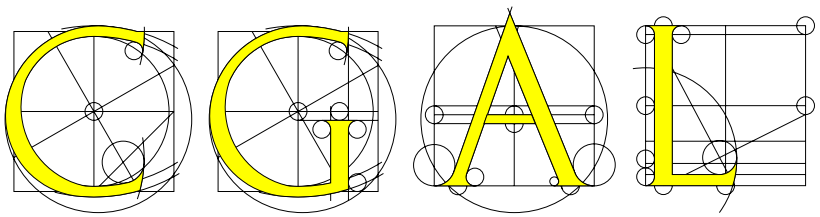

Installation Guide

Release 2.0, June1999

Authors

Michael Hoffmann, ETH Zürich.
Wieger Wesselink, Utrecht University.

Acknowledgement

This work was supported by the ESPRIT IV Long Term Research Projects No. 21957 (CGAL) and No. 28155 (GALIA).

Contents

1 Introduction

CGAL is a *Computational Geometry Algorithms Library* written in C++, which is developed in an ESPRIT LTR project. The goal is to make the large body of geometric algorithms developed in the field of computational geometry available for industrial application.

This document describes how to install CGAL on your system. Besides that, you will find some information about the makefile structure of CGAL and the support for simultaneously using CGAL and LEDA, the Library of Efficient Datatypes and Algorithms[1], and/or the GNU Multiple Precision library GMP.

2 Prerequisites

In order to build the CGAL libraries you need a C++ compiler. Currently only a small number of recent compilers on unix platforms are supported. The reason is that most compilers do not conform to the ISO 14882 standard for C++[2] and some of them have so many limitations/bugs that we could not work around all of them.

More precisely, `CGAL-2.0` supports the following Unix compilers/operating systems:

compiler	**operating system**
SGI Mips(Pro) `CC` 7.2.1 (n32)	IRIX 6.5
GNU `g++` 2.8.1	IRIX 6.5 / Solaris 2.6 / Linux 2.0
CYGNUS `egcs` 1.1	IRIX 5.3 / IRIX 6.5 / Solaris 2.6 / Linux 2.0

There will be support for PC compilers (MICROSOFT Visual `C++` 6) in the future, but there is no such support at the moment. The SUNPRO `CC` 4.2 compiler is not supported anymore, please stay with CGAL-1.2 if you have to use it. Support for SUNPRO `CC` 5 will be reconsidered as soon as an acceptable degree of standard-conformance is reached.

3 Getting CGAL

The CGAL library can be downloaded in two different ways: using ftp or using WWW. If you have a WWW connection, the easiest way to download CGAL is via the CGAL homepage:

```
http://www.cs.uu.nl/CGAL
```

and go to the 'Software' section.

Just follow the instructions on this page to obtain your copy of the library. The CGAL library can also be downloaded using FTP. The library can be found at the following location:

```
ftp.cs.uu.nl
```

in the directory `/pub/CGAL`. This directory contains release 2.0 of the CGAL library. There is also a `README` file that contains descriptions of the files in this directory. An example of an FTP-session is given below.

[1] see `http://www.mpi-sb.mpg.de/LEDA/index.html` for information.
[2] see e.g. `http://reality.sgi.com/austern/std-c++/faq-new.html#PartB` for information

```
$ ftp ftp.cs.uu.nl
Name (ftp.cs.uu.nl:<your username>): anonymous
Password: <type your email address here>
ftp> cd pub/CGAL
ftp> get README
ftp> binary
ftp> get CGAL-2.0.tar.gz
ftp> quit
```

After you have downloaded the file containing the CGAL library, you have to decompress it. For the zipfile use the command

```
unzip <filename>.zip
```

and for the gzipped file use the commands

```
gunzip <filename>.tar.gz
tar xvf <filename>.tar
```

