24783 Advanced Engineering Computation Project

1 Important Dates

Team Preference	2/27 (Mon) by E-Mail
Short Presentation of Project Proposal	3/13 (Mon) Git Server & in Class
Finalize Project Proposal	3/17 (Fri) Git Server
Component Design	3/24 (favorite) Git Server
Alpha Version	4/3 (Mon) Git Server
Final-Goal Calibration Presentation	4/10 (Mon) Git Server
Beta Version	4/17 (Mon) Git Server
Final Version and Presentation	4/26 (Wed)

Weight: 28%

2 Project Goal

The goal of the project is to let you practice software development in a team environment, which requires an additional skill than individual programming. The goal is NOT to develop something that looks nice.

You may use external libraries in the project. However, it should not provide majority of the functionality. There are thousands of external libraries available for free, which allows you to build a nice-looking program quickly. However, if an existing library does majority of the work, and you write only small lines of code, it kills the whole purpose of the project.

You have two options this year.

Option A: Find and solve an engineering problem with C++. It can be a problem that you deal with on a day to day basis. Or, it can be part of your research project. For example, you may be doing repetitive and painful manual operations over and over again for your research, which may not be a focus of your research. Such manual operations may be made more efficient by writing a program that automates the process. Another example can be an modeling tool specialized for your problem. We often use a general-purpose CAD program to generate test cases, sample models, etc. However, a general-purpose CAD program may not be most efficient for your needs. Writing a program that quickly create those models may expedite overall research. Those are good candidates for your Option-A project.

Option B: Port a classic arcade game from around 1990. Pick a legendary classic arcade game and port it to Windows, macOS, and Linux. The graphics and audio do not have to be exact, as long as your program replicates an essence of the game. If the game play continues more than five minutes, your program needs to replicate at least first 5 minutes of the game play.

If you have a favorite game that you want to port, and it is not from 1990s, I'm happy to discuss about it.

3 Team Member and Team Name Preference

Form a team of 4 to 6 students, and send an email to our teaching assistants. You can CC to me, but make sure your email is sent to our TAs.

The email subject needs to begin with "24783 Team Preference". If you want to include your team name, append it to the subject, but make sure the subject begins with "24783 Team Preference"

The email needs to include (1) list of students (Name and Andrew IDs), (2) who is the team leader, and (3) team name.

Unless you provide with your team name, our TAs will give a strange name to your team. So, please include your team name.

Our TAs will set up your team Git repository.

4 Short Presentation of Project Proposal

Once you form up a team, you need to discuss and choose a project topic. Please make a five-minute short presentation of your project topic. The presentation must include (1) what your program will do, (2) why you choose this topic, and (3) why you think it is of reasonable difficulty for this course project.

Please be ambitious in the proposal. You'll have a chance to calibrate your final goal before the final presentations.

When you prepare a presentation, you always think about the purpose and audience. In this case, the main purpose is to persuade me that it is a good project topic. Therefore, the main audience is me. Side purpose is to show everyone what everyone else is going to do for the project.

Upload your presentation package (can be Power Point of PDF) to your team Git repository.

5 Finalize Project Proposal

Make corrections and finalize your project proposal based on the feedback from the project-proposal presentation.

Write 1- or 2-page final proposal in PDF format and submit to your team Git repository.

6 Component Design

Before jumping on to programming, think about what components are needed and the behavior of them. A component can be a class or a set of functions. If you are going to use an external libraries (such as OpenCV) describe it as a component. The component-design document must include for each component:

- Description of the component
- Who is in charge (or is an imported external library)
- Minimum 3 unit tests

The description of a component can be just a text, a text plus flow chart, a text plus state-transition diagram, a text plus C++ class definition, or anything comfortable for you. The important point is it should work as a blueprint of your program. Also, it is a good idea to make clear purposes of the component.

There have been numerous methodologies for designing computer programs. While many people say flow chart is obsolete, we still use flow chart. State-transition diagram is good in general, but in many cases a simple text describing what states that a component can take and conditions for a transition is clear enough, and the advantage is you can write it as comments in your code.

Also think how your component can be tested. List minimum 3 ways to test your component individually.

Ideally one person is in charge of one to two components. For each component, please write who is in charge (or is an imported external library).

Since it should be a blueprint, too-long a description will lose its purpose. I would say the description of one component should be between half-page to 1 page.

You do not have to stick to this design. It is always possible that things get clearer as you write code, and you find that the initial design needs to be modified, which is perfectly fine. The goal here is to have something to start with.

Please upload your component-design document (one file PDF) to your team Git repository.

7 Alpha Version

Alpha version does not have to be fully functional. What is important is the final product should be taking shape, and have a clear path to the goal. Ideally the alpha version runs with limited-functionality, but some components can still to be linked.

You do not have to upload a document as an alpha-version. Instead, tag your team Git repository as ”Alpha” and push to the server.

8 Final-Goal Calibration Presentation

In industry, under-promise and over-delivery keep customers happy and good for the business. However, in reality, the project easily fall into over-promise and under-delivery. What if you cannot deliver what you promised? You need to first persuade your supervisor, and then your customer to convince that your product is valuable.

If you are missing your goal, it is your chance to make your case and justify why it is reasonable to lower the bar.

Please include your presentation (1) current progress of your project, (2) what you expect to achieve, and (3) what probably need to be dropped and why.

The presentation should be up to 5 minutes plus 3 minutes Q&A.

9 Beta Version

Beta version should be almost-functional program. Each component must have minimum 3 unit tests that can be run from ctest, and all must pass.

You do not have to upload a document as a beta-version. Instead, tag your team Git repository as ”Beta” and push to the server.

10 Final Version and Presentation

We will set up tables for each group, and let everyone try every team's product, and hopefully we get some free food as well. Please set up two computers, one shows a presentation slides, and the other for live demo.

Presentation slides need to be short concise. It should explain what your product does, summary of how to use your product, your product's advantage (why it is fun? how it solves a problem?), also (for grading purpose) tell who did what part of the development.

Imagine you are at a convention and trying to find a new customer. Tell your potential customer the advantages of your product, what is the fun point, how technologically advanced, etc.

The final version also should be uploaded to a Git repository where everyone has read access, so that we can download and run on our own computers.

Add the presentation package to your Git repository, tag your repository as ”Final”, and push to the server.