

CS488/688

Winter 2012

Course Outline

University of Waterloo

Department of Computer Science

Instructor(s): Abdallah Rababah

January 12, 2012

Contents

1	CS488/688 W12	General Information	5
1.1	Course Description		6
1.2	Course Objectives		6
1.3	Important Dates		7
1.4	Important Skills		7
1.5	Text, References, and Documentation		7
1.6	General overview of topics to be covered		8
1.7	Programming Environment		9
1.8	Electronic Resources		9
1.9	Marking Algorithm		10
1.10	Assignment Evaluation		10
1.11	Rules for group work		11
1.12	Late and missed assignments		11
1.13	Submitting assignments and getting them back		11
1.14	Exams		11
1.15	FIPPA		11
1.16	Questions		12
1.17	University mandated information		12
2	CS488/688 W12	Evaluation	15
2.1	Instructions		15
2.2	Points to Cover		15
3	CS488/688 W12	Assignment Overview	17
3.1	Computing Environment		17
3.2	Password Changes and Customization		18
3.3	Tools		18
3.4	Accessing Multiple Machines		18
3.5	Lab Etiquette		18
3.6	The OpenGL Graphics API		19
3.7	Assignment Outlines		19
3.7.1	Assignment 0: An Introduction		20
3.7.2	Assignment 1: OpenGL		20

3.7.3	Assignment 2: Transformations and the Graphics Pipeline	20
3.7.4	Assignment 3: Hierarchical Modelling and OpenGL Rendering	20
3.7.5	Assignment 4: Raytracer	20
3.7.6	Assignment 5: The Project	20
3.8	Ambiguities or Omissions in the Assignments	21
3.9	Code Credit	21
3.10	Cheating	22
3.11	Previously Written Code, Use of Other People's Code	22
3.12	Feedback	22
3.13	Additional information	23

Chapter 1

CS488/688 W12

General Information

University of Waterloo

Term and Year of Offering: Winter 2012

Course Number and Title: CS488/688, Introduction to Computer Graphics

Instructor: Name, Office, Phone, email, Office Hours

Abdallah Rababah DC3519 x33602 arababah@uwaterloo.ca MWF 12:30-1:30

TA(s): Name, Office, email, Office Hours

Tiffany Inglis DC 2303 t3chao@cgl.uwaterloo.ca TBA

Alex Pytel DC 2303 apytel@cgl.uwaterloo.ca TBA

Grace Yao DC 2303 z7yao@uwaterloo.ca TBA

Course Newsgroup: uw.cs.cs488

Course Email: cs488@cgl.uwaterloo.ca

Course URL: <http://www.student.cs.uwaterloo.ca/~cs488/>

Lecture Times: 1:30-2:20 MWF

Lecture Room: MC4060

Laboratory, Phone: MC3007 x36560

Programming Environment: C++, GNU/Linux, OpenGL, GTK+, Lua

Required Texts:

CS488/688 Course Notes and Overview (which you are reading).

Recommended Texts:

Hearn, Baker and Carithers, *Computer Graphics with Open GL*, Prentice Hall.

References:

Jackie Neider, Tom Davis, Mason Woo, OpenGL Architecture Review Board, *OpenGL Programming Guide*, Addison Wesley (and Silicon Graphics).

1.1 Course Description

Software and hardware for interactive computer graphics. Implementation of device drivers, 3-D transformations, clipping, perspective, and input routines. Data structures, hidden surface removal, colour shading techniques, and some additional topics will be covered.

1.2 Course Objectives

At the end of the course you should be able to:

- write interactive 3D computer graphics programs;
- understand how linear and perspective transformations are used in modeling and rendering in 3D computer graphics;
- understand the process of rendering, lighting, hidden surface removal, and other computer graphics techniques;
- write a simple ray tracer.

