AP CS Principles Pilot at University of California, Berkeley

Authors: Daniel D. Garcia, Brian Harvey, and Luke Segars
Course Name: CS10: The Beauty and Joy of Computing
Pilot: Autumn semester 2010 (and other offerings cited)

a) How CS10 Fits In At UC Berkeley:

The course is intended for non-CS majors. For students in the College of Letters and Sciences, it fulfills the "Quantitative Reasoning" breadth requirement. It is not required for CS majors, but some intended CS majors with no prior programming experience decide to take it as preparation for our first course for CS majors. In addition, many non-CS majors enjoy CS10 enough to continue with the sequence for majors.

The department, on our recommendation, stopped offering our two other venerable (Scheme programming-only) non-CS major classes so these students would all flow through CS10. We have since seen a dramatic growth in enrollment, from 80 students in both fall 2010 and spring 2011 terms to 240 in both fall 2011 and spring 2012 terms. In the spring of 2011, the course was chosen by the *UC Online Instruction Pilot Program (OIPP)* as an online course pilot, and we hope to go live with the online version of CS10 in the fall of 2012.

b) Course Specifics

- **Lectures**: Two 50-minute periods per week, MW
- **Labs**: Two 110-minute closed labs driven by Moodle activities with Teaching Assistant (TA) and Lab Assistant supervision, TuTh
- **Discussion**: One 50-minute recitation period led by TA (used for lecture review, programming problem work in small teams, reading discussions, CS Unplugged exercises, role playing, computer disassembly, group design of video games, project work, etc.)
- **Format in a given week**: Lecture - Lab - Lecture - Lab - Discussion
- **Course**: 14-week semester, yielding 7 * 14 = 98 contact hours
- **Credit Hours**: 4 semester credit hours
- **Fulfills Requirements**: Quantitative Reasoning
- **Attendance**: 80 started; 76 finished
- **Programming Language**: *Scratch* and *Build Your Own Blocks* (BYOB) based on *Scratch.* Soon to be browser-based *Snap!* that works on mobile devices.
- **Grading**: Weekly reading quizzes and homework (15%), 2-3 page paper [later evolving into a 1-page blog with 3 mandatory response paragraphs to other students posts] (15%), Midterm Project (15%), Final Project (15%),

Quest [early, sanity-check exam, halfway between a "quiz" and a "test"] (5%), Midterm (15%), and Final (20%).

c) Course Design

The name of the course originated from Grady Booch's SIGCSE 2007 Keynote in which he exhorted us to share the "Passion, Beauty, Joy and Awe" (PBJA) of computing. We simplified the name (fearing that "awe" may imply more "fear" than "wonder," and that "passion" is internal, not taught) to "Beauty and Joy of Computing" (BJC). Our whimsical logo captures the spirit of the course, while paying homage to its roots (the colors of the BJC letters come from BYOB and the *Blown to Bits* book).

Between the course title, the idea that this may be the last computing course many students ever take, and the need to adequately prepare them for CS61A, several key design principles emerged (many directly from the SIGCSE panels!):

- As much as possible, show beautiful and joyful examples of computing. This includes leveraging (and extending via BYOB) the incredibly well-designed, simple-yet-powerful graphical development environment of Scratch, delightful fractals to teach recursion, elegant higher-order functions that use abstraction to hide messy details of control flow, showing how climate simulations are saving the planet, the internet is wonderfully enabling, etc.
- *Half of the course should be the societal implications of technology*. We assigned an incredible book, *Blown to Bits* (Abelson, Ledeen, and Lewis), which we supplement with videos and other readings.
- Programming should be joyful, so allow students to choose *any project they wish* .
- Deep learning happens by *doing*, not by *listening*, so use the lab-centric model of instruction (here, 4 hours of Moodle-driven labs per week).
- Show relevance of computing to the students. *Every lecture* begins with a "technology in the news" discussion, selected to be most relevant to the student demographic. They are culled from *Slashdot*, *Technology Review*, *NY Times Technology*, *CNN Technology*, *Digg*, etc.
- Lecture should be engaging, so employ *peer instruction* through clickers that are provided as free loaners to all students.
- Invite guest speakers from industry to share "behind the scenes, how the technology behind our business works" talks, to reinforce & ground the importance of the big ideas. We've been fortunate to have accessible and inspiring talks from engineers Raffi Krikorian from Twitter™ and Tao Ye from Pandora™.
- Exciting computing research areas should be presented, by experts in that area. These should be a broad "here's the entire field" overview, not a

narrow "here's my research" talk. We invited departmental faculty Bjoern Hartmann and Armando Fox to speak on HCI and Cloud Computing, and course instructors added their survey of the field of artificial intelligence.

