Developer documentation: libGaze gaze tracking framework version: 1.0.0

Sebastian Herholz

June 11, 2009

Contents

1 Introduction

2 Development Environment

Because libGaze is written in C, it is possible to compile and run it on different operating systems such as: Linux, Windows or MacOSX. To be able to compile libGaze a development environment has to be set up. This environment contains a C compiler and a set of libraries used by libGaze. This section will explain stepwise, how such an environment can be set up on a Linux (e.g. Ubuntu 8.10) and Windows (e.g. XP) systems.

2.1 Windows: MinGW

Windows itself does not come with a C compiler. To be able to compile C code under a Windows system a C compiler must be to be installed. There are two famous C compiler under Winder:

Visual Studio C Compiler The Visual Studio C compiler is the proprietary C compiler from Microsoft and comes with the Visual Studio Development Kit or express edition.

GNU C Compiler The GCC is "the" open source C compiler, which is available for about every platform.

This section describes how to set up a development environment under Windows using the GCC compiler. As environment we choose the MinGW packages which includes a set of GNU tools need for developing C code (including: GCC, Make, Binutils etc.). A packages containing the newest version of the GCC compiler and some nice additional features is the TDM MinGW package, which can be downloaded from http://www.tdragon.net/recentgcc/.

To install MinGW download and run the latest TDM MinGW installation package from the TDM website (e.g. tdm-mingw-1.95.0-4.4.0-2.exe). When the setup wizard pop up click "Create" to create a new MinGW installation. As installation directory choose "$C : /MinGW$". In the future we will refer to this directory as the "MinGW folder". During installation the wizard will ask you, which components it should install, make sure to select all listed components. When the installation is finished you have a working development environment for libGaze installed on your Windows system. As last step you have to define a system variable *MINGWPATH* to the MinGW folder. This step has to be done, so that the libGaze makefiles can find the dependent include files and libraries.

2.2 Linux

Many Linux based systems already come with full development environment for C projects, including the GCC compiler, "Make" and "binutils". On Debian based systems such as Ubuntu all needed packages are installed when the package "build-essential" is installed. If this package is not installed you can easily install it by running following command in the console.

```
sudo apt-get install build-essential
```

3 CMake Building System

CMake is a build system, which generates cross-platform make and project-files for different development environment like "MinGW" and "Unix" makefiles or "Eclipse CDT 4.0" and "Visual Studio" project files. A project specifies its build process platform-independent in the CMake listfile "CMakeLists.txt". The cmake generator uses this listfile to generate from this listfile the specific make-/project-files for the build system of the user. More information about the CMake build system can be found on the CMake website http://www.cmake.org.

3.1 Windows

To install the CMake build system on a windows system you need to download the CMake installer from the CMake website (e.g. "cmake-2.6.4-win32-x86.exe"). After downloading install CMake by running the installer. When the installer ask for the "Install Options" make sure to select "Add CMake to system PATH for all user". As installation directory you can choose any location you want (e.g. "C:/CMake/").

3.2 Linux

Under Debian based Linux systems such as Ubuntu you can install CMake by running following command in the console.

```
sudo apt-get install cmake
```

4 Dependencies

After the we have set up an development environment we need to install the development packages of the dependency libraries used by libGaze.

4.1 GSL

The Gnu scientific library is used for efficient vector and matrix operations and for solving linear equations (http://www.gnu.org/software/gsl/).

4.1.1 Windows/MinGW

To install the developer package for GSL under Windows/MinGW download the "Developer files" package from this website http://gnuwin32.sourceforge.net/packages/gsl.htm. Next you have to extract the zip file and copy content of the "lib" and "include" folders into the "lib" and "include" folders under the MinGW folder.

4.1.2 Linux

Under Debian based Linux systems such as Ubuntu you can install the development package of gsl by running following command in the console.

```
sudo apt-get install libgsl0-dev
```

4.2 GLib 2.0

The GLib2.0 framework is used to make the libGaze framework platform independent. It is used for loading dynamic libraries and for handling threads on the different supported platforms (http://www.gtk.org/).

4.2.1 Windows/MinGW

To install the developer package for GLib 2.0 under Windows/MinGW download the "Dev" package for GLib form the project page. Next you have to extract the zip file and copy content of the "lib", "include" and "bin" folders into the "lib", "include" and "bin" folders under the MinGW folder.

