Winter 2003 ICS 186A
Lab Assignment 0

Due: Jan 10, 2003, 5:00 PM PDT

1 Lab Grading Policies

Before introducing this lab, let us review the policies which we will be enforcing for all lab assignments this quarter.

You are required to program in C and/or C++ using the GLUT toolkit and the OpenGL API. You may not use Java. All your assignments will be graded on Sun Ultra 60 workstations running the Solaris 8 operating system using either the GNU C/C++ compiler or Sun's Forte C/C++ compiler.

Since this course is using GLUT and OpenGL, you should be able to complete your lab assignments on any machine that has GLUT and OpenGL installed. There are twelve Sun Ultra 60 workstations in CS 364 for your use in this course. However, you can use any platform you like. This includes Microsoft Windows PCs, Macintoshes, Linux PCs, and any other Unix platform. The ICS labs have many Microsoft Windows PCs available with OpenGL and GLUT installed as well as additional Sun workstations with unaccelerated graphics hardware. Claiming to lack access to a computer to complete the lab assignments is not an adequate excuse to gain an extension on the due date of a lab.

However, it is not uncommon for you to have *minor* problems compiling your lab on Solaris, if the work was done on a different platform. This can not be used as an excuse to gain partial credit or an extension on the due date. Please be advised to complete your assignments early enough to allow you enough time to test your code on Solaris.

Furthermore, all programming assignments will be submitted electronically using the `turnin_ics186a` command available on all ICS Solaris accounts. Do not submit your assignments via email or to the ICS NT drop boxes.

As a rule of thumb, if nothing is submitted, you will earn no credit. If something is submitted and it compiles with no warnings or errors, this can earn you *up to* 50% of the total lab score. Should a submitted assignment exit abnormally, i.e., segmentation fault or bus error, the submitted assignment can earn *up to* 10% of the total lab score. If something is submitted but it does not compile, this can earn you *up to* 10% of the total lab score.

It goes without mention that you should test your code, rigorously.

Every file that is submitted must contain a header including your full name, email address, student ID number, and any other pertinent information. Every submitted project must contain a `README` file in plain text format. The file must be called `README`. It must also include a header. The information that must be covered in your `README` are what was completed in this lab, the existence of bugs, any additional features that you added to the project which went above and beyond the scope of the assignment. Also bear in mind as you compose your `README` that the use of complete sentences is always appreciated.

That last point is very important. If you would like to gain extra credit for figuring out something really cool or doing something quite fantastic, make sure you point it out in your `README`. This *may* earn you extra points.

(Of course, cheating/copying/academic dishonesty is not permitted. Please do not volunteer to be an example of what happens when someone is caught committing an unethical act.)

2 Overview

This lab assignment is to familiarize yourself with the programming environment and procedures that you will be using for this course. Your assignment will not involve any programming on your part, but is solely an exercise to ensure that any questions that you may have are answered now rather than the day before an actual lab is due.

This lab can be roughly broken up into five parts.

1. Logging in
2. Making sure your account works
3. Where to get help
4. Compiling & running a demonstration program
5. Submitting a completed lab

3 Logging In

You will need an ICS Solaris account for this course. This is different from your NACS EA Unix account. On top of this, you will need to confirm that your account is enabled to use the Sun Ultra 60s in CS 364.

To do this, go to the computer lab in CS 364 (aka 'the 3rd floor lab'). Locate the Sun Ultra 60s which are located near the networking racks. Ask the lab attendant if you have difficulty locating the Ultra 60s.

Once you have found them, log into one of them using your ICS Solaris account username and password. If you can not log in, send an email to support@ics.uci.edu informing them of your full name and your ICS login. Explain in the email that you are enrolled in ICS 186a and need access to the graphics workstations. Please 'cc' your instructor on this email, (gopi@ics.uci.edu). Once your account is enabled, you can continue with this lab.

4 Making Sure Your Account Works

If you can log in, we will need to edit your `.cshrc` to make sure you have the proper module loaded for this course. Using your favorite text editor open your `.cshrc`. For example, I would type this at the command line:

```
prompt% module load emacs
prompt% emacs .cshrc &
```

This will pop up a new window with your `.cshrc` ready for editing.

