CS193P: iPhone Programming

Course Overview

The goal of CS193P is to teach you how to write object-oriented applications for iPhone and iPod touch, using the Cocoa Touch framework on Mac OS X. The language we will be using is Objective-C. We will primarily be using Mac OS X 10.6 (aka SnowLeopard) in lecture, but 10.5 (aka Leopard) can also be used.

Who We Are and Where We'll Be

Instructors: Al Cannistraro <accannis@stanford.edu>
Josh Shaffer <joshh@stanford.edu>

Teaching Assistants: Paul Salzman <paulsalz@stanford.edu>
David Jacobs <dejacobs@stanford.edu>

Admin Support: Paul Marcos <pmarcos@stanford.edu>

Lecture Location: Education 128
Lecture Hours: Tues & Thurs 4:15 pm – 5:30 pm
Office Hours: TBD

Email Address: cs193p@cs.stanford.edu
Website: http://cs193p.stanford.edu

Prerequisites

Students should have completed either CS106B or CS106X. Previous experience with object-oriented programming will be helpful but not strictly required. Obj-C is a simple language that's easy to learn. The Cocoa Touch frameworks are based on the Mac OS X Cocoa frameworks which are mature and provide a wonderful environment for developing real-world object-oriented applications. Don't be afraid! If you're comfortable with C, you'll be fine. You should be familiar with C constructs such as pointers, arrays, memory allocations and debugging C programs.

Lecture

Lectures will be held in Education 128 on Tuesday and Thursday from 4:15 PM – 5:30 PM. We expect to have an additional section on Fridays although the time and location has not yet been determined. Lectures will be recorded and available on iTunes U. Typically it takes a couple days before lectures are posted and are sometimes delayed. Please do not rely on iTunes U videos for lecture material!

Web Site

The web site URL is http://cs193p.stanford.edu. We'll keep the syllabus, current assignment information, all the handouts, slides from lecture, project files and any announcements on the web site. Office hours and contact information will always be available there as well.

Textbook

There is no required textbook for 193P. Instead, we'll rely heavily on the documentation available at Apple's iPhone Dev Center. Once you've installed the SDK you'll have the documentation available locally in Xcode or you can view it online at developer.apple.com.

Assignments

We'll start out with an introductory assignment to get familiar with the tools and the Objective C language. After that we'll move to an a couple assignments to get our feet wet with the Cocoa Touch frameworks including a number of the design patterns and UI elements available. Next we'll spend several weeks developing a more complicated application with more features and covering more of the various aspects of the iPhone SDK. There will be a series of weekly assignments that work through various object oriented techniques and areas of functionality in the Cocoa Touch frameworks. We will focus on the Model-View-Controller design that is fundamental to all Cocoa Touch (and Cocoa on Mac OS X) applications. Once we've finished with our application we'll move on to a multi-week final project of your choice.

The final projects are your chance to strut your newfound iPhone stuff! Let your imagination run wild and come up with a great project that has a great user interface and leverages the OO strengths of Cocoa Touch and Mac OS technologies.

There will not be a final exam for this class. Instead, you will do a brief presentation of your final project in what would be our final exam slot. During the final exam time slot we'll ask you to demo your application for us. Details will be provided as that time approaches.

All projects will be submitted electronically.

Grading

Each assignment will be graded on a very simple scale: ✓, ✓+ or ✓-. The weekly assignments will count for roughly 60% of your grade and the final project will count for the remaining 40%. What does this really mean? Here's an idea of what these mean:

✓+	Given to a solid submission that completes all the required functionality, usually with extra credit included. Attention has been paid to things like correct memory management, well organized object-oriented code, good MVC design, doesn't crash and only contains trivial errors.
✓	Given to a submission that fulfills the vast majority of the required functionality of the assignment. The submission might contain one or more major errors, or enough minor errors to not warrant a ✓+.
✓-	Given to a submission that either doesn't fulfill the required functionality or has significant problems with the code or execution.
Zero	Given to a submission that is never turned in.

If a student receives a solid string of ✓'s that would generally yield a grade in the B+ / A- range. Getting ✓+'s would obviously raise that, while ✓-'s would lower it. We will sometimes use intermediate grades such as ✓/✓+ or ✓++ to help clarify when a submission is on the fence between buckets, or for exceptionally good (or poor) submissions.

Handouts

All handouts and lecture slides will be posted to the class web page. Printed copies of handouts and slides will usually not be available in lecture.

Late Policy

We strongly encourage you to start the assignments early to avoid having to turn assignments in late. Regardless, we realize that unexpected circumstances arise and to accommodate this every student can use 3 late days. Note that these are calendar days. These are yours to use at

your discretion; you can use 1 late day on three different assignments or use 3 late days on a single assignment. Late days may not be used on the final project. Once your late days are used up, late work will receive one lower grade per day late. That is, a ✓+ becomes a ✓ when turned in 1 day late and a ✓ becomes a ✓- when turned in 2 days late.

Frequently Asked Questions

Q: Will the class be televised?
A: Yes, via iTunes U. The class is not available via SCPD.

Q: I have my own Macintosh and I'm running 10.4 Tiger, can I use that for the assignments?
A: No. The minimum requirement for using the iPhone SDK is Mac OS X 10.5 Leopard or 10.6 SnowLeopard.

Q: I don't have an iPhone or iPod touch, can I still take this class?
A: Yes. We expect to be able to provide loaner iPod touches for students who need them. Regardless, all of the work we'll be doing can be done using the iPhone simulator running on Mac OS X. If you have your own iPhone or iPod touch you can use that for development.

