24-780 Sec B – Engineering Computation

Fall 2022

Course Syllabus

Mon/Wed 4:40-6:40pm

1. Objectives

The practice of engineering in the 21st century relies heavily on the use of computing software. From preliminary design using "back-of-the-envelope" calculations, through modeling, analysis and prototyping, into the production of design documentation, all stages of the engineer's work are supported by software. Unfortunately, many engineers have only a marginal acquaintance with the programming required to create the software packages they depend on to solve the technological challenges they must overcome. This course seeks to provide engineers with the knowledge of how software is developed and with the skills to develop software solutions that can be brought to bear on the unique problems they will encounter.

The skill of computer programming is part knowledge, part ingenuity, part sweat, and part art. With your technical background, you are perfectly positioned to be a good programmer if you rely on your problem-solving ability and judiciously apply your organizational skills. Your experience in design will also serve you well as you proceed through the software implementation process.

In this course, you will expand on your ability to apply computing technology to engineering practice by:

- Identifying what parts of an engineered solution can benefit from a programming approach;
- Creating data models that best fit the engineering domain being implemented;
- Breaking down a large problem into workable parts and then integrating an overall solution;
- Effectively communicating design intent through documentation and programming style;
- Becoming adept at using online sources to enhance your coding prowess (life-long learning); and
- Visualize results using computer graphics (OpenGL).

Prerequisites: none

2. Instructors

Nestor Gomez, Ph.D., P.E.
nestor@cmu.edu
Wean Hall 4303 (412) 944-6515
https://cmu.zoom.us/my/nestorgomez
Office hours: Mon-Thurs, 10:30am-2:30pm
Or any other reasonable hours except
MW 3:00-7:00pm, F 12:00-3:30pm

T.A.: Yijie Chen
yijieche@andrew.cmu.edu

T.A: Tanmay Chinchanikar
tchincha@andrew.cmu.edu

3. Textbook and References

The recommended (**not required**) textbooks for the course are listed here, with my full knowledge that you will likely *not* use them.

Stroustrup, Bjarne. *The C++ Programming Language*.

This is the classic reference book, written by the creator of the C++ programming language.

Shreiner, Dave and Mason Woo, Jackie Neider, and Tom Davis. *OpenGL programming Guide: The Official Guide to Learning OpenGL* Version 2, 5th Edition

The official reference for OpenGL may be useful to have on hand.

In addition, it is likely that you may also want/need to use manuals and/or instruction guides for programming, especially when you forget a small detail or need a little push. Here is a list of recommended sources (printed and electronic):

http://www.google.com -> most questions are answered in first 5 hits.

http://www.codeguru.com -> more directed to Microsoft Visual C++, but good all-around too.

http://www.stackoverflow.com -> a discussion group where senior programmers answer questions.

https://www.khronos.org/registry/OpenGL/index_gl.php -> OpenGL documentation, a bit raw.

4. Attendance

You are expected to participate in every lecture session. Although there will be no "roll call" and no grade will be assigned to attendance, missing lectures will surely reflect on your performance in problem sets and projects.

Class participation will be an integral part of the learning experience so *get your money's worth; come to lecture.*

5. Grade Breakdown

Problem Sets 11 assignments, 5% each	**55%**
Individual Project (Demo)	**15%**
Team Project	**30%**
Total	**100%**

6. Problem Sets

Learning to program does *not* need to be intellectually problematic and emotionally torturous, as is often believed and stereotyped. The analogy of learning to code with learning a new language is, I believe, erroneous in that modern computer languages can be used to reflect the human way of thinking, rather than having humans adapt to the computer's limitations.

Since hands-on experience is an absolute must to truly internalize the use of a programming language and apply it to the design process, all of your graded effort in the course will consist of problem sets and projects. There will be 11 assignments in the course, mostly requiring the creation of code. Towards the end of the course, you will work in small teams to develop your original software creations.

