

CSI 422/502 – Computer Graphics

Project Grading Policies

Instructions for Turning In Project Programs

The command to use (until further notice) to compile openGL programs for this course on UAlbany's ITSUNIX system is:

```
gcc -g -lglut -lGLU -lGL -lXft -lXi -lm [one or more .c file names]]
```

You can run the resulting program with the command: `a.out`

You can also write your programs in C++. Just use the command:

```
g++ -g -lglut -lGLU -lGL -lXft -lXi -lm [one or more .cpp file names]]
```

An “X-server” is absolutely necessary for OpenGL and other graphics applications to work on Unix. Unless you know what you are doing, I expect you to use the ITS Public User Room PC's (LC-3 and LC-4). There, you must use the following steps:

1. Do Windows login with your NETID and password.
2. From the “Start” button, select the “Programs” menu. Find the “XWin-32” menu entry...it is near the bottom, so you will have to “extend” menu one or two times. The XWin-32 selection has a submenu.
3. First, select the “Start Server” item of that submenu.
4. Second, from the “Start” button (again), select the “SSH” item of the XWin-32 submenu.
5. Log in to ITSUNIX with SSH. Select password authorization if asked.

Writing the program so its modules (collections of functions and declarations that taken together have a common logical purpose) are expressed by separate files is encouraged and might be required for the more complex projects. A quick and dirty way to compile them is to put all AND ONLY the programs `.c`, `.cpp` and header files together in one directory; and in that directory, use the compile command with the file names given by `*.c` or `*.cpp` This will cause the shell to form a list of all the files with names ending in `.c` or `.cpp` and pass that list to the compile command.

A better way to give the compile command over and over is to put it in a script (like `build.sh`—see my CSI310 course) or in a Makefile (that is a CSI333 topic).

You are welcome to use your own computers instead of ITSUNIX to develop your project programs. OpenGL is pretty but not perfectly portable, so you probably won't have trouble if you use openGL properly. However:

1. Obtaining, installing, and maintaining the needed computer, OS, C/C++ compiler and openGL library implementation is FULLY your responsibility. That means failure of any of these for any reason will not be accepted as an excuse for a delay in completing an assignment. I will give you advice on this to the extent I can, with no guarantees, if you ask.

2. Your submitted work will be compiled and tested for grading on the ITSUNIX system with the commands given above (unless otherwise specified). If you develop it on a different system, you must find out how to transfer the files to ITSUNIX. Then, **COMPILE AND TEST IT YOURSELF** on ITSUNIX, fix problems if necessary, and **COMPILE AND TEST** again if you make any change to any of the files! Do this **no matter how small or innocuous** you think the change is, even if it is to improve formatting or comments! One space somewhere in a comment (like between the /'s in "//) can make a program wrong!

1 Grading

Only submitted source (.c, .cpp and header files) files will be graded. If your submission includes an object or executable files, a 5 point penalty will be deducted and they will all be removed before we try to compile your submission. If your submission contains a “core” file (a large file name “core”) we might delete your whole submission to reclaim file space quota (and email to your ITSUNIX account), and if not, deduct 25 points. “Core” files are written when your program crashed and they are useful if you know how to use a debugger.

The programming homework assignments are really “projects”. That means they entail analysis of the problem parts, learning topics relevant to their solutions, planning for incremental development, design, coding, testing, debugging and the solution of unanticipated problems spread out over the two or so week period over which the project is assigned. There will be pitfalls and problems to solve that you will not know about until you work through the project details! Most students who do no substantial work on a project before a couple of days before the due date will encounter frustration and failure.

The due dates for each project will be specified as a pair of dates, like Feb. 8-10. Projects submitted by 11:59PM of the first due date are not late. Late projects submitted after the end of the last date provide no credit towards your project score, but may be taken into consideration by the professor when deciding marginal grades. A project submitted during the “grace period” between those times will have its grade multiplied by the following lateness factor:

$$\text{LatenessFactor} = 100\% - \frac{\text{TimeSubmitted} - \text{TimeDue}}{\text{Time within grace period}} * (50\%)$$

For example, for the dates Feb. 8-10, the grace period is 2 days (48 hours); the formula means roughly 1% will be subtracted for each hour late. The calculation will be done by computer so accurately (< 1%) that roundoffs will make no difference in your final letter grade. (The penalty rate of 25% per day may be reduced and the grace period of 48 hours increased for the longer projects later in the semester.)

Project grade reports will be emailed to your `itsunix` Unix account. Consult the helpdesk or online web material for help to make sure you will receive this email.

Some projects may consist of several parts where the main result of each part is a complete program. Each part will be counted separately for grading. **A part that does not compile and link to an executable file that can be run gets no credit. Never make any**

change, not even a comment, between testing and submission. See the advice for final testing in the section on submission.

