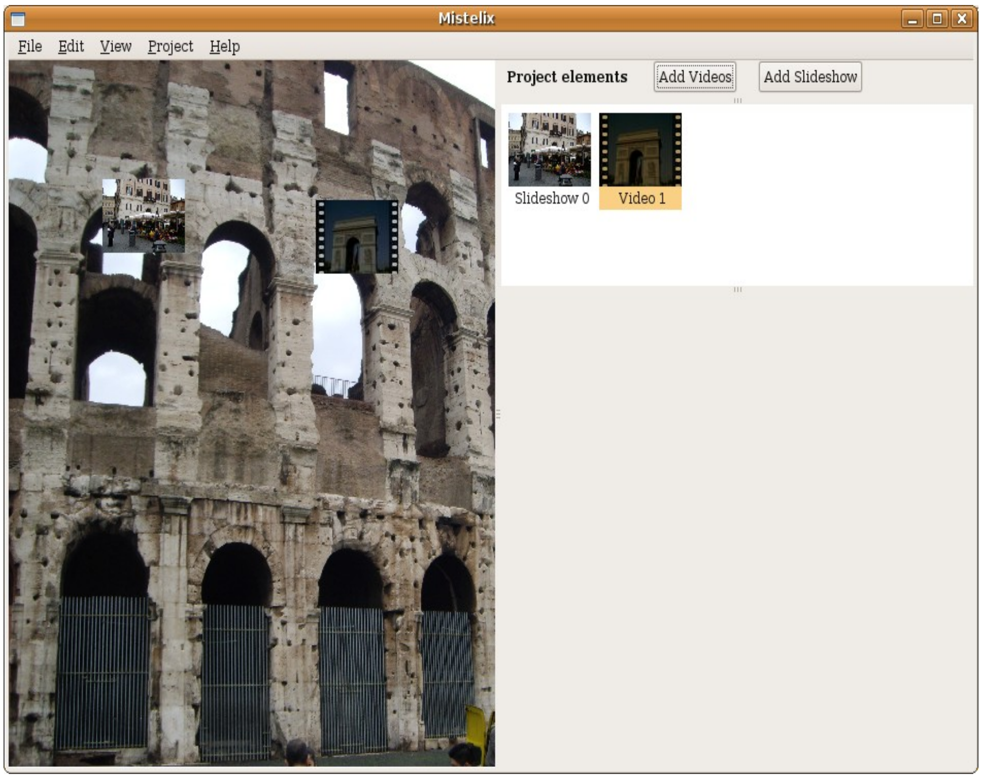


Mistelix's select them dialog box

The theme determines the aspect of the main DVD menu of the authored project.

5.6 Authoring the menus

The main window is Mistelix main authoring area. This is the place where the user can drag the project elements to create the final authored DVD.



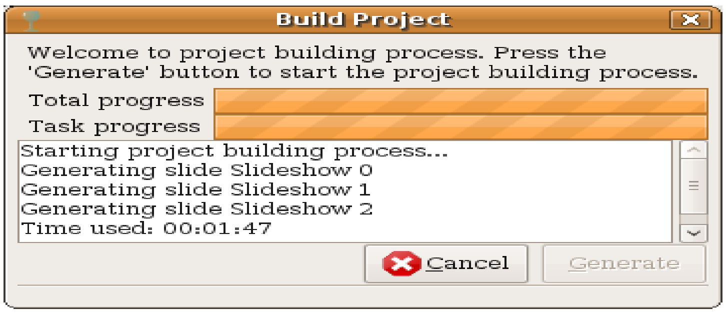
Mistelix's authoring menus

The authoring area contains the the following elements:

- The right section of the screen capture (where it says *Project elements*) the screen is divided in two additional subareas:
    - The actions toolbar at the top (where is says *Add Videos*, *Add Slideshow*) help the user to add new items to the list of project elements.
    - The project element view (in the screen captures shows tree thumbnails) shows a preview of the different project elements that you can drop into the project authoring view.
- The left section of the screen capture (where you can see the Coliseum picture in the screenshot) is where the user drops the elements that will be part of the DVD.

5.7 Building the project

Once you have finished the authoring the project, you can proceed to build the project. This process includes generating the necessary videos for the slideshows, menus and so on. From the main application menu, the option is available at *Project → Build*.



Welcome to project building process. Press the 'Generate' button to start the project building process.

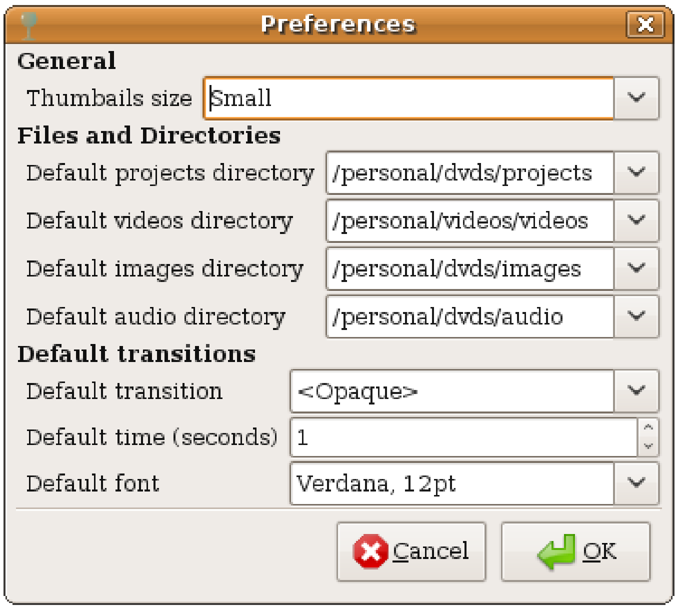
```
Starting project building process...
Generating slide Slideshow 0
Generating slide Slideshow 1
Generating slide Slideshow 2
Time used: 00:01:47
```

Mistelix's build project dialog box

When the user presses the *Generate* button the generation process starts and creates the necessary output files in the project's output directory.