There will be a problem set due essentially every week and each should be turned in on time through the Canvas. Problem sets are typically due at 11:59pm on Tuesdays. As a deterrent from untimely submittal, there will be a 30% decrease in the homework's score for every day (including weekends and holidays) that the homework is late (e.g., 4 hours late = 23 hours late = 30% off, 25 hours late = 47 hours late = 60% off, 49 hours late = 90% off, so why turn it in?). Extenuating circumstances will excuse at least part of the deduction, but these should be discussed ahead of time.

I *fervently* advice that you start working on the assignment as soon as it is posted, rather than waiting to the last minute. Carefully re-read the previous sentence. I am not advising that you finish the assignment early since I am realistic about how student life actually works. What I am suggesting is that you START early, even if you cannot schedule/prioritize enough time to finish until closer to the deadline. 30 minutes of work on a Thursday afternoon can save you three hours of frustration on a Tuesday night.

7. Individual Project

In the individual project, you will create a 1-minute long demo program to be anonymously presented in lecture. A demo program is a non-interactive presentation of graphical and audio programming techniques. See Demoscene (http://www.demoscene.info/the-demoscene) for more details. We'll discuss the demo project in more detail later in the semester but rest assured that this is a fun/creative activity.

8. Team Project

In software development, collaborative coding is the norm. Thus, the primary focus of the team project is to experience writing a program in a team environment. The project will require your team (3 to 5 members) to develop a program in an engineering field (with perhaps some leanings towards education or entertainment, but don't forget who you are). Commercial-grade software is absolutely NOT expected.

9. In-class Exercises

Since we cannot cover every topic in the problem sets and we do not want to miss out on the hands-on experience for learning programming skills, we will be performing in-lecture exercises sporadically throughout the semester. You should join each lecture ready to program (i.e., smart phone will not be enough) so you can take full advantage of these exercises (although not all exercises require actual coding).

10. Examinations

None.

11. Individual Work

Unless specially noted (team project), assignments you submit in this course must be your own individual work. Note how this statement, although quite discouraging of it, does not necessarily rule out working in groups from time to time since even in a group's collaborative setting, your own work can come through. However, I strongly suggest that you at least attempt the assignments individually and resort to groups only when you get stuck and you cannot get hold of me or the TAs.

Some other warnings about working in groups:

- Don't let yourself or anyone else "be carried". No one is doing anybody any favors that way. A person who asks you for "a quick look just this once" has been asking for the same thing since 2nd grade.
- Limit the group's size to 3 members. Larger groups have the tendency to lose their focus and make it easier for the clueless to get by unnoticed.
- Use groups **only** in a cooperative mode, never in a "divide and conquer" mode. In other words, be sure that everyone in the group works out every part for the problem instead of splitting it up among the group members and then synthesizing an answer.

Now for a note on cheating as it refers to programming courses specifically. We all know that there is a lot of sample code on the internet. The best advice is that you only use code that you truly understand and never just blindly copy/paste into your solutions. Remember that you owe it to yourself to actually learn something from this course.

CMU maintains an honesty policy regarding cheating. Specific policies can be found at:
http://www.cmu.edu/policies/student-and-student-life/academic-integrity.html

12. Work Quantity

It is not my intention to have you spend more time on problem sets than is necessary for the learning process. As a 12-unit course with 4 hours of lecture time, you should be spending about 8 hours a week of your own time working on problem sets (a bit more or a bit less based on experience, errors, time management, perfectionism, etc.). If you find that you are devoting an excessive amount of time on this course, please let me know as soon as possible. There is a good chance that you have misunderstood the requirements rather than the subject matter. We are here to learn, not to test your endurance or sleep-deprivation capabilities.

As a graduate student, you are likely entering the phase of life in which you establish the work-life balance that you'll keep for most of your career. Be sure that the choices you make about how to achieve your academic goals do not adversely impact your ability to keep your body and mind healthy.

13. Students with Disabilities

Students requesting classroom accommodation must first register with the Dean of Students Office. The Dean of Students Office will provide documentation to the student who must then provide this documentation to the Instructor when requesting accommodation.

