AP CS Principles Pilot at University of Washington

Author: Lawrence Snyder
Course Name: CSE120 Computer Science Principles
Pilot: Winter Quarter 2011

a) How CSE120 Fits In At UW:

UW's Computer Science and Engineering department has offered CSE100, *Fluency with Information Technology* for a decade in collaboration with the University's Information School. That class is taught to 150 students (typically freshmen) per quarter from across campus. It is a required course for a few departments. FIT100 is consistently over-subscribed and further expansion is resource limited. The course is general interest, and is considered preliminary to the standard introductory sequence, *Computer Programming I & 2*.

CSE120 *Computer Science Principles* was added as a new preliminary class. It is distinguished from *Fluency* as follows: Both classes are concepts classes that teach similar content: *Fluency* emphasizes concepts to make students more facile with computation; *Principles* emphasizes concepts as science and stresses the clever and amazing ideas of CS as a science.

b) Course Stats

- **Lectures**: 3 50-minute periods, MWF
- **Labs**: 2 50-minute closed labs with TA instruction, TTh
- **Course**: 10-week quarter, yielding 50 contact hours
- **Credit Hours**: 5
- **Fulfills Requirements**: General Ed, Quantitative & Symbolic Reasoning
- **Attendance**: 22 started; 22 finished
- **Programming Language**: Processing and XML with some HTML/CSS
- **Grading**: 20 homework assignments, 5 lab exercises, midterm, final

c) Class Content The class followed the schedule shown below.

Lectures were a combination of conventional Powerpoint slides, online demos and discussion. Labs were generally designed to give students hands-on experience with material illustrating or implementing concepts discussed in the preceding lecture. Some of the lab exercises were graded.

In general a homework assignment would be given at each lecture to be turned in prior to the next lecture. The exercises were varied, and doable within two hours by

---

© ACM, 2012. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in ACM Inroads, {VOL 3, ISS 2, (June 2012)} http://doi.acm.org/10.1145/ 2189835.2189856

a typical student. Students were asked to spend one hour per day engaged in class work, as opposed to two hours in one sitting. Eventually the assignments became larger and spanned several classes. Lab time was allocated so the TA could answer questions. Though students preferred the less frequent assignments, at the start the relentless every-other-day schedule brought everyone quickly up to speed with basic concepts, esp. programming ideas.

The class emphasized using CS Principles in other majors, rather than trying to motivate students to major in CS. (UW's program has severely limited enrollment and accepts far fewer than the number of qualified students.) Two guest lectures from departmental faculty emphasized the use of fundamental CS for non-technical purposes.

- Richard Ladner: Accessibility – Using Computing To Help The Handicapped
- Zoran Popovic: FoldIt and Other Games To Make The World Better

Students found both lectures extremely inspirational. Regarding majoring in CS specifically, there was an "Ask The Advisor" reception (with pizza) before the first Watson Jeopardy Match. Several students have started the path to be CS majors.

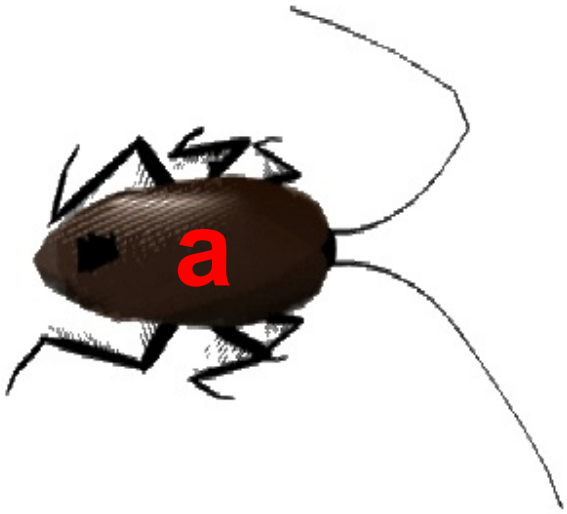
Topic List for UW's CS Principles in Time Sequence

