Chapter 1

CS488/688 F17 Assignment Format

"I take off marks for anything..."

– A CS488 TA

Assignments are due at the beginning of lecture on the due date specified. More precisely, all the files in your assignment directory *must* be dated before the beginning of lecture, and the documentation *must* be handed at the beginning of class.

The documentation for the assignment will be submitted in class. In addition, we will examine the files under your `cs488` subdirectory. The portion of the assignment that is physically handed in must be arranged in the assignment format documented below. Failure to follow any of the procedures specified here will result in deductions from your assignment mark.

There is a checklist available at the end of this section. It is *strongly* suggested that you make a few photocopies of this checklist, and step through it carefully while preparing your submission. However, you should *not* hand in your checklist with your submission.

Invalid Excuses for Late Assignments

The following excuses will **NOT** be accepted for late assignments:

"The computer was down."
: The computer often goes down shortly before assignments are due. Take this into consideration when working on your assignment.

"The print queue was too long."
: The print queue is *always* too long shortly before assignments are due. Print your documentation early.

"I ran out of money on my print account."
: It doesn't cost much to print the assignments. Just make sure you have enough money in your account.

Assignment Specification Format

Each assignment specification has the following sections:

Topics: A list of topics covered in the assignment

Statement: A statement of what you need to do for the assignment. This may be followed by additional sections that give more detail on particular topics.

Donated Code: This is code that we give you to get you started on the assignment and/or some routines that we supply for your use. Note that we will only list `.cpp` files here; however, there are often associated `.h` files.

Deliverables: This is a list of files that you must submit (in addition to what's listed in the next section) and details what you should be sure to include in your documentation, plus anything else you need to submit with an assignment.

Objectives: This is a list of objectives on which your assignment will be graded. Each objective has equal weight in your grade. You should submit a copy of this page with your written documentation. *Be sure to sign this sheet at the bottom!*

Do not check off objectives on the sheet you submit—the TA's fill in the blanks next to each objective when determining the mark for your assignment.

Directory Structure

The TA(s) will execute your program and look at the source code in your `cs488` subdirectory. Special group ownership and permissions are necessary for us to grade your assignment. We will change the permissions when we grade, so it is important that your directory is set up correctly by the `setup` programs in `/u/gr/cs488/bin`. If you register for the course late, it may take a while for MFCF to get your account established. Until your account is set up, we cannot grade your assignments.

Note that the `grsubmit` program sets file permissions to allow the TAs access to your files. It also generates a checksum. You must submit a hardcopy of this checksum (the output of `grsubmit`) with your documentation. No files are transmitted to us via `grsubmit`; you can run it multiple times with no problems. However, after running the `grsubmit` program for the final time **DO NOT MAKE CHANGES TO THE FILES OR DIRECTORIES, OR TO THEIR PERMISSIONS.**

Note to grad students: You will not have an account on the SGI workstations until you register, so it is important that you register for the course as soon as possible. Once you register, you will get a `cs688` directory in the *cs688* group. The `setup` command will create a symbolic link between `cs488` and `cs688`.

If you are not yet registered, you should first make the `cs688` directory yourself. Eg, type `mkdir ~/cs688` before running `setup`.

When you have finished working on an assignment, you should remove all unnecessary files (i.e., core files, temporary files, `.o` files, etc). Only the Python scripts and C++ source code, the dynamic libraries, and any input or output should remain in the directory.

For some assignments, the assignment specification (under the Deliverables section) will name certain files in which to place your code, etc. You must use these names no matter how much you may disapprove of them. In addition to what is listed in each assignment specification, the `cs488/handin/A`*n* directory should contain the following entries (and nothing else):

1. The executable(s) required for the assignment, *named as indicated in the assignment handout.* Regardless of how much you may disapprove of these names, you are expected to use them.

2. A subdirectory `handin/An/src`, which should contain all source files.

3. Your mainline Python script(s). Subroutines that are imported should be under the `src` or `data` subdirectories.

4. A file called `README`. This file should indicate on which machine each executable(s) was compiled and linked, should explain *briefly* how to run each program, should explain what assumptions you made (if any), *and should contain nothing else.*

5. A subdirectory `handin/An/data`, which should contain any data files, and subdirectories other than those specified in the assignment that were used to test your program. If you have gone to the trouble of creating some interesting data files of your own, make reference to them in the `README` file.

6. One or more files called `screenshotXX.png` that are screenshots of your program. You must have at least one screenshot file called `screenshot01.png`. You may have additional screenshots, which should be number consequetively following 01. For most assignments, you will likely only want a single screenshot. However, for the project, you may wish to have more screenshots. For the raytracer (assignment 4), your `screenshot01.png` file should be your best image. In general, you should take a screenshot that reasonbly illustrates that your program works.

The README file and all source files (e.g., `.cpp`, `.h`, and `.py` files) should contain your name, userid and student number near the top of each file.

DO NOT REMOVE OR MODIFY ANY OF THE ABOVE FILES UNTIL WE HAVE RETURNED YOUR GRADED DOCUMENTATION! The TA(s) will look at these files and check that the modification dates match those given by the checksum you submit with the documentation.

Documentation Submission

You will also be required to submit some documentation. This documentation should include the items listed in the following subsections. The documentation should be ordered in the same sequence as the following subsections, and all pages should be stapled together. **MARKS WILL BE DEDUCTED IF THE DOCUMENTATION IS NOT STAPLED TOGETHER.** Paperclips are **not** acceptable.

