

grabui Reference Manual

2

Generated by Doxygen 1.5.1

Fri Jul 27 12:47:59 2007

Contents

1	grabui	1
2	grabui Hierarchical Index	2
3	grabui Class Index	2
4	grabui File Index	3
5	grabui Page Index	3
6	grabui Class Documentation	4
7	grabui File Documentation	11
8	grabui Page Documentation	12

1 grabui

The graphical user interface application that should facilitate remote controlling supported digital cameras in a microscopy environment

1.1 Requirements:

- working gphoto2 (for firing cameras)
- dcrw (optional, for dealing with RAW images)
- libtiff-tools (opt., for converting to TIFF)
- ffmpeg (opt., for making movies)
- stereozoom2 (opt., for viewing stereo pairs)
- GIMP (opt., for postprocessing)

1.2 Usage:

The UI is very easy to deal with. There are four tabs, where settings can be modified.

- Main: Where to store data that are about to come + parameters of the XML file
- Capture: Which ports should be used, how stacks should be captured
- Download: How files should be named, when to download
- Postprocess: What to do with downloaded images after they are downloaded

Once you've made up your mind, just hit Start. The application generates capture script, runs it, tries to get some information from the output for the download script, then generates download script and postprocess script and launches them and checks for errors. Scripts remain along photos in the dedicated directory. If desired(default), the XML file describing shooting circumstances is generated and saved alongside.

1.3 Internals (for developers and curious persons):

The class that user deals with is the [Shooter](#) class. Generating the capture script is only and only in the competence of the [Shooter](#). Then, examination of the output of gphoto2 when shooting desired batch is performed, and that action is supervised by [Reporting](#) class. [Reporting](#) works through the output and the clears it so that only meaningful information are passed back to [Shooter](#), that needs to make that download script. Next, some magic is performed to assemble the post script. Then, everything is launched again and [Reporting](#) told to report everything (mainly remaining time and what's going on). Ca y est - fini!

2 grabui Hierarchical Index

2.1 grabui Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

App	4
Dialogus	4
Shooter	10
exc_directory_exists	5
exc_gphoto2_capturing	6
exc_gphoto2_downloading	6
Exc_gphoto2_noname	7
exc_illegal_name	7
Process	7
Reporting	8

3 grabui Class Index

3.1 grabui Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

App (The main application class, nothing more)	4
Dialogus (The foundation of the main form. Quite lowlevel, ensures that every form interaction stuff works as e)	4

exc_directory_exists (Exception class for dealing with the error caused by trying to overwrite contents of an existing directory)	5
exc_gphoto2_capturing (Exception class for dealing with errors arising during the process of capturing)	6
exc_gphoto2_downloading (Exception class for dealing with errors arising during the downloading - much less fatal)	6
Exc_gphoto2_noname (Exception class that saves the day when gphoto2 dosen't report the name of the saved picture)	7
exc_illegal_name (Exception class for dealing with the problem of using undesireable characters in filenames)	7
Process (Overloaded wxProcess class - mainly because of the OnTerminate function)	7
Reporting (The frame that reports about the progress of the current task)	8
Shooter (The shooting interface of the form. Reads input, generates scripts, launches them and observes them)	10

4 grabui File Index

4.1 grabui File List

Here is a list of all documented files with brief descriptions:

App.h	??
Dialogus.h	11
exceptions.h	??
Shooter.h	??
tinyst.h	??

5 grabui Page Index

5.1 grabui Related Pages

Here is a list of all related documentation pages:

Todo List	12
---------------------------	----

6 grabui Class Documentation

6.1 App Class Reference

The main application class, nothing more.

```
#include <App.h>
```

Public Member Functions

- virtual bool [OnInit](#) ()
Called on the very beginning.

6.1.1 Detailed Description

The main application class, nothing more.

The documentation for this class was generated from the following files:

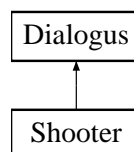
- App.h
- App.cpp

6.2 Dialogus Class Reference

The foundation of the main form. Quite lowlevel, ensures that every form interaction stuff works as e.

```
#include <Dialogus.h>
```

Inheritance diagram for Dialogus::



Public Member Functions

- [Dialogus](#) (wxWindow *parent=(wxWindow *) NULL)
- void [Fill_combos](#) ()
Fills combos on the main tab with data hints provided in the XML file.
- void [Prepare_gimp](#) ()
Fills GIMP hint combo with data from another XML file.

Protected Attributes

- std::vector< wxString > [Script_templates](#)

Hints about GIMP script-fu functions.

