GStreamer SDK

Andoni Morales Alastruey (ylatuya)
GStreamer Conference, 27th of August 2012 - San Diego

The **GStreamer SDK** is a joint initiative from **Fluendo and Collabora** to provide a **stable and tested** distribution of GStreamer for the most common platforms (**Linux, Windows and OS X**).

- The first release of the SDK, **Amazon** 2012.5, was done the end of May.
- A new bugfix release for Amazon is scheduled within the next weeks.
- The next release will be named **Brahmputra**, with support for **Android**.

Topics:

- Goals of the SDK
- SDK's components
- Developments done for the SDK
- Building the SDK
- Packaging the SDK
- Integration with the native OS development tools (VS, XCode)
- cerbero: the build system used to build and package the SDK

1. Goals of the SDK

gStreamer SDK

FLUENDO Influencing the Multimedia World

Lower the barrier to entry for new GStreamer adopters

- GStreamer is a huge project and it's sometimes hard to get started with
- The current documentation is very good, but it can be overwhelming for developers starting with GStreamer.
- It's hard to evaluate GStreamer as a solution for a project in a quick way.

✔ A new **documentation** site where GStreamer is described in a more generic way for newcomers.

✔ Create a new set of **tutorials to introduce progressively** new developers in the GStreamer concepts.

Promote GStreamer as a multi-platform multimedia framework

- GStreamer is the best multimedia framework on Linux (isn't it?), but still not very popular in other platforms.

- It's a cross-platform framework, but we don't provide installers/packages for Windows and OS X

- Developing and distributing GStreamer apps is rather tricky on Windows and OS X (no package managers like yum or apt-get to resolve deps ☹)

✔ Provide **installers with pre-compiled binaries**.

✔ Provide a **complete development environment** to write GStreamer apps in all the platforms that integrates with the system development tools.

✔ Make it **easier to distribute** GStreamer apps.

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Stable distribution of GStreamer with the same features in all the platforms

- It's hard to debug issues when there are several platforms with several distributions, each one using different versions.

- Upgrades are usually very problematic and requires many hours in testing.

✔ Using the **same software versions** across platforms.

✔ **Smooth upgrades** trying to reduce regressions as much as possible with a good testing framework like insanity.
(Easier in the 0.10 branch now that it is in maintenance mode ☺)

✔ Give some love to the **system plugins**:
  - Navigation interface and force-aspect-ratio implemented in all the video sinks.
  - Audio passthrough working in all audio sinks.
  - osxvideosink working from the command line.

Supported Platforms and Distributions:

- Linux:
  - Debian
  - Ubuntu
  - Fedora
- Windows:
  - XP
  - Vista
  - Windows 7
- OS X:
  - Snow Leopard
  - Lion
  - Mountain Lion (needs more testing)
- Android (next release)

2. SDK components

- The first version of the SDK is focused on **multimedia playback**
- Using the **0.10** branch as it's more mature and stable than the upcoming 1.0:

  - Local and network sources to support local and remote media playback
  - Filter elements, decoders and demuxers
  - Audio and video sinks
- Also provide a GUI toolkit, language bindings and a showcase app:

  - Gtk+ (2.0)
  - Python bindings
  - Snappy

The SDK is modularized in several packages:

- A single big bundle doesn't fit everybody's use cases.
- Application developers can decide which parts of the distribution to include.
- Allows to separate conflictive plugins (licenses or patents)

The GStreamer components in the SDK are **split in packages by functionality** rather than the current -base -good -bad and -ugly.

It makes more sense for bundling GStreamer with applications.

It was hard to split everything, but that's the best we could do...

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Packages installed by default:

- **base-system**: Base system libraries such as glib, libtiff, libz, ...
- **gstreamer-core**: GStreamer core (core libraries and utils)
- **gstreamer-codecs**: Decoding and demuxer plugins
- **gstreamer-playback**: Playback elements (eg: playbin, uridecobin)
- **gstreamer-effects**: Effects, instrumentation and misc plugins (eg: deinterlace, videobox, alphcolor)
- **gstreamer-networking**: Networking plugins (eg: souphttpsrc, rtspsrc)
- **gstreamer-capture**: Capture plugins (eg: v4l2src, ksvideosrc, osxaudiosrc)
- **gstreamer-visualizers**: Visualization plugins (eg: goom)
- **gstreamer-editing**: Editing plugins (gnonlin, ges)
- **gstreamer-python**: Python bindings
- **gstreamer-clutter**: Clutter support
- **gtk+**: Gtk+-2.0
- **gtk-python**: Gtk+ python bindings
- **gobject-python**: pygobject
- **snappy**: Snappy video player

Optional packages:

- **gstreamer-codecs-restricted**: Codecs plugins with potential patents issues (eg: a52dec mpeg2dec)
- **gstreamer-codecs-gpl**: Codecs plugins that depends on GPL libraries
- **gstreamer-networking-restricted**: Networking plugins with potential patents issues (eg: mmssrc)
- **gstreamer-dvd**: DVD plugins
- **gstreamer-ffmpeg**: FFmpeg plugin

3. Developments for the SDK

- Testing Framework:
  - Insanity
  - Collect sample for the testig framework
  - Manual and automated test suites for:
    - Local media: container/codecs discovery, seeking, trick modes, frame stepping, deinterlacing
    - Live streaming: RTP/RTSP, HTTP, HLS
    - DVD
    - Container and codecs: video decoders, audio decoders, subtitles
- Video base clases:
  - Finish the video base and move them to Base
  - Port x264enc, mpeg2dec, gst-ffmpeg, png and jpeg plugins
- Audio sinks:
  - direcsoundsink: fixed audio passthrough
  - osxaudiosink: audio passthrough support
- Video sinks:
  - d3dvideosink: stabilization
  - osxvideosink: navigation, make it work with gst-launch, force-aspect-ratio
- HLS demuxer
- Open Source Fluendo's ADPCM and iLBC plugins
- Improved support for static builds

4. Building the SDK

Building a fully featured GStreamer is very similar to assemble a mini Linux distribution, with lots of packages dependencies introduced by the plugins.

