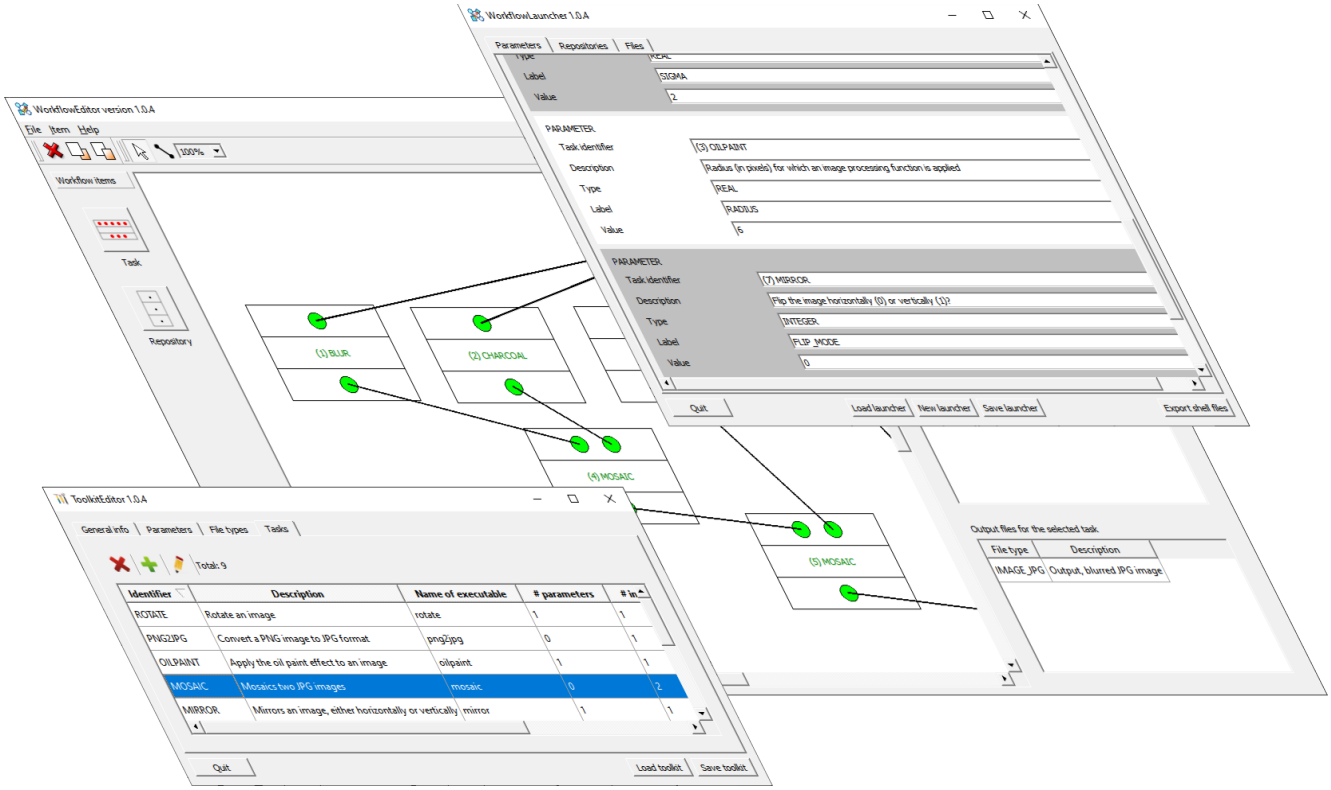
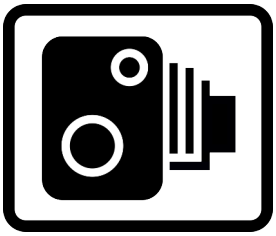
# Building the samples

For WorkflowMaker version **1.0.8** and later
Windows 10 & 11 only
64-bit only



# Table of contents



# List of figures

# 1 The goal

This document explains how the image processing sample software is organized, describing its distribution across different folders and sub-folders, and indicating what can be found in each one. Additionally, it provides instructions for building the sample from the source code, for both Windows and Linux. It specifies the tools to be used; in the case of Linux, it also indicates the procedure for installing said tools.

Due to the limited resources available, the build instructions for Linux systems provided in the next sections are guaranteed to work only on Debian-based Linux operating systems. Furthermore, and since version 6 of the Qt framework (see section 3) is required, only relatively new distributions such as Ubuntu 24.10 or Linux Mint 22 are supported.

# 2 The tools

To build the image processing sample directly from source the tools below are required; note that these are the same needed to build WorkFlowMaker itself.