In both cases the directory `CGAL-2.0` will be created. This directory contains the following subdirectories:

directory	**contents**
`auxiliary`	packages that can optionally be used with CGAL
`config`	configuration files for install script
`demo`	demo programs (some of them need LEDA)
`doc_html`	documentation (HTML)
`doc_pdf`	documentation (PDF)
`doc_ps`	documentation (Postscript)
`examples`	example programs
`include`	header files
`lib`	(shared) object libraries
`make`	files with platform dependent makefile settings
`scripts`	some useful scripts (e.g. for creating makefiles)
`src`	source files

4 Installing CGAL

The directory `CGAL-2.0` contains a Bourne shell script called `install_cgal`. The script can be run in two modes: a menu-driven interactive mode and a non-interactive mode. Normally you should use the interactive mode, but in case you run into problems with it or do not like it for some reason, you can still use the non-interactive mode.

We first describe a sample installation in section 5. This provides you with an overview on how the interactive installation works. If you want more detailed information about specific menus and their options, take a look at section 6. Finally, for the non-interactive mode refer to section 7.

If you want to use LEDA together with CGAL, have a look at section 11.

5 A Sample Installation

In this section we sketch an example installation on a SUN running Solaris 2.6 with the CYGNUS `egcs` 1.1.1 compiler. For a complete description of the different menus and their options refer to section 6.

5.1 Starting the script

Go to the `CGAL-2.0` directory and enter the command

```
./install_cgal -i
```

You get a message indicating the CGAL version you are going to install and that you are running the interactive mode. Then it takes some time while the script locates a number of utility programs. You will not get informed about this[3], but see some dots written to the screen indicating progress.

```
-------------------------------------------------------
  This is the install script for CGAL 2.0
-------------------------------------------------------

starting interactive mode - one moment, please
......

  Choosing compiler GNU egcs-2.91.60.
  Testing for STL ... ok.
  ...
  <tests for several compiler features>
  ...
  Saving current setup ... done.
.
```

If there is any compiler installed on your system and accessible through your `PATH` environment variable that is supported by CGAL, one of these compilers is chosen and a number of tests are done to check whether your compiler supports certain language constructs respectively has specific bugs. There are quite a number of these tests, so this step may take some time. For each test you should get a message what particularly is tested at the moment and what the result is. If there is more than one compiler installed on your system (and supported by CGAL), you may later choose to use a different compiler from the compiler menu.

A menu similar to the following will appear on your screen.

```
  ************************************************************
  **              CGAL 2.0 Installation Main Menu           **
  **              ------------------------------            **
  **                                                        **
  **   OS:                  sparc_SunOS-5.6                 **
  **   Compiler:            GNU egcs-2.91.60                **
  **   LEDA:                not supported.                  **
  **   GMP:                 not supported.                  **
  **                                                        **
  **   Compiler is supported by CGAL.                       **
  **   The setup has been tested ok.                        **
  **                                                        **
  **   There are no libs for this os/compiler.              **
  **                                                        **
  **   <C>  Compiler Menu                                   **
  **   <L>  LEDA Menu                                       **
  **   <G>  GMP Menu                                        **
  **   <T>  Test (and save) setup                           **
  **   <A>  Run all setup tests (no cache)                  **
  **                                                        **
  **   <B>  Build CGAL Libraries                            **
  **                                                        **
  **   <Q>  Back to OS                                      **
  **                                                        **
  **   Your Choice:                                         **
  **                                                        **
  ************************************************************
```

The first lines below the headline contain some kind of status report: current OS and compiler, are LEDA and GMP supported etc. Moreover you can see that the current setup has been tested and that there do not exist CGAL libraries for this OS/compiler combination in the CGAL lib directory by now.

The fact that your setup has been tested implies that the current settings have been saved to a file in the directory `CGAL-2.0/config/install`. Thus, if you run the install script a second time for this OS/compiler, you will not have to go through the whole config-/test cycle again, but the configuration will be retrieved from the corresponding config file instead.

[3] If you are that curious what happens exactly, have a look at the file `CGAL-2.0/install.log`.