5.8 Preferences

Mistelix has also an application preferences dialog box that allows the user to customize some of the main application parameters.



Preferences

General

Thumbails size: Small

Files and Directories

Default projects directory: /personal/dvds/projects

Default videos directory: /personal/videos/videos

Default images directory: /personal/dvds/images

Default audio directory: /personal/dvds/audio

Default transitions

Default transition: <Opaque>

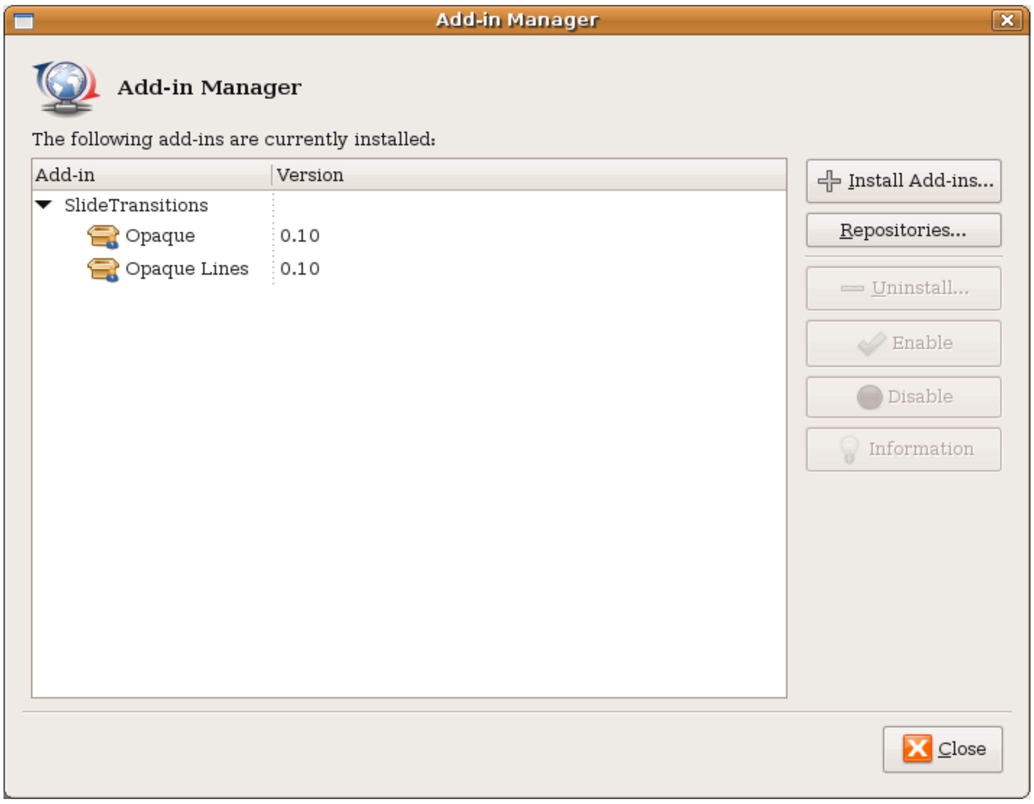
Default time (seconds): 1

Default font: Verdana, 12pt

Mistelix application preferences

5.9 Extensions

Mistelix can be extended using external extensions. From the main application menu, users can select the *Edit* → *Manager extension* option and the Extensions Management interface is shown.



Add-in	Version
SlideTransitions	
Opaque	0.10
Opaque Lines	0.10

Mistelix Add-in manager dialog box

In the sample screenshot there are two extensions shown. These are bundled as part of Mistelix: *Opaque* and *OpaqueLines*. These extensions can be enabled, disabled or the user can check the information about them. The *Install add-ins* button allows to install additional extensions.

6. Implementation

6.1 Project hosting

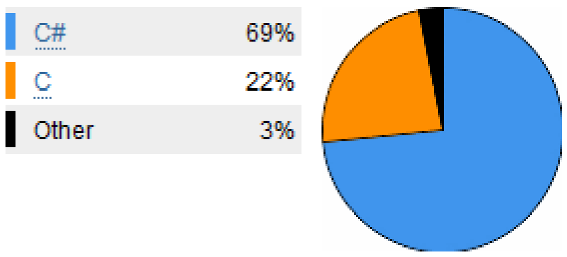
The full Mistelix's development process has been done in an open manner. The project is hosted at Google Code[48], a project hosting service that provides revision control using Subversion, a bug tracking system, a wiki for documentation, and a file download feature.

All the changes done to the project are documented in the source control system change log[49].

6.2 Statistics from Ohloh

Since the project source code of Mistelix is publically available, I have registered the project at Ohloh[50]. This service provides statistics about the development of projects by retrieving data from revision control repositories.