2 Cheating

Your programs are supposed to be entirely your own work. You may discuss problem and programming issues with others, but the code you submit must be written and typed in by you, except exactly as specified in the project assignment, since some projects will begin with sample programs from openGL texts or software distributions. NO OTHER WEB or other MATERIAL MAY BE COPIED.

If two programs look to me like they are too much alike to be a coincidence, both get zeros. We may use a computer program to compare submissions, then check by hand to be sure. That is not an empty threat – Computer Science professors do it several times a year, and some students never graduate because their major grades have been lowered because of cheating. All cases of cheating are reported to the Office of Undergraduate (or Graduate) Studies according to the expectation of faculty under the “Penalties and Procedures for Violations of Academic Integrity” in the Albany Undergraduate Bulletin.

Everybody’s account is initialized so nobody else can read its files. Do not change the protections (unless it’s of home and `public_html` directories to make a web page, and you know what you are doing there): Your homework may become world-readable. It is your job to keep your work safe from prying eyes.

3 Turning in Project Work

1. **Very Important (for your grade!!)** Do a final test that our command builds the submitted software. Ideally, a full regression test (using ALL test cases used in development and in debugging) should be done next. Finally, **extremely important**, carefully make sure there are no “core” files (named “core” and typically a few megabytes long) in the directories you are going to submit.

It is our experience that SPARC Solaris, compared to Linux, has more “garbage” values (fewer zeros) in uninitialized dynamic and local variables. (Standards conforming C/C++ systems are allowed to leave such variables uninitialized, for efficiency reasons.) Therefore, we have seen many student C/C++ programs that apparently work under IA-32 Linux to fail under SPARC Solaris.

2. Very carefully remove all “core” files, objects and executable files from all the directories (the specified one on down) you will submit. What is safer is to plan ahead and create a `cleanup.sh` script that automates the proper `rm` commands. That way, will you avoid the mistakes you are more likely to make when you do an unusual thing when you are under stress. (Yes, I know what kinds of course work here is stressful.)

The core must be removed so the class account storage quota is not exceeded. Object and executable files must be removed so our grading is based on the sources we can

see. Penalties will be subtracted if we receive them, especially for the core files.

3. Note the **project or exercise turnin name from the assignment**. and section number of 422 or 502 depending on which course you are in. Suppose for illustration's sake that the turnin name for the project or exercise is "proj1". (This is the actual turnin name for the first project of Fall 2005 CSI422.) When you are ready to submit your work, go (use `cd`) to the directory *directory-name* **ABOVE** the one you plan to submit. Type the command

```
turnin-csi422 -p proj1 -c csi422 directory-name
```

where *directory-name* stands for the directory to be turned in. (Seek help if the command wasn't found: It's in `/usr/local/bin`)

After you issue this command, the system responds with

```
The sections of csi422 are:
```

```
422
```

```
502
```

```
Enter your section:
```

Now type 422 if you are taking 422, but if you are taking the graduate version, figure out what to do yourself. The system responds with

```
Your files have been submitted to csi422, proj1 for grading.
```

Please note that using the turnin program as above is the **ONLY** acceptable way of submitting programming assignments in this course. You should **NOT** mail the files to the instructor or the TAs.

4. Pay attention to all error messages printed by `turnin`: An error message means your submission will not be received. Recheck the instructions, repeat until `turnin` use with `-v` described below shows successful receipt, and see a TA if positive acknowledgment is not obtained.

An error message indicating a quota is exceeded is usually caused by other students having submitted core files. If you get this message, email the professor and TAs immediately. We will investigate as soon as possible, fix the problem, and get very angry at all the students who submitted core files.

5. If you use the `turnin` command above again at a later time, then everything you submitted previously **will be replaced** by the newly submitted files. (This allows you to resubmit a program if the previous submission was erroneous. But it wipes out the submission time record of the previous submission, so your revised submission will be more late.)
6. For the bigger projects, if you have a good reason to turn in a revised or corrected version of your work without wiping out the record of your previous version, get permission from the professor or any TA to send the revision to an alternate "**-extra**" project. Permission generally will be given, but late version will be penalized for some lateness.

7. If you run into a programming jam, you can email the professor (`sdc@cs.albany.edu`) with a message describing the **specific** problem, and if you wish, a note indicating that you have turned in ALL THE FILES one needs to compile enough to reproduce the problem. Do the turnin to project `debug`.

I try to respond to email in a reasonable time, but make no guarantees. I will also ignore messages that are general complaints or not specific enough for another person to provide any help.

In particular, you are expected to carefully read **every line** in the project assignment, the published C++ function and header files for the project, relevant text readings and lecture materials from the Web. A request for clarification of the assignment will be rejected or ignored if it doesn't cite the **page, paragraph, or line** where you think there is an ambiguity, error or need for clarification.

8. **Strongly Recommended!!** After any submission, run `turnin-csi422 -c csi422 -v -p projectname` and observe the report of what files were received. If the file(s) you meant to send are not listed, check and redo the instructions. See a TA if you cannot get a positive acknowledgment.