5.2 Building the CGAL libraries

We are now ready to build the CGAL libraries. Just type "b" to start compilation. Building consists of three steps:

1. writing the include makefile,
2. compiling the static libraries *and*
3. compiling the shared libraries.

The include makefile encapsulates the OS– and compiler-specific settings and should be included (hence the name) in all makefiles that compile CGAL applications. If everything went ok, the output should look as follows (if not, you should have a look at the error messages from compiler or linker).

```
*************************************************************
**                                                         **
**                    Compiling CGAL 2.0                   **
**                    ------------------                   **
**                                                         **
*************************************************************

OS:           sparc_SunOS-5.6
COMPILER:     GNU egcs-2.91.60
LEDA:         not supported
GMP:          not supported

Generating Makefiles ... done.
Building CGAL_lib ... done.
Building CGAL_sharedlib ... done.

*************************************************************
**             Please press <ENTER> to continue.           **
*************************************************************
```

That's all, it's done. Press "<ENTER>" to return to the main menu and proceed by installing for a different compiler (go to the compiler menu and choose "c" to get a list of supported compilers detected on your system) or with LEDA or GMP support (go to the LEDA resp. GMP menu) or simply quit the install script by typing "q". When leaving the script, you get a list of successful builds during the session.

Now it would be a good idea to print and read the document `getting_started.ps` that can be found in the `doc_ps` directory.

6 The interactive mode

To run the install script in the interactive mode, go to the `CGAL-2.0` directory and enter the command

```
./install_cgal -i
```

After initialization during which certain utility programs are located and your system is searched for compilers supported by CGAL, you get into the CGAL installation *main menu* (see page 3 for a picture).

From the main menu you can reach a number of different sub-menus, of which the most important maybe is the *compiler menu*. This is where you can choose the compiler you want to work with and setup custom compiler or linker options. The compiler menu is described in section 6.2.

If you want to use LEDA or GNU GMP with CGAL, you will have to go to the *leda menu* described in section 6.3 resp. to the *gmp menu* described in section 6.4.

Finally you can build the CGAL libraries by typing `b`. However, it is recommended to run the *setup test* which is available in all menus as option `t` before. The setup test includes an STL test, a LEDA test and a GMP test, but not all tests are performed always; e.g. the LEDA test is only done, if you enabled LEDA support. The install script keeps track of the tests passed and only tests again, if you change the setup in a way that might affect the test result. If you want to redo *all* tests, you have to choose option "a" from the main menu. This also retests for LEDA or GMP installation in system directories which otherwise is only done the first time you enable LEDA/GMP support for an OS/compiler combination.

6.1 Files created during installation

The install script stores all relevant settings for an OS/compiler combination in the directory

`CGAL-2.0/config/install/<`*CGAL-OS-description*`>`

where <*CGAL-OS-description*> identifies your OS/compiler combination in a way specified in section 8. [4] This saves you typing everything again, if you upgrade CGAL or another package that makes recompiling the CGAL libraries necessary.

Besides the config files, `install_cgal` uses several temporary files during interactive installation. Most of them are removed after use, but some are not, since it might be helpful to keep some information about the last run. You can keep or delete them as you like, as they are not needed anymore once the script terminated. It follows a list of these files (all are plain ASCII and reside in `CGAL-2.0`).

filename	**content**
`install.log`	detailed overall protocol
`install.completed`	list of systems for which CGAL libraries have been built
`compile.log`	output of the last compiler call

6.2 The Compiler Menu

Here is the place to setup the compiler specific options, such as the compiler to use (if more than one has been detected) and custom compiler or linker flags.

Compiler Menu

<**C**> Choose the compiler to be used from the list of detected compilers.

<**F**> Set custom compiler flags. These are the first flags given to the compiler in every call. Under normal circumstances there should be no need to set any such flag.

<**L**> Set custom linker flags. These are the first flags given to the linker in every call. Under normal circumstances there should be no need to set any such flag.