Global statistics on languages used by Mistelix



Taken from Ohloh Mistelix's analysis page1

Detailed statistics on languages used by Mistelix

Language	Code Lines	Comment Lines	Comment Ratio	Blank lines	Total lines
C#	3891	1190	23.4%	977	6058
C	1235	167	11.9%	314	1716

48 http://code.google.com/p/mistelix/
49 http://code.google.com/p/mistelix/source/list
50 http://www.ohloh.net/

XML	821	1	0.1%	535	1357
Automake	218	0	0.0%	71	289
Shell script	139	7	4.8%	20	166
Autoconf	91	7	7.1%	40	138

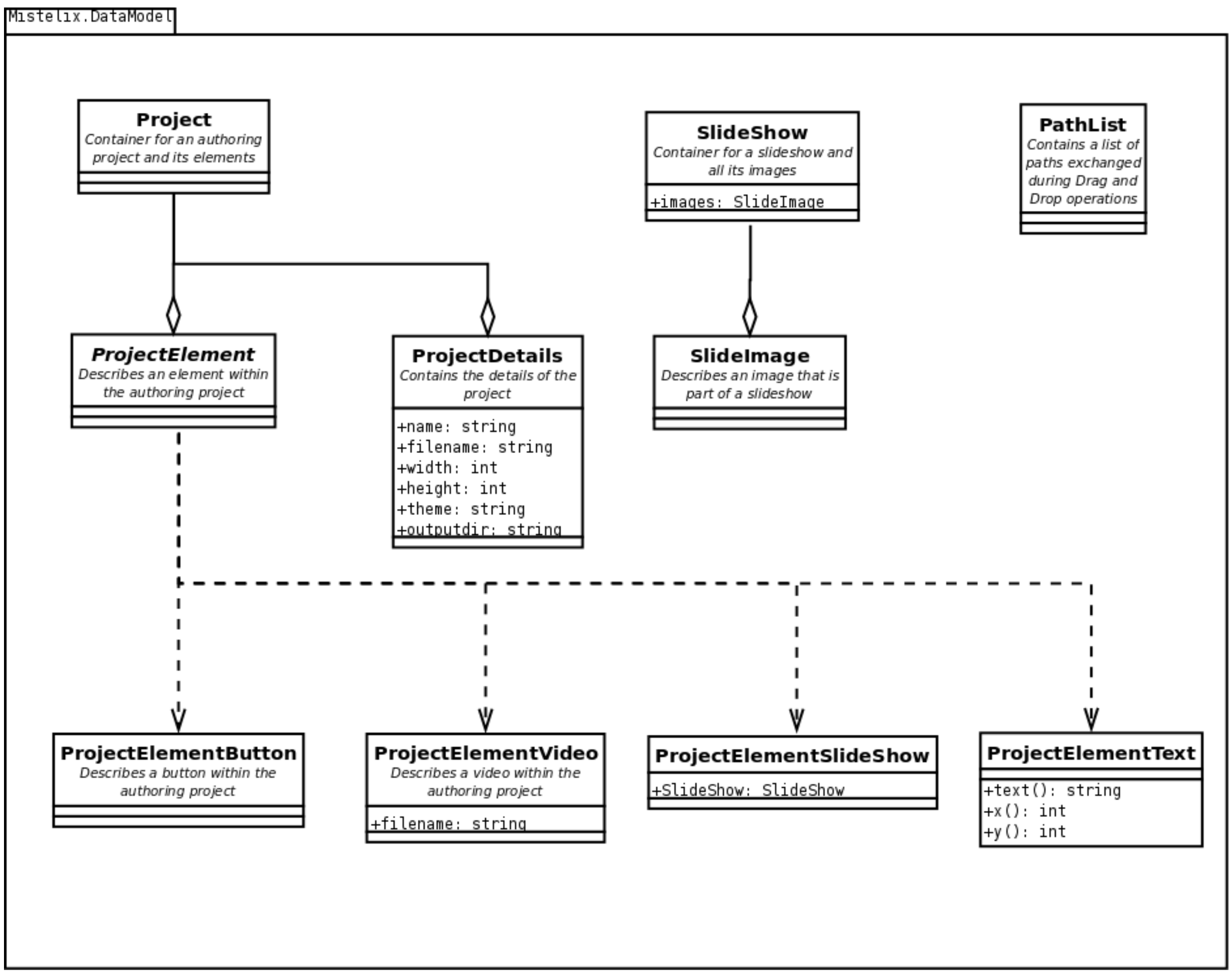
Taken from Ohloh Mistelix's analysis page[51]

6.3 Mistelix classes

This a description of the main classes that group Mistelix functionality.

6.3.1 Mistelix.Datamodel namespace

The *DataModel* namespace contains classes that describe the data model used by Mistelix.