- Windows
  - MSVC 2022 Community Edition (https://visualstudio.microsoft.com/downloads/).
  - Qt version 6 (https://www.qt.io/download-open-source). Download the prebuilt binaries for the MSVC 2022 compiler (msvc19_64).
  - The NSIS (Nullsoft Scriptable Install System (https://nsis.sourceforge.io/Download).
  - The HM NIS EDIT editor (https://hmne.sourceforge.net/). This editor is used to build the Windows installer together with NSIS.
  - doxygen (https://www.doxygen.nl/download.html) and graphviz (https://graphviz.org/) to build (manually) the documentation of the `simple_options_file_parser` library included with the samples.
- Linux
  - gcc 13.
  - The dpkg and dpkg-dev packages (for building .deb packages and finding package dependencies).
  - Qt version 6.
  - doxygen and graphviz, to build (manually) the documentation of the `simple_options_file_parser` library.

For Linux, all tools should be available using the built-in package manager.

On Linux **do not** download version 6 of the Qt libraries from Qt's website; use the ones available in the system repositories instead. Deploying a Linux application relying on the libraries included in the official Qt's distribution is much more complicated than using the system libraries; furthermore, the size of the resulting deb package is notably bigger.

# 3 The samples source code

Figure 1 depicts the organization of the source code once the samples have been installed.

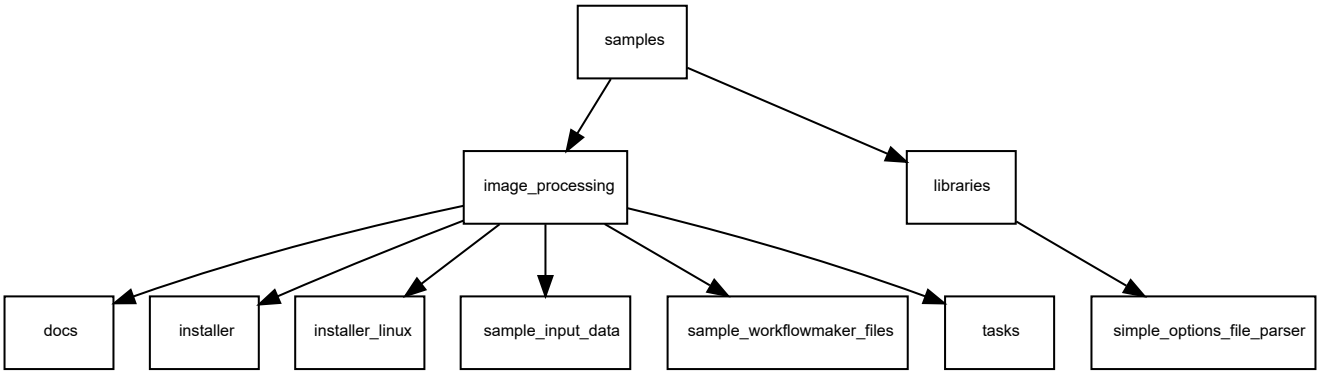



*Figure 1: Folder hierarchy for the source code of the samples.*

The source code is located at the `samples` sub-folder of the **WorkflowMaker** source code directory. Refer to [RD1] for more information about the WorkflowMaker source code folder structure.

It is organized as follows:

- `image_processing` – Main folder for a sample toolkit dealing with image processing. In the future, there might be extra folders like this to include more example toolkits.
  - `docs` – The user guides (as this one) relative to this example toolkit are stored here.
  - `installer` – This folder includes all the resources needed to build a Windows installer making possible to install the example toolkit easily.
  - `installer_linux` – And here, some resources needed to build a deb package for Debian-based Linux computers are stored.
  - `sample_input_data` – This folder contains input data files to test the toolkit (images).
  - `sample_workflowmaker_files` – This is the folder that will save most time to users. There, the formal definition of the image processing toolkit has been copied (file: `image_processing_toolkit.xml`). Furthermore, an example workflow (`image_processing_workflow.xml`) and a launcher (`image_processing_launcher.xml`). Thanks to these example files users will be able to start testing WorkflowMaker immediately.

    - `tasks` – The implementation of the several applications making the toolkit. These are provided in source code form, so it is possible to see how they are implemented, and how these adhere to the restrictions that any WorkflowMaker-compliant console application must respect. The tasks have been implemented using C++, Qt6 and ImageMagick.

- `libraries` – In this folder, all the libraries on which the tasks above rely are included.

    - `simple_options_file_parser` – This is a simple C++ library used to parse text files containing label / value pairs. It is used by all the tasks to read their respective option files.