6.3 The LEDA Menu

This is the place to setup LEDA specific options, if you plan to use LEDA together with CGAL (see also section 11). In order to enable LEDA support in CGAL, LEDA has to be installed on your system.

If LEDA support is enabled the first time, the script tests whether LEDA is installed in standard system directories. If this test does not succeed, you have to supply directories containing the LEDA header files (`LEDA_INCL_DIR`) and LEDA libraries (`LEDA_LIB_DIR`). Even if the tests are passed, you still have the option to set these directories differently.

[4] Note that these files are only OS/compiler specific, i.e. there are no different files for with and without LEDA support.

LEDA Menu

<**E**> Enable/Disable LEDA support in CGAL.

<**I**> (*present if* LEDA *support is enabled)* Set the include directory for LEDA.

<**J**> (*present if* LEDA *support is enabled,* LEDA *headers have been found in a system include directory and* `LEDA_INCL_DIR` *has been set)* Use LEDA header from system include directory.

<**L**> (*present if* LEDA *support is enabled)* Set the directory containing the LEDA libraries.

<**M**> (*present if* LEDA *support is enabled,* LEDA *libs have been found in a system lib directory and* `LEDA_LIB_DIR` *has been set)* Use LEDA libraries from system lib directory.

6.4 The GMP Menu

This menu is to setup GMP (GNU Muptiple Precision Library) specific options, if you plan to use GMP together with CGAL. In the `auxiliary` directory you can find a GMP distribution, if you do not already have it installed on your system. This menu contains an option to install GMP in you CGAL directory tree, but of course you can also install it independently from CGAL.

If GMP support is enabled the first time, the script tests whether GMP is installed in standard system directories or in the CGAL tree. If this test does not succeed, you have to supply directories containing the GMP header files (`GMP_INCL_DIR`) and GMP libraries (`GMP_LIB_DIR`). Even if the tests are passed, you still have the option to set these directories differently.

GMP Menu

<**C**> Install the GMP distribution shipped with CGAL in the CGAL directory tree.

<**G**> Enable/Disable GMP support in CGAL.

<**I**> (*present if GMP support is enabled)* Set the include directory for GMP.

<**L**> (*present if GMP support is enabled)* Set the directory containing the GMP libraries.

<**M**> (*present if GMP support is enabled, there is a GMP installation in system directories or in the* CGAL *tree and* `GMP_INCL_DIR` *or* `GMP_LIB_DIR` *have been set)* Use GMP installation from system directories / CGAL tree.

7 The non-interactive mode

To run the install script in the non-interactive mode, go to the `CGAL-2.0` directory and enter the command

```
./install_cgal -ni <compiler>
```

where `<compiler>` is the C++ compiler executable.
You can either specify a full path, e.g. `/usr/local/bin/g++`, or just the basename, e.g. `g++`, which means the script searches your `PATH` for the compiler location. If your compiler call contains whitespaces it has to be quoted, e.g. `./install_cgal -ni "CC -n32"`. The options given this way become part of your CGAL-OS description (see section 8) which is useful e.g. to distinguish between different compilers using the same frontend (SGI Mips(Pro) `CC` on `IRIX6`).

There are a number of additional command line options to customize your CGAL setup which are discussed below. You should read the corresponding paragraphs before you continue, especially if one or more of the following conditions apply to you:

- you want to use LEDA together with CGAL (section 7.1) *or*
- you want to use GNU GMP together with CGAL (section 7.2).

Once you started the script, it should give you a message indicating the CGAL version you are going to install and that you are running the non-interactive mode. Then it proceeds by locating some utility programs, determining your OS and compiler version and displaying the settings you gave via command line. Your compiler is also checked for a number of bugs resp. support of certain language features; a message `ok` always indicates that your compiler works as it should i.e. a feature is supported resp. a bug is *not* present. On the other hand `no` resp. `unfortunately` indicate a lack of support resp. presence of a bug.

