Computer Science Principles

CSE 120 Winter 2017

Instructor:

Justin Hsia

Teaching Assistants:

Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

Technology trends changing the world as we look ahead

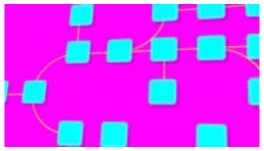
"There's an age old saying that humans tend to overestimate what can be accomplished in one day, but underestimate what can be accomplished in one year. As 2016 comes to a close, it is a good time to zoom out the lens, and get reflective on what has happened this year, and predictive about what we are excited about for the coming 3-5 years."

- https://techcrunch.com/gallery/technology-trends-changing-the-world-as-we-look-ahead/

Commercial Drones

Intelligent Apps

Virtual Assistants

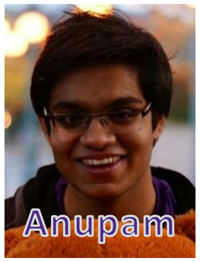

Blockchain (currency transfers)

Autonomous Vehicles

VR / AR

Who: Course Staff

- Your Instructor: just call me Justin
  - From California (UC Berkeley and the Bay Area)
  - I like: teaching, the outdoors, board games, and ultimate
  - Excited to be teaching CSP for the 1st time at UW!
- 4 TAs:



  - Available during lab, in office hours, and on Piazza
  - An invaluable source of information and help
- Get to know us
  - We are here to help you succeed
  - And to make the course better – with your help

Who: You!

- 52 students registered
  - Undergrads from *many* different majors
- This class is intended for students without significant previous experience with computing/programming
- Get to know each other and help each other out!
  - Learning is much more fun with friends
  - Working well with others is a valuable life skill
  - Diversity of perspectives expands your horizons
- Submit Introduction Survey so we can find out more

Why Study Computer Science?

- Increasingly useful for *all* fields of study and areas of employment
  - Art – computer-aided design, animation
  - Drama – lighting, sound, ticket sales, advertising
  - Lumberjacking – mapping, tracking size & # of forests

- Massive impact on our lives and society as a whole

Amazon

Commercial Drones

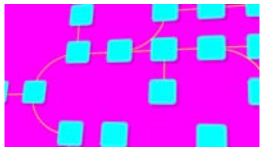
Fit bit
Google Calendar

Intelligent Apps

Siri
Cortana
Alexa

Virtual Assistants

Bitcoin


Blockchain (currency transfers)

Google
Tesla

Autonomous Vehicles

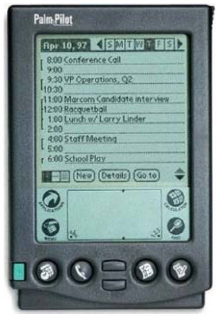
Pokemon Go
Occulus Rift

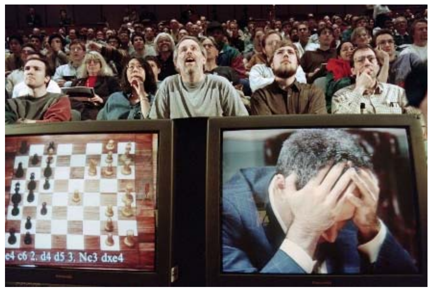
VR / AR

The Hottest Tech 20 Years Ago (1997)

- Sharp MiniDisc Player
- Sony PlayStation
- Grand Theft Auto
- WebTV
- Palm Pilot 1000
- DVD Players

- Al Gore: "Atari Democrat"
- Deep Blue
- Sony Mavica MVC-FD5
- Motorola StarTAC
- Windows 95
- MP3s





https://www.cnet.com/pictures/this-was-the-hottest-tech-20-years-ago-in-1997/

Computing in Your Future

- Computing and its data are inescapable
  - You generate "digital footprints" all the time
- Computing is a regular part of *every* job
  - Use computers and computational tools
  - Generate and process data
  - Dealing with IT people
  - Understanding the computation portion of projects
- Our goal is to help you make sense of the "Digital Age" that we now all live in

What This Course Is

- This course is split into two major themes:

1) Computational Thinking
   - How can you use computers to solve problems
   - Using programming as a tool

2) Computational Principles
   - The “big ideas of computing” that we think everyone should know
   - *e.g.* bits can represent anything and everything, what a computer can and can’t compute, how do websites and the Internet work, social implications of computing

What This Course Is NOT

- Preparation for CSE142: Computer Programming I
  - This is not just a programming course
  - But great if you feel motivated to continue afterward!
- Trivial
  - Supposed to be material you haven't seen before
  - A technical class that asks you to read and write and be creative
- Boring or back-breaking
  - Assignments intended to be fun, interesting, and reasonable

About Programming

- programming ≠ computational thinking
  - *Computational thinking* is knowing how to break down and solve a problem in a way that a computer can do it
  - *Programming* is the tool you use to execute your solution
  - We use programming in this course as a way of teaching computational thinking
- Can be learned, just like any other skill
  - It’s not black magic; there’s no such thing as a “coding gene”
  - Yes, at first it may be challenging and mind-bending – just like learning your first non-native language
  - My hope is that you will think differently after this course

