CMSC 23700 **Introduction to Computer Graphics** **Handout 2**
Fall 2003 **September 30**

Lab tips

This handout provides an introduction to some of the tools you will use to complete the programming projects.

Getting Started

You will need an account on the CS machines (this is different from your harper account). If you do not already have one, you can request one at

`www.cs.uchicago.edu/info/services/account_request`

You may work on your projects using either Linux or MacOS X. Both Linux and MacOS X machines with good-quality graphics cards are available for use in the CS Instructional Computing Laboratory (MacLab) located on the A-Level of Regenstein Library. There are some advantages to the MacOS X environment as it provides OpenGL performance monitoring tools, but feel free to work in the environment that you are most comfortable with.

You will be expected to use CVS for your projects. We will set up CVS repositories for you. Projects will be collected for grading directly out of your CVS repository.

Using OpenGL and GLUT functions in your C programs

In order to use OpenGL and GLUT functions in your program you will need to include the appropriate header files. The file `glut.h` header file includes the OpenGL header files (`gl.h` and `glu.h`), so it is the only one you will need to include. Unfortunately, Linux and MacOS X differ in where they put the `glut.h` file. The following bit of preprocessor code will allow your program to compile on both platforms:

```
#if defined(__APPLE__) && defined(__MACH__)
#  include <GLUT/glut.h>
#else
#  include <GL/glut.h>
#endif
```

Compiling under Linux

On Linux systems, you should use **gcc** version 3.0, which accessed under the name `gcc-3.0`. To compile and link an OpenGL program under Linux, you must use the following linking options:

```
-lglut -lGL -lGLU -lm -L/opt/xfree86/default/lib
```

Compiling under MacOS X

MacOS X also uses **gcc** as its default C compiler. Apple uses a different set of linking flags from Linux. To link an OpenGL program, you need the following linker flags:

```
-framework GLUT -framework OpenGL -framework Foundation
```

The MacLab machines also have two IDEs installed: Metrowerks **CodeWarrior** and Apples **ProjectBuilder** (the latter uses **gcc** as its compiler). You may use these systems to develop and debug your projects, but please include a makefile in you submissions.

Makefiles

For each of your projects, you should include a makefile in your submission. We will provide a skeleton makefile for you, but you are responsible for maintaining it. For a simple project, such as Project 0, that contains only a single source file, the following makefile will suffice:

```
SHELL =                 /bin/sh

ifeq ($(shell uname -s),Darwin)
  CC = cc -std=gnu99
  LDFLAGS = -framework GLUT -framework OpenGL -framework Foundation
else
  CC = /usr/bin/gcc-3.0 -std=gnu99
  LDFLAGS = -lglut -lGL -lGLU -lm -L/opt/xfree86/xfree86-4.3.0/lib
endif

project0:        main.c
        $(CC) $(CFLAGS) -o project0  main.c $(LDFLAGS)

clean:
        rm -rf project0
```

This makefile works on both Linux and MacOS X by setting the LDFLAGS make variable based on the host OS. If you have not used **make** before, you should take a look at the documentation. Information about make is available at `www.gnu.org/software/make` and online documentation can be found at `www.gnu.org/manual/make/html_chapter/make.html`.

Using CVS

You are expected to keep the source code of your projects in a CVS repository that we will setup for you. If you CS account user ID is `joebob`, then the *root* of you repository will be

```
cvs.cs.uchicago.edu:/stage/cmsc237/students/joebob
```

For each project, we will create a CVS *module*. In the rest of this section, we give a quick introduction to using CVS.

To use CVS, you will first need to set the following environment variables (using **csh** syntax):

```
setenv CVS_SERVER /usr/local/bin/cvs
setenv CVS_RSH ssh
setenv CVSROOT cvs.cs.uchicago.edu:/stage/cmsc237/students/joebob
```

The last of these assumes that you login ID is "`joebob`." Instead of setting the `CVSROOT` variable, it is also possible to use the "`-d`" command line option when issuing CVS commands (see below).

You are now ready to checkout your project. A CVS repository with a module called "`project-0`" will already have been created for you. To *checkout* a copy of this module, run the following command:

```
cvs co project-0
```

or, if you didn't set `CVSROOT`,

```
cvs -d cvs.cs.uchicago.edu:/stage/cmsc237/students/joebob co project-0
```

This command will create a directory called `project-0`. In this directory is another directory called `CVS`, which holds various metadata about this copy of the repository — *you should not change anything in this directory*. You will also find a file named `Makefile`, which we have provided for you. All the files related to your project should live in the `project-0` directory.

Now suppose you create a file called `main.c` in your `project-0` directory. In order for CVS to keep track of it, it needs to be added to the repository. You do this by the following command:

```
cvs add main.c
```

You should see a message like:

```
cvs add: scheduling file ‘main.c’ for addition
cvs add: use ’cvs commit’ to add these files permanently
```

This command records the fact that `main.c` has been added to the repository, but file will only be added when you commit your changes. To do so, type the following command:

```
cvs commit
```

to add the files permanently to the repository. You will be prompted to enter a log message in an editor. To specify a particular editor for entering log messages, set the `CVSEDITOR` environment variable. You can also avoid editors altogether by typing your log message on the command line with the `-m` flag:

```
cvs commit -m "added files"
```

After you have entered your message, you will see a message like the following:

```
/tmp/cvsCBrFgq: 9 lines, 337 characters.
Checking in main.c;
/stage/cmsc237/students/joebob/project-0/main.c,v  <--  main.c
initial revision: 1.1
done
```

Changes you make to your files are recorded in the repository every time you do a `cvs commit`. Before you make changes to your files, you can ensure that you have a current version, by running `cvs update`. This fact is not of tremendous significance for individual projects, but matters when more that one person can modify the same files.

Not all the files in your project directory need to be in the repository. For example you should not put your executble files in the repository — these can always be recreated (hopefully!) by compiling the source.

The "`cvs diff`" command is for comparing differences between versions. If no files (or options) are specified, all working files are compared to their last committed versions, otherwise only the specified files are compared. There are also flags to compare other versions, see the man pages or the online manual for details.

Useful resources

There are links to some useful Computer Graphics resources on the course web page at

`www.classes.cs.uchicago.edu/archive/2003/fall/23700/`

Information about make is available at `www.gnu.org/software/make/` and online documentation can be found at `www.gnu.org/manual/make/html_chapter/make.html`.

The CVS home page is at `www.cvshome.org/`. Official documentation is at `www.cvshome.org/docs/manual/`. There is also a nice introduction to CVS at `www.cvshome.org/docs/blandy.html`.