- Make the entire course free to students, so the class can be exported to high schools easily. Everything from the IDE to the book is available for free.
- Learning to program can be deeply frustrating, so encourage both their three-week projects to be done in pairs (even the occasional teams of three).
- It sometimes takes a while for this material to "click" and students often have a bad day, so allow a high score on a later exam to overwrite a poor performance on an earlier exam.

Lecture Topics (and number of lectures on that topic)

- **Abstraction** (1) – Abstraction as a reasoning and problem solving tool
- **Video Games** (1) – The development of video game platforms and technology, games with a purpose (GWAP)
- **3D Graphics** (1) – Fundamentals of 3D graphics (a fun, "How It Works" explanation)
- **Algorithms** (2) – Divide and conquer algorithms and analysis techniques
- **Functions** (1) – How to create functions in BYOB and the beautiful implications
- **Programming Paradigms** (1) – The benefits and trade-offs of each
- **Concurrency** (1) – Dealing with concurrency on one computer, both historical and BYOB-based
- **Distributed Computing** (1) – Dealing with concurrency between computers, both historical and BYOB-based
- **Recursion** (3) – Emphasis on graphical recursion (fractals) and non-linear recursion since looping constructs are available
- **Social implications of computing** (2) – Discussion of the ethical, economic, security and privacy implications of computing (Education, privacy, risks, Patents and copyrights, war, community, etc.)
- **Applications that changed the world** (1) – Commentary and technical details of some applications of computing that have changed the world (Transistor, PC, WWW, Internet, Search Engines, Google Maps, Video Conferencing, Social Computing, Cloud Computing, etc.)
- **Computing in Industry** (1) – Industrial guest (Twitter, in Fall 2011 we added Pandora) explains how they use computing
- **Saving the World with Computing (CS + X)** (1) – Guest from Lawrence Berkeley National Labs talks about climate simulation, protein folding, digital humans, and other supercomputer activities that are saving the world

- **Higher Order Functions and Lambda** (2) – Higher order functions and lambda. The same programming ideas introduced in CS3 (the beauty and power of functions as data, anonymous functions)
- **Cloud Computing** (1) – Fundamentals of cloud computing (a "How it Works" explanation)
- **Game Theory** (1) – Deep Blue and general game strong solving techniques
- **Artificial Intelligence** (1) – Historical progress and current directions and innovation in AI
- **HCI** (1) – Historical progress and current directions and innovation in Human-Computer Interfaces
- **Limits of Computing** (1) – Revisit analysis techniques and discuss NP problems, computers are finite (for representations)
- **Future of Computing** (1) – Discussion of current and future areas of innovation and research
- **Wrap-up** (1) – Project demos and a sneak peak at EECS undergrad course offerings

Lab Topics (and number of labs on that topic)

- **Learn Scratch** (3)
- **Learn BYOB** (1) – Learn how to create "functions" or "Blocks" in BYOB
- **Lists** (2)
- **Algorithms** (2)
- **Concurrency** (1)
- **Distributed Computing** (1)
- **Recursion** (3)
- **Higher Order Functions and Lambda** (2)
- **Hash Table** (1)
- **Simulations** (1)
- **Work on projects** (6)

Reading List: See the course web site (below) for a complete list.