The ImageMagick library, although needed by all tasks, has not been included in the source code distribution. Users must obtain this library prior to building the samples. See section 5.1 for further details.

Figure 2 below depicts the organization of the `tasks` folder. There, the nine console applications making the toolkit are shown.

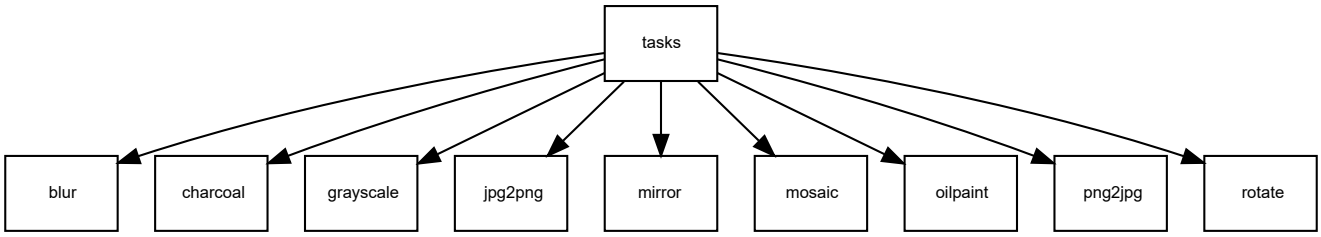



*Figure 2: Folder organization of the several tasks (applications) included in the toolkit.*

The source code of all tasks has been organized in the same way. These are the files included in every task folder:

- *xxxxx*`.cpp`. The source code implementing the logic of the application. A common trait is that all these `xxxxx.cpp` files create an object of the corresponding `xxxxx_options_file_reader` class to read the parameters controlling its behavior and then make use of the ImageMagick library to apply some kind of effect to the input image(s).

- *xxxxx*`_options_file_reader.cpp` + `xxxxx_options_file_reader.hpp`. These files implement specific options file readers for each of the tasks, taking care of searching for the appropriate labels and then reading their values. Since each task reads a different set of labels, a specific options file parser is required. For instance, the labels to parse for the `mirror` task are INPUT_FILENAME, OUTPUT_FILENAME, and FLIP_MODE. In the case of mosaic, these are LEFT_IMAGE_FILENAME, RIGHT_IMAGE_FILENAME, and MOSAIC_FILENAME.

- *xxxxx*`.pro`. The Qt project file to build the application.

- `postbuild.bat` and `postbuild.sh`. The post build scripts for Windows and Linux computers respectively. These are run automatically after the tool is built to copy files to the appropriate folders that will be used to build a Windows installer or a deb package.

In the list above *xxxxx* stands for the name of any of the five tasks included. In the case of the `mirror` task, these filenames are `mirror.cpp` + `mirror_options_file_reader.cpp` + `mirror_options_file_reader.hpp`.

As required by WorkflowMaker, all the tasks included in the toolkit read an input options file stating the values of their keyboard parameters as well as the names of their input and output files. This is so common a task that a C++ library helping to read such files have been provided as an additional component of the source code distribution of the example toolkit. It is the `simple_options_file_parser` library, which is targeted at facilitating the parsing of options files including label + value pairs (as requested by WorkflowMaker). All tasks rely on this library to parse their corresponding options files. This library is self-contained and depends on no other components.

# 4 Installing the tools

Please refer to [RD1] for instructions on how to install the required tools – these are the same for both WorkflowMaker and the samples.

# 5 Building the samples

The following subsections explain how to build the samples for both Windows and Linux computers.

## 5.1 The ImageMagick library

Prior to building the samples, it is necessary to have a working copy of the ImageMagick library. The procedure to do so differs depending on whether the target computer runs the Windows or Linux operating systems. In all cases, however, version 7 of said library is required.

### 5.1.1 Building ImageMagick on Windows

To obtain and build the ImageMagick library for Windows, follow the instructions that may be found here:

**https://github.com/ImageMagick/ImageMagick-Windows**

When building the ImageMagick library, please select option `Static Multi-threaded DLL runtimes`. Although this is not the optimal choice, it matches the requirements to build the image processing toolkit on Windows.

Once the library has been built, some of its components must be copied to the samples folder tree. More precisely, a new **`imagemagick`** folder (as well as some sub-folders) must be created and

organized as shown in Figure 3. The folders (and sub-folders) to create are shown in red. Please, keep the names of the folders exactly as shown in the figure (even the case).

Once created, these folder must be **populated** using files that must be retrieved from the ImageMagick source tree. For the sake of simplicity, **we will assumed that such root folder is located at `<imagemagick>`.** In the instructions below, replace `<imagemagick>` by the actual path to this folder.

