COMPUTER SCIENCE AT CAL

The True Story (ish)

Jon Kotker, jo_ko_berkeley@berkeley.edu

BEFORE WE BEGIN…

WHO AM I?

Jonathan "Jon" Kotker, EECS sophomore, CS3 TA, Sometimes All-Around Nice Guy

WHY AM I TALKING TO YOU?

Because I can.

NO, REALLY.

To explore different aspects of Computer Science and EECS at Berkeley.

FUNNY PICTURE TO WAKE YOU UP

Jon Kotker, jo_ko_berkeley@berkeley.edu

WHY CS/EECS AT ALL?

- Get to make products that everyone can use
- CS is a design discipline – has elements common with architecture and art
- Programming is not the only profession that CS leads into; can also be
  - System Administrator
  - Project Management
  - User Interface Designer
  - Lawyer (not kidding!)

WHY CS/EECS AT ALL?

- Similarly, not all current employees in the Computer Science field come from the Computer Science major!
  - Economics
  - Statistics
  - Applied Mathematics
  - Cognitive Science
- Academia
  - Teach CS courses as a professor – or take my job
  - Perform Computer Science research

CS61A: Fundamentals

- Requires recursion (you've got it!)
- Exposure to several programming paradigms, high-level ways to organize programs, related areas of computer science
- Like and unlike CS3:
  - Scheme is used; UCWISE is not.
  - CS3 topics are covered in the first 3-4 weeks
- One cool thing learnt: How to make a chat client in Scheme

CS61B: Data Structures

- Requires 61A (with a B- or higher)
- Wide coverage of dynamic data structures: queues, trees, arrays, strings, hash tables, etc.
- A *lot* of programming in this course
- Language used: Java
- One cool thing learnt: Final project is made from scratch; teaches beginning software engineering: design, coding, testing, debugging, and analysis.

Jon Kotker, jo_ko_berkeley@berkeley.edu

CS61C: Machine Structures

- Low-level programming (i.e., what happens when you read data from a disk drive, or hit a key on the keyboard)
- Learnt about machine architecture, how operating systems actually work, and even do a little low-level programming
- Language(s) used: C, MIPS, Verilog
- One cool thing learnt: Design your own processor!

Jon Kotker, jo_ko_berkeley@berkeley.edu

CS70: Discrete Math/Prob. Theory

- Requires Math 1B
- Mainly work with proofs, logic problems, and algorithms
- Touches on many different areas in CS, such as cryptography, networking efficiency, and search/sort algorithms
- No programming in this course, but weekly problem sets, a lot of proof and deduction problems
- One cool thing learnt: How to obtain information from corrupted packets

Jon Kotker, jo_ko_berkeley@berkeley.edu

THIS IS A COMPUTER

Jon Kotker, jo_ko_berkeley@berkeley.edu

THIS IS COMPUTER SCIENCE AT CAL

•ALGORITHM STHEORY **(CS70)**

•GAMES **(CS188)**
•GRAPHICS **(CS184)**
•SECURITY **(CS261)**

•USER INTERFACE **(CS160)**
•EDUCATION **(CS301)**

•MEMORY, PROCESSORS **(CS61C, CS150)**
•DATA STORAGE **(CS61B)**
•OPERATING SYSTEM **(CS162)**

Jon Kotker, jo_ko_berkeley@berkeley.edu

UPPER DIVISION COURSES (Descriptions from a Graduate)

CS150 (Digital Systems)

"Design and implement your own processor!"

CS152 (Computer Architecture)

"Parallel processing awesomeness!"

CS160 (User Interface Design)

"Design, create and determine usability of a theoretical application, like a Facebook app!"

CS161 (Security)

"Learn about viruses and cool hacks!"

CS162 (Operating Systems)

"Make your own functional operating system (NachOS)!"

Jon Kotker, jo_ko_berkeley@berkeley.edu

UPPER DIVISION COURSES (Descriptions from a Graduate)

CS164 (Compilers)

"Write something that actually runs the code people write!"

CS169 (Software Engineering)

"Make something – anything – that you think people might want to use! (We made something that tells you about local concerts in your area when you listen to those bands.)"

CS170 (Algorithms)

"Mathy awesomeness!"