Finally the current setup is summarized, system specific directories for makefiles and libraries are created (if they did not exist before) and a new include makefile is written into the makefile directory. If there already exists a makefile for the current OS/compiler combination, it is backuped and you should get a corresponding message.

To compile the CGAL libraries go now to the `src` directory. Then type `make -f makefile_lib` to compile the CGAL object library and `make -f makefile_sharedlib` to compile the CGAL shared object library. If you want to make changes to the makefiles first, see section 9 for an explanation of the makefile structure of CGAL.

When this is finished it would be a good idea to print and read the 'Getting Started with CGAL' document `getting_started.ps` that can be found in the `doc_ps` directory.

7.1 Setting up LEDA support

See also section 11. By default there is no support for LEDA, but you can change this easily by use of the command line option "`-leda`". If LEDA is installed in system directories on your system, you should indicate this by setting the flags "`--leda-sys-incl`" resp. "`--leda-sys-lib`". If this is not the case, you have to supply the directories containing the LEDA header files ("`--LEDA_INCL_DIR` <*dir*>") resp. the LEDA libraries for your compiler ("`--LEDA_LIB_DIR` <*dir*>").

7.2 Setting up support for GMP

By default there is no support for GMP, but you can change this easily by use of the command line option "`-gmp`". If GMP is installed in system directories on your system, you are already done now. If this is not the case, you have to supply the directories containing the GMP header files ("`--GMP_INCL_DIR` <*dir*>") and the GMP library ("`--GMP_LIB_DIR` <*dir*>").

7.3 Other Options

There are some less important features of the install script we will summarize here.

First of all you can get the version number of `cgal_install` with option "`--version`". Note that all other options are ignored in this case.

Second there is an option "`-os` <*compiler*>" where <*compiler*> is your C++ compiler. This allows you to determine your CGAL-OS description (see section 8). The compiler can either be given by an absolute path like

```
./install_cgal -os /usr/local/gcc-2.8.1/sun/bin/g++
```

or just by denoting its basename, as long as it is on your path:

```
./install_cgal -os CC
```

The option is intended for testing purposes and automatic detection of the correct include makefile (see also section 9).

Finally there exists an option "`--verbose`" that can be set in interactive mode as well as in non-interactive mode. When set you get a detailed summary of error messages occurring during *any* compiler test (determining STL version etc.). Normally you only get these messages, if a required test (such as the general STL test) fails, otherwise you are just informed, *if* it succeeded or not. This option is not recommended for general use, but it can be useful to check why a certain test fails that was expected to be passed.

8 Identifying OS and Compiler

Since CGAL supports several different operating systems and compilers, this is also reflected in the structure of the CGAL directory tree. Each OS/compiler combination has its own lib directory under `CGAL-2.0/lib`) (and analogously its own include makefile in `CGAL-2.0/make`) named as determined by the following scheme.

<*arch*>_<*os*>-<*os-version*>_<*comp*>-<*comp-version*>[_LEDA]

<**arch**> is the system architecture as defined by "`uname -p`" or "`uname -m`",

<**os**> is the operating system as defined by "`uname -s`",

<**os-version**> is the operating system version as defined by "`uname -r`",

<**comp**> is the basename of the compiler executable (if it contains spaces, these are replaced by "-") *and*

<**comp-version**> is the compiler's version number (which unfortunately can not be derived in a uniform manner, since it is quite compiler specific).

The suffix _LEDA is appended to indicate LEDA support.

We call the resulting string CGAL-OS description.
Examples are `mips_IRIX-6.2_CC-7.2` or `sparc_SunOS-5.5_g++-2.8.1_LEDA`.
You can use the install script to get your CGAL-OS description, see section 7.3.