1.3 Important Dates

- Assignment 0 due: Monday, January 9th [Week 2]
- Assignment 1 due: Wednesday, January 18th [Week 3]
- Assignment 2 due: Wednesday, February 1st [Week 5]
- Assignment 3 due: Wednesday, February 15th [Week 7]
- **Midterm Exam: Wednesday, February 29th [Week 8]**
- Assignment 4 due: Friday, March 9th [Week 9]
- Assignment 5, First Goal due: Monday, March 5th [Week 9]
- Assignment 5, Second Goal due: Wednesday, March 14th [Week 10]
- Assignment 5, Third Goal due: Monday, April 2nd [Week 13]
- Demonstrations: TBA [Week 13]
- **Final Exam: TBA**

1.4 Important Skills

- Reading.
- Read the newsgroup.
- Read the course notes.
- Last but not least, read and follow instructions.

1.5 Text, References, and Documentation

Our lecture notes are fairly extensive. Even though copies of these notes are included with the course notes (which you are required to purchase), you are still required to purchase the course text.

In previous terms the text book has been either *Computer Graphics* by Foley, van Dam, Feiner, and Hughes; a “textbook” version of this text, *Introduction to Computer Graphics*, by Foley, van Dam, Feiner, Hughes, and Phillips; or *Computer Graphics* by Hearn and Baker. The first of these is a thick tome with a white cover, the second has a red cover. The last has either a white or a grey cover, depending on the edition. Any of these text books would be sufficient for the course. Check the used book store. If you plan to work in the computer graphics field, we would strongly recommend investing in a current edition of the first text, *Computer Graphics* by Foley, van Dam, Feiner, and Hughes—although it is a bit encyclopedic for the purposes of this course.

One additional reference book is recommended. The *OpenGL Programming Guide* documents the 3D graphics programming interface you will be using. Copies of this reference will be available in the lab **AND SHOULD NOT BE REMOVED**.

If you buy both this book, it will be much easier to do the assignments, but obviously this could be expensive.

A great deal online documentation for GTK+ and Lua is also available through the course web page, and from the web.

We are also recommending Jim Blinn's book. This book is a collection of his columns from CG&A. Most of what he discusses is at a lower level than we are concerned with. However, there are a few good project ideas in the book, and he gives all the dirt on many low level graphics operations. If this sort of thing interests you, consider buying the book. However, you are not expected to know any material from this book (unless it is covered elsewhere or in lecture).

1.6 General overview of topics to be covered

- **The Graphics Environment** (4 hours)

Overview of a representative processing sequence that connects application programs with the images they display on screen. Outline of the graphics library to be used and the hardware of the graphics workstation.

- **Mathematical Underpinnings** (4 hours)

A review of concepts and tools: points, vectors, lines, planes, matrices, dot and cross products, vector space, affine space, projective space, etc.

- **Transformations** (4 hours)

2- and 3-dimensional translation, rotation, and scaling as matrix operations. Homogeneous coordinates. Clipping, windowing, and viewing with perspective.

- **Interrupting, Picking, Polling, Callbacks** (3 hours)

The management of picking, selecting, and control tasks through the use of event queues, interrupts, and device polling. Windowing systems and user interface toolkits.

- **Hidden Surfaces and Shading** (4 hours)

Standard lighting models and their implementation. Hidden-surface elimination using depth buffering, scanline coherence, and subdivision. Polygon filling.

- **Ray Tracing** (4 hours)

Basic ray tracing techniques for generating shadows, mirror reflections, and refraction. Constructive solid geometry models.

- **Physically Based Rendering** (4 hours)

Radiosity, bi-directional path tracing, global illumination.

- **Discretionary Topics** (5 hours)

Chosen at the discretion of the instructor. Possibilities include: more depth on any of the foregoing, as well as human vision, colour theory, anti-aliasing, database amplification, animation, scientific visualization, graphics hardware support, higher-order curves and surfaces, and dynamic simulation.

1.7 Programming Environment

Programming assignments for this course will be implemented in C++ under the UNIX operating system, using the X window system and OpenGL. The completion of a significant independent project is required (see Assignment 5). Some of the assignments make use of the Lua scripting language for parsing scene descriptions.

Some handouts relating to UNIX in the MFCF context, and MFCF procedures, can be obtained from the MFCF consultants in the MC Building. A collection of UNIX documentation may be purchased at IST/CHiP on the first floor of the MC. Standard references for the C programming language can be purchased at the UW Bookstore. However, it is *strongly* recommended that students taking this course have prior experience in C.