Q: Can I audit this class if I am not admitted?
A: Yes, you are welcome to sit in on the class. We try to support auditors as best we can, but our primary responsibility is to the students enrolled in the class so we ask you to respect this when it comes to visiting TA office hours or emailing the staff.

Honor Code

What follows are the guidelines regarding the Stanford's Honor Code. The policy we are using in this class is effectively the same as the one from CS106.

Since 1921, academic conduct for students at Stanford has been governed by the Honor Code, which reads as follows:

> THE STANFORD UNIVERSITY HONOR CODE
>
> A. The Honor Code is an undertaking of the students, individually and collectively:
>
> (1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
>
> (2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.
>
> B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid as far as practicable, academic procedures that create temptations to violate the Honor Code.
>
> C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

In the Computer Science Department, we take the Honor Code seriously and expect you to do the same. The good news is that the vast majority of you will do so. The bad news is that all historical evidence indicates that some students in computer science will submit work that is not their own, shortchanging not only their own learning but undermining the atmosphere of trust and individual achievement that characterizes Stanford's academic community. Each year, the Computer Science Department accounts for somewhere between 20 and 40 percent of all Honor Code cases, even though our courses represent only about seven percent of the student enrollment.

The purpose of this handout is to make our expectations as clear as possible in the hope that we will reduce the number of Honor Code violations that occur. The basic principle under which we operate is that each of you is expected to submit your own work in this course. In particular, attempting to take credit for someone else's work by turning it in as your own constitutes plagiarism, which is a serious violation of basic academic standards.

From the attention that the department pays to the Honor Code, some of you will get the idea that any discussion of assignments is somehow a violation of academic principle. Such a conclusion, however, is completely wrong. In computer science courses, it is usually appropriate to ask others—the TA, the instructor, or other students—for hints and debugging help or to talk generally about problem-solving strategies and program structure. In fact, we strongly encourage you to seek such assistance when you need it. The important point, however, is embodied in the following rule:

Rule 1: You must indicate on your submission any assistance you received.

If you make use of such assistance without giving proper credit, you may be guilty of plagiarism.

In addition to providing proper citation—usually as part of the comments at the beginning of the program—it is also important to make sure that the assistance you receive consists of general advice that does not cross the boundary into having someone else write the actual code. It is fine to discuss ideas and strategies, but you should be careful to write your programs on your own. This provision is expressed in the following rule:

Rule 2: You must not share actual program code with other students.

In particular, you should not ask anyone to give you a copy of their code or, conversely, give your code to another student who asks you for it. Similarly, you should not discuss your algorithmic strategies to such an extent that you and your collaborators end up turning in exactly the same code. Discuss ideas together, but do the coding on your own.

The prohibition against looking at the actual code for a program has an important specific application in computer science courses. Developing a good programming assignment often takes years. When a new assignment is created, it invariably has problems that require a certain amount of polishing. To make sure that the assignments are as good as they can be, Stanford's department—like most others in the country—reuses assignments over the years, incorporating a few changes each time to make them more effective. The following rule applies in all computer science courses:

Rule 3: You must not look at solution sets or program code from other years.

Beyond being a clear violation of academic integrity, making use of old solution sets is a dangerous practice. Most assignments change in a variety of ways from year to year as we seek to make them better. Each year, however, some student turns in a solution to an assignment from some prior year, even though that assignment has since changed so that the old solution no longer makes sense. Submitting a program that solves last year's assignment perfectly while failing to solve the current one is particularly damaging evidence of an Honor Code violation.

Whenever you seek help on an assignment, your goal should be improving your level of understanding and not simply getting your program to work. Suppose, for example, that someone responds to your request for help by showing you a couple of lines of code that do the job. Don't fall into the trap of thinking about that code as if it were a magical incantation—something you simply include in your program and don't have to understand. By doing so, you

will be in no position to solve similar problems on exams. The need to understand the assistance you receive can be expressed in the following rule:

Rule 4: You must be prepared to explain any program code you submit.

In accordance with Stanford's judicial policy, we are required to tell you that we use plagiarism detection tools to help identify possible violations. We archive all submissions, both from this quarter and past quarters, and cross-compare for unusual resemblance. We do not target specific students, all assignments are subject to the same scrutiny. Any similarity detected by the tools is then examined more closely by our staff and, where appropriate, cases are referred to Judicial Affairs. The tools are very adept at identifying all variants of improper collaboration, from major to minor.

Rule 5: All submissions are subject to automated plagiarism detection.

In summary
Although you should certainly keep these rules in mind, it is important to recognize that the cases that we bring forward to Judicial Affairs are not those in which a student simply forgets to cite a source of legitimate aid. Most of the students we charge under the Honor Code have committed fairly egregious violations. Students, for example, have rummaged through paper recycling bins or undeleted trash folders to come up with copies of other students' programs, which they then turn in as their own work. In many cases, students take deliberate measures—rewriting comments, changing variable names, and so forth—to disguise the fact that their work is copied from someone else. Despite these cosmetic changes, it is usually easy to determine that a copy has been made. Programming style is highly idiosyncratic, and the chance that two submissions would be that similar is vanishingly small.

We have no desire to create a climate in which students feel as if they are under suspicion. The entire point of the Stanford Honor Code is that we all benefit from working in an atmosphere of mutual trust. Students who deliberately take advantage of that trust, however, poison that atmosphere for everyone. As members of the Stanford community, we have a responsibility to protect academic integrity for the benefit of the community as a whole.