9 The CGAL makefile structure

The CGAL distribution contains the following makefiles:

- `CGAL-2.0/src/makefile_lib` for compiling the CGAL object library `libCGAL.a`,
- `CGAL-2.0/src/makefile_sharedlib` for compiling the CGAL shared object library `libCGAL.so`,
- `CGAL-2.0/src/makefile_geomview` for compiling a library for geomview support,
- `CGAL-2.0/examples/makefile` as sample makefile *and*
- `CGAL-2.0/examples/*/makefile` for compiling the CGAL example programs.

All these makefiles are generic: they can be used for more than one compiler. To achieve this, the first section of each makefile contains an include statement that looks as follows:

```
CGAL_MAKEFILE = /users/jannes/CGAL-2.0/make/makefile_<CGAL-OS description>
include $(CGAL_MAKEFILE)
```

The file CGAL_MAKEFILE is an include file with platform dependent makefile settings. The abbreviation <CGAL-OS description> (see section 8 for details) is used to identify the operating system and compiler for which the settings hold. For example, the file makefile_mips_IRIX64-6.5_CC-n32-7.2.1.1m contains makefile settings for the IRIX 6.5 operating system and the SGI Mips(Pro) CC 7.2.1 compiler. These include files are automatically generated by the install_cgal script and they are all located in the CGAL-2.0/make directory. For convenience, the install_cgal script will substitute the include makefile that was generated most recently.

If you want to compile an application or an object library with a different compiler, the only thing you need to do is to substitute another include makefile for the CGAL_MAKEFILE variable. An alternative way to do this is to create an environment variable CGAL_MAKEFILE. To pass the value of the environment variable to the makefile you can either comment out the CGAL_MAKEFILE line in the makefile or use an appropriate command line option for the make utility. A comfortable way to set CGAL_MAKEFILE is by using install_cgal -os (see section 7.3). E.g. if your compiler is g++, you would type

CGAL_MAKEFILE=‘<*insert your CGAL-2.0 dir*>/install_cgal -os g++‘

in bourne shell resp.

setenv CGAL_MAKEFILE ‘<*insert your CGAL-2.0 dir*>/install_cgal -os g++‘

in csh derivatives.

Tip: Include the setting of CGAL_MAKEFILE into your shell startup script (e.g. .(t)cshrc for (t)csh or .bashrc for bash).

All makefiles contain sections with compiler and linker flags. You can add your own flags here. For example, you might want to add the flag -DCGAL_NO_PRECONDITIONS to turn off precondition checking. The flags $(CGAL_CXXFLAGS) and $(CGAL_LDFLAGS) should never be removed.

The default extension for CGAL source files is .C. The last section of the makefiles contains a suffix rule that tells the compiler how to create a .o-file from a .C-file. If you want to use the default rule that is defined by the make utility, you may want to remove this suffix rule. However, note that this may have consequences for the makefile variables CGAL_CXX and CXXFLAGS.

10 Compiling a CGAL application

The directory CGAL-2.0/examples contains a small program (example.C) and a sample makefile with some comments. The CGAL_MAKEFILE variable in this makefile (see section 9) is automatically substituted by the install_cgal script and equals the most recently generated include makefile in the CGAL-2.0/make directory. After the installation of CGAL this sample makefile is ready for use. Just type 'make example' to compile the program example.C. There is a script for conveniently creating makefiles for CGAL applications, see section 16.1.

Furthermore the directories CGAL-2.0/examples and CGAL-2.0/demo contain many subdirectories with non-graphical and graphical example programs. In all these directories you will find a makefile that is ready for use.

11 Using CGAL and LEDA

This section describes how to use CGAL and LEDA simultaneously.

11.1 Support for LEDA

CGAL supports LEDA in the following ways:

1. There are support functions defined for the LEDA number types `big_float`, `integer`, `rational` and `real` (see the files `<CGAL/leda_*>`).

2. CGAL defines the following LEDA-related compiler flags:

   - When LEDA is used, the flags `CGAL_USE_LEDA` and `LEDA_PREFIX` will be set.
   - When LEDA is used, the LEDA memory management (LEDA handles) will be used for geometric primitives in CGAL. This can be turned of by setting the flag `CGAL_NO_LEDA_HANDLE`. In that case CGAL memory management will be used (see `<CGAL/Handle.h>`).