1. 1 Welcome & Orientation
2. "What I Value" writing exer. (lab)
3. Lightbot: Game or Programming?
4. FTP Review & Portfolio Page (lab)
5. Digitization: History of 0 & 1
6. Computing in Social Setting
7. Bits: Counting in bases 2, 16 (lab)
8. Basic Processing Ideas
9. Q & A on Processing (lab)
10. Digital Representation
11. MLK Holiday – no class
12. Writing Processing Pgms (lab)
13. Pac-Man – Abstract in Proc'g
14. Populating Portfolio Page (lab)
15. If's and For's in Processing
16. Color, RGB, binary arithmetic
17. Practice w/binary adding (lab)
18. Fetch/Executing – How it works
19. Work on Own Programs (lab)
20. Sports Timer: Abstraction Layers
21. Digital, Analog, Bits "R" Universal
22. Practice writing Functions (lab)
23. Complexity, Universality Principle
24. Work on Turtle Game HW (lab)
25. Midterm Review and Recap
26. Midterm Exam – In class 1 hr
27. Lab Canceled
28. Recursion – Examples/Uses
29. Practice Recursion in Proc'g (lab)
30. Reasoning: Algorithm Correctness
31. Artificial Intelligence – Jeopardy
32. Pair Programming Intro (lab)
33. Internet Structure + WWW
34. Pair Programming Project (lab)
35. DNS and Large Scale Systems
36. Presidents Day
37. Guest Lec: Ladner, Accessibility
38. Searching, Page Rank, Big Data
39. DIY Instruction in HTML (lab)
40. Debugging Principles
41. Guest Lec: Popovic Games 4 Good
42. DIY Instruction in CSS (lab)
43. Data, Meta-data, XML
44. Try Out XML Structuring (lab)
45. Database Structure
46. Crypto + Steganography

47. XML & XSL practice (lab)
48. Review and Recap
49. Help w/Final Project (lab)
50. Party Hats & Pizza

d) **Evidence of Student Work** Creativity is the #1 "Big Idea" and it was #1 in the class. As soon as the rudiments of programming in Processing were presented, the students were assigned the task of inventing two interactive "art works." (See the class web site, linked below.) Later in the course students were asked to invent and implement a game, in order to practice the basics of working in pairs. Though they were not using game development software, Processing is effective enough that they could focus on the concepts, even if totally glitzy features were not available.

An example was Matar Gokiburi developed by two women, one a native Spanish speaker (for whom matar is "kill"), and one a native Japanese speaker (for whom gokiburi are "cockroaches"). Analogous to a typing drill, the player must type the letter fast as the cockroaches scurry around the screen.



e) **What Worked And Didn't**

The Language. UW's CS Principles was an outlier among the pilots in that it did not use a drag and drop programming language. The Processing decision was motivated by preferring a "real" tool used in practice. Whether that is advantageous can be debated. (Students who later took CSE142 *Computer Programming in Java I* said it "was no problem … we'd seen most of it in *CS Principles*.") But the main criticism against symbolic programming– the so called "syntax barrier" – did not present any significant difficulties. The Processing IDE is very helpful, "Did you forget a semicolon?", and students quickly got past those issues. Indeed, there was wide agreement that Processing is just plain fun.

Lightbot. Using the video game Lightbot 2.0 as a first-day exercise to introduce basic computational ideas – specify-then-execute, sequential specification/sequential

execution, limited instruction repertoire, functions, repetition, etc. – was enormously effective, and smoothed the way into Processing well.

HTML/CSS. Because of the sequencing glitches mentioned below, teaching HTML and CSS wasn't fitting into the schedule, so students were given a cheat sheet and a 3-slide introduction, and asked to learn it on their own. This turned out to be highly successful. Students did learn it on their own. But, they were in effect learning the point that they could teach themselves technology. That turned out to be an unintended, but very valuable result.

Abstraction. It's the #2 idea in the "Big Ideas" list, and it was a topic addressed in some way most weeks. Though I didn't test closely for their understanding, my perception based on the discussions suggests a good level of understanding by the 2/3 point of the class.

Sequencing. The rational order for the topics was revised after a few weeks in order to prepare for the debut of the Watson system and the *Jeopardy* competition. Though it was sensible to do this, it seemed like the course lost momentum for a week or two after that.

Concept Gradient. Using Lightbot and the supportive world of Processing, it was possible to introduce concepts without going too fast, but that didn't prevent me from assigning a couple of "killer" homework exercises. Most students survived them, but they need to be fixed. Curiously, these mistakes didn't ruin most students' self-confidence.

f) Links, Resources, Acknowledgments

The pilot was co-taught with Susan Evans, a Seattle high school teacher; she gave a few lectures, designed projects and vetted all assignments. The teaching assistant was Brandon Blakeley, a graduate student in CSE.

The class Web Site: www.cs.washington.edu/cse120

Blog written while teaching CS Principles: csprinciples.cs.washington.edu/blog/

Class documents zipped together:

www.cs.washington.edu/education/courses/cse120/11wi/dist/distribute.html

The Processing Home Page – notice the Exhibition and Learning Links

www.processing.org

Lightbot 2.0 – it's accessible from many sites- armorgames.com/play/6061/light-bot-20