Private Member Functions

- virtual void **Stereozoom_browse** (wxCommandEvent &event)
- virtual void **Check_ports** (wxCommandEvent &event)
Polls the ports and gets them to the combo for the user to choose.
- virtual void **Place_GIMP_template** (wxCommandEvent &event)
When user chooses a GIMP function that he would like to use in the script, the corresponding template is copied into the text window.
- virtual void **Time_interval_trigger** (wxCommandEvent &event)
Just greying out spinbox when the checkbox is unchecked.
- virtual void **Terminate** (wxCloseEvent &event)
In order to shut down application properly.

6.2.1 Detailed Description

The foundation of the main form. Quite lowlevel, ensures that every form interaction stuff works as e.

Implementing Dialog_grabui

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Dialogus::Dialogus (wxWindow *parent = (wxWindow *) NULL)

Constructor

The documentation for this class was generated from the following files:

- [Dialogus.h](#)
- [Dialogus.cpp](#)

6.3 exc_directory_exists Class Reference

Exception class for dealing with the error caused by trying to overwrite contents of an existing directory.

```
#include <exceptions.h>
```

Public Member Functions

- int **Ask_overwrite** (wxWindow *parent)
Function attempts to ask user whether to wipe out contents of the old directory.

6.3.1 Detailed Description

Exception class for dealing with the error caused by trying to overwrite contents of an existing directory.

The documentation for this class was generated from the following file:

- exceptions.h

6.4 exc_gphoto2_capturing Class Reference

Exception class for dealing with errors arising during the process of capturing.

```
#include <exceptions.h>
```

Public Member Functions

- [exc_gphoto2_capturing](#) (wxString message)
Just a simple constructor, that just accepts error report.
- void [Show_message](#) (wxWindow *parent)
Displays an OK window with problem description.

Private Attributes

- wxString [Message](#)
The displayed error message.

6.4.1 Detailed Description

Exception class for dealing with errors arising during the process of capturing.

The documentation for this class was generated from the following file:

- exceptions.h

6.5 exc_gphoto2_downloading Class Reference

Exception class for dealing with errors arising during the downloading - much less fatal.

```
#include <exceptions.h>
```

Public Member Functions

- int [Ask_continue](#) (wxWindow *parent)
Function attempts to ask user whether to continue downloading or not.

6.5.1 Detailed Description

Exception class for dealing with errors arising during the downloading - much less fatal.

The documentation for this class was generated from the following file:

- exceptions.h

6.6 Exc_gphoto2_noname Class Reference

Exception class that saves the day when gphoto2 dosen't report the name of the saved picture.

```
#include <exceptions.h>
```

Private Member Functions

- void **What** ()

6.6.1 Detailed Description

Exception class that saves the day when gphoto2 dosen't report the name of the saved picture.

The documentation for this class was generated from the following file:

- exceptions.h

6.7 exc_illegal_name Class Reference

Exception class for dealing with the problem of using undesireable characters in filenames.

```
#include <exceptions.h>
```

6.7.1 Detailed Description

Exception class for dealing with the problem of using undesireable characters in filenames.

The documentation for this class was generated from the following file:

- exceptions.h

6.8 Process Class Reference

Overloaded wxProcess class - mainly because of the OnTerminate function.

```
#include <Dialogus.h>
```

Public Member Functions

- [Process \(Reporting *parent\)](#)
A not very special constructor.

- bool [Get_data](#) (wxString &in, wxString &err)
This function.
- virtual void [OnTerminate](#) (int pid, int status)
This function just tells its parent that it terminated and deletes the process.

Private Attributes

- [Reporting](#) * **Parent**
- int **Pid**

6.8.1 Detailed Description

Overloaded wxProcess class - mainly because of the OnTerminate function.

The documentation for this class was generated from the following files:

- [Dialogus.h](#)
- [Dialogus.cpp](#)

6.9 Reporting Class Reference

The frame that reports about the progress of the current task.

```
#include <Dialogus.h>
```

Public Member Functions

- [Reporting](#) (wxWindow *parent, int count)
A not very special constructor.
- **Reporting** (const [Reporting](#) &rhs)
- int [Start_process](#) (wxString command)
Executes following command (calls Current_report with corresponding params).
- wxString [Exploit_capturing](#) (wxString script_path)
Method that takes care of everything concerning running capture script.
- void **Supervise_downloading** (wxString script_path)
- void **Process_ended** ()
- virtual void **User_abort** (wxCommandEvent &event)

Public Attributes

- void(Reporting::* [Current_report](#))()
Pointer to the method that deals with input and error stream.