The include makefiles in the `CGAL-2.0/make` directory corresponding to LEDA can be recognized by the suffix "`_LEDA`".

12 Troubleshooting

This section contains some remarks about known problems and the solutions we propose. If your problem is not listed here, please have a look at the CGAL homepage:

`http://www.cs.uu.nl/CGAL`

or send an email to `cgal@cs.uu.nl`.

12.1 The "Long-Name-Problem" (Solaris only)

The system assembler and linker on Solaris 2.5 and 2.6 cannot handle symbols with more than 1024 characters. But this number is quickly exceeded where one starts nesting templates into each other. So if you encounter strange assembler or linker errors like

```
/usr/ccs/bin/as: "/var/tmp/cc0B5iGc.s", line 24:
error: can't compute value of an expression involving an external symbol
```

there is a good chance that you suffer from this "long-name" problem.

A solution is to install the GNU-binutils[5] and to tell the compiler that it shall use the GNU– instead of the native tools. From the compiler-menu (described in section 6.2) you can set the corresponding option through the custom compiler flags, i.e. for `gcc/egcs` you would add

```
-B/my/path/to/gnu/binutils/bin
```

assuming you installed the GNU-binutils executables in `/my/path/to/gnu/binutils/bin`.

If you cannot (or do not want to) install GNU-binutils, there is a workaround that lets you compile, link and run your programs, but it prevents debugging, since the executables have to be stripped. In short the workaround is to compile with `-g` and to link with `-z nodefs -s`. In order to still have portable makefiles (see section9), we define flags `LONG_NAME_PROBLEM_CXXFLAGS` and `LONG_NAME_PROBLEM_LDFLAGS` in the include makefiles which are empty except for the Solaris platform where they are set as stated above. In order to use these flags, edit your application makefile and add the flags to `CXXFLAGS` resp. `LDFLAGS` as indicated below.

```
#--------------------------------------------------------------------#
#                     compiler flags                                 #
#--------------------------------------------------------------------#
```

[5] see `http://www.gnu.org/software/binutils/`

```
# The flag CGAL_CXXFLAGS contains the path to the compiler and is defined
# in the file CGAL_MAKEFILE. You may add your own compiler flags to CXXFLAGS.

CXXFLAGS = $(LONG_NAME_PROBLEM_CXXFLAGS) $(CGAL_CXXFLAGS)

#--------------------------------------------------------------------#
#                    linker flags
#--------------------------------------------------------------------#
# The flag CGAL_LDFLAGS contains common linker flags and is defined
# in the file CGAL_MAKEFILE. You may add your own linker flags to CXXFLAGS.

LDFLAGS = $(LONG_NAME_PROBLEM_LDFLAGS) $(CGAL_LDFLAGS)
```

12.2 LEDA and STL conflicts

If you are using an old version of LEDA, the combination of LEDA and STL may give some problems. In order to avoid them, it is highly recommended to use the latest LEDA release[6], since this is what we test CGAL with.

13 Compiler Optimisations

You may have noticed that we do not set optimizer flags as `-O` by default in the include makefiles(see section9 for a description of the makefile structure in CGAL). The main reason for not doing this is that compilers run much more stable without. On the other hand, most if not all CGAL programs will run considerably faster when compiled with optimisations! So if you are going for performance, you should/have to add `-O`, `-O3` or maybe more specific optimizer flags (please refer to the compiler documentation for that) to the `CXXFLAGS` variable in your application makefile:

```
#--------------------------------------------------------------------#
#                    compiler flags
#--------------------------------------------------------------------#
# The flag CGAL_CXXFLAGS contains the path to the compiler and is defined
# in the file CGAL_MAKEFILE. You may add your own compiler flags to CXXFLAGS.

CXXFLAGS = $(CGAL_CXXFLAGS) -O
```

14 Upgrading a CGAL 1.* installation

In case you have CGAL 1.0/1.1/1.2 installed on your system, you might like to reuse your configuration files and GMP installations. Simply use the following command to copy them into the right place:

```
./install_cgal --upgrade <OLD_CGAL_DIR>
```

where `<OLD_CGAL_DIR>` is your CGAL 1.* root directory (e.g. `/pub/local/CGAL-1.2`).