4.2.2 Linux

Under Debian based Linux systems such as Ubuntu you can install the development package of glib2.0 by running following command in the console.

```
sudo apt-get install libglib2.0-dev
```

4.3 parseconflib

The parse_conf library is used for parsing the INI configuration files, which specify the configuration of the different modules (http://sourceforge.net/projects/parseconflib/). Because the parseconflib project is not distributing binary packages for the parse_conf library, we have to compile this library on our own.

4.3.1 Windows/MinGW

Because it is more complicated to compile the parse_conf library under Windows. We support a 32Bit binary package for Windows/MinGW . This package can be downloaded from ftp://ftp.tuebingen.mpg.de/kyb/sherholz/libGaze/parse_conf-1.0-win32.zip. After you downloaded the package extract it and copy th content of the ”lib“ and ”include“ folders into the ”lib“ and ”include“ folders under the MinGW folder.

4.3.2 Linux

To install parseconflib under Linux you need to download the sources from the parseconflib project page (current version 1.0). after you have downloaded the ”parse_conf-1.0.tar.gz“ file you can compile and install the library by using following command.

```
tar -xf parse_conf-1.0.tar.gz
cd parse_conf
./configure
make
sudo make install
```

5 libGaze

5.1 Code tree structure

This section describes how to structure the source-code of libGaze if you want to compile it or one of its modules. If you want to develop new modules for libGaze you need place the source-code of your module in the specific sub folder for this module type (see 6.3 for more information about developing new modules for libGaze).

```
devel\
devel\libGaze\
devel\eyetracker_modules\
devel\headtracker_modules\
devel\calibration_modules\
devel\display_modules\
devel\...
```

devel: the main development folder, where all source-code for libGaze and its modules is stored in.

libGaze: the folder of the libGaze source-code.

eyetracker_modules: This folder contains the source-code for the different eyetracker modules (e.g. "eyetracker_modules/eyelink_2_module").

headtracker_modules: This folder contains the source-code for the different headtracker modules (e.g. "headtracker_modules/vicon_module").

calibration_modules: This folder contains the source-code for the different calibration modules (e.g. "calibration_modules/stampe_calibration_module").

display_modules: This folder contains the source-code for the different display modules (e.g. "display_modules/standardDisplay_module").

...: Additional folders can contain the source-code for APIs for different programming languages (e.g. "pyGaze", "jGaze"), documentation or demos.

5.2 Getting the Sources

5.2.1 release package

5.2.2 svn

```
svn co https://libgaze.svn.sourceforge.net/svnroot/libgaze libgaze
```

5.3 Compiling libGaze

5.3.1 Windows/MinGW

```
cmake . -G "MinGW Makefiles"
mingw32-make
```

creating a package

```
mingw32-make package
```

5.3.2 Linux

```
cmake .
make
```

creating a package

```
make package
```

6 libGaze modules

6.1 Module code srtucture

6.2 How to compile modules

6.3 How to write modules

7 Appendix

7.1 Using Eclipse IDE

Using the eclipse IDE for compiling libGaze and its modules the CDT plugin for eclipse has to be installed. If you want to work on an svn checkout you might also want to install the subclipse plugin to be able to commit your changes directly to the SVN repository. If the CDT plugin is installed,

CMake can be used to create an eclipse CDT project-file and makes it possible to include this project into an eclipse workspace. Before generating the eclipse project files for a part of the libGaze framework make sure that all source-code is stored in the file-system as described in 5.1. To generate the eclipse project files go to the folder of the libGaze or module source-code you want to work on and execute the following command.

7.1.1 Windows/MinGW

```
cmake -G "Eclipse CDT4 - MinGW Makefiles" .
```

7.1.2 Linux

```
cmake -G "Eclipse CDT4 - Unix Makefiles" .
```

After the project files for eclipse are created by CMake the project can be included into an eclipse workspace. To include the project into the eclipse workspace select "File - Import". In the opening wizard select "General - Existing Projects into Workspace" and press "Next". As "root directory" select the folder in which you generated the "Eclipse CDT 4.0" project files. The name of the project should now appear in the projects list. Make sure to un-select the check box "Copy projects into workspace" (see figure 7.1.2). The import procedure is finished by clicking on the "Next" button.