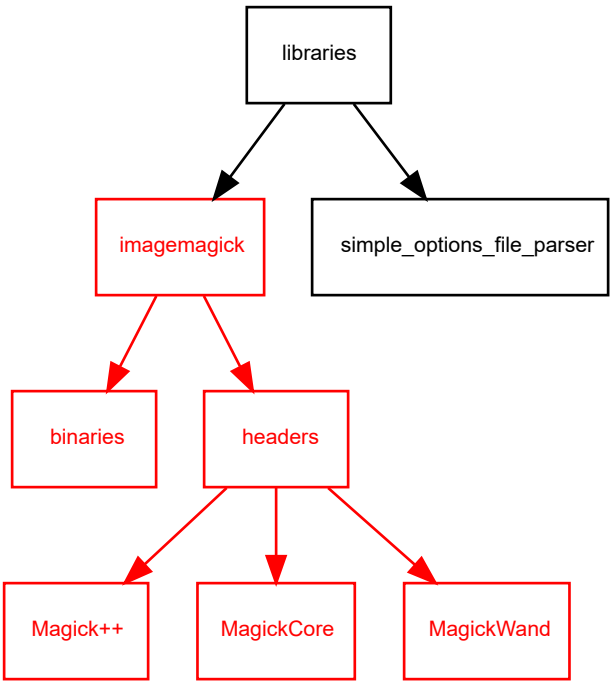



*Figure 3: Folder structure for the ImageMagick library on Windows.*

- ***In sub-folder `binaries`*** – Copy all the files with extension `.lib` from folder `<imagemagick>/Artifacts/lib`
- ***In sub-folder `headers`*** – Copy the following files and folders:
  - From `<imagemagick>ImageMagick/Magick++/lib` copy
    - File `Magick++.h` and
    - Sub-folder `Magick++`.
  - From folder `<imagemagick>ImageMagick` copy
    - Sub-folder `MagickCore` (only files with the .h extension) and
    - Sub-folder `MagickWand` (only files with the .h extension).

In short, folder `binaries` must keep the libraries themselves (`*.lib`), folder headers will have three subfolders, namely `Magick++`, `MagickCore` and `MagickWand` all of them containing only header files (`*.h`). Additionally, folder headers will also store a single header file named `Magick++.h`.

### 5.1.2 Obtaining ImageMagick on Linux

Installing ImageMagick on Linux is far more simple. Just open a terminal and type the following commands:

```
sudo apt update
sudo apt install imagemagick libmagick++-dev
```

The user's password will be requested during the installation process.

## 5.2 Building the samples on Windows

This sections explains the steps to build the samples on a Windows computer.

**Open a MSVC 2022 x64 terminal.**

Beware that a regular Windows terminal is not enough to build WorkflowMaker. Instead, a specific MSVC 2022 terminal to build x64 applications must be used. These may be started from the Visual Studio menu (in the Windows start menu). Its precise name is "*x64 Native Tools Command Prompt for VS 2022*".

**Change the default directory to the samples root folder.**

```
cd <samples>
```

In the command above <*samples*> stands for the full path of the `samples` folder in the WorkflowMaker folder hierarchy; therefore, in the command below, replace <*samples*> by the actual path to this folder.

**Run the build script.**

```
build_samples.bat
```

**When the build process finishes,** a new installer is available in the folder named

```
<samples>/image_processing/installer
```

The name of the installer is `WorkflowMaker_image_processing_samples-`*`x.y.z`*`-setup.exe` where *`x.y.z`* stands for the actual version of the samples reported by the installer script (file `image_tools.nsi`, located in the same folder than the installer above).

## 5.3 Building the samples on Linux

As already stated, due to the limited resources available, the build instructions provided in the next sections are guaranteed to work only on modern Debian-based Linux operating systems (such as Ubuntu 24.10 or Linux 22) since version 6 of the Qt framework must be available in the *software repositories* of these operating systems (not downloaded from the Qt website).

To build the samples follow the steps below:

**Open a terminal**

**Change the default directory** to <*samples*> where <*samples*> stands for the full path of the `samples` folder in the WorkflowMaker folder hierarchy; therefore, in the command below, replace <*samples*> by the actual path to this folder.

```
cd <samples>
```

**Grant execution privileges** to the file `build_samples.sh`:

```
chmod +x build_samples.sh
```

**Run** the build script:

```
./build_samples.sh
```

Once the process is finished **two new folders have been created** in `<samples>/installer_linux`:

- `workflowmaker-image-processing-samples_`*`x.y-zz`*`_amd64` – This folder has been structured to match the needs of a deb package. Rename it so *`x.y-zz`* is replaced by the actual version (as, for instance, 1.0-1) reported *in the control file located in the DEBIAN sub-folder of this folder* and then follow the steps required to build a deb package as described in [RD1]. The result of this step is a new deb package that may be used to install the samples.