Private Member Functions

- void [Exploit_capture](#) ()
filters the string in so it contains only filenames
- void **Supervise_download** ()
- void **Debug** ()
- virtual void **OnIdle** (wxIdleEvent &event)

Private Attributes

- [Process](#) * [Current_process](#)
The (only one) process that is in use now.
- volatile bool [Processing](#)
Whether the process is still active.
- wxString [Raw_input](#)
A string as a raw captured output of the running process.
- wxString [Raw_error](#)
A string as a raw captured error output of the running process.
- wxString **Buff_input**
- wxString **Buff_error**
- wxString [Input](#)
A string as a processed output of the running process.
- wxString [Error](#)
A string as a processed error output of the running process.
- long **Process_pid**
- bool **Aborting**
- int **Tasks_num**
- int **Tasks_done**

Friends

- class [Process](#)

6.9.1 Detailed Description

The frame that reports about the progress of the current task.

The documentation for this class was generated from the following files:

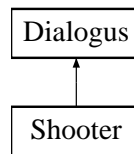
- [Dialogus.h](#)
- [Dialogus.cpp](#)

6.10 Shooter Class Reference

The shooting interface of the form. Reads input, generates scripts, launches them and observes them.

```
#include <Shooter.h>
```

Inheritance diagram for Shooter::



Public Member Functions

- virtual void [Start](#) (wxCommandEvent &event)
The main function called when user presses the big red button :-).
- void [Make_XML](#) ()
Function makes the XML summary of desired information and saves them into a file to the current shooting directory.

Private Member Functions

- void [Settle_download](#) (wxString &script)
Generates the download script according to preferences. Explotation of capture script first is needed.
- void [Settle_postprocessing](#) (wxString &script)
Generates the postprocessing script, though some tasks have to be carried out in the download script.
- void [Settle_intro](#) ()
Basic initialization of the application environment.
- void [Settle_finnish](#) ()
End of everything, just launching the download and capture script - and observing.
- void [Settle_delete](#) (wxString &script)
Function takes care about the "delete part" of the capture script.
- void [Settle_capture](#) (wxString &script)
Function takes care about the "capture part" of the capture script.
- void [Append_capture_script](#) ()
Writes the capture script onto the disc.
- void [Append_download_script](#) ()
Writes the download script onto the disc.
- void [Append_postproc_script](#) ()

Writes the postproc script onto the disc.

- wxString [Exploit_capture_script](#) ()
Runs and processes the capture script in order to generate DL script.
- void [Set_current_path](#) ()
(re)sets Current_path to its right value (= <date>/<user-spec dirname>)

Private Attributes

- wxString [Current_path](#)
Path to the directory where produced files are stored.
- wxString [Script_filename](#) [NUM_PATHS]
Collection of individual script names.

6.10.1 Detailed Description

The shooting interface of the form. Reads input, generates scripts, launches them and observes them.

6.10.2 Member Function Documentation

6.10.2.1 void Shooter::Settle_download (wxString & script) [private]

Generates the download script according to preferences. Exploitation of capture script first is needed.

TODO: Some non-fatal exception (no name given) should be handled here

6.10.2.2 void Shooter::Settle_finnish () [private]

End of everything, just launching the download and capture script - and observing.

todo: Exploit this

The documentation for this class was generated from the following files:

- Shooter.h
- Shooter.cpp

7 grabui File Documentation

7.1 Dialogus.h File Reference

```
#include <vector>
#include <wx/process.h>
#include "grabui.h"
#include "exceptions.h"
```

Classes

- class [Dialogus](#)
The foundation of the main form. Quite lowlevel, ensures that every form interaction stuff works as e.
- class [Process](#)
Overloaded wxProcess class - mainly because of the OnTerminate function.
- class [Reporting](#)
The frame that reports about the progress of the current task.

Defines

- #define [WXS_TO_ASCII](#)(wxst) ((const char*)(wxst).mb_str(wxConvUTF8))
Auto converts a wxString to const char.*
- #define [PREPARE_COMMAND](#)(command) (command) = _("bash -c \"\\\"\\\"") + (command) += _("\\\"\\\"";\n")
Passes a script to execute to bash as a parameter.

Enumerations

- enum { **LEFT, RIGHT** }

7.1.1 Detailed Description

Subclass of Dialog_grabui, which is generated by wxFormBuilder.

Todo

Add your event handlers directly to this file.

8 grabui Page Documentation

8.1 Todo List

File [Dialogus.h](#) Add your event handlers directly to this file.

Index

App, [3](#)

Dialogus, [4](#)

 Dialogus, [5](#)

Dialogus.h, [11](#)

exc_directory_exists, [5](#)

exc_gphoto2_capturing, [5](#)

exc_gphoto2_downloading, [6](#)

Exc_gphoto2_noname, [6](#)

exc_illegal_name, [6](#)

Process, [7](#)

Reporting, [7](#)

Settle_download

 Shooter, [10](#)

Settle_finnish

 Shooter, [10](#)

Shooter, [9](#)

 Settle_download, [10](#)

 Settle_finnish, [10](#)