

Programming Languages and Techniques (CIS120)

Lecture 1

January 11, 2012

Course Overview and Logistics
Introduction to Program Design

Introductions

- Instructor: Dr. Stephanie Weirich*
 - Levine Hall 510
 - sweirich@cis.upenn.edu
 - <http://www.cis.upenn.edu/~sweirich/>
- Course Staff:
 - Laura Fox – course administrative assistant – Levine 308
 - Grace, Lewis, Cory, Jeremy, Rafe, Harmony, Boris, Xiang, Andrew, Jessy, Max, Nick, Nathan, and Jerry – lab leaders (*CIS 120 spirit guides, student champions, and all around defenders of the universe*)

*Pronounced phonetically as: “why rick”. I won’t get upset if you mispronounce my name (really!). I will answer to anything remotely close, or, you can call me Stephanie or just Professor. Whatever you feel comfortable with.

Course website

- Web site: <http://www.cis.upenn.edu/~cis120>
- No required textbook. All materials online.
 - Lecture slides, lecture notes
 - Programming style guides
 - Homework assignments, submission, labs, exams
 - Supplementary resources
- Piazza: <http://piazza.com/class#spring2012/cis120>
 - Course discussion group for questions, clarifications, etc.
 - Allows *anonymous* posting, do not abuse

Prerequisites

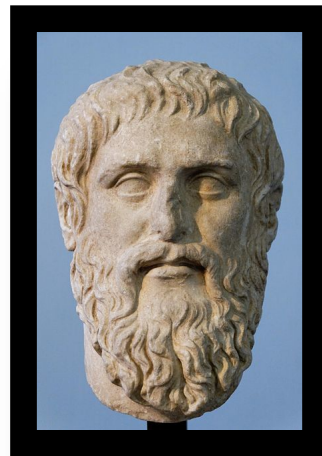
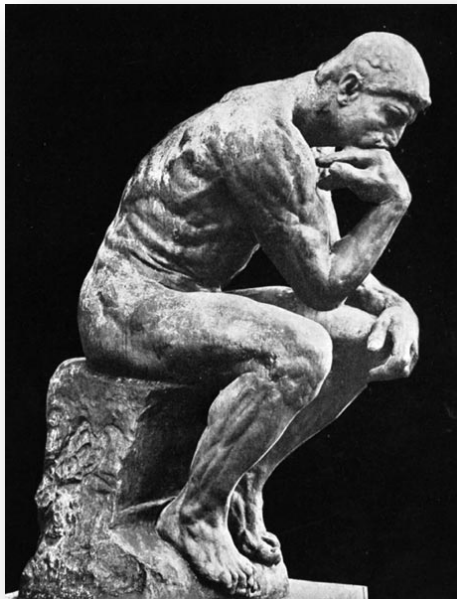
- We assume you can already write 10 to 100-line programs in some imperative or OO language
 - Not necessarily Java or any other particular language
 - CIS 110 or AP CS is typical
 - But you should be familiar with using a compiler, editing code, and running programs you have created
- CIS 110 is an alternative this course
 - If you have doubts, come talk to me to figure out the right course for you

Course Goals

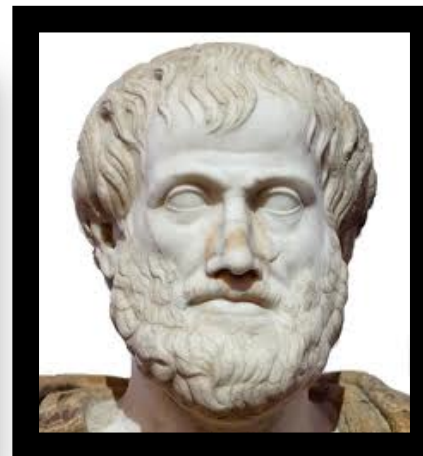
- Ability to write larger (~1000 lines) programs
 - increased independence ("working without a recipe")
- Fluency in program design
 - test-driven programming
 - modular decomposition
 - working with different programming idioms
- Fluency in core Java
- Firm grasp of principles of Computer Science

Philosophy

- Programming
 - ... is *fun, useful, and rewarding* in its own right
 - ... is also a conceptual foundation for all of computer science
 - ... involves tools that are often large and complex, for good reasons!
 - ... takes lots of practice to master