OpenGL is a standard graphics applications programming interface (API) that was developed as an extension and standardization of Silicon Graphics Incorporated's (SGI's) proprietary IRIS GL API. Lua is an interpreted scripting language intended to be embedded in other programs (in our case C++ programs). It is particularly widespread in games but used for many other tasks. GTK+ is a Graphical User Interface toolkit, which provides many useful widgets. `gtkmm` is a C++ wrapper around GTK+, which by itself is written in C.

The *OpenGL Programming Guide* includes little information on how to program in OpenGL under X with all but the most trivial of interfaces. This was intentional, since OpenGL is supposed to be window system independent. However, it makes the kind of "small-scale" programming we do in assignments a little difficult.

We have worked around this problem by, for the most part, hiding¹ window system details under GTK. Since GTK has been officially ported to the Mac and Windows environments, your programs could *in theory* be run on any machine with OpenGL. Note, however, that we only explicitly support UNIX/X11 machines in this course, and your programs *must* be tested on the machines in MC3007.

1.8 Electronic Resources

The course URL can be used with to access the course web pages. These web pages have a large amount of information, including but not limited to this overview, the lecture notes and overheads, a glossary, sample exam questions, additional reference material, pointers to other Internet sources, and specifications for all the assignments. Much of this material is in hypertext.

Subscribe to the course newsgroup `uw.cs.cs488` using the command `rn` or your favourite news browser. Read the comments in your `.cshrc` file, and include your news browser in the `.cshrc` file

¹ For completeness, we have included additional documentation on using OpenGL under X as PostScript documents accessed through the course web pages. There are also a number of sample programs available online.

in a place consistent with those comments. **It is your responsibility to read the course news on a regular basis.** You should definitely read the news before major events such as exams and assignment due dates. Changes to the material in the course web will be posted to the newsgroup, but the newsgroup is also intended as a public forum for discussion of the assignments. Feel free to post to the newsgroup using the command `postnews` or your news reader. The TAs and instructor will monitor the newsgroup and answer any relevant technical or policy questions. You should check the Frequently Asked Questions (FAQ) web page before posting a question which you suspect may have been asked (and answered!) before.

Material on reserve will be listed under “CS 488”; the instructor is permanently listed as “Stephen Mann” regardless of who is really teaching the course.

1.9 Marking Algorithm

The final mark for the term will be computed according to the following weighting:

- Assignments: 36%
- Proposal: 4%
- Project: 20%
- Midterm Test: 10%
- Final Exam: 30%

You must average at least a 50% in *both* the programming portion of the course and in the examination portion of the course to pass the class. If you average less than 50% in either category, you will receive a course grade no higher than 45%. The instructor reserves the right, where appropriate, to adjust raw marks downward in the case of cheating and upward in other situations.

1.10 Assignment Evaluation

Each assignment will be accompanied by a list of objectives. The TA(s) will grant one mark for each objective met satisfactorily. The TA(s) may grant one bonus mark per assignment for such things as additional programming features, innovative approaches, etc. The TA(s) may also deduct marks for poor user interfaces, poor documentation, or other problems.

Assignments are due at the *beginning* of the lecture on the date specified. More precisely, the portion submitted electronically *must* be dated before the beginning of the lecture, and the paper documentation *must* be given to the instructor at the *start* of the lecture period. Assignments will be returned at the end of the following lecture. **CREDIT WILL NOT BE GIVEN FOR LATE ASSIGNMENTS.**

If, on receiving a marked assignment, you have complaints or requests for clarification, these must be made to the TA(s) within one week of the return of the marked assignments to you. Beyond this time, the mark will not be altered.

Graduate students will be expected to prepare a short technical report, with bibliography, for their course project, in addition to the items requested in the project statement.

1.11 Rules for group work

No group work is allowed; all work on the assignments and project must be done by each individual student. However, students are allowed to discuss the ideas and concepts that are to be implemented.

1.12 Late and missed assignments

No late assignments will be accepted. Any assignment missed or submitted late will be assigned a mark of 0.

1.13 Submitting assignments and getting them back

Assignments should be submitted in class during the first five minutes of the lecture on the day they are due. Assignments will be returned in class after they have been marked.