You can then build all libraries for the actual operating system that existed in your CGAL 1.* installation with

```
./install_cgal --rebuild-all
```

If you want to install CGAL for more than one operating system in the same directory structure, you have to run the latter command (`rebuild-all`) once on each operating system.

[6] At the moment this is LEDA 3.8.

15 Compiler workarounds

In CGAL a number of compiler flags is defined, all of them start with the prefix CGAL_CFG. These flags are used to work around compiler bugs and limitations. For example, the flag CGAL_CFG_NO_MUTABLE denotes that the compiler does not know the keyword mutable.

For each compiler a file <CGAL/compiler_config.h> is defined, with the correct settings of all flags. This file is generated automatically by the install_cgal script. For this the test programs in the directory CGAL-2.0/config/testfiles are used. The file <CGAL/compiler_config.h> and the test programs contain a description of the problem, so in case of trouble with a CGAL_CFG flag it is a good idea to take a look at it.

The file <CGAL/config.h> manages all configuration problems of the compiler. This file includes the file CGAL/compiler_config.h. It is therefore important that the file <CGAL/config.h> is always included before any other CGAL source file that depends on workaround flags. In most cases you do not have to do anything special for this, because many CGAL files already take care of including <CGAL/config.h>. Nevertheless it would be a good idea to always start your CGAL programs with including <CGAL/config.h> (or <CGAL/basic.h>, which contains some more basic CGAL definitions).

16 Scripts

16.1 create_cgal_makefile

The bourne-shell script create_cgal_makefile is contained in the CGAL-2.0/scripts directory. It can be used to create makefiles for compiling CGAL applications. Executing create_cgal_makefile in an application directory creates a makefile containing rules for every *.C file there.

In order to use this makefile, you have to specify the CGAL include makefile (see section 9) to be used. This can be done be either setting the environment variable CGAL_MAKEFILE or by editing the line

```
# CGAL_MAKEFILE = ENTER_YOUR_INCLUDE_MAKEFILE_HERE
```

of the created makefile. First remove the "#" at the beginning of the line and then replace the text after "=" by the location of the include makefile.

Finally type make to compile the application programs.

16.2 use_cgal_namespace

The perl script use_cgal_namespace is contained in the CGAL-2.0/scripts directory. It can be used to convert CGAL-1.* application sourcecode to the CGAL-2.* format. Basically, it replaces CGAL_ prefixes by CGAL:: namespace qualifiers. You have to give the files to convert as arguments, e.g.

```
use_cgal_namespace my_great_file1.C *.h
```

The original files are kept with the suffix .bck.

In order to use it, you first have to set the perl path in the first line correctly, i.e. replace /net/bin/perl5 by the path to perl on your system (try which perl(5), if you do not know). Alternatively, you can type

```
perl -wi.bck -- use_cgal_namespace <FILES>
```

16.3 replace_headers

The perl script redirect_headers is contained in the CGAL-2.0/scripts directory. It can be used to replace oldstyle include directives for standard headers by their standard ISO counterparts, e.g.

```
#include <algo.h>
```

is replaced by

```
#include <algorithm>
```

You have to give the files to convert as arguments, e.g.

```
use_cgal_namespace my_great_file1.C *.h
```

The original files are kept with the suffix `.hrbck`.

In order to use it, you first have to set the perl path in the first line correctly, i.e. replace `/net/bin/perl5` by the path to perl on your system (try `which perl`(5), if you do not know). Alternatively, you can type

```
perl -wi.hrbck -- replace_headers <FILES>
```