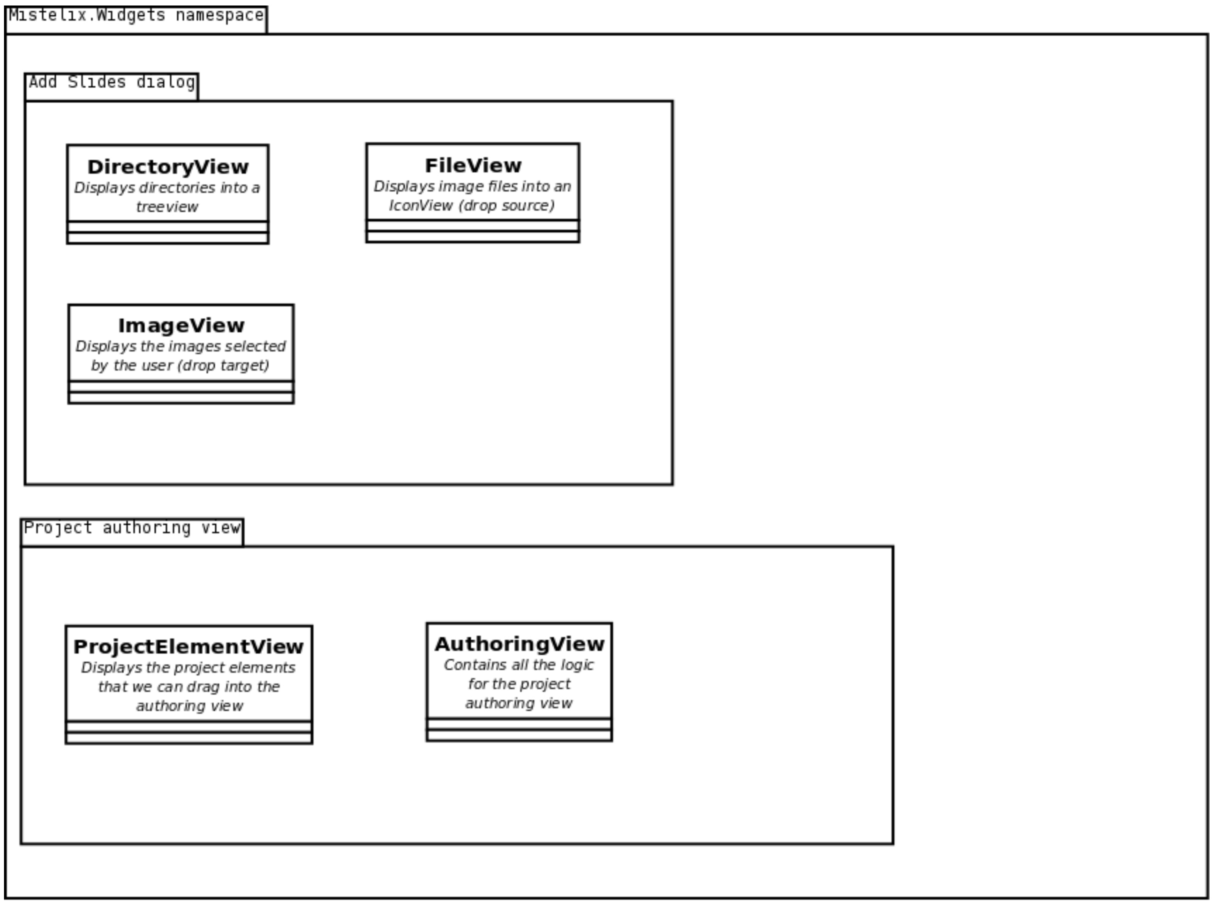
Mistelix.DataModel
Project
Container for an authoring project and its elements
SlideShow
Container for a slideshow and all its images
+images: SlideImage
PathList
Contains a list of paths exchanged during Drag and Drop operations
ProjectElement
Describes an element within the authoring project
ProjectDetails
Contains the details of the project
+name: string
+filename: string
+width: int
+height: int
+theme: string
+outputdir: string
SlideImage
Describes an image that is part of a slideshow
ProjectElementButton
Describes a button within the authoring project
ProjectElementVideo
Describes a video within the authoring project
+filename: string
ProjectElementSlideShow
+SlideShow: SlideShow
ProjectElementText
+text(): string
+x(): int
+y(): int