1.14 Exams

The midterm test will be held on the date listed under the heading **Important Dates**, during class time, in the class lecture room. It will cover material pertinent to all lectures given, and all assignments due, before the midterm date. The final exam will be scheduled by the Registrar. It will cover material from the entire course.

Questions from past midterm tests and final exams will be made available on reserve and on the course web page. These questions serve the faculty as suggestive material for making up new exam questions, and it is strongly suggested that you work through the pertinent ones to prepare for exams. All exams will be closed book.

For the midterm, all complaints or requests for clarification must be made to the instructor within one week of the day that the exams are returned. Beyond this time, the mark will not be altered.

1.15 FIPPA

The Freedom of Information and Privacy Protection Act says that a student's name can not be connected with anything else related to that student without their consent. In the course, you will complete assignments (A3,A4,project) in which you may have pictures that we want to put in the course web page gallery. We will implicitly assume consent to put these material on the course web page with your name associated with them. This will also identify the term in which you took the course, and the image will likely have a file name based on your uw email address.

If you do NOT want your name to appear in the web page, you should let us know and we will not put images from your assignments in the gallery. If at any time in the future, you want us to remove an image that has your name on it (as well as your name etc.), just let us know and we will remove it.

1.16 Questions

All questions about assignments, due dates, or anything else related to the course should be directed to the instructor during class, to the instructor and TA(s) during their office hours, to the course newsgroup, or to cs488@cgl.uwaterloo.ca via electronic mail. Office hours will be posted to uw.cs.cs488 at the start of term and will be recorded in the course web pages.

The TA(s) will be available during posted office hours to handle questions. Please *do not* bother the TA(s) at other times, in person, on the phone, or by direct e-mail, except in dire emergencies—TA(s) are students, too, and have other commitments in addition to CS488/688!

Note: confidential questions (about your marks, etc.), should be sent from your University of Waterloo account. Regardless, we will send confidential information such as grades **only** to your account on student.math. You should make a habit of reading your mail on that account or setting up a forward file from that account to where ever you read your email.

1.17 University mandated information

Academic Integrity: In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility. [Check www.uwaterloo.ca/academicintegrity/ for more information.]

Grievance: A student who believes that a decision affecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. Read Policy 70, Student Petitions and Grievances, Section 4,

www.adm.uwaterloo.ca/infosec/Policies/policy70.htm.

When in doubt please be certain to contact the department's administrative assistant who will provide further assistance.

Discipline: A student is expected to know what constitutes academic integrity [check

www.uwaterloo.ca/academicintegrity/

] to avoid committing an academic offence, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offence, or who needs help in learning how to avoid offences (e.g., plagiarism, cheating) or about 'rules' for group work/collaboration should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean. For information on categories of offences and types of penalties, students should refer to Policy 71, Student Discipline,

www.adm.uwaterloo.ca/infosec/Policies/policy71.htm.

For typical penalties check Guidelines for the Assessment of Penalties,

www.adm.uwaterloo.ca/infosec/guidelines/penaltyguidelines.htm.

Appeals: A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances) (other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals)

www.adm.uwaterloo.ca/infosec/Policies/policy72.htm.

Note for Students with Disabilities: The Office for persons with Disabilities (OPD), located in Needles Hall, Room 1132, collaborates with all academic departments to arrange appropriate accommodations for students with disabilities without compromising the academic integrity of the curriculum. If you require academic accommodations to lessen the impact of your disability, please register with the OPD at the beginning of each academic term.

Chapter 2

CS488/688 W12

Evaluation

2.1 Instructions

The faculty who teach CS488/688 are interested in your opinions concerning the course. Please write your comments on a *separate* piece of paper, to be turned in at the final examination, with an *unsigned* short evaluation which addresses the points listed below. Don't feel you have to answer each question individually, but please cover all of the items. Do feel free to add any other constructive comments that you have concerning the course.

These evaluations will not be looked at until after the grading is complete, even though they are unsigned.

