CS 3: Introduction to Software Design — Spring 2022

Course Syllabus

Information At-A-Glance

Instructor	
Name:	**Adam Blank**
E-mail:	`blank@caltech.edu`
Office:	ANB 115
Office Hours:	By private meeting.

Course Website

`https://sof.tware.design`
Visit early. Visit often.

Lecture

Beckman Institute Auditorium on M
01:00 PM – 01:55 PM

Course Overview

This course covers large program design in the C programming language. Prerequisite: CS 2.

CS 3 is a practical introduction to designing large programs in a low level language. Heavy emphasis is placed on documentation, testing, and software architecture. Students will work in teams in two 5-week long projects. In the first half of the course, teams will focus on testing and extensibility. In the second half of the course, teams will use POSIX APIs, as well as their own code from the first five weeks, to develop a large software deliverable. Software engineering topics covered include code reviews, testing and testability, code readability, API design, refactoring, and documentation.

Grading Policy

Your grade will be composed of several categories with varying weights: 7 labs (30%), project0 (5%), a physics engine (30%), a game design document and proposal (5%), a game or simulation (30%). **If you do not get at least a ✓ on every week in the physics engine, you will receive an F in the course. Also, if you miss more than two weeks of code reviews, you will receive an F in the course. If none of the members of your group attend a code review in a given week, the corresponding project will receive a zero.**

- Labs will be graded on a correctness basis. We will check that you've correctly completed lab n a week after the lab is released. If you have completed it, you'll get full credit; otherwise, you will get no credit.
- project0 and the physics engine will be graded based on three dimensions: functionality, testing, and design. The percentages of these categories will vary drastically for each project, but we will tell you explicitly what they are. You will only receive testing and design points for functionality that is implemented correctly. In other words, we will multiply the design and testing components by the fraction of functionality points you received.

  Functionality points will be assigned based on gitlab tests and demos during code review. Testing and design points will be assigned on a $+/\checkmark/-/0$ scale after code reviews. To convert these grades to points we will use the following scale: $+ \mapsto 100$, $\checkmark \mapsto 80$, $- \mapsto 60$, $0 \mapsto 0$.

  We expect you to improve your testing and design scores on most assignments by implementing the feedback provided by your code reviewer. This will allow your grade in each section to move one rank up on the scale.
- Your game design document and proposal will be graded on a rubric we will provide in the assignment write-up.
- Your game will be graded based on three parts: individual contribution, group result, and application of the guidelines for good software discussed throughout the course.

Late Policy

In this course, you have the opportunity to *earn* up to 10 "late tokens". You may not use partial tokens, however. The following activities will net you partial tokens:

- Attending a lecture will net you 1 late token.
- Setting up a meeting with Adam to discuss the course, the option, or whatever will net you $\frac{1}{3}$ of a late token.

You may use these tokens at your discretion subject to the following rules:

- You may not use more than one token per week
- Each token gets you an entire extra weekend, but you must set up a Code Review on the following Monday with Adam
- If you are in a group, everyone in that group must use a late token

Projects will never be accepted more than three days late under any circumstances.

Getting Help

Please don't be afraid to ask for help if you don't understand something.

During office hours or lab hours time, you can ask for clarification on a lecture (or for a *repetition* of the lecture!) either via the ticketing system or discord. You can ask for help with a frustrating part of the homework. You can even show up just to tell us you're frustrated and vent.

Collaboration & Academic Integrity

You may not look at other students' code outside of your group. No one except TAs may look at your code outside of your group. You may use the Internet, but you may NOT look at online code repositories that were not okayed by course staff. You may discuss design decisions outside of your group. You may NOT discuss code or pseudocode outside of your group. You may discuss the way C works with anyone. You may NOT discuss the way C works in the context of the project outside of your group. We reserve the right to modify or clarify this policy as needed.