Plato



Aristotle



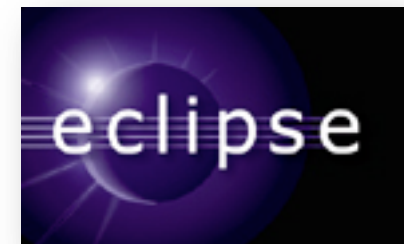
Al-Kwarizmi

Philosophy

- Programming
 - ... is *fun, useful, and rewarding* in its own right
 - ... is also a conceptual foundation for all of computer science
 - ... involves tools that are often large and complex, for good reasons!
 - ... takes lots of practice to master
- Teaching introductory computer science
 - Start with basic skills of “algorithmic thinking” (AP/110)
 - Develop more systematic design and analysis skills in the context of larger and more challenging problems (120)
 - Practice with industrial-strength tools and design processes (120, 121, and beyond)
- About CIS120
 - Teach Java *after* setting up the context needed to understand why Java and OO programming are good tools
 - Concentrate on the process of *designing* programs
 - Give a taste of the breadth and depth of CS

CIS 120 Tools

- OCaml
 - Mature functional programming language
 - Lightweight, approachable setting for learning about program design
 - Levels the playing field at the start of the course
- Java
 - Industry-standard OO language
 - Many tools/libraries/resources available
- Eclipse
 - Popular open-source integrated development environment (IDE)
- We'll help you get these set up on your laptops in lab today and tomorrow



Who uses OCaml?



facebook



JANE STREET

LexiFi



CITRIX

Microsoft

MLstate



my life

SimCorp



Why two languages?

- Pedagogic progression
- Perspective
- Disparity of background
- Confidence in learning new tools

“[The OCaml part of the class] was very essential to getting fundamental ideas of comp sci across. Without the second language it is easy to fall into routine and syntax lock where you don't really understand the bigger picture.”

---Anonymous CIS 120 Student

“[OCaml] made me better understand features of Java that seemed innate to programming, which were merely abstractions and assumptions that Java made. It made me a better Java programmer.”

--- Anonymous CIS 120 Student

Course Components

- Lectures
 - Presentation of ideas and concepts
 - Interactive demos
- Recitations (8% of final grade)
 - Practice and discussion in small group setting
 - Grade based on participation
 - **START TODAY**
- Homeworks (50% of final grade)
 - Practice, experience with tools
 - Exposure to broad ideas of computer science
 - Grade based on automated tests + style
- Exams (42% of final grade)
 - In class, pencil and paper
 - Do you understand the terminology? Can you model computation?
Can you synthesize solutions?

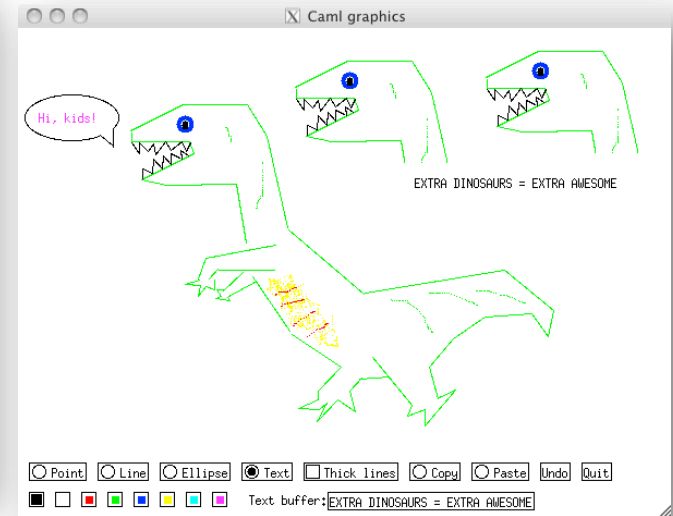
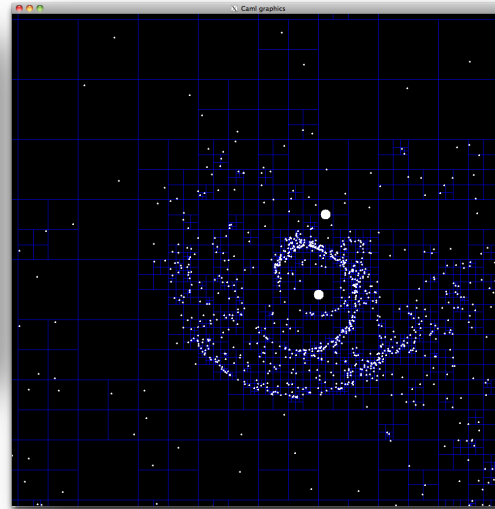
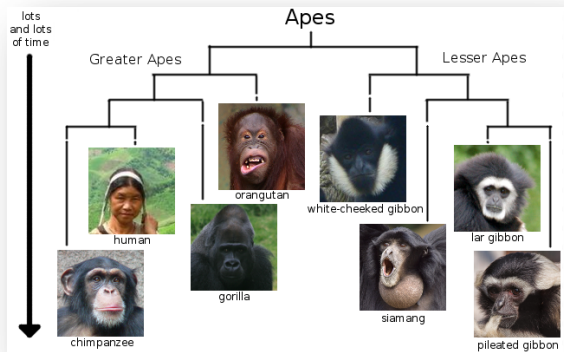
Lecture Policy