14. Mutual Respect

Each of us is responsible for creating a safer, more inclusive environment, with respect for diversity in all its forms: race, ethnicity, nationality, gender, creed, disability, socio-economic status, and everything else that we can and cannot yet think of. In addition to seeking fairness and justice, maintaining a strong pro-diversity, pro-equity, pro-inclusion stance allows us to tap into our differing backgrounds and perspectives to create and innovate in unique ways as we solve the problems that challenge all of humanity. We recognize that we are in this together and that our future is forged by our combined tolerance and open-mindedness.

15. Use of Mobile Devices and Laptops

All mobile devices should be turned off during class. You may follow the course material or take notes on your laptop, but you must turn the sound off so that you do not disrupt other students' learning. I should mention, however, that research has shown that handwritten notes lead to better comprehension of class material (http://www.npr.org/2016/04/17/474525392/attention-students-put-your-laptops-away).

16. *Approximate* Course Schedule

Lect	Date (2022)	Topic/Event	Problem Sets
1	Mon, Aug 29	Intro to C++ Language. Steps to write a computer program. Using a C++ compiler.Console Input and Output. Basic Data Types and Variables.	PS1 out
2	Wed, Aug 31	Functions, Arrays, Loops, Type Casting. Switch and case. High-Low game. Writing without dialects	
	Mon, Sep 05	**Labor Day Holiday (no classes)**	
3	Wed, Sep 07	Range-based for. Shuffling. Math library. Binary numbers	PS2 out
4	Mon, Sep 12	Macros. C-style Text String. Intro to State Transition (Word Counting). Bubble Sort. Merging Two Pre-Sorted Arrays (Pre-requisite for merge sort)	
5	Wed, Sep 14	Compiling multiple source files. Intro to OpenGL. Animation. Modern graphics hardware. Computational Cost (Big-O Notation)	PS3 out
6	Mon, Sep 19	Interactive Program (A shooting game). Euler's Method. Visual effects in OpenGL	
7	Wed, Sep 21	Intro to Pointers. Parsing. File Input / Output	PS4 out
8	Mon, Sep 26	Text-drawing functions. Dealing with Window Resizing. Sub-second timer. Mouse event and mouse state	
9	Wed, Sep 28	Introduction to audio programming.	PS5 out.
10	Mon, Oct 03	Intro to Object-Oriented programming. Difference between C and C++. House cleaning using C++ classes. Writing a C++ program in a team.	
11	Wed, Oct 05	More about member functions. Splitting the shooting-game into multiple .cpp / .h files	PS6 out
12	Mon, Oct 10	Constructor and destructor. Protected members. Dynamic memory allocation.	
13	Wed, Oct 12	Bitmap. Texture mapping. 'const' qualifier. Constructor with parameters.	PS7 out
	Mon, Oct 17	**Fall Break (no classes)**	
	Wed, Oct 19	**Fall Break (no classes)**	

Lect	Date (2022)	Topic/Event	Problem Sets
14	Mon, Oct 24	TextInput class. Image Processing. Inheritance. Templates. **Last day to Register to Vote**	
15	Wed, Oct 26	Intro to 3D graphics. Coordinate transformation. Z-buffering. Z-fighting problem. Writing a 3D graphics framework. Fly-through. Orbit	PS8 out
16	Mon, Oct 31	Project Introduction. More examples of 3D graphics	Demo Project Due
17	Wed, Nov 02	**Demo Night**	
18	Mon, Nov 07	Virtual functions and Polymorphism.	
	Tue, Nov 08	**Election Day**	
19	Wed, Nov 09	textString class (How std::string works). Copy constructor and copy operator.	PS9 out
20	Mon, Nov 14	Template functions and classes. ‘decltype’. std::function and std::bind	
21	Wed, Nov 16	Runtime Type Information (RTTI) Dynamic cast. Recursion. Singly-linked and doubly-linked list	PS10 out
23	Mon, Nov 21	Binary Tree. Merge Sort	
	Wed, Nov 23	**Thanksgiving Break. No Classes**	
24	Mon, Nov 28	GUI Development	
25	Wed, Nov 30	Parallel processing. std::thread and std::mutex	PS11 out
26	Mon, Dec 05	Project Presentations.	
27	Wed, Dec 07	Project Presentations. Last Day of Classes	