We are currently building arround **100 projects**, and the list is growing.

The first big question was, **what build system should we use?**

Maintaining several build systems, one for each platform, was completely out of scope.

The main requirement was a cross-platform build system, aware of host and target **platform**, **architecture** and **distribution**

This leaves out very good build systems like:

- jhbuild
- macports
- homebrew

The perfect candidate:

- Entropy Wave's build system (also used to build a SDK for GStreamer)

The main problem is that it's shell script, so it would be harder to extend and add new features, like native packaging.

And cerbero was born...

- Written in python
- Cross-platform: aware of host and target platform, architecture and distribution
- Support for cross-compilation (either to a different arch or to a different platform)
- Projects are described with recipes files, a single one shared across platforms.
- Recipes are python classes, so it's easy to extend and adapt them to the buggy build systems.
- Support for creating native packages
- Source backends: git, svn, tarballs (backends are very easy to add)
- Build backends: autotools, automake, CMake
- Bootstrapping
- More in the last section...

Build system minimum requirements:

- A toolchain:
  - compiler (GCC)
  - binutils
- Build systems:
  - autotools
  - cmake
- libtool
- pkg-config
- python
- git

How do we get them for each platform?

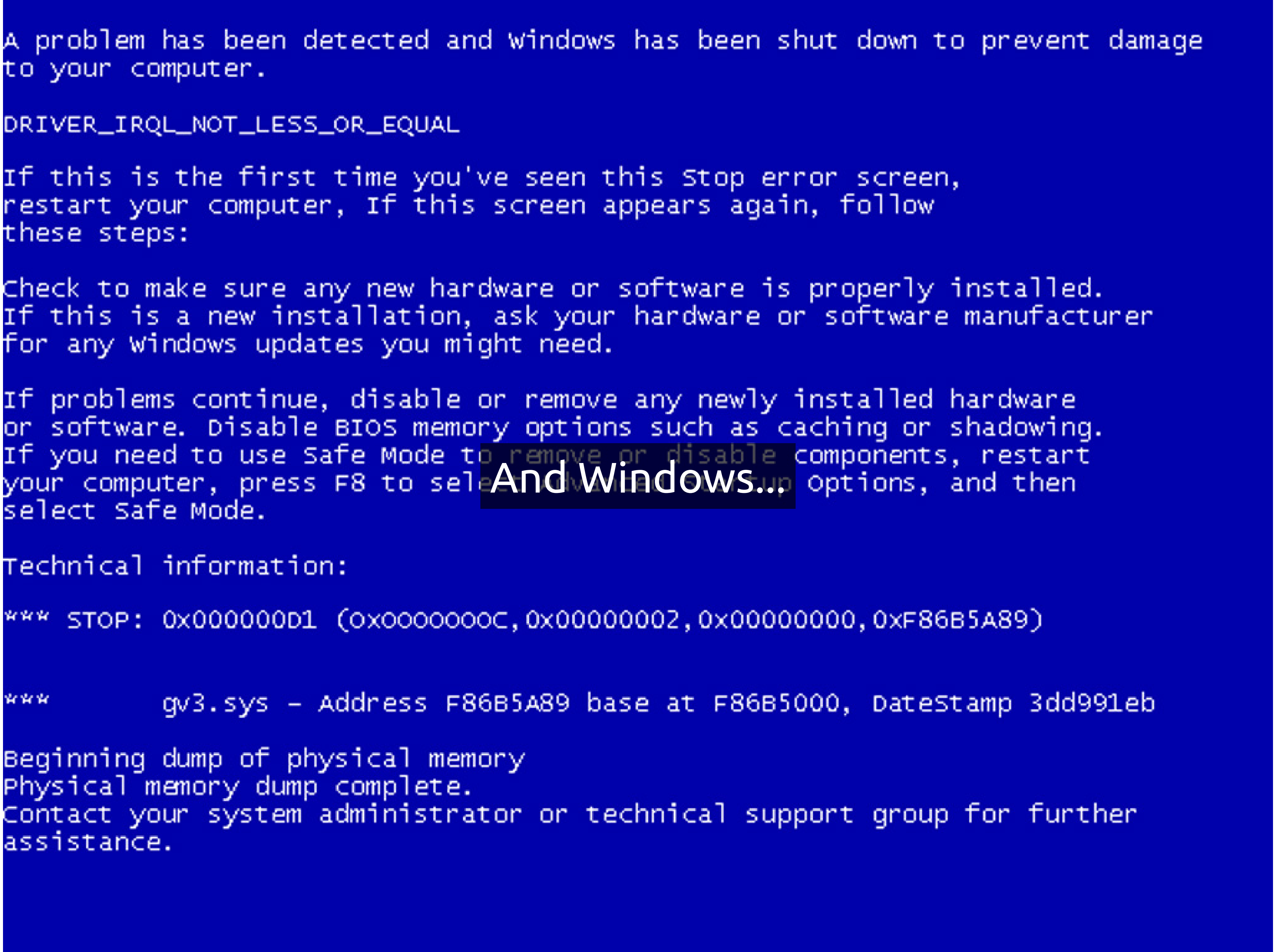
Linux distributions make our live **much** easier!