Programming in CSE120

- Use a language called Processing
  - Text-based language that is good for visuals and interaction
  - We will use Java syntax
  - At the end of the day, the language you use doesn't matter as long as you develop computational thinking skills

- Examples:
  - Jumping robot
  - Ripples
  - Constellations

Big Ideas of Computing

- Exposure to a broad range of topics in computer science
  - Not going to dive into the details
  - These are the motivations & the applications for programming (the tool)
  - Focus on what to be aware of to navigate the digital world

- **Goal:** become "literate" in computing
  - As new innovations arise, can you read about it, understand its consequences, and form your own opinion?
  - This course will ask you to *read*, *discuss*, and *write* about computing

Lecture Outline

- Course Introduction
- **Course Policies**
  - http://courses.cs.washington.edu/courses/cse120/17sp/policies.php
- Abstraction

Communication

- Website: http://cs.uw.edu/120
  - Calendar, schedule, policies, labs, links, assignments, etc.
  - Grade book and assignment submissions via Canvas

- Discussion: http://piazza.com/washington/spring2017/cse120
  - Ask and answer questions – staff will monitor and contribute
  - *ALL* questions on course material should go here

- Office Hours: spread throughout the week
  - Can also email to make individual appointments

- Anonymous feedback form

Weekly Schedule

- Lectures are Mon, Wed, Fri (3 hr)
  - Friday lectures will generally be reserved for "Big Ideas"
- Weekly reading is due before lab on Thursday
  - All readings online, complete "reading check" to prep
- Labs on Tue, Thu (2 hr)
  - Work time for labs & assignments w/help from TAs
  - 15-20 minutes at start of Thu lab will be spent discussing the weekly reading
- Can be a demanding schedule, but should be fun!

Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Weekends**
Lecture	Lab	Lecture	Lab (Reading)	Lecture	Assignments

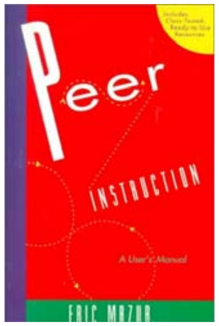
Course Components and Grading

- **Programming Assignments** (40% total)
  - Labs and Assignments (mostly Processing)
  - Website portfolio
  - Mid-course "Creativity Assignment" (2 mini-projects)
- **Final Project** (20%)
  - Use your newfound skills to make a project *of your choosing!*
- **Written Assignments** (15% total)
  - Reading Checks
  - Living Computer Museum Report ($)
  - Innovation Blog
- **Exams:** Midterm (10%) and Final (10%)
  - Double-check understanding of concepts and big ideas
- **EPA:** Effort, Participation, and Altruism (5%)

EPA

- Encourage class-wide learning!
- Effort
  - Attending labs and office hours, completing all assignments
  - Keeping up with Piazza activity
- Participation
  - Making the class more interactive by asking questions in lecture, office hours, and on Piazza
  - Peer instruction voting
- Altruism
  - Helping others in lab, during office hours, and on Piazza

Peer Instruction



- Increase real-time learning in lecture, test your understanding, increase student interactions
  - Lots of research supports its effectiveness
- Multiple choice question at end of lecture "segment"
  - 1 minute to decide on your own
  - 2 minutes in pairs to reach consensus
  - Learn through discussion

- Vote using Poll Everywhere
  - Use website (https://www.polleverywhere.com) or app
  - Linked to your UWNetID

Peer Instruction Question

- Which of the following statements is FALSE?
  - Vote at http://PollEv.com/justinh

A. The weekly readings are intended to prepare you for the "big ideas" lecture on Friday readings are on the Friday lecture topic

B. Your participation both in person and online count towards your class grade lecture lab, OH Piazza

C. Effective communication is an important skill for this class writing and discussing!

$\boxed{\text{D.}}$ The two major themes for this class are ~~programming~~ and computational principles computational thinking

How to Get an A (I Promise!)

- Attend class everyday
- Complete your assignments on time
- Reach out to us if you ever feel stuck or overwhelmed
- If you miss ANY deadline, don't ignore it—come talk to us and tell us what is going on
- **Persistence is important**: a lot of things will seem new and confusing at first, but you can figure them out – stick with it and don't give up!
    - You learn best from your mistakes

Make a good-faith effort to *try everything* and *think* about what you do!

What to Expect From Us

- We will put forth a good faith effort to present the material in the clearest possible way
- We will teach topics that are interesting and enjoyable – if it's not working for you, let us know
- We will be respectful, cooperative, and understanding
- We will provide help whenever you ask, both online and 1-on-1

What We Expect From You

- Come to every class ready to learn
- Make a sincere effort to understand the material
- Do a little bit of work for this class each day
- Turn in work on time, and communicate with us if you have special circumstances that require extensions
- Submit your own work… please don't cheat ☹

- Be respectful to us and other students
  - Everyone has different past experiences and learns at their own pace

Hooked on Gadgets

- Gadgets reduce focus and learning
  - Bursts of info (*e.g.* emails, IMs, etc.) are *addictive*
  - Heavy multitaskers have more trouble focusing and shutting out irrelevant information
    - http://www.npr.org/2016/04/17/474525392/attention-students-put-your-laptops-away
  - This applies to all aspects of life, not just lecture
- NO audio allowed