HoloVibes – User manual

Jeffrey Bencteux Thomas Kostas Pierre Pagnoux

Contents

About HoloVibes

Holovibes is a software program that allow to make holographic videos. It is developped in `C/C++` languages.

Authors

HoloVibes has been rewritten from scratch in September 2014 by:

- Jeffrey BENCTEUX jeffrey.bencteux@epita.fr
- Thomas KOSTAS thomas.kostas@epita.fr
- Pierre PAGNOUX pierre.pagnoux@epita.fr

The previous version of Holovibes has been written by:

- Romain CANCILLIERE
- Antoine DILLÉE

Developers rules

This page contains important informations that a developer must keep in mind. It explains how to use the versioning tool, what is required to code on Holovibes, great advices for C/C++ programming and the following rule.

If you do not respect these rules, you will have to pay pizzas to your entire team. ;-)

Dependencies

- Visual Studio 2013 Professional
- Doxygen
- Git
- Qt
- Qwt
- nVidia CUDA

Git

If you do not know how to use git, please have a look at the following tutorial.

- Git - Documentation
- Bitbucket Git FAQ

Clone the repository

You can use a GUI tool as SourceTree or use git in command line.

```
git clone https://username@bitbucket.org/PierrePagnoux/holovibes.git
```

Git rules

To let the versioning tool consistent, you have to respect these rules.

1. `master` branch **must** be *clean* and *compile*.

- Never push generated files.
- Use branch for everything. For example to develop a new feature : `new/myfeature`.
- Prefer use rebase when pulling changes in your own branch (it avoids *merge commits*).

- Use merge when pushing your changes to another branch.
- Never commits on master branch directly (without the acknowledge of your team mates).
- Commit messages: use keywords as 'add, fix, remove'
- Git rules - Code review
- Git names conventions

Set up your IDE

The project is based on Visual Studio 2013 solution.

Visual Studio merge tool

Edit ~/.gitconfig and add this gist. Make sure that the path is correct (Visual Studio 2013 is 12.0).

There is a little mistake in this gist : `trustexistcode = true` should be `trustexitcode = true`.

Useful plugins

- Codemaid
- Productivity Power Tools 2013
- DoxyComment

Configuration

Codemaid This plugin is very useful. It automatically format the source code.

- Uncheck `Remove end of line trailing new line`.

DoxyComment It helps to generate documentation and brings syntax highlighting for Doxygen. Install it from Visual Studio : *Tools* and *Extensions and Updates*.

Tabs According to the rules, the use of tabs is not allowed.

In *Tools*, *Options*. Set your *Text Editor* parameters for `C/C++` languages. The Holovibes source code use only whitespaces.

- Set tab size to 8, to emphasize the use of tabs (which is forbidden).
- Set indent size to 2.
- Select the option *Insert spaces* to use only whitespaces.

Disable *output window* Each time you *build* the project, the *output window* is messing up.

In *Tools*, *Options*, section *Projects and Solutions*.

- Uncheck `Show Output window when build starts`.Qt is the Graphical User Interface (GUI) used by Holovibes.

C/C++ coding styles

You should know good pratices given by big software companies.

- Google
- Mozilla
- GNU
- Linux Kernel
- EPITA *This is a bit old, but it helps to understand.*

Moreover, you should have a look at the **C++ FAQ**.

Coding style in Holovibes

- Use the *WIN32 API* syntax (capital letters everywhere!) is *allowed* only in *Windows* dependent modules (such as specific *Camera* implementation that use external libraries, or the *WIN32* GUI).
- Use standard portable code: for example, prefer use `malloc` (in C source file) or `operator new()` (in C++ source file) instead of `HeapAlloc`.
- The code must have *2 white spaces* as indentation. It must not contain *tabs*.
- No trailing white spaces are allowed.
- No blank lines at the beginning or the end of the file.
- Prototypes for exported function must appear in header files and must not appear in source files. The source file which defines an exported function MUST include the header file containing its prototype.

Test your code

- Do unit tests for everything, to ensure your not breaking features.

Document your code

Source code

The documentation is written in source code with Doxygen syntax.

- Most of the documentation must remains in *headers*.
- Each exported function should have a documentation.
- Write documentation simultaneously.
- Comment your source code with parsimony (especially in function's body). Your function should already be understandable by reading its prototype. Unless if it does tricky things.