- Laptops *closed*... minds *open*
 - Although this is a computer science class, the use of laptops, cell phones, mobile devices, iPads, etc., in lecture is *prohibited!*
- Why?
 - Laptop users tend to surf/chat/e-mail/game/etc.
 - They also distract those around them
 - You will get plenty of time in front of your computers while working on the course projects
- Instead:
 - Pay attention, ask questions, and take notes!

Recitations

- First recitation, TODAY and TOMORROW
- Goals of first meeting:
 - Meet your TAs and classmates
 - Set up tools (OCaml, eclipse) your laptops
 - Get to see a bit of OCaml
- If you need to register/switch recitation sections, contact me (sweirich@cis.upenn.edu)

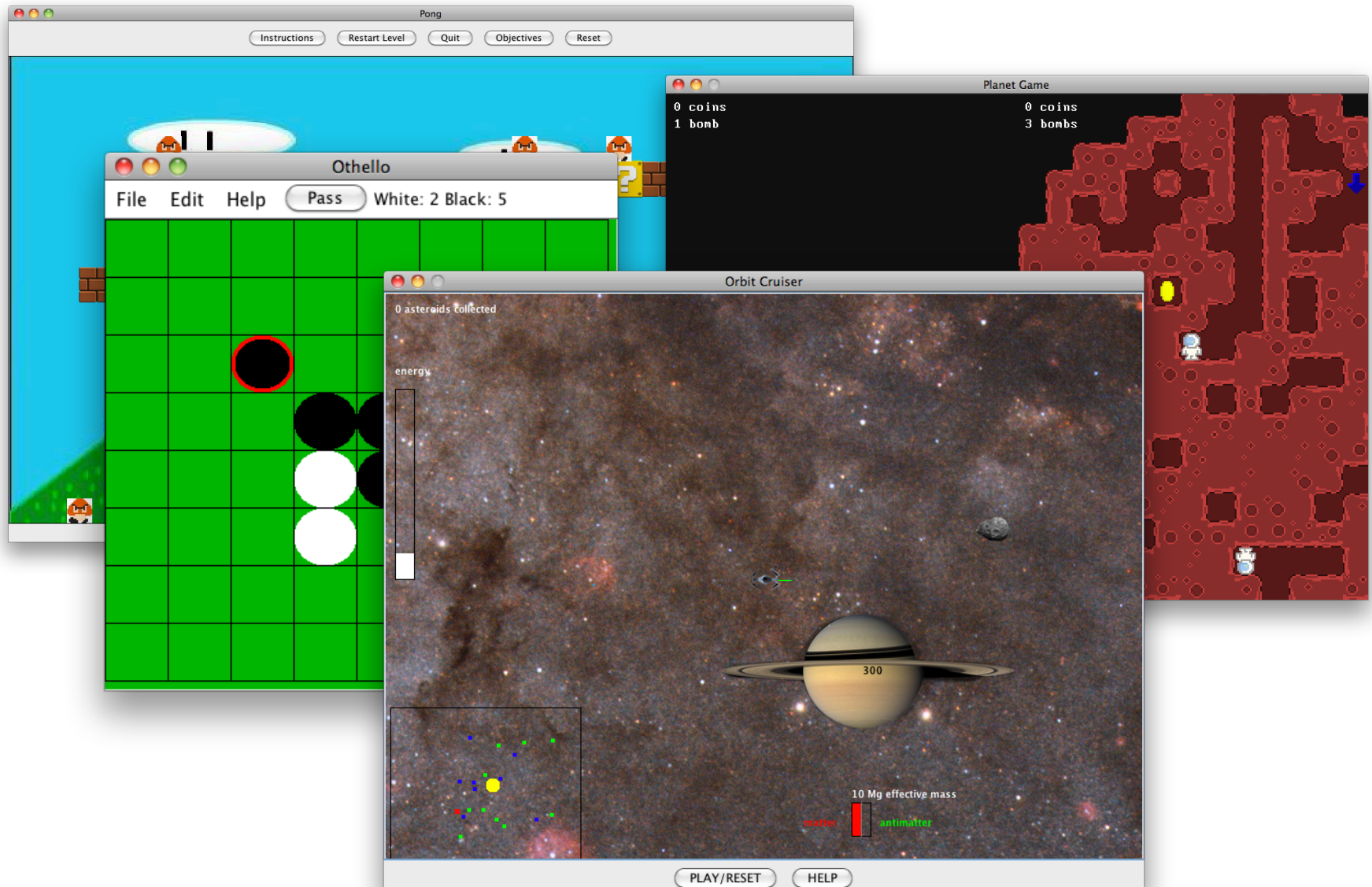
(Selected) CIS 120 Homeworks



```
Welcome to the Adventure Game.  
Type "help" at any time to get a list of available commands.  
  
You are in the ballroom.  
There are exits to the south and east.  
You have 10 health and 7 coins.  
>>>
```