Mistelix's Datamodel namespace

51 http://www.ohloh.net/p/mistelix/analyses/latest

6.3.2 Mistelix.Widgets namespace

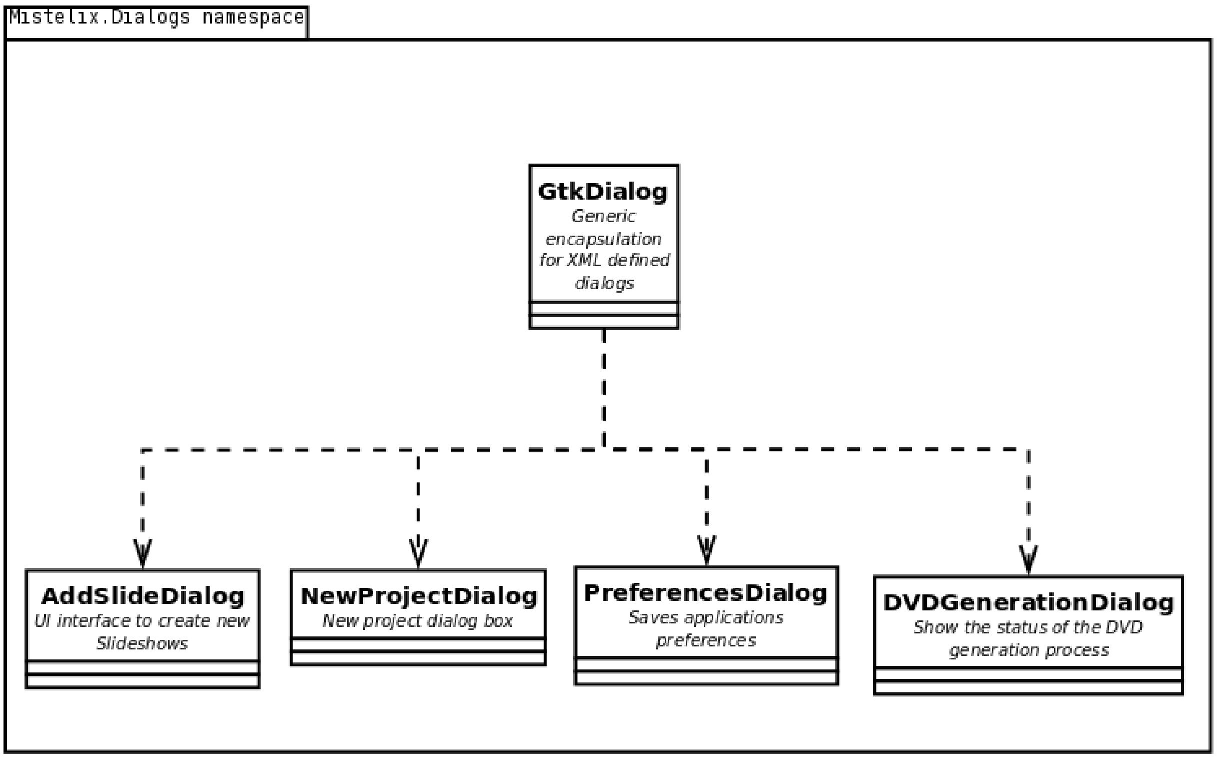
The *Mistelix.Widgets* namespace contains all the classes that encapsulate the widgets user interace.


Mistelix.Widgets namespace
Add Slides dialog
DirectoryView
Displays directories into a treeview
FileView
Displays image files into an IconView (drop source)
ImageView
Displays the images selected by the user (drop target)
Project authoring view
ProjectElementView
Displays the project elements that we can drag into the authoring view
AuthoringView
Contains all the logic for the project authoring view

Mistelix's Widget namespace

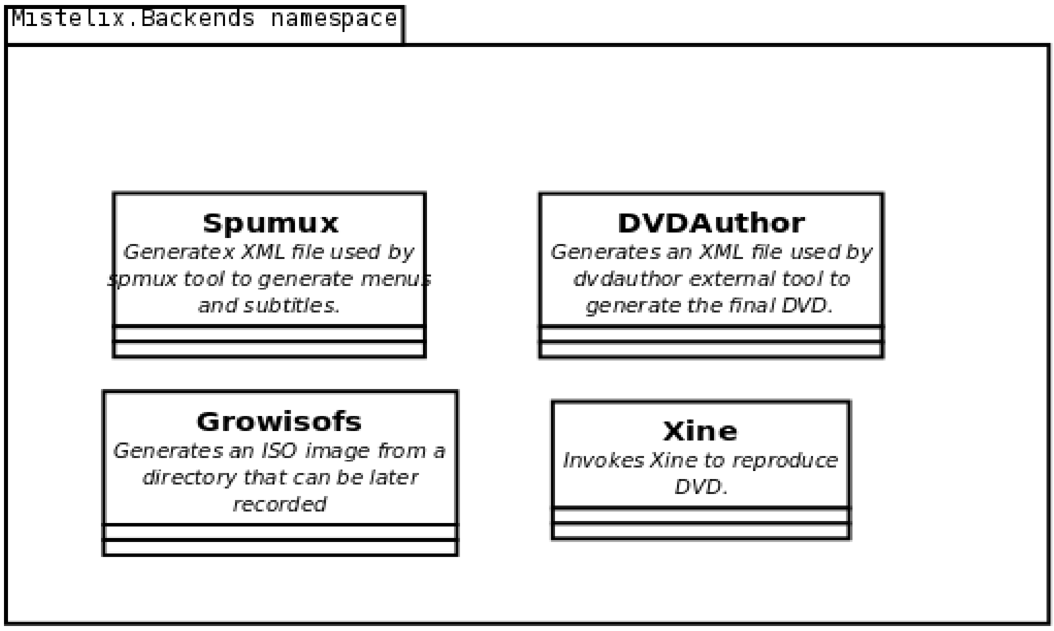
6.3.3 Mistelix.Dialogs namespace

The *Mistelix.Dialogs* namespace contains the classes that encapsulate the dialog boxes user interface.


Mistelix.Dialogs namespace
GtkDialog
Generic encapsulation for XML defined dialogs
AddSlideDialog
UI interface to create new Slideshows
NewProjectDialog
New project dialog box
PreferencesDialog
Saves applications preferences
DVDGenerationDialog
Show the status of the DVD generation process
Mistelix's Dialog namespace

6.3.4 Mistelix.Backends namespace

The *Mistelix.Backends* namespace contains classes that encapsulate the different external tools.


Mistelix.Backends namespace
Spumux
Generatex XML file used by spmux tool to generate menus and subtitles.
DVDAuthor
Generates an XML file used by dvdauthor external tool to generate the final DVD.
Growisofs
Generates an ISO image from a directory that can be later recorded
Xine
Invokes Xine to reproduce DVD.
Mistelix's Backends namespace

7. Further development and plans

Until now we have been covering Mistelix's deliverables for the UOC final year project. However, the plans from the very beginning included to further develop Mistelix and make it available to the GNU/Linux community.

