Objectives and Outcomes

This course explores the computer science subfield known as interaction design (IxD), also called computer-human (or human-computer) interaction (CHI/HCI). IxD is concerned with how human beings interact with computing systems, provides measures for its effectiveness, and explores techniques and theories for achieving effective interaction. Long after the course concludes, my hope is that you will:

1. Know and understand the art and science of interaction design, particularly its first principles and key metrics.
2. Apply this knowledge by studying, comparing, and evaluating the user interfaces of actual systems.
3. Know the fundamentals behind implementing user interfaces with working knowledge of technologies such as HTML/CSS/JavaScript, Ajax, jQuery, and Bootstrap.

Although there are no absolute prerequisites to this course, intermediate to advanced programming proficiency in any language will be helpful. Students concurrently taking CMSI 386 Programming Languages will benefit from exposure to common language concepts with varying syntax. Some of the material in this course carries directly into CMSI 371 Computer Graphics.

Materials and Texts

- Assorted handouts, articles, and sample code to be distributed throughout the semester.

The following text, especially Chapters 6–8, can serve as a programming tutorial and reference:


In addition, do not hesitate to look for further information regarding the concepts, techniques, tools, and paradigms that we will discuss.

Course Work and Grading

This course uses standards-based grading: your proficiency in each course objective is directly evaluated according to the outcomes shown on page 4 of this syllabus. Proficiency is measured according to the following key:

| + | Advanced proficiency |
| | | | Appropriate proficiency |
| / | Approaching appropriate proficiency |
| − | Needs practice and support |
| O | Not yet evaluated |

Your submitted work is used to evaluate these outcomes (see below). Letter grades are then assigned as follows:

| + | − |
| A | many | none |
| B | many | few |
| C | some | few |
| D | some |
| F | many |

A–, B+, B–, C+, and C– grades may be assigned based on “close calls” along the proficiency measure thresholds and qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course. You may inquire at any time about the proficiency measures that I currently have on record for you.
**Homework**

Homework consists of questions, exercises, and programming assignments to be given throughout the semester. Homework is one mechanism for demonstrating the proficiencies expected of the course. You will be given feedback on these proficiencies, and may resubmit your homework throughout the semester in order to improve upon them.

With great flexibility comes great accountability. First, you must submit your homework on time. The assignment due date is encoded in the homework number. Late homework detracts from outcome 4f (Meet all designated deadlines).

**Quizzes and Tests**

Some outcomes are best demonstrated by answering questions or doing exercises in class. These resemble traditional quizzes and tests, but, like homework, they are evaluated according to standards and do not produce a numerical score. They are typically spontaneous and unannounced.

Questions may include content-oriented elements as well as forward-looking, applicative portions (i.e., “use this knowledge to resolve this situation”). Tests are open-paper-everything; no sharing. “Open computer” might be allowed depending on the circumstances. You may neither solicit nor give help during a quiz or test. Late or missed tests are handled case-to-case; in all instances, talk to me.

**Term Portfolio**

Your accumulated homework and tests for the semester comprise the term portfolio—the final, definitive artifact that demonstrates the proficiencies you have reached for each course outcome. The term portfolio provides you with an opportunity to polish the work done throughout the semester; it is how you show that you learned from your mistakes or improved on already established knowledge.

Throughout the semester, you may improve your work based on received feedback and show it to me for re-evaluation. Improvements in proficiency are recorded and give you a good idea of how your term portfolio will fare long before its final version is submitted.

The final version of the term portfolio is due on December 14. Late portfolios detract from outcome 4f.

**Extra Credit**

In terms of standards-based grading, “extra credit” takes on a different meaning: it indicates work that, if successfully performed, would indicate advanced proficiency (+). Extra credit tasks may be assigned for homework, quizzes/tests, or the final term portfolio. Accomplish them successfully to rack up those +’s. You do not need to perform extra credit work to show advanced proficiency; it merely demonstrates such proficiency more readily.

**Version Control**

Version control is an indispensable part of today’s computer science landscape in industry, the academy, and the open source community. We use version control heavily in this course: make sure that you get the hang of it.

**Attendance**

Attendance at all sessions is expected, but not absolutely required. If you must miss class, it is your responsibility to keep up with the course. The last day to add or drop a class without a grade of W is August 31. The withdrawal or credit/no-credit deadline is November 2.

**University Policy on Academic Honesty**

Loyola Marymount University expects high standards of honesty and integrity from all members of its community. All students are expected to follow the LMU Honor Code and Process, as stated in the LMU Undergraduate Bulletin.

**Americans with Disabilities Act**

Students with special needs as addressed by the Americans with Disabilities Act who need reasonable modifications, special assistance, or accommodations in this course should promptly direct their request to the Disability Support Services (DSS) Office. Any student who currently has a documented disability (physical, learning, or psychological) needing academic accommodations should contact DSS (Daum 224, x84535) as early in the semester as possible. All discussions will remain confidential. Please visit http://www.lmu.edu/dss for additional information.
# Topics and Important Dates

Correlated outcomes are shown for each topic. Specifics may change as the course progresses. University dates (italicized) are less likely to change.