Seriously... all distributions have a decent package manager to install everything.

OS X is not that bad.

It comes with a good toolchain and most of the required software.

We still need to install automake, autoconf, aclocal, libtool and pkg-config with macports, but just because they are not yet built by cerbero.



```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)


***        gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```

And Windows…

We need everything:

- **shell**: msys shell installed with mingw-get (provides also auto* and libtool)
- **toolchain**: built with the great MinGW-w64 project
- **python**
- **git**
- **pkg-config**

To make things easier cerbero installs the toolchain as part of the bootstrap, so you can build the sdk your self with 2 simple commands:

```
$ cerbero bootstrap
$ cerbero package gstreamer-sdk
```

mingw-get, python and git must be installed manually.

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Builds are **10x~20x** slower compared to linux:

- no native 'fork', emulated by msys which makes configure scripts **very slow**
- strings manipulations in 'make' mapping unix paths to windows paths.

Speed up builds:

- using a unique cache file for all the builds (./configure --cache-file=/path/to/cache)
- ccache

Cross-compilation is much faster, but the SDK can't be completely cross-compiled because some bugs in Wine affects WiX, the packaging tool used for Windows.

Getting a working toolchain was painful.

MinGW-w64 doesn't provide a pre-built toolchain with a stable GCC version so we had to build it.

The build process of the toolchain is now integrated with cerbero:
  1) Build a cross-compiler toolchain for windows (32 and 64 bits), mingw-w64 and binutils
  2) With this new toolchain, cross-compile a native windows toolchain, mingw-w64 and binutils

There is still a problem with the GCC builds on Windows, they don't provide debug symbols for VS.

In the future we will also try to provide builds with Microsoft's compiler.

gStreamer SDK

FLUENDO
Influencing the Multimedia World

For all the platforms we provide builds for 32 and 64 bits (even on windows)

For OX X, in the first release we provided 2 separate builds for i386 and x86_64, but we will ship **universal builds** in the next release.

The support for universal builds in cerbero is hackish.
The sources are configured twice in separate build tree, for the 2 target archs and then merged into universal mach-o binaries using a tool named **lipo**
We should avoid this for projects that supports universal builds.