To make this happen, further work has to be carried developing the program, packing it for different GNU/Linux distributions to easy its dissemination and installation, and more important, a few legal challenges have to be overcome.

The current plan is to publish Mistelix version 0.1 during May 2009.

7.1 Publishing Mistelix 0.1

This the list of tasks to be performed before making Mistelix publically available.

7.1.1 Software enhancements

Building on top of the work done for the UOC final year project, the following development will be done to make Mistelix usable to a wider number of users and suitable for distribution.

Major enhancements

- Add support for creating slideshows targeting using Theora[52] and Vorbis[53] open formats. These video and audio formats have no know patent claim issues and make possible for GNU/Linux distributions to include Mistelix with a minimum of functionality. If the users want to extend the capabilities to generate a DVD compatible media, they will require an extra plug-in.
- Plug-ins capabilities analysis. Show the user which capabilities are available with the current set of plug-ins and external tools installed within the user's system. Suggest which next steps the user has to perform to enable currently disabled functionality.

Minor enhancements

- All the pre-thumbnails should be of the same size that the final image and self-drawn using Cairo.
- Enhance current theme definition to include audio support.
- Support for aspect ratios (16:9, 4:3). Aspect ratio is the ratio of width to height of a television set. Traditional television sets have a 4:3 aspect ratio. Wide screen television sets have a 16:9 aspect ratio.

This development should take about one and half months of a full time developer.

52 http://www.theora.org/
53 http://www.vorbis.com/

7.1.2 Packaging

Proper packaging is key to software dissemination in the open and free software world. I have previous GNU/Linux distribution packaging experience since I packaged software before, most notably the game gbrainy[54] available for all major GNU/Linux distributions.

For legal reasons (see the section below called **Legal strategy**) Mistelix will be distributed in two separated parts:

- Mistelix, the main application able to author projects, but only able to build export them in Theora and Vorbis formats. These are open formats free of patents claims. This part of the application can be included in any GNU/Linux distribution.
- An additional plug-in that when installed enables MPEG2 capabilities and allows to produce DVD formats. This part of the application cannot be included in GNU/Linux distributions and will be distributed from the web site www.mistelix.org.

Initially, Mixtelix will be packaged for the following GNU/Linux distributions:

- Opensuse and Fedora using the free Opensuse build service[55]. This is an automatic build system that with the right configuration files produces RPM packages for Fedora and Opensuse distributions. See for example the gbrainy's packages that I produced[56] using this service. Additionally, when the files are published using this system, they become automatically available to any Opensuse distribution.
- Ubuntu and Debian packages will be created manually by Siegfried-Angel Gevatter Pujals[57], a Debian and Ubuntu developer that has produced already packages for other pieces of software that I wrote, like gbrainy.

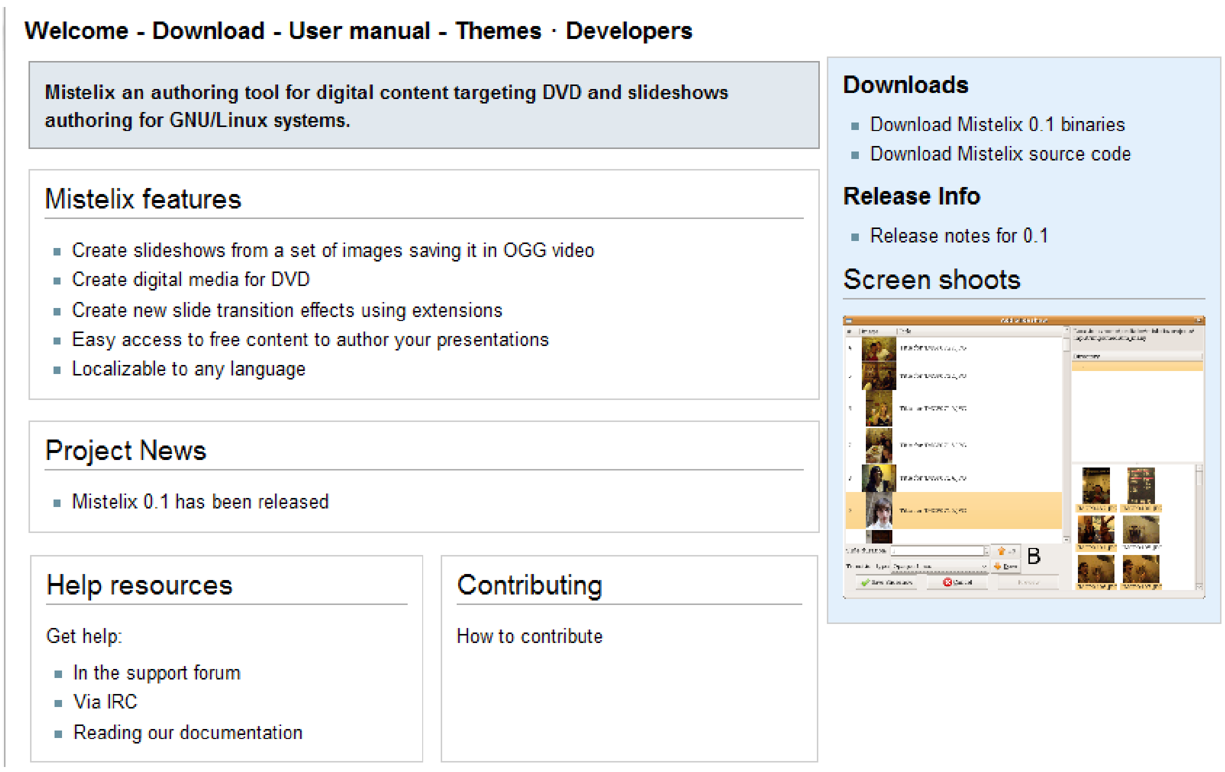
The plug-in that enables MPEG2 capabilities and allows to produce DVD formats will be available only from www.mistelix.org web site.