Title Page

The first page should be the title page. The title page should list your name, your userid, your student number, and the assignment's number. Leave blank space for the TAs' comments.

Objective List

Each assignment specification includes an **objective list** as the last page. You should include a copy of this sheet as the second page of the document you submit. The objective list indicates how the assignment is to be marked, and serves as a marking aid for the TA(s). Usually, there will be ten objectives worth one mark each. If you don't have time to complete the entire assignment, try to achieve as many of the objectives as you can in the time you have.

Be sure to fill in the information at the top of the Objective sheet, even though this information should also be on the title page, and make sure you read and sign the declaration at the bottom of the page. *No credit will be given for the assignment if you have not signed the Objectives sheet.*

Manual

This should NOT be a restatement of the assignment specification. You will be given strict procedures to follow on how to begin execution, what data is used and what interaction with the program is required. It is not necessary to reiterate this in great detail.

The manual should include any details or assumptions that you have made that were not specified in the specification. These could include additional commands, options or features or any additional data files that you may have generated. If you have made any assumptions about the assignment specification or the objectives, be sure you state and justify them.

You should also note in your manual any objectives that you did not complete. If you wrote code for objectives that you did not get completed, you should request code credit in your manual. See the section on code credit for details on what to put here.

Hardcopy

Hardcopy refers to any paper documentation requested, such as PostScript output or other diagrams.

Checksum

You will need to submit a hardcopy of the output of the program `/u/gr/cs488/bin/grsubmit`. To run this program, change directory to your `cs488/handin` directory and run `/u/gr/cs488/bin/grsubmit`. with the current assignment as an argument. For example, for Assignment 1 you would type

```
grsubmit A1
```

(assuming `/u/gr/cs488/bin` is on your path) and submit the resulting output. Again, **DO NOT MODIFY ANY OF YOUR FILES FOR THE ASSIGNMENT ONCE YOU HAVE RUN THIS PROGRAM!**

Program Execution

Each assignment handout specifies the names of the executable program(s) and scripts to be generated, and the names of supplied data files and scripts provided under `/u/gr/cs488`.

Follow the instructions in the assignment specification for naming conventions for data files and access. If the assignment specification says that the standard input `stdin` or a filename specified on the command line will be used, make sure to implement this functionality. This is because the TA's may use additional test scripts for marking.

Similarly, any textual output required should be sent to the standard output (`stdout`) or to a filename provided on the command line or though a graphical user interface, as specified. Output should not be sent to a file whose name has been hard coded into the executable.

Your final submission should NOT send debugging output to `stdout` or `stderr`. Debugging output should be turned off for the submitted versions.

README vs Manual

You will notice that you need to prepare both a `README` and a Manual for each assignment. In essence, the `README` tells us how to run your program, while the manual tells us what you did and why you chose to do it that way. Thus, the `README` should contain the following information:

- Which machines the executable(s) were compiled on;
- How to run your program, including the following information:
  - How to invoke your program.
  - How to use the parts of your user interface that are not specified in the assignment description.
  - Any assumptions about how the user will interact with your program that, if violated, might cause your program to fail.

Note that the `README` does not have to be handed in as hardcopy. We will read it online. You will find a skeleton README file in `/u/gr/cs488/data/README.skel`.

The manual should contain justification for your decisions and extra information about the implementation, including:

- Justification of any choice you make, whether in choosing your user interface, selecting a data structure, etc.
- Command-line options.
- Extra features or data files.
- Mention any objectives you did not complete, and (possibly) request code credit.

CS488 Assignment Checklist

Paper submission has:

[__] Title page with name, student ID number, course (CS488 or 688), assignment number, and userid. If there are multiple sections of the course in the semester, be sure to include the name of your professor and/or the section number as well.

[__] Second page is an objective list:

- Fill in the information at the top
- SIGN IT!
- DON'T check off objectives yourself

[__] Third page is a MANUAL:

- Describe anything unusual about your particular assignment.
- Ask for code credit.
- Indicate what doesn't work.
- Indicate any "interpretations" of spec.
- Don't restate the assignment specification.
- Be concise.

[__] Last page is a checksum printout from `grsubmit`.

[__] All of pages are STAPLED together in the top left.

Code has:

[__] Executable and mainline scripts are in:

```
~userid/cs488/handin/A?/*
```

[__] All filenames are as per spec.

[__] Running the executable or mainline script from this directory works and produces NO DEBUGGING OUTPUT

[__] README file indicating UI details, "how to run", is in:

```
~userid/cs488/handin/A?/README
```

[__] Screenshot file(s) of your program, is in:

```
~userid/cs488/handin/A?/screenshot01.png
```

(additional screenshots maybe provided if desired).

[__] C source code and Python scripts are in:

```
~userid/cs488/handin/A?/src
```

[__] Data files (i.e., scenes) are in:

```
~userid/cs488/handin/A?/data
```

[__] Running the executable conforms to objectives 1-10 unless otherwise indicated in the MANUAL. Always re-read the spec WHILE running the final program to make sure that all details are correct.

The most common mistake is to forget to sign the objective list. By checking this list before handing in everything, you can avoid losing marks for these nit-picky details.