Using the arrow keys on the keyboard, move down to the line where it begins like

```
module load std-unix ics-default...
```

On this line, add the following modules at the end of the list if you have not already added them: `sun-ccs`, `sun-lang/6.1`, `openwin`, `glut`, `tkman` and `ssh`.

Log out and then log back in. If you have errors logging back in, ask the person next to you for help.

To make sure everything is working correctly, run the following commands to see if you get the same output:

```
prompt% which xlogo
/usr/openwin/bin/xlogo
prompt% which cc
/opt/forte-6.1/SUNWspro/bin/cc
prompt% where make              # For make, any one of the following is OK
/opt/contrib/bin/make
/usr/ccs/bin/make
/opt/gnutools-1.1/bin/make
prompt% which ssh
```

```
/opt/ssh-2.4.0/bin/ssh
prompt% which tkman
/opt/contrib/tkman-2.1/bin/tkman
```

If you do not get the same output as the above, ask someone next to you for help.

5 Where To Get Help

The worse thing is to be stuck. The best way to stay un-stuck is to know where to find information. Students usually get stuck in one of several ways; *how do I do this?*, *how do I use this?*, *I'm lost - heaven help me.*

To answer the *how do I do this?* question, you have several resources available to you. You may only need to see an example of how to do something very basic in order to jump start your own creative thinking. From one of the Ultra 60 workstations you can look in this directory: `/opt/glut-3.7/progs`. In this directory, there are collections of pre-built sample programs complete with source code to help you understand how to use OpenGL and GLUT. The `redbook` directory contains the simplest of the examples and are pulled from the OpenGL redbook (aka The OpenGL Programming Guide).

For example, log into one of the Ultra 60s and from the command line issue the following commands:

```
prompt% cd /opt/glut-3.7/progs/redbook
prompt% ls              # This will actually list a lot of files
aaindex.c   aargb.c     accpersp.c  alpha3D.c   bezcurve.c
aapoly.c    accanti.c   alpha.c     anti.c      bezmesh.c
prompt% ./bezmesh &
[1] 4689
prompt%
# This will pop up a window with a grey curvey surface.
prompt% less bezmesh.c
# This will display the contents of bezmesh.c to your window. Use the
# space bar to go forward and the 'b' key to go backward.
```

Should you have more fundamental questions, then perhaps referring to a reference book would be best. If you don't already own the redbook (The OpenGL Programming Guide), then you can browse it online at http://kate.ics.uci.edu/library/SGI_bookshelves/SGI_Developer/books/OpenGL_PG/sgi_html/index.html. This link only works from within UCI. That means if you use Cox Cable or Pacific Bell as your ISP, you will not be able to view this page.

If you find yourself stuck because you can not figure out how to use a particular tool or function call, then the first place to look for help are the *man* pages. The *man* pages are the online Unix manuals which cover almost everything. The first man page that you should look at is the man page on *man*. So at the prompt, type the following command:

```
prompt% man man
```

This will display a very terse description of how the *man* command works.

You can even look up function calls. Lets say for example you can not remember how to use the OpenGL function `glVertex3fv`. With the command `man glVertex3fv` a manual page will be displayed reminding you of the proper syntax. (This page will only display if you correctly added the `openwin` module.)

Some of you may prefer to have a graphical man page browser. There is one available at ICS called `tkman`. To start `tkman`:

```
prompt% tkman &
```

This will bring up a new window which will let you browse all the different man pages.

The FreeBSD project also has a nice web front end to many different man pages that you can find at http://www.FreeBSD.org/cgi/man.cgi?manpath=sunos5.

In particular, if you need to know how to use OpenGL function calls the two best places to look are the OpenGL man pages and the bluebook (The OpenGL Reference Manual). The bluebook is essentially a reprint of the man pages. You can also find the bluebook online at http://kate.ics.uci.edu/library/SGI_bookshelves/SGI_Developer/books/OpenGL_RM/sgi_html/bk02.html. Again, this web page is only accessible from within UCI.