7.1.3 Mistelix.org web site

As part of Mistelix software project a web sited called www.mistelix.org will be created. This site plays an important role on the distribution of the software. The objectives of the web site are:

- Provide information to Mistelix users about how to user the application (e.g user manual, how to, etc.)
- Distribute the MPEG-2 codecs that can be a source of patent claims
- Build a developers community around Mistelix project.
  - Information about how to participate in the development
  - Information on how to build extensions

54 http://live.gnome.org/gbrainy/
55 http://en.opensuse.org/Build_Service
56 http://download.opensuse.org/repositories/home:/jordimas/
57 https://wiki.ubuntu.com/RainCT



Welcome - Download - User manual - Themes · Developers

Mistelix an authoring tool for digital content targeting DVD and slideshows authoring for GNU/Linux systems.

Mistelix features

- Create slideshows from a set of images saving it in OGG video
- Create digital media for DVD
- Create new slide transition effects using extensions
- Easy access to free content to author your presentations
- Localizable to any language

Project News

- Mistelix 0.1 has been released

Help resources

Get help:

- In the support forum
- Via IRC
- Reading our documentation

Contributing

How to contribute

Downloads

- Download Mistelix 0.1 binaries
- Download Mistelix source code

Release Info

- Release notes for 0.1

Screen shoots

Mock-up of a design proposal of Mistelix web site

Mistelix.org will be hosted at Universitat de Lleida network after sign up and agreement with their information technology department.

7.1.4 Legal strategy

As described in the **project risks section** at the introduction of this document, the major challenge that Mistelix faces for its distribution is legal (patent claims).

These potential claims can affect three kinds of different people:

- Developers. People that writes Mistelix, initially me but also other people that may join the project in the future, including also translators or extensions developers.
- Users. People using the software. The patent legislation varies from country to country and users responsibilities vary depending on the country that the user resides.
- The people hosting the web site mistelix.org, in this case Universitat de Lleida.

Current strategy for distributing Mistelix

Mistelix will be packed for the most popular distributions (Debian, Ubuntu, Fedora and OpenSuse). The packaged version will not contain the MPEG-2 encoders, only Theora and Vorbis support. The idea is:

- The user gets Mistelix from the standard distribution repositories for his GNU/Linux distribution. This version cannot export DVD since it has no MPEG2 encoders. It can only create open Theora and Vorbis open formats.
- When the user wants to export into a DVD, Mistelix will ask to download additional software in binary form from www.mistelix.org web site. The downloaded software adds MPEG-2 encoding capabilities.
- The web site www.mistelix.org will contain a package with "ffmpeg with MPEG-2 encoding support enabled" that will enable this functionality in Mistelix.
- The web site www.mistelix.org server will be placed in Universitat de Lleida (http://www.udl.es) in Spain.

The rationale is that since the server is located in Spain and software patents are not supposedly to be legal in Europe, developers and users are safe from patent claims.

I have been working with the **Software Freedom Law Center**[58] to analyze the best distribution strategy for Mistelix. This center provides free of charge legal representation and related services to protect and advance free and open source software. They work mainly on the areas of License Defense and Litigation Support and Legal Consulting.

The Software Freedom Law Center has a very reputed legal team[59] led by Eben Moglen[60], a very well know free software advocate.

According emails exchanged with Daniel B. Ravicher, Legal Director at Software Freedom Law Center "*This strategy does pose risk (everything you do poses risk), but it seems to be a reasonable way to proceed, at least until you receive a direct communication from a patent holder that they object to such.*"

I have decided to proceeded with the publication of Mistelix 0.1 after the necessary enhancements described as next steps (Software enhancements, packing, publication of the web site, etc) are completed.

7.2 Future versions (beyond 0.1)

These is a list of major features from a high level respective that can be consider for future versions of the application.

Major features

- Add support for multilevel DVD menus. Currently one single main menu is supported.
- DVD Subtitle support.
- Voice recoding for slideshows.

58 http://www.softwarefreedom.org/
59 http://www.softwarefreedom.org/about/team/
60 http://en.wikipedia.org/wiki/Eben_Moglen

- Adobe Flash export export support (probably built on top of swfmill[61]) for web publishing.
- Microsoft Silverlight export support (built using Moonlight[62]) for web publishing.
- Support for 3D effects based on OpenGL

61 http://osflash.org/swfmill
62 http://www.mono-project.com/Moonlight

8. Conclusions

On the tools used

The flexibility of the Microsoft .Net framework has been invaluable. Writing the logic of Mistelix using a high level language like C#, leveraging on the very power library class that the .Net framework provides plus been able to interface with low level libraries using marshalling[63] and DllImports[64] have been prove to be a very efficient platform to deliver a desktop application.