2.2 Points to Cover

- What do you think of using Python/Tkinter for the interface? Was it confusing to do half the assignment in Python and the other half in C?
- What was your opinion of: the notes; the recommended textbooks; the reference texts; the course URL and other online material (How useful was it? What could be added? How could it be better organized?); the lectures (Have you any stylistic suggestions? Organizational suggestions?); and the assignments (Too difficult? Too easy?).
- What was your opinion of the teaching assistant(s)? Were assignments carefully marked? Did you visit the teaching assistant(s) for help, and if so, how productive were your visits?
- How do you think CS488/688 could be improved? Were there any topics covered that you think should be omitted? Were there any topics not covered that you think should have been included?
- If you are interested in taking the advanced graphics course (CS 788), what topics would you like to see covered there?
- What was your overall opinion of CS488/688?

Chapter 3

CS488/688 W12

Assignment Overview

No food or drink is allowed in the lab.

There will be four programming assignments and a programming project for this course, which will be spread over a thirteen week term. These programs are worth a substantial portion of the final mark.

There will be a preliminary assignment that will be marked, but this mark will not be used in computing your final grade. This assignment is optional, but it gives a “no fault” trial to become familiar with the computing environment and the organisation of handing in and marking assignments. **We strongly suggest you do this assignment.** The TA(s) will assume that you have this familiarity for the regular assignments and the project, and they will make **no allowances** for lack of familiarity.

3.1 Computing Environment

No food or drink is allowed in the lab.

There is a special lab for CS 488/688 in MC3007. There are several Linux PCs in this room. Some of this equipment is extremely expensive. **Therefore, there is no food or drink allowed in the lab.** Anyone caught eating or drinking in the lab will be penalized by loss of marks on the next assignment (or final project!) from 1 mark to all marks at the instructor’s discretion.

The computing environment will be under the UNIX operating system. The programming languages will be C++, and for some of the assignments, Lua.

You will be using C++ to build most of your user interface using the `gtkmm` wrapper around the C-based GTK+ library. You will also be extending the Lua language to interpret 3D scene description files by adding extra functionality in C++.

The Operating System will be Debian GNU/Linux on the PCs.

All assignments must be submitted on the Linux PCs, and must run on one of these machines in MC 3007.

Graphics output will be through the PNG file format, and the OpenGL application programming interface (API).

You may obtain general UNIX and C++ help from the CSCF Consultants and specific help

on the course from the TA(s), the instructor, and the newsgroup. There is also a great deal of information available through the course web, including a list of frequently asked questions.

Assignments will only be accepted if they execute on one of the Linux PCs.

3.2 Password Changes and Customization

There is one password file shared by all the machines. One of the machines is the master password keeper. Any time you change your password, the master is updated, and it will spread the change to the other machines, but not immediately.

3.3 Tools

You are required to review GNU `make`, and use it.

3.4 Accessing Multiple Machines

We recommend setting up an appropriate `ssh` public key to allow access between all the machines easily. If you have not yet made an `ssh` key, you can do the following (on any of the lab machines):

```
$ ssh-keygen -t rsa # (accept the defaults)
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

This will allow you to `ssh` from any lab machine to any other lab machine without a password (unless you supplied one when generating the `ssh` key). This will also have the same effect for all of the other CSCF undergraduate hosts.

3.5 Lab Etiquette

“Can we make people take a shower before they come in the lab?”

– An olfactorily sensitive student.

Near due dates, the lab will be busy. **You should NEVER xlock one of the lab machines to reserve it for yourself while you step out of the lab.** If the instructor finds a student in the class has xlocked one of the machines, the student may be given a mark of 0 on the next assignment that is due (or a 0 on the final project if that is the only thing left to handin).

If all the lab machines are in use, you may be able to run your program from a different machine using `ssh`. To do this, connect to the machine you wish to use (first asking the person sitting at that machine for permission using `ssh -X glXX`, where `glXX` is the machine you wish to use remotely. You should then be able to run your application on that machine. OpenGL may or may not work depending on the environment (and will probably be slow regardless). This is also possible from home.

At most two courses use the Graphics Laboratory and the machines in MC 3007 in any term. You will be made aware of which courses these are. You are encouraged to recognize legitimate

students. If you see someone in the lab whom you do not know, please introduce yourself, state which course you are in, and ask them to do the same, particularly if they are doing something that is unfamiliar in the context of the course work you are doing. We have had trouble with “crashers” in the past, students who cracked the lab combination. They frequently came into the lab to play with the machines and have been annoying and troublesome. Even worse, we have had machines stolen, so if we seem paranoid, it’s because we have reason! If it becomes clear that someone has no business being in the lab, please request that he/she leave. If the request is refused or ignored, you should call UW Security to have him/her removed.

3.6 The OpenGL Graphics API

Some assignments will require *real-time, interactive* 3D graphical output. Programs may therefore make use of the OpenGL graphics library, an Application Programming Interface (API) that provides an interface to hardware accelerated graphics. Writing a program that uses OpenGL will involve including the file `GL/gl.h`, `GL/glu.h` and sometimes `GL/glx.h`. A typical OpenGL source file will look something like this:

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glx.h>
ogl_doNeatStuff()
{
    /* wildly wonderful OpenGL code here */
}
main (...) {...}
```