Perhaps you mutter to yourself 'heaven help me - what is all this madness!' It is better to say this right at the beginning rather than three weeks into the course. If you are in this situation there is a course newsgroup where you can post questions of a general nature. Here you can get help from the TA and your peers. It is best to keep up to date with the discussions on the newsgroup since many of your questions will be answered even before you know about the question.

To access the newsgroup, you need to login into EEE and can use this URL:
http://eee.uci.edu/toolbox/noteboard/index.php?board=1054k

If you have problems reading and posting to the newsgroup, ask one of your classmates to help you.

As a part of this lab, post a message to the newsgroup. Confirm that the message is there.

If you are in dire straits, email your TA (kartic@ics.uci.edu) or professor to set up an appointment. Email is a great communication tool, but please remember that although professors, and to a lesser degree TAs, are blessed with uncanny intellectual abilities, they are not mind readers. Writing clear and concise email can greatly improve the quality of the response you receive.

6 Compiling & Running A Demonstration Program

The final portion of this lab is to make a directory in your home directory, copy some files into that directory, compile them and finally run the resulting binary.

The following will be a set of Unix shell commands that, if followed exactly, will make a directory and copy some files into that directory. (Everything after the `#` is a comment.)

```
prompt% cd                         # Change directory to your home directory
prompt% mkdir ics186               # Make a new directory
prompt% ls ics186                  # List the contents of the new directory
                                   # Nothing is printed meaning its empty
prompt% cd ics186                  # Change directories into the new directory
prompt% mkdir asgt0                # Make another new directory
prompt% cd asgt0                   # Change into the new directory
prompt% cp -r ~graphics/teaching/opengldemo/* .# See below for the comment
prompt% ls                         # List the contents of this directory
Makedepend           alternate_makefiles/  opengldemo*
Makedepend.bak       cube.c                sceneModule.c
Makefile             inputModule.c         sceneModule.h
RCS/                 inputModule.h         viewModule.h
```

The `cp -r` command means copy all the files and directories recursively. Without the `-r` only the files will be and not the directories.

Now that we have copied over the files, you can use the `make` command to build the project. First, we clean the project to get rid of any old object files. Then we build the entire project. Follow the example below to build the project.

```
prompt% make clean                 # Delete all the object files and other junk
rm -f  cube.o inputModule.o sceneModule.o core opengldemo.core *~
prompt% ls                         # List the files
Makedepend           alternate_makefiles/  opengldemo*
Makedepend.bak       cube.c                sceneModule.c
Makefile             inputModule.c         sceneModule.h
RCS/                 inputModule.h         viewModule.h
prompt% make                       # Build the entire project
cc -xCC -g +w2 -xarch=v9 -I/opt/glut-3.7/include -L/opt/glut-3.7/lib/sparcv9\\
 -L/usr/openwin/lib/sparcv9 -L/usr/lib/sparcv9 -c cube.c
cc -xCC -g +w2 -xarch=v9 -I/opt/glut-3.7/include -L/opt/glut-3.7/lib/sparcv9\\
 -L/usr/openwin/lib/sparcv9 -L/usr/lib/sparcv9 -c inputModule.c
cc -xCC -g +w2 -xarch=v9 -I/opt/glut-3.7/include -L/opt/glut-3.7/lib/sparcv9\\
 -L/usr/openwin/lib/sparcv9 -L/usr/lib/sparcv9 -c sceneModule.c
CC -g +w2 -xarch=v9 -I/opt/glut-3.7/include -L/opt/glut-3.7/lib/sparcv9\\
 -L/usr/openwin/lib/sparcv9 -L/usr/lib/sparcv9 -o opengldemo  cube.o\\
 inputModule.o sceneModule.o -lglut -lGL -lGLU -lXmu -lXext -lX11 -lm
prompt% ls                       # Notice the .o files and the updated binary
Makedepend            cube.c                opengldemo*
Makedepend.bak        cube.o                sceneModule.c
Makefile              inputModule.c         sceneModule.h
RCS/                  inputModule.h         sceneModule.o
alternate_makefiles/  inputModule.o         viewModule.h
prompt% ./opengldemo             # Run the project
```

Use the escape key to quit the program.