Working with the Mono platform has been an enriching experience. The level of conformance of the implementation compared to the official ECMA standards 334[65] (C# Language Specification) and 335[66] (Common Language Infrastructure) is almost perfect. Also, the level of conformance with the Microsoft C# 2.0 compiler and runtime is extremely high. The only shortcoming of the Mono platform is the lack of a debugger what forces you to use other methods that are widely documented[67]. However, a debugger will be available during early 2009.

During the development of Mistelix I used Monodevelop[68] as development IDE. It has proved to be a complete IDE suitable for developing complex applications. On top of the features that you expect from a modern IDE, Monodevelop also has very good User Interface designed called Stetic[69] that has been very valuable during the creation of Mistelix's user interface.

The fact the tools used during the development are open source is very convenient since it allowed me to inspect the source code when reading the documentation of an API is not clear enough. It also made possible to contribute back fixes for some bugs that I found. For example, during the development of Mistelix I noticed that one of the signatures of the C# bindings for accessing the Cairo libraries was wrong. I did prepare a fix for the issue and send it to the Mono Subversion source control repository (see revision 120.299[70] in the Mono project).

Using Google Code has been also positive. During the development of this project I use their Subversion source control system to develop Mistelix that had produced 86[71] major code changes during its development. This has allowed me to keep track of the changes on the source code and help me when I found regressions bugs.

On the challenges faced

During the development of this project I faced several challenges. Let me summarize the most important ones:

- Selecting the right multimedia stack. There are many open source libraries that initially seemed to provide the necessary multimedia features to implement a DVD authoring

63 http://en.wikipedia.org/wiki/Marshalling_(computer_science)
64 http://msdn.microsoft.com/en-us/library/aa984739(VS.71).aspx
65 http://www.ecma-international.org/publications/standards/Ecma-334.htm
66 http://www.ecma-international.org/publications/standards/Ecma-335.htm
67 http://www.mono-project.com/Debugging
68 http://www.monodevelop.com
69 http://monodevelop.com/Stetic_GUI_Designer
70 http://archive.netbsd.se/?ml=mono-patches&a=2008-11&m=9203502
71 http://code.google.com/p/mistelix/source/list

system. However only very few of them can provide the necessary support required for DVD authoring. Additionally, I have been careful about the different licenses that these libraries have and if they are compatible with Mistelix license.

- Making sense of the DVD standards. There are many DVD standards ranging from different physical disc formats to different encoding mechanisms to store video and audio. Selecting the appropriate formats and parameters to use to make sure that the produced DVD can be played in the larger number of DVD players has represented lots of work.
- Understanding Gstreamer. Gstreamer is a complex and very large multimedia infrastructure. A part of the documentation available in their site[72], there is very little additional documentation. For example, there are no books available or advanced tutorials. Additionally, some of the more than 1.000 plug-ins available are very poorly documented, including gst-ffmpeg[73] that Mistelix uses. To overcome this limitation, I have invested many hours browsing and analyzing the Gstreamer source code and analyzing how some open source applications like Banshee[74] audio player or Totem[75] video player use it.
- Legal issues. As mention during the project, to overcome possible patent claims has been one of the challenges of this project. It could affect its distribution and have legal consequences for its developers and users. Finally, I have crafted a distribution strategy that minimizes the rights of patents claims and that has been acknowledged by Software Freedom Law Center as one of the most low risk strategies possibles.
- Feature set and user interface. The DVD Video standard offers many features. To select a small set of features that can be delivered with an easy a solid user interface has proved to be not trivial. Focusing on the use cases of authoring DVD slideshows and videos using a modern drag and drop interface I think has been the right option.

On lessons learned

Despite the fact that I already had experience on some of the following areas, this project has helped me to consolidate or learn new things in the following areas:

- **How to design and plan a full development**. Mistelix represents a complete software project, covering a full development cycle, including functional description, architecture, implementation or the deployment among users.
- **What is the the state of art of the multimedia frameworks for Linu**x. As mentioned before, there are many multimedia frameworks, understand which frameworks exist and which functionality they provide.
- **Understand better Microsoft .Net framework and C#**. This actually a platform that is not learned during the studies at UOC, since the teaching focuses on the Java platform. It is always good to extend or enhance your knowledge to other platforms.
- **More experience with open source tools**. Work with the most common open source

72 http://gstreamer.freedesktop.org/
73 http://gstreamer.freedesktop.org/modules/gst-ffmpeg.html
74 http://banshee-project.org/
75 http://projects.gnome.org/totem/

libraries, tools and frameworks is an enriching experience that I can apply to my day per day job or future jobs.

On future development

On top of overcome these challenges and the learning experience, I think that is also important to highlight that this project will be developed further and it will be made available to the GNU/Linux community.

To make this happen, further work will be carried developing the program, packing it for different GNU/Linux distributions to easy its dissemination and installation. The current plan is to publish Mistelix version 0.1 during May 2009.

9. Bibliography

This is the list of books used during this project

Taylor, Jim; Johnson, Mark, Crawford Charles. DVD demystified (third edition). McGraw Hill 2006.

Taylor, Jim; Johnson, Mark, Crawford Charles. High Definition DVD handbook. McGraw Hill 2007.

Lirio, Antonio. Adobe Encore DVD 2.0. Anaya 2006.

Albahari, Joseph. C# 3.0 in a Nutshell: A Desktop Quick Reference. O'Reilly 2007.