Wiki

Most of the big/global documentation should remain in the wiki.

Documentation

The Holovibes project documentation is written directly in sources files using doxygen. The configuration file of the project is at its root and the generated documentation is located in the /doc folder.

Install doxygen

You can easily install doxygen by downloading it from the official website. Follow the instructions (install is really easy for all platforms).

Configuration file

A configuration file is necessary to use doxygen, you can generate it using the following command:

```
doxygen -g <config-file>
```

Generation

Doxyfile documentation can easily be generated using:

```
doxygen <config-file>
```

Manual

Doxygen manual can be found here, it teaches how to write documentation in source files.# Developers Tools

Libraries

This page contains an updated list of API used in Holovibes.

- Boost 1.55.0 build2
- XiAPI XiQ Camera XIMEA API V4.01.80
- Driver IDS V4.41
- AVT Vimba V1.3
- pco.pixelfly usb 2.0 driver V1.04
- pco.edge 4.2 usb 3.0 driver V1.08
- Qt OpenSource Windows x64 MSVC OpenGL 5.3.2

Boost

Holovibes use the well-known C++ library: boost.

Installation for Microsoft Visual C++

- Download the automatic installer from Sourceforge Boost Binaries page
- Holovibes currently use Boost 1.55 (but should use the lastest release)
- The installer will by default install boost in : `C:\local\boost_1_55_0`.
- Notice that Holovibes' Visual Studio Solution expect to find boost at this place.
- The installation contains headers and compiled libraries.

Add Boost headers to the compiler's include path

In `Configuration Properties : VC++ Directories : Include Directories` add : `C:\local\boost_1_55_0`

Add Boost libraries to the linker's library path

In `Configuration Properties : Linker : General : Additional Library Directories` add : `C:\local\boost_1_55_0\lib64-msvc-12.0`

The linker will find automatically needed libraries (following Debug/Release mode: it do not load the same library files).

Verification

Check if your solution compile in *debug* and *release* modes.

Qt

Holovibes require a different Qt version than the default one offered when you go on the Qt project download page, the following steps are here to help:

- Go on Qt project download page.
- Choose "community" release
- Choose "view all downloads"
- Choose "windows host" -> "Qt 5.4.0 for Windows 64-bit (VS 2013, OpenGL, 709 MB)"

As the link above may be dead in some time, be sure to always choose the 64-bit, OpenGL version.

Install

- Follow the steps given by the installer. Installing Qt at `C:\` location is recommended.
- Change the windows path variable (start -> Control Panel -> System -> Advanced system settings -> Environment variables -> Path -> Edit) : add a semicolon ';' and then the path to Qt bin folder location (e-g:`C:\Qt\Qt5.4.0\5.4\msvc2013_64_opengl\bin`)
- Log out and in to apply changes.

Information

- There will be a "GeneratedFiles" folder in your holovibes sources, don't bother, it's Qt compiler generated files for our GUI implementation.

Qwt

What is it ?

Qwt is the library Holovibes uses to display plots. It's based on Qt and under LGPL licence.

Official documentation here.

Download

- Go on the project page, click on qwt.
- Download the latest version (tested until now with qwt 6.1.2).

Installation

As we are currently using windows and because qwt doesn't have an installer, you will need to build it from its sources. This means it is necessary to have Qt installed (to use Qt tools) and Visual Studio installed (to use nmake).

- Extract the previously downloaded archive (.zip) at `C:\`
- Start menu > All programs > Qt your.version > your.version > command line
  - Go to the path you've extract qwt sources
  - Type `qmake qwt.pro`
- Start menu > All programs > Visual Studio your.version > Visual Studio Tools > VS your.version x64 Native Tools Command Prompt
  - Go to the path you've extract qwt sources
  - Type `nmake`, it takes approximately 10 mn, don't worry
  - Type `nmake install`
- Change the windows path variable (start -> Control Panel -> System -> Advanced system settings -> Environment variables -> Path -> Edit) : add a semicolon ';' and then the path to Qwt library folder location (e-g: `C:\qwt-6.1.2\lib`)
- Log out then log in to apply path changes

Information

- This helper is based on the official one
- All the documentation is available at qwt project page, examples of what it is possible to do are also available on the website.
- Be sure to have the most recent Qt version (Qt 5.4 for qwt-6.1.2), as we are compiling from the sources, some functions/classes may have been updated or removed.