CS184 (Graphics)

Demonstration from Ketrina Yim

CS186 (Databases)

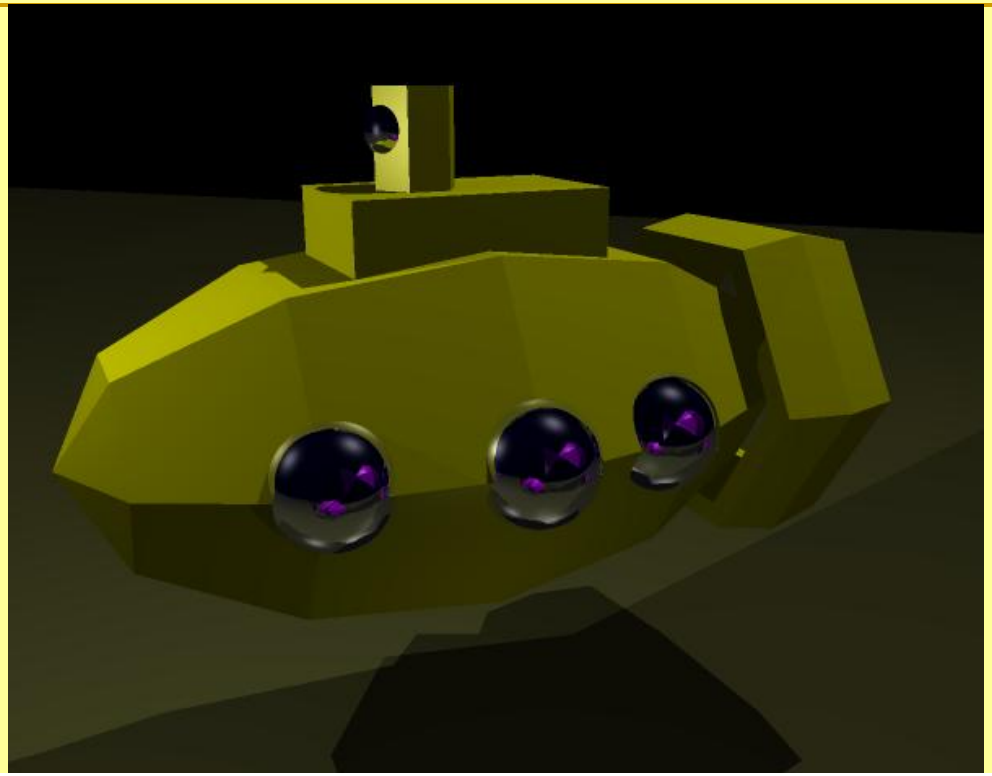
"Learn the super useful skill of working with databases!"

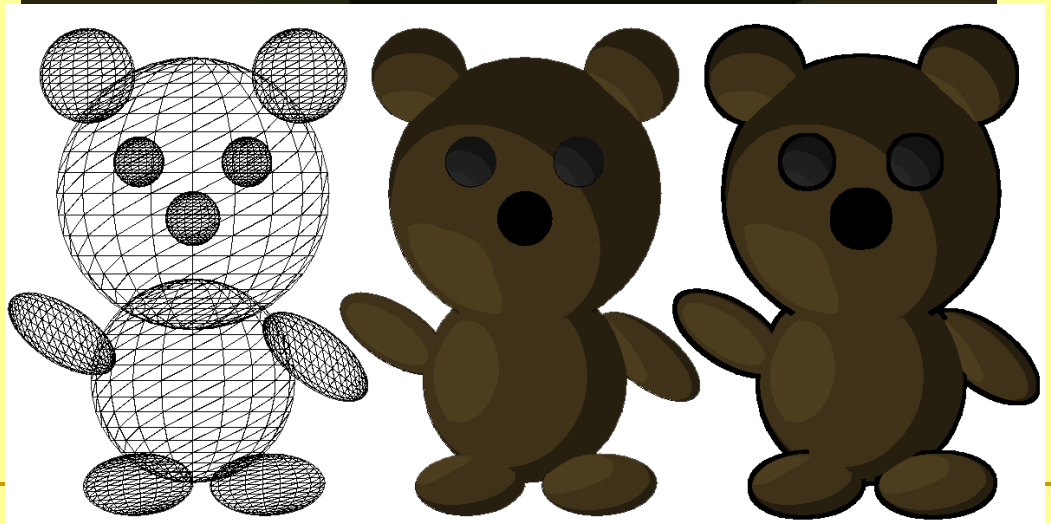
CS 188 (Artificial Intelligence)

Demonstration from Ramesh Sridharan

Jon Kotker, jo_ko_berkeley@berkeley.edu

CS184





Jon Kotker, jo_ko_berkeley@berkeley.edu

EECS OR CS?

EECS over CS

- I'm in EECS. What else do you need?
- More depth, more math; deals with actual physical entities, like circuits
- Can pursue EECS with emphasis on either EE or CS – great if you dislike programming, but love circuits or making Transformers
- Can focus on a lot of sub-fields

EECS OR CS?

CS over EECS

- CS is uncapped; easier to get into as a major
- CS allows more flexibility over choice of courses; allows pursuit of more non-CS courses
- Can double-major in another non-technical field
- Less math

TIPS FOR SUCCESS

- Don't fall behind.
  - CS is hard enough as it is.
  - Pay attention in lecture and/or read the book.
- Use your resources.
  - Engage your TAs.
  - Go to office hours.
  - Check out http://hkn.eecs.berkeley.edu for ratings.
- Form study groups.
  - Work gets done faster in groups. More importantly, it gets done more correctly.
  - (Plus, it's more fun with friends.)