4. Packaging

- The SDK is packaged using the native packaging tools available for the target platform/distribution
- The smallest packaging unit are the packages mentioned previously (base-system, gstreamer-core, etc...)
- Each package is split in a runtime and a development version.
- Meta packages groups packages to create the final installer

On Linux:

- Debian/Ubuntu: dpkg-buildpackage
- Redhat/Fedora: rpmbuild
- Packages -> rpm/deb regular packages
- Meta Packages -> rpm/deb metapackage
- Install Path -> /opt/gstreamer-sdk

On OS X:

- PackageMaker: a tool to create installer packages (.pkg)
- Packages -> installer package
- Meta Packages -> Also called Meta package (installer package with selection step)
- Install Path -> /Library/Frameworks/GStreamer.framework

On Windows:

- WiX: Windows Installer XML toolset
- Packages -> Merge Modules (.msm)
- Meta Packages -> MSI installer (.msi)
- Install Path -> c:\gstreamer-sdk\0.10\
- Environment variables: GSTREAMER_SDK_ROOT_X86 and GSTREAMER_SDK_ROOT_X86_64
- GSTREAMER_SDK_ROOT_X86 is not added to PATH (avoid the "DLL Hell")

gStreamer SDK

FLUENDO Influencing the Multimedia World

5. Integration with the operating system's IDE and tools

One of the goals of the SDK was to provide a complete development environment that could be used with the system tools, being XCode in OS X and VS in Windows.

People that comes from the Linux world are very comfortable within a shell, but I don't see a pure Windows developer using MinGW.

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Linux is great…

- We have a shell
- We have the autotools! (yes, we love them…)
- We have libtool! (and yes.. we also love it)
- We have pkg-config

A shell script, named gst-sdk-shell, sets the right environment to target the SDK, and that's it ☺

gStreamer SDK

FLUENDO
Influencing the Multimedia World

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Windows

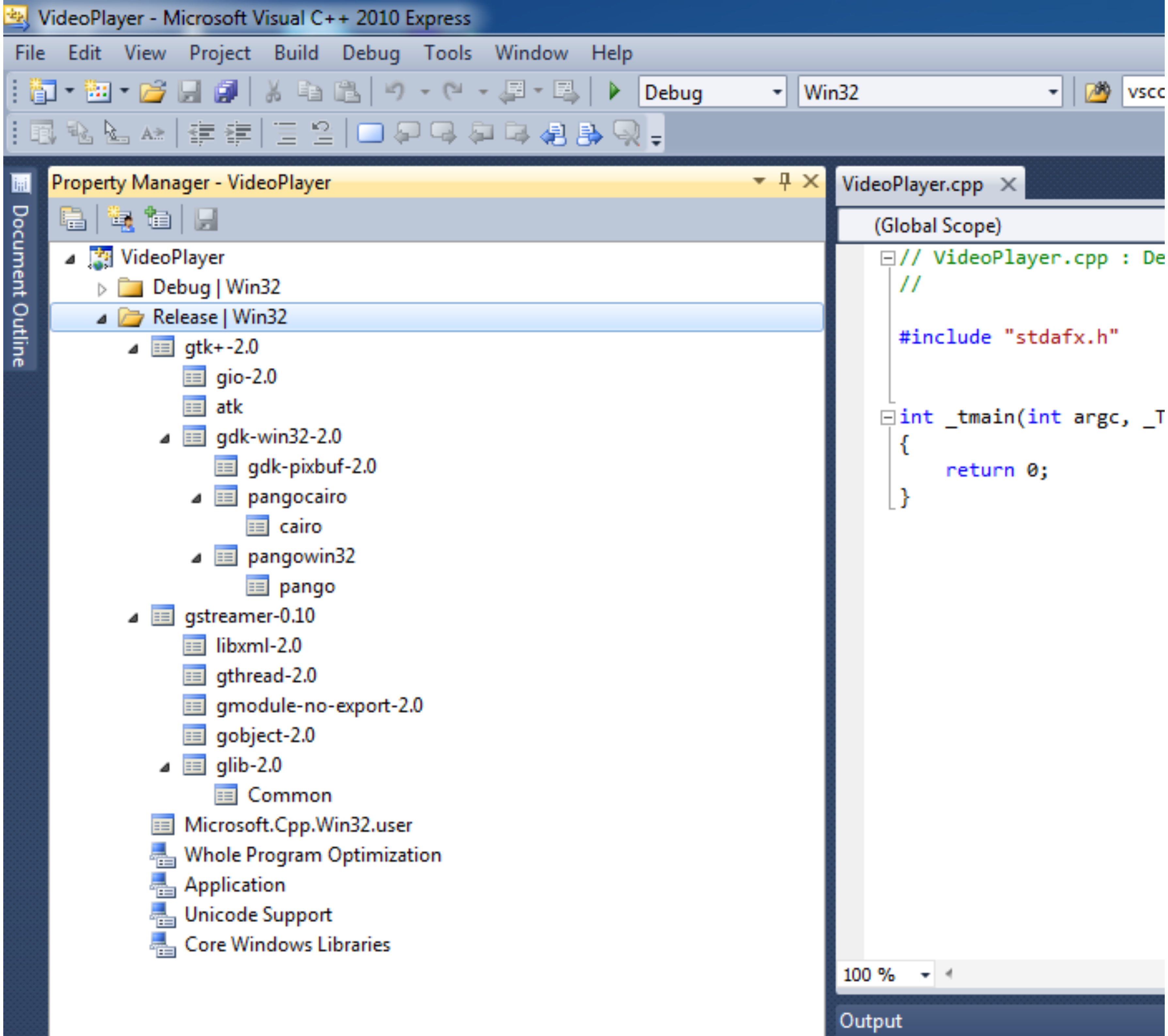
To link the DLL's with Visual Studio on Windows an **import library (.lib)** is needed.
A post build step in cerbero takes care of that, so for each shared library a .lib is created

We wanted to make it **very easy to use** any of the available libraries in the SDK from VS, like it's done on Linux with pkg-config and the -libs and --cflags options.

VS projects can include property sheets that can sets the paths for headers and libraries and the libraries to link.

cerbero includes a tool that **parses .pc files and creates an equivalent .props file** that can be easily included in VS.