<table>
<thead>
<tr>
<th>August</th>
<th>Background and history of interaction design ($1a$, $2a$); version control setup ($4e$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 31</td>
<td>Last day to add or drop a class without a grade of W</td>
</tr>
<tr>
<td>September</td>
<td>Usability metrics ($1b$, $2b$, $2c$); cognitive psychology talk ($1a$); guidelines, principles, and theories ($1c$, $2c$); introduction to modern web apps ($3a–3c$, $4a–4e$)</td>
</tr>
<tr>
<td>October</td>
<td>Overview of interaction styles ($1d$, $2c$); menus, forms, and dialogs ($1d$, $2c$); implementation in HTML/CSS/JavaScript ($3a–3d$, $4a–4e$)</td>
</tr>
<tr>
<td>November</td>
<td>Direct manipulation ($1d$, $2c$); affordances ($1e$, $2c$); implementation in HTML/CSS/JavaScript ($3a–3d$, $4a–4e$)</td>
</tr>
<tr>
<td>November 2</td>
<td>Withdraw/credit/no-credit deadline</td>
</tr>
<tr>
<td>November 21–23</td>
<td>Thanksgiving; no class</td>
</tr>
<tr>
<td>December</td>
<td>Portfolio improvement workshops ($1a$–$4e$); miscellaneous topics (varies)</td>
</tr>
<tr>
<td>December 14</td>
<td>Term portfolios due</td>
</tr>
</tbody>
</table>

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course web site (it’s too long to provide in writing).

If necessary, this syllabus and its contents are subject to revision. Students are responsible for any changes or modifications announced in class.
**Course Outcomes**

1. **Know and understand the art and science of interaction design, particularly its first principles and key metrics.**

   - 1a. Know and understand how interaction design relates to mental models.
   - 1b. Know and understand the five usability metrics.
   - 1c. Know and understand interaction design guidelines, principles, and theories.
   - 1d. Know and understand interaction styles.
   - 1e. Know and understand affordances.

   This is derived mainly from Don Norman’s big picture view of interaction design, as expressed in *The Design of Everyday Things*. For outcomes 1b to 1e, “knowing and understanding” means being able to list, define, explain, and give examples of each.

2. **Apply this knowledge by studying, comparing, and evaluating the user interfaces of actual systems.**

   - 2a. Map real-world interaction design cases and/or situations to how mental models are expressed and communicated.
   - 2b. Prioritize the five usability metrics for a given application.
   - 2c. Effectively use usability metrics, interaction design guidelines, principles, and theories, interaction styles, and affordances to make appropriate, well-founded interaction design decisions.

   This involves identifying what parts of a situation correspond to Norman’s big picture view and showing how they interact. This includes effectively explaining your prioritization as well.

3. **Know the fundamentals behind implementing user interfaces with working knowledge of technologies such as HTML/CSS/JavaScript, Ajax, jQuery, and Bootstrap.**

   - 3a. Know and understand how user interfaces are constructed.
   - 3b. Know and understand event-driven programming.
   - 3c. Know and understand the model-view-controller (MVC) paradigm.
   - 3d. Break down a high-level user action into a sequence of lower-level user or system events.

   These outcomes are all demonstrated by writing programs that involve one or more of these areas. Thus, some specific set of technologies, languages, and libraries must be learned and used—for this course, we focus on the aforementioned web technologies. However, it must also be understood that these concepts are general and technology-independent: when called for, one should be able to transfer this knowledge to other platforms.

4. **Follow academic and technical best practices throughout the course.**

   - 4a. Write syntactically correct, functional code.
   - 4b. Demonstrate proper separation of concerns, especially MVC.
   - 4c. Write code that is easily understood by programmers other than yourself.
   - 4d. Use available resources and documentation to find required information.
   - 4e. Use version control effectively.
   - 4f. Meet all designated deadlines.

   Code has to compile. Code has to work. No errors, no bugs. Use unit tests as much as possible.

   This is the basis of good software design. It makes software easier to maintain, improve, and extend. Proper separation of concerns includes but is not limited to correct scoping of variables & functions and zero duplication of code.

   This outcome involves all areas of code readability and clarity, including but not limited to documentation & comments, spacing & indentation, proper naming, and adherence to conventions or standards.

   The need to look things up never goes away. Remember also that the course instructor counts as an “available resource,” so this outcome includes asking questions and using office hours.

   In addition to simply using version control correctly, effective use also involves appropriate commit frequency and descriptive commit messages.
Sample Standards Achievement Report

Based on these proficiencies, the student will get a B–.

1. **Know and understand the art and science of interaction design, particularly its first principles and key metrics.**

   1a. Know and understand how interaction design relates to mental models.
   1b. Know and understand the five key usability metrics.
   1c. Know and understand interaction design guidelines, principles, and theories.
   1d. Know and understand interaction styles.
   1e. Know and understand affordances.

2. **Apply this knowledge by studying, comparing, and evaluating the user interfaces of actual systems.**

   2a. Map real-world interaction design cases and/or situations to how mental models are expressed and communicated.
   2b. Prioritize the five usability metrics for a given application.
   2c. Effectively use usability metrics, interaction design guidelines, principles, and theories, interaction styles, and affordances to make appropriate, well-founded interaction design decisions.

3. **Know the fundamentals behind implementing user interfaces with working knowledge of technologies such as HTML/CSS/JavaScript, Ajax, jQuery, and Bootstrap.**

   3a. Know and understand how user interfaces are constructed.
   3b. Know and understand event-driven programming.
   3c. Know and understand the model-view-controller (MVC) paradigm.
   3d. Break down a high-level user action into a sequence of lower-level user or system events.

4. **Follow academic and technical best practices throughout the course.**

   4a. Write syntactically correct, functional code.
   4b. Demonstrate proper separation of concerns, especially MVC.
   4c. Write code that is easily understood by programmers other than yourself.
   4d. Use available resources and documentation to find required information.
   4e. Use version control effectively.
   4f. Meet all designated deadlines.