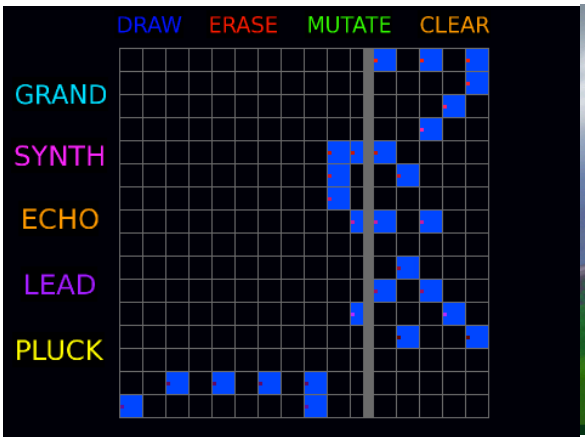
d) Evidence of Student Work

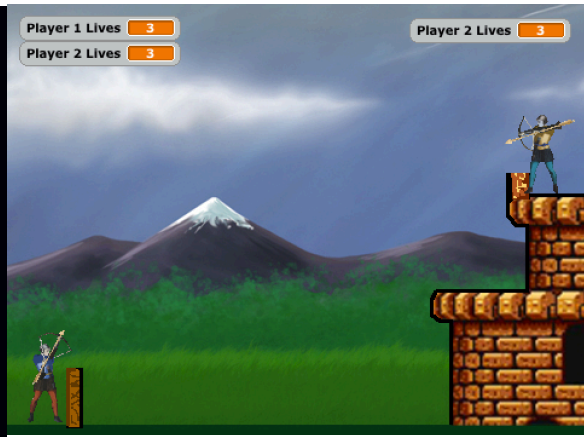
Student papers were topical and interesting. Samples titles: Net neutrality, Prosthetic Arms, Record Companies No Longer Rule the Roost, Computers in Astrophysics, News for the Information Age, AI & Transportation and Electronic Privacy.

Three programming projects (across three consecutive years of teaching the course) were chosen as the best that students ever produced (in only three weeks):

Picoboard Guitar Hero (Pierce Vollucci and Sina Saeidi, fall 2009); these students used a USB PicoBoard sensor board and a keyboard and built their own guitar, then wrote a *Guitar Hero–style* game to train them to play it! The musician would press keyboard keys for different notes and pluck one of the wire frets, which would complete a circuit, and the PicoBoard would let BYOB know what note to play. They performed "The Battle Hymn of the Republic" during the final presentations!

Magic Machine 3.0 (Max Dougherty and Victor Lymar, fall 2010); this incredible program featured an editable 2D grid; time vs instrument (with higher-pitched instruments at the top, drums at the bottom). Clicking in a grid location placed a blue square there, meaning that note should be played at that time. A grey vertical bar slid to the right (and reset on the left after it reached the end), and played the multi-track loop over and over. The user could change the instrument using the words on the left column and change editing modes using the words on the top.





Kinect Archery (Jim Knudstrup, Amanda Atkinson and Vivian Lo, fall 2011); these students hooked up a Microsoft Kinect to BYOB, and authored a very playable archery game that used the Kinect to aim the bow, and physics to control the trajectory of the arrows. Video available at http://youtu.be/wlHwbCcujvo

e) What Worked And Didn't

Sufficient development time, outstanding team, small half pilot … worked. The course was developed over the eighteen months prior to the official AP CS pilot by three teaching-track faculty and ten of the strongest graduate and undergraduate teachers. By the time we launched with the full course pilot in the fall of 2010, we felt we had debugged most of the issues with the software and course content.

Peer Instruction … worked. Students responded that they were delighted they didn't need to pay for clickers (as in all their other classes) and that it made lecture more interesting and dynamic.

Videotaping lectures … mostly worked. We learned that it's hard to have a well-lit instructor as well as a dark screen, and to keep lecturers from pacing the floor.

BYOB … mostly worked. Several key bugs were squashed during the fall 2009 "half pilot" as development continued. The from-the-ground rewrite now in progress (called *Snap!*) will address the remaining speed problems and will run in a browser and on every smart mobile device.

2-3 Page Paper … didn't scale, and didn't take advantage of the social component. The care and effort we gave to the papers for the half-pilot (each of the co-instructors read every paper) clearly didn't scale as we moved to 80 students for the pilot. We replaced this assignment with a one-page blog where students were required to read three other students' blogs (two assigned by us, the other of their choice) and write at least a single paragraph of reflected commentary for each.

f) Links, Resources, Acknowledgments

The fall 2010 pilot was co-taught by Dan Garcia and Brian Harvey, with help from teaching assistants Luke Segars and Jon Kotker. The fall 2009 pre-AP course was co-developed by Dan, Brian, Jon, Mike Clancy, Colleen Lewis, George Wang, Glenn Sugden, Stephanie Chou, Brandon Young, Wayland Siao, Gideon Chia, and Daisy Zhou, with collaboration of local high school teachers Ray Pedersen, Eugene Lemon and Josh Paley.

The entire course curriculum is available at http://bjc.berkeley.edu. We offer teacher preparation workshops to groups of 20 or more teachers in a locality; participants get a stipend and we bring an experienced BJC instructor to your site. See the web site for more details.

The current class Web Site:

http://inst.eecs.berkeley.edu/~cs10/

Build Your Own Blocks (BYOB) based on Scratch, and Snap!

http://byob.berkeley.edu

Testimonials about CS10 : The Beauty and Joy of Computing

http://www.youtube.com/playlist?list=PL48C2AF5762B2D32D

Fall 2010 High-Definition lecture videos:

http://www.youtube.com/playlist?list=PLECBD29A17AAF6EF9

ensemble Computing Portal for commenting and rating BJC learning items

http://www.computingportal.org/bjc_collection