GLX is a set of utilities for interacting with X Windows that provides some important functionality not found in OpenGL itself. OpenGL (unlike the SGI’s older GL), does not provide built-in window support. One must use X or an X toolkit. The GTK+ toolkit (using the `gtkmm` wrapper) will be used for the assignments. You are welcome to use this or any other installed toolkit for the project, but for most projects a simple library such as GLUT or SDL will be sufficient.

You will rarely need to write entire programs from scratch. In particular, we will be giving you a bunch of code that will take care of most of the grungy interface hacking. This is in the interests of time only, so we can give you more interesting graphics assignments.

The *OpenGL Programming Guide* has been specified as a reference text. We will be providing some instruction in class, but you are expected to learn the details of OpenGL on your own time. The lectures will only direct you to the sections you will need to know to complete the assignment. The course TA(s) are the primary consultants on OpenGL.

3.7 Assignment Outlines

In this section we will give brief descriptions of each of the assignments.

3.7.1 Assignment 0: An Introduction

This assignment is primarily designed to introduce you to the computing environment and make sure you understand how assignments are to be submitted. You will be modifying the user interface of a simple graphics program. This assignment will be critiqued but will not count in your final grade. **AGAIN, WE STRONGLY SUGGEST YOU DO THIS ASSIGNMENT.** In the past, students who skipped this assignment regretted it.

3.7.2 Assignment 1: OpenGL

This assignment will introduce you to 3D graphics programming under OpenGL and to the GTK+/`gtkmm` toolkit in C++.

3.7.3 Assignment 2: Transformations and the Graphics Pipeline

You will build a transformation pipeline to visualize the difference between model, world, and view space. Your pipeline will only support wireframe rendering of a cube and a few gnomons, but will have to directly support perspective and 3D line clipping. You will use `gtkmm` to build a graphical user interface to this program; your implementation effort will consist of the GTK+ interface, and C++ code to implement a graphics pipeline using only line drawing routines that will be provided to you.

3.7.4 Assignment 3: Hierarchical Modelling and OpenGL Rendering

You will build a hierarchical modelling and rendering system on top of OpenGL. The program you will create will parse a description of a hierarchical model and render it, and then will allow 3D picking and interactive manipulation. Your renderer will support Phong lighting models and Gouraud shading.

The hierarchical model will be described in a scene description “language” defined as an extension to Lua. This extension will be reused and extended in the next assignment.

3.7.5 Assignment 4: Raytracer

Using the hierarchical modeller you defined in the last assignment, you will build a raytracer. A raytracer is a rendering algorithm which can create extremely realistic images, but is not suited for real time implementation. Your raytracer will support spheres, cubes, meshes of planar convex polygons, and shadows. You will also be required to implement one additional feature of your own choice. This feature can be an acceleration technique, a new model type or construction technique, or an additional visual effect.

3.7.6 Assignment 5: The Project

This assignment will be a project of your own devising. A proposal is required, which we will critique and return to you. The project is worth a substantial part of your mark, and should have a significant 3D graphics component.

3.8 Ambiguities or Omissions in the Assignments

No assignment specifies completely what is to be done; some things are left to your discretion. Wherever something is ambiguous or omitted, you are to use your own judgment. The essential thing is to

IDENTIFY AND DESCRIBE WHAT YOU ARE DOING AND WHY.

Document your decisions and especially any added features you implement if you want them to be recognized.