```
<?xml version="1.0" ?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
        <ImportGroup Label="PropertySheets">
                <Import Condition="$(glib_2_0Imported)!=true" Project="glib-2.0.props"/>
                <Import Condition="$(gobject_2_0Imported)!=true" Project="gobject-2.0.props"/>
                <Import Condition="$(gmodule_no_export_2_0Imported)!=true" Project="gmodule-no-export-2.0.props"/>
                <Import Condition="$(gthread_2_0Imported)!=true" Project="gthread-2.0.props"/>
                <Import Condition="$(libxml_2_0Imported)!=true" Project="libxml-2.0.props"/>
                <Import Condition="$(CommonImported)!=true" Project="Common.props"/>
        </ImportGroup>
        <PropertyGroup Label="UserMacros"/>
        <PropertyGroup>
                <gstreamer_0_10Imported>true</gstreamer_0_10Imported>
        </PropertyGroup>
        <ItemDefinitionGroup>
                <ClCompile>
                        <AdditionalIncludeDirectories>$(GSTREAMER_SDK_ROOT_X86)\include\gstreamer-0.10;%(AdditionalIncludeDirectories)</AdditionalIncludeDirectories>
                </ClCompile>
                <Link>
                        <AdditionalLibraryDirectories>;%(AdditionalLibraryDirectories)</AdditionalLibraryDirectories>
                        <AdditionalDependencies>gstreamer-0.10.lib;%(AdditionalDependencies)</AdditionalDependencies>
                </Link>
        </ItemDefinitionGroup>
        <ItemGroup/>
</Project>
```



VideoPlayer - Microsoft Visual C++ 2010 Express

Property Manager - VideoPlayer

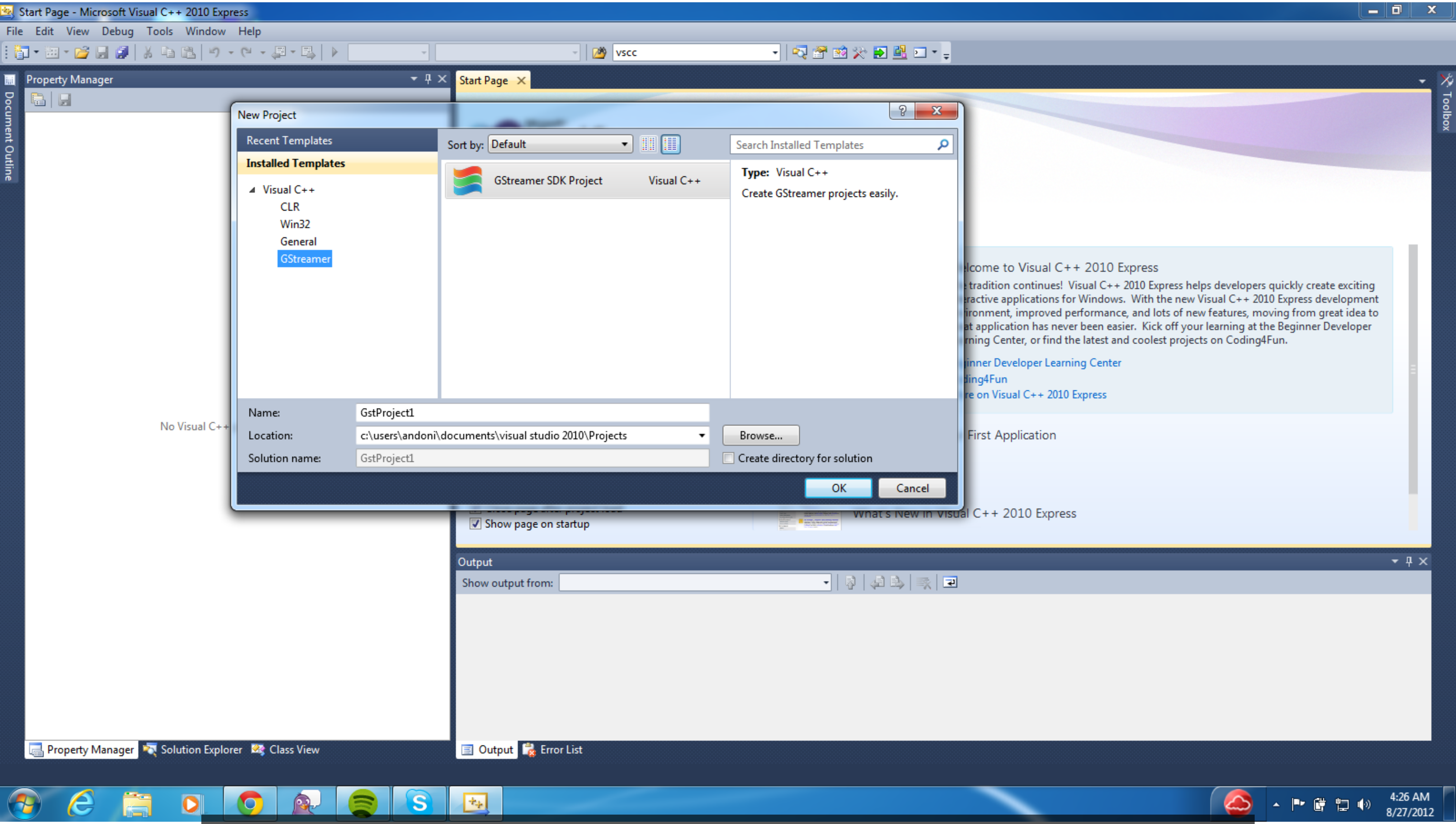
```
VideoPlayer
    Debug | Win32
    Release | Win32
        gtk+-2.0
            gio-2.0
            atk
            gdk-win32-2.0
                gdk-pixbuf-2.0
                pangocairo
                    cairo
                pangowin32
                    pango
        gstreamer-0.10
            libxml-2.0
            gthread-2.0
            gmodule-no-export-2.0
            gobject-2.0
            glib-2.0
                Common
        Microsoft.Cpp.Win32.user
        Whole Program Optimization
        Application
        Unicode Support
        Core Windows Libraries
```

VideoPlayer.cpp