Open one of the files and take a look at it. Find the line in `cube.c`'s `main` where the window's title is set. Change the name of the window to your login name. Save the file and rebuild the project using the `make` command. Notice that if you only made a change to `cube.c`, `make` only re-compiles `cube.c` and links the object files.

7 Submitting A Completed Lab

The next step is to package up your project to be submitted to the grader. To do this you will use two commands, `tar` and `turnin_ics186a`.

The first step is to clean the project. So from the project directory do the following command.

```
prompt% make spotless
rm -f  cube.o inputModule.o sceneModule.o core opengldemo.core *~
rm -f opengldemo
prompt% ls
Makedepend            alternate_makefiles/  sceneModule.c
Makedepend.bak        cube.c                sceneModule.h
Makefile              inputModule.c         viewModule.h
RCS/                  inputModule.h
```

Now the directory is completely clean and only the source files are left for you to package. Then use the following commands to package the project into a tar file.

```
prompt% pwd                     # Print the current directory's name
/home/gopi/ics186/asgt0
prompt% cd ..                   # Go up one directory
prompt% pwd                     # Print the current directory's name
/home/gopi/ics186
prompt% tar cvf asgt0.tar asgt0/* # See the comment below
asgt0/Makedepend
asgt0/Makedepend.bak
asgt0/Makefile
asgt0/RCS/
asgt0/RCS/Makefile,v
asgt0/RCS/cube.c,v
asgt0/RCS/inputModule.c,v
asgt0/RCS/inputModule.h,v
asgt0/RCS/sceneModule.c,v
asgt0/RCS/sceneModule.h,v
```

```
asgt0/RCS/viewModule.h,v
asgt0/alternate_makefiles/
asgt0/alternate_makefiles/RCS/
asgt0/alternate_makefiles/RCS/Makefile.GNU,v
asgt0/alternate_makefiles/Makefile.GNU
asgt0/cube.c
asgt0/inputModule.c
asgt0/inputModule.h
asgt0/sceneModule.c
asgt0/sceneModule.h
asgt0/viewModule.h
prompt% ls                    # List the files, notice the asgt0.tar file
asgt0/      asgt0.tar
```

The `tar` command takes three arguments. The first argument tells tar what to do. Its made of three different letters. The first letter tells tar to *c*reate a new tar file. The second letter tells tar to be *v*erbose, that is print out all the files being added to the tar file. The third letter tells tar what the *f*ilename of the tar file is going to be. The next argument is the name of the tar file. And the last argument is the place where all the source files are stored.

The last step is to submit the resulting tar file. Submit your tar file with the following example.

```
prompt% ls                   # List the files
asgt0/      asgt0.tar
prompt% ~graphics/bin/turnin_ics186a asgt0.tar  # See the comment below
turnin_ics186a: your assignment has been accepted
```

Using the `turnin_ics186a` command takes the tar file and puts it in a safe place for the grader to look at later. The name of the tar file does not matter since the turnin command will change the name of the tar file automatically.

However, you can only submit your files once. If you try to resubmit your tar file you will see the following.

```
prompt% date
Wed Oct  2 19:40:29 PDT 2002
prompt% ~graphics/bin/turnin_ics186a asgt0.tar
turnin_ics186a: you turned in your assignment on Wed Oct  2 19:36:43 2002
```

You can see that the turnin command will just tell you when you originally turned in your homework. If you would like to resubmit your homework, all hope is not lost. Just email your TA and everything will be taken care of. But try to avoid it as much as possible.

8 Conclusion

After reading all this and doing all the exercises listed here you should be well prepared for the next nine weeks. Please be sure that you have completed the following:

1. Your ICS Solaris account works
2. You can use the graphics workstations in CS 364
3. You have changed the window title of the OpenGL demo program
4. You have posted and read a message on ics.186a
5. You have submitted a tar'ed project