- `workflowmaker-image-processing-samples_x.y-zz_doc_and_samples` – This folder contains the user guides (as this one) and some sample WorkflowMaker files (the image processing toolkit, a workflow and a launcher). This folder, compressed, may be distributed together with the deb package above.

# 6 The API documentation

The tasks include no API documentation, since they are so simple that it is should not be needed. However, the `simple_options_file_parser` library is much more interesting, especially if the readers are willing to develop its own WorkflowMaker-compatible tasks using C++.

The API documentation for this library is built automatically when building the samples. It may be found in folder

```
<samples>/libraries/simple_options_file_parser/doc_html
```

Open the file `index.html` to view it.

# 7 Installing the tools

To install the tools on Linux, please use the deb package created in section 5.3. For Windows computers run the installer built in section 5.2.

**It is important to highlight some aspects regarding how the Windows installer works.**

If the standard options are accepted when running the installer, the install path is:

```
C:\Program Files\cttc\WorkflowMaker_image_processing_samples
```

Besides the tools' executable files themselves, the installer also copies the source code of both the tools and libraries to some `src` folder which is located just under the installation directory. Figure 4

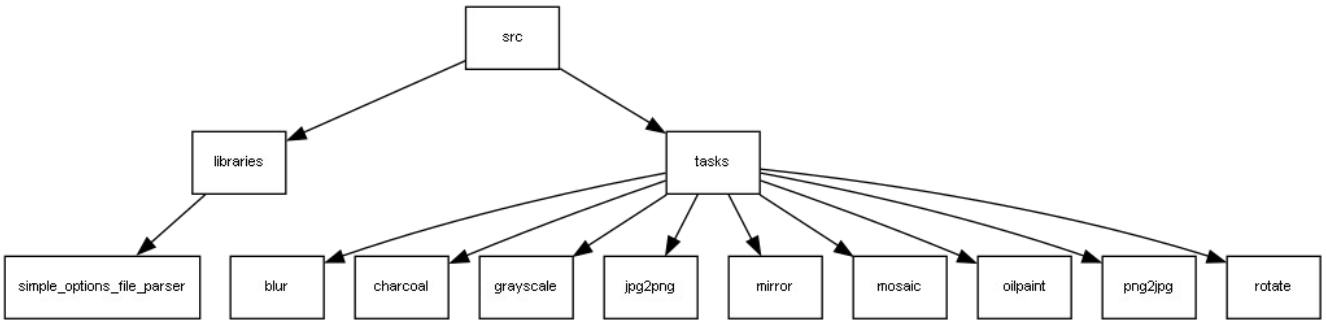



*Figure 4: The folder structure created by the samples' Windows installer.*

It is possible to access the `src` folder by means of the Windows start menu. There, an entry named `WorkflowMaker_image_processing_samples` may be found. Unfolding this entry, another one, named `WorkflowMaker_image_processing_samples Source code folder` may be found. Selecting this entry, the `src` folder will be opened.

There are also entries in said menu to open a folder including the user guides.

# 8 Open Source licenses

The WorkflowMaker samples rely on several open source libraries and it is also distributed as open source. In the following subsections, `<workflowmaker>` stands for the root folder where the WorkflowMaker source code has been downloaded.

## 8.1 The rapidxml library

Thie samples use the rapidxml library, which is available under the Boost Software License. For more information, visit https://www.boost.org/LICENSE_1_0.txt. A copy of this license may be found in the `<workflowmaker>/doc/licenses` folder.

## 8.2 The Qt 6 framework

The samples rely on the Qt 6 framework, which is available under the LGPL v3 license. For more information, visit https://www.qt.io/licensing and https://www.gnu.org/licenses/lgpl-3.0.html. A copy of the LGPL v3 license may be found in the `<workflowmaker>/doc/licenses` folder.

## 8.3 ImageMagick

All samples make use of on the ImageMagick library, which is available under ImageMagick license. For more information, visit https://imagemagick.org/script/license.php. A copy of the ImageMagick license may be found in the `<workflowmaker>/doc/licenses` folder.

## 8.4 WorkflowMaker

This project is licensed under the MIT License. See the LICENSE file in the root folder of the source code distribution for details.

# Reference documents

All the documents listed below are part of **WorkflowMaker's documentation**.

[RD1] CTTC's Geomatics Research Unit (2024). Building WorkflowMaker.