If you think something is broken or brain-damaged, it may well be. Let us know **WELL BEFORE** the assignment is due and we may be able to fix it or modify the specification and benefit everyone in the class.

3.9 Code Credit

Code that is not efficient, well written, well structured, and/or well documented may result in the **deduction** of points. On the other hand, it is possible for non-functioning code to be given credit. Credit for code that does not work **must be requested in writing** in the manual that accompanies your handin; TA(s) will not grant code credit automatically.

What you are actually doing when requesting code credit is stating which objectives do not work properly and asking for the TA(s) to examine code to give you up to 1/2 credit for each non-functioning objective. For each missed objective, your request must state the following explicitly:

1. What code files do not work, and the relevant line numbers.
2. What execution features of the code in these files do not work (i.e. describe the current behaviour).
3. What you think the probable cause of the problem is.
4. What steps you have undertaken and will/would continue to undertake to track the problem down and correct it.

If you fail to provide this information, the TA may not give you credit for your code.

The code in question must be stylistically good, well organised, heavily commented, and clearly documented. You must make it easy for the TA to understand what you are doing in your code. If the TA cannot understand your code, you cannot expect them to give you any credit for it.

3.10 Cheating

The programming assignments are designed to give practice on the topics presented in lectures. It is important that you learn all you can by doing them.

One activity that promotes learning is discussing the assignments, in class, with the instructor, with the TA(s), and with others in the class. These discussions are to be **PRELIMINARY** in nature, exploring the possible tools, techniques, and issues involved in the solution. The discussions can take place in class, on the newsgroup, and in small groups of other class members. Feel free to participate.

The second activity that promotes learning is that each student, **individually**, works out a solution and composes **individual code** to execute the assignment. You will discover that each assignment is to be handed in with an “Objectives” sheet that contains the following statement:

Declaration:

I have read the statements regarding cheating in the CS488/688 course handouts. I affirm with my signature that I have worked out my own solution to this assignment, and the code I am handing in is my own.

Signature:

Only assignments with the student’s signature affixed to this declaration will be graded. If anyone is discovered not to have abided by their declaration, severe disciplinary action will be taken.

The minimum penalty for cheating of any kind on an assignment is a mark of -100% on that assignment, and a letter of notification to the Dean, who will enter a letter of reprimand into your file. Further action by the Mathematics Faculty Disciplinary Committee may also be undertaken; note that the penalties are severe, and can easily include expulsion.

If you have any questions about what may or may not constitute cheating, please ask the instructor so you don’t end up cheating “by mistake”.

3.11 Previously Written Code, Use of Other People’s Code

The code you submit should be code you wrote. For the project, it may make sense for you to use code written by someone else to assist you in your work. Further, you may wish to reuse code you wrote in previous terms. In both cases, you should check with the instructor before hand, and credit the code (even if it was code you wrote in previous terms) in your documentation for the assignment/project. However, you do not need to give credit for code that we provide for you. See the discussion of projects for more information on using code you previously wrote as part of your project.

3.12 Feedback

The course was offered using SGI Indigo workstations for several years. The Indigo’s were replaced with SGI Octanes, which are much faster machines. X workstations were added a few years ago. Eventually the whole lab became full of Linux PCs. Furthermore, the course has evolved from

using C with Tcl/Tk, to C++ with Python/Tk and most recently C++ with GTK+ and Lua. The programming environment has thus been modified each term. It's still not perfect, and the constant modification gives entropy a chance to work. *Please* let us know as soon as possible if you spot any inconsistencies in the documentation or the provided code.

In addition, the instructor and TA(s) will strongly welcome any *constructive* criticism or suggestions you may have for making the programming environment better. We will be asking you to fill in an anonymous evaluation form for each assignment, whose contents help us to improve the assignments from term to term.

3.13 Additional information

- Make sure you don't change the specified default behaviour if your program has additional features. Let your extra features be optional and mention them in the README file.
- You can lose marks for the things that are not explicitly stated in the assignment objectives. Submission of poorly written code has already been mentioned, for example. Failure to put the executable(s) in the proper place is another example, and it will also result in deduction from the assignment mark. You can also lose a mark for submission of a program with a poor user interface.