10 homework assignments total,
weighted equally

Final projects



Homework Policies

- Projects will be (mostly) automatically graded
 - We'll give you some tests, as part of the assignment
 - You'll write your own tests to supplement these
 - Our grading script will apply additional tests
 - Your score is based on how many of these you pass
 - Your code must compile to get *any* credit
- You will be given your score (on the automatically graded portion of the assignment) immediately
- Multiple submissions *are allowed*
 - First *few* submissions: no penalty
 - Each submission after the first few will be penalized
 - Your final grade is determined by the *best* raw score
- Late submissions
 - 10 point penalty if less than 24 hours late
 - 20 point penalty if 24-48 hours late
 - *Submissions not accepted after 48 hours past the deadline*

Exam Dates

- 12% First midterm: Wed 15 Feb, in class
- 12% Second midterm: Fri 30 Mar, in class
- 18% Final exam: Tue 8 May, 9AM (UGH!)

Academic Integrity

- All submitted homework must be individual work

Not OK:

- Copying / sharing of code
- Discussions of specific homework problems with other students

OK / encouraged:

- “High level” discussions of concepts from lecture

- Course staff will check for copying. Violations will be treated seriously.
- *If in doubt, ask us!*

Penn's code of academic integrity:
<http://www.vpul.upenn.edu/osl/acadint.html>

Where to ask questions

- Course material
 - Piazza Discussion Boards
 - TA email (tas120@lists.seas.upenn.edu)
 - TA office hours, schedule on webpage
 - Tutoring, Sunday and Monday evenings
 - Prof office hours: Mon. 3:30-5:00pm, or by appt
Today: 3:30 – 5 PM
- HW/Exam Grading: see webpage
- About the CIS majors
 - Ms. Jackie Caliman, CIS Undergraduate coordinator
- Otherwise: email professor (sweirich@cis.upenn.edu)

Program Design

Design

Design is the process of translating informal specifications (“word problems”) into running code.

1. **Understand the problem**
What are the relevant concepts and how do they relate?
2. **Formalize the interface**
How should the program interact with its environment?
3. **Write test cases**
How does the program behave on typical inputs? On unusual ones? On erroneous ones?
4. **Implement the required behavior**
Often by decomposing the problem into simpler ones and applying the same recipe to each

A design problem

Imagine the owner of a movie theater who has complete freedom in setting ticket prices. The more he charges, the fewer people can afford tickets. In a recent experiment the owner determined a precise relationship between the price of a ticket and average attendance. At a price of \$5.00 per ticket, 120 people attend a performance. Decreasing the price by a dime (\$.10) increases attendance by 15. Unfortunately, the increased attendance also comes at an increased cost. Every performance costs the owner \$180. Each attendee costs another four cents (\$0.04). The owner would like to know the exact relationship between profit and ticket price so that he can determine the price at which he can make the highest profit.