```
// VideoPlayer.cpp : De
//

#include "stdafx.h"


int _tmain(int argc, _T
{
    return 0;
}
```


Start Page - Microsoft Visual C++ 2010 Express

New Project

Recent Templates

Installed Templates

- Visual C++
  - CLR
  - Win32
  - General
  - GStreamer

Sort by: Default

GStreamer SDK Project	Visual C++

Type: Visual C++

Create GStreamer projects easily.

Name: GstProject1

Location: c:\users\andoni\documents\visual studio 2010\Projects

Solution name: GstProject1

Create directory for solution

OK Cancel

And there is also a wizard for GStreamer projects!

6. Cerbero: the build system for the GStreamer SDK

gStreamer SDK

FLUENDO Influencing the Multimedia World

- Written in python
- Cross-platform: aware of host and target platform, architecture and distribution
- Support for cross-compilation (either to a different arch or to a different platform)
- Projects are described with recipes files, one per project and platform
- Recipes are python classes, so it's easy to extend and adapt them to the buggy build systems.
- Support for creating native packages
- Source backends: git, svn, tarballs (backends are very easy to add)
- Build backends: autotools, automake, CMake
- Bootstrapping

Commands:

- **build**: Builds a recipe
- **buildone**: Re-build a single recipe
- **package**: creates a package
- **cleanone**: cleans a recipe
- **check**: runs check on a recipe
- **list**: list available recipes
- **list-packages**: list available packages
- **shell**: starts new shell with the build environment
- **run**: runs a command with the build environment
- **bootstrap**: bootstrap the build system

Recipes:

- Describes a project, the way it's built and the files it provide
- Files with the **.recipe** extensions loaded from the recipes directory
- Python classes inheriting from *cerbero.build.build.Recipe*
- Common Fields:
  - **name**: name of the recipe
  - **version**: version of the recipe
  - **sources**: url of the sources
  - **deps**: recipe dependencies
  - **platform_deps**: platform recipe dependencies
  - **stype**: source backend (Git, SVN, Tarball, Custom)
  - **btype**: build backend (Autotools, CMake, Custom)
  - **licenses**: recipe licenses
  - **files_$CATEGORY** : list of files for a given category
  - **platform_files_$CATEGORY** : platform list of files for a given category
- Backend Fields: source or build backend specific fields

Building GStreamer:

```
$ cerbero build gstreamer gst-plugins-base gst-plugins-good gst-plugins-bad gst-plugins-ugly
```

gStreamer SDK

FLUENDO Influencing the Multimedia World

```
(file:atk.recipe)

class Recipe(recipe.Recipe):
    name = 'atk'
    version = '2.4.0'
    licenses = [License.LGPLv2Plus]
    deps = ['glib']

    files_libs = ['libatk-1.0']
    files_devel = ['lib/pkgconfig/atk.pc', 'include/atk-1.0']
    files_lang = ['atk10']
```

gStreamer SDK

FLUENDO
Influencing the Multimedia World

File categories helpers for:

- **binaries** -> *files_bins*: Handles the different extensions in several platforms

- **libraries** -> *files_libs*:
    - Different location (lib/ or bin/)
    - Different extensions(.so, .dylib, .dll)
    - Different version naming scheme (libfoo-1.so.2 vs libfoo-1.1.dylib).
    - Automatically adds development files to the development package (libfoo-1.a, libfoo-1.la, libfoo-1.dll.a, libfoo-1.so, libfoo-1.dylib)

- **python** -> *files_python*: Add byte-compiled files if present and handles different python version

- **lang** -> *files_lang*: For translation files

- **man (to be added)** -> *files_man*: manual pages

- **doc (to be added)** -> *files_doc*: documentatio

gStreamer SDK

FLUENDO Influencing the Multimedia World

```
(file:pango.recipe)

class Recipe(recipe.Recipe):
    name = 'pango'
    version = '1.30.1'
    autoreconf = True
    autoreconf_sh = 'ACLOCAL="$ACLOCAL $ACLOCAL_FLAGS" autoreconf -ivf '
    licenses = [License.LGPLv2Plus]
    configure_options = '--with-included-modules '
    deps = ['cairo', 'fontconfig', 'freetype']
    platform_deps = {
        Platform.WINDOWS: ['gtk-doc-windows'],
        Platform.LINUX: ['gtk-doc'],
        Platform.DARWIN: ['gtk-doc'],
    }


    files_libs = ['libpangocairo-1.0', 'libpango-1.0', 'libpangoft2-1.0']
    files_bins = ['pango-querymodules', 'pango-view']
    files_devel = ['include/pango-1.0',
        'lib/pkgconfig/pangoft2.pc', 'lib/pkgconfig/pango.pc',
        'lib/pkgconfig/pangocairo.pc']
    platform_files_libs = {
        Platform.WINDOWS: ['libpangowin32-1.0'],
        Platform.LINUX: ['libpangox-1.0'],
    }
    platform_files_devel = {
        Platform.WINDOWS: ['lib/pkgconfig/pangowin32.pc'],
        Platform.LINUX: ['lib/pkgconfig/pangox.pc'],
    }

    def prepare(self):
        if self.config.target_platform == Platform.DARWIN:
            self.configure_options += ' --without-x '
```

Packages:

- Describes a packages.
- Files with the **.package** extension loaded from the packages directory
- Fields:
  - **name**: name of the package
  - **shortdesc**: Short description of the package
  - **longdesc**: Long description of the package
  - **version**: version of the package
  - **codename**: codename of the release
  - **uuid**: unique id for this package
  - **license**: package license
  - **vendor**: vendor for this package
  - **org**: organization for this package (eg: net.foo.bar)
  - **url**: url for this pacakge
  - **sys_deps**: system dependencies for this package
  - **sys_deps_devel**: development system dependencies for this package
  - **ignore_package_prefix**: don't use the package prefix set in the config
  - **resources_license**: filename of the .txt license file
  - **resources_license_unwrapped**: filename of the .txt license file withouth the 80 chars wrapping
  - **resources_license_rtf**: filename of .rtf license file
  - **resources_icon**: filename of the .ico icon
  - **resources_icon_icns**: filename of the .icsn icon
  - **resources_backgound** = filename of the background image
  - **resources_preinstall** = filename for the pre-installation script

Packaging the SDK:

```
$ cerbero package gstreamer-sdk
```

4 different types of packages:

- **Package**: smallest packaging unit.
- **SdkPackage**: installer that combines several *Package* and take cares of setting up the environme for the SDK (env variables and registry entries on Windows. OS X framework for OS X)
- **InstallerPackage**: installer that combines several *Package*. Similar to *SdkPackage*, except that it only install files.
  Can be used to extend the SDK, for example with a codec pack.
- **App** (work in progress): Creates an installer for applications, either bundling the SDK or depending on an installed version of the SDK's runtime. On Windows, it creates links in the Home Menu.
  On OS X, it creates an Application bundle, relocating the shared libraries if the SDK is bundled.

gStreamer SDK

FLUENDO
Influencing the Multimedia World

```
(file:gstreamer-core.package)

class Package(package.Package):

    name = 'gstreamer-core'
    shortdesc = 'GStreamer core'
    url = "http://www.gstreamer.com"
    version = '2012.9'
    codename = 'Amazon'
    license = License.LGPL
    vendor = 'GStreamer Project'
    org = 'com.gstreamer'
    uuid = '32fe67c2-4565-411f-8287-e8faa892f853'
    deps = ['base-system']

    files = ['gstreamer', 'gst-plugins-base:bins:libs:core:lang',
            'gst-sdk-shell',
            'gst-plugins-good:core:lang',
            'gst-plugins-bad:core:lang',
            'gst-plugins-bad:core:lang',
            'gst-plugins-ugly:core:lang']
    files_devel = ['gstreamer-static', 'gst-plugins-base-static:core_devel',
            'gst-plugins-good-static:core_devel', 'gst-plugins-bad-static:core_devel']
    platform_files = {Platform.DARWIN: ['gstreamer-osx-framework']}
```

gStreamer SDK

FLUENDO
Influencing the Multimedia World

```
(file:gstreamer-sdk.package)

class SDKPackage(package.SDKPackage):

    name = "gstreamer-sdk"
    shortdesc = "GStreamer SDK"
    longdesc = "GStreamer SDK"
    title = "GStreamer SDK"
    url = "http://www.gstreamer.com"
    version = '2012.9'
    sdk_version = '0.10'
    codename = 'Amazon'
    license = License.LGPL
    uuid = '3ffe67b2-4565-411f-8287-e8faa892f853'
    vendor = "GStreamer Project"
    org = "com.gstreamer"
    ignore_package_prefix = True
    resources_wix_installer = 'installer.wxs'
    packages =[
               # (name, required, selected)
               ('gstreamer-core', True, True),
               ('gstreamer-system', False, True),
               ('gstreamer-playback', False, True),
               ('gstreamer-codecs-gpl', False, False),
               [...]
               ('gstreamer-codecs-restricted', False, False),
               ('gstreamer-tutorials', False, True),
              ]
    platform_packages = {
            Platform.WINDOWS: [('vsintegration', True, False)],
            Platform.DARWIN: [('gstreamer-xcode-integration', False, True)],
            }

    install_dir = {
        Platform.WINDOWS: 'gstreamer-sdk',
        Platform.LINUX: '/opt/gst-sdk/',
        Platform.DARWIN: '/Library/Frameworks/GStreamer.framework/'}
    root_env_var = 'GSTREAMER_SDK_ROOT_%(arch)s'
    osx_framework_library = (reamer', 'lib/GStreamer')
```

```
(file:longomatch.package)

class App(package.App):

    name = 'longomatch'
    app_name = 'LongoMatch'
    shortdesc = 'LongMatch Video Analysis'
    url = "http://www.longomatch.org/"
    version = '0.17.5'
    license = License.GPL
    vendor = 'LongoMatch Project'
    uuid = '5f444646-4165-511f-8287-e8f2a895f853'
    org = "org.longomatch"
    app_recipe = 'longomatch'
    deps = ['base-system', 'gstreamer-core', 'gstreamer-codecs', 'gstreamer-system',
            'gstreamer-capture', 'gstreamer-libav', 'gstreamer-editing', 'gstreamer-codecs-gpl',
            'gstreamer-codecs-restricted', 'gstreamer-playback', 'gtk+-2.0', 'mono']
    commands = [('LongoMatch', 'bin/LongoMatch', False, None)]
```

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Cerbero configuration files:

- files with python syntax
- User config files: uses the '.cbc' extension
    - Main config: ~/cerbero/.cerbero.cbc
    - User config: a file passed using the command line *--config filename*
- Platfrom config files: uses the '.config' extension. Loaded based on the target platform. (windows.config, darwin.config, linux.config, android.config) Used to set the correct environment for the toolchain.

Load sequence:

1) Main configuration
2) User config
3) Platform config

```
$ cerbero -c cross-win32.cbc bootstrap
```

(file:cross-win64.config)

```
import os
from cerbero.config import Platform, Architecture, Distro, DistroVersion

target_platform=Platform.WINDOWS
target_arch=Architecture.X86_64
target_distro=Distro.WINDOWS
target_distro_version=DistroVersion.WINDOWS_7
```

gStreamer SDK

FLUENDO
Influencing the Multimedia World

```
(file:linux.config)

import os
from cerbero.config import Architecture

if target_arch == Architecture.X86_64:
    prefix=prefix or os.path.expanduser('~/cerbero/dist/linux_x86_64')
    sources=sources or os.path.expanduser('~/cerbero/sources/linux_x86_64')
    cache_file= cache_file or 'linux_x86_64'
elif target_arch == Architecture.X86:
    prefix=prefix or os.path.expanduser('~/cerbero/dist/linux_osx_i686')
    sources=sources or os.path.expanduser('~/cerbero/sources/linux_i686')
    cache_file=cache_file or 'linux_i686'

for f in ['CFLAGS', 'CCASFLAGS', 'CXXFLAGS', 'LDFLAGS', 'OBJCFLAGS']:
    os.environ[f] = os.environ.get(f, '')

os.environ['CFLAGS'] += ' -g -O2'
os.environ['CXXFLAGS'] += ' -g -O2'
os.environ['OBJCFLAGS'] += ' -g -O2'
os.environ['am_cv_python_pyexecdir'] = '%s/%s/site-packages' % (prefix, py_prefix)
os.environ['am_cv_python_pythondir'] = '%s/%s/site-packages' % (prefix, py_prefix)

if use_ccache:
    os.environ['CC'] += 'ccache gcc'
    os.environ['CXX'] += 'ccache g++'
```

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Directories:

- **prefix_dir**: sets the output directory for the builds (~/cerbero/dist/$platform_$arch/)
- **sources_dir**: directory where the sources are extracted (~/cerbero/sources/$platform_$arch/)
- **local_sources_dir**: cache directory for the sources (~/cerbero/sources/local/)

Build environment

- cerbero commands are run in shell with an environment that tries to isolate the build
- a set of environment variables are set relative on the build prefix
    - LD_LIBRARY_PATH
    - LDFLAGS
    - C_INCLUDE_PATH
    - CPLUS_INCLUDE_PATH
    - DYLD_FALLBACK_LIBRARY_PATH
    - PATH
    - MANPATH
    - INFOPATH
    - PKG_CONFIG_PATH
    - PKG_CONFIG_LIBDIR
    - GI_TYPELIB_PATH
    - XDG_DATA_DIRS
    - XDG_CONFIG_DIRS
    - XCURSOR_PATH
    - ACLOCAL_FLAGS
    - ACLOCAL
    - PERL5LIB
    - MONO_PREFIX
    - MONO_GAC_PREFIX
    - GST_PLUGIN_PATH
    - GST_REGISTRY
    - PYTHONPATH

Build process

The build process is divided in 2 big steps:

- Fetching the sources -> Source backend
- Building the project -> Build backend

The sources step is split in 2 sub-steps:

- **fetch**: fetch the sources
- **extract**: extract the sources

The build step is split in 4 sub-steps:

- **configure**: configures the project
- **compile**: compiles the project
- **install**: installs the project
- **post_install**: runs a post-install step

Backends are python classes that implements these steps.
A recipe inherits from the backend bases defined by the 'btype' an 'stype' fields using python metaclasses.

```
class Recipe(package.Recipe)
    stype = Git                          ----> class Recipe(package.Recipe, Git, Autotools)
    btype = Autotools
```

gStreamer SDK

FLUENDO
Influencing the Multimedia World

Build command

```
$ cerbero build recipe_name
```

1) Loads configuration
2) Load recipes from the recipes and dir and call prepare() with the current config
3) Resolve dependencies
4) Build the list of recipes in order

Cerbero uses a cache that stores the build state of each recipe, recording the last successful step. The state of recipe is reset when it's recipe file is edited.

Package command

```
$ cerbero build package_name
```

1) Loads configuration
2) Load recipes from the recipes and dir and call prepare() with the current config
3) Load packages from the packages and dir
4) Get the list of packages dependencies
4) Get list of recipes to build
5) Resolve dependencies
6) Build the list of recipes in order
6) Package packages dependencies
6) Creates the final package

Extending recipes and packages with external repos

- cerbero includes a default set of recipes and packages that can be extended, adding a couple of lines to the user configuration files:

```
external_recipes = {'repo_name': repo_path, priority}
external_packages = {'repo_name': repo_path, pripority)}
```

cerbero-extras

- Extends cerbero with new recipes and packages

```
$ git clone https://github.com/ylatuya/cerbero-extras.git
$ cerbero -c cerbero-extras/extras-x86.cbc list
```

- Contains:
  - mono
  - gtk-sharp
  - longomatch

And very soon:
  - totem
  - pitivi
  - banhsee
  - transmageddon
  - and many more ....

GStreamer SDK

FLUENDO Influencing the Multimedia World

Questions ?

Useful links:

- Landing page: http://www.gstreamer.com
- Documentation: http://docs.gstreamer.com
- GStreamer SDK repo: http://cgit.freedesktop.org/gstreamer-sdk
- Bugzilla: https://bugs.freedesktop.org/enter_bug.cgi?product=GStreamer%20SDK
- Fluendo: http://www.fluendo.com/
- Collabora: http://www.collabora.com/

Many thanks to Fluendo and Collabora for making the GStreamer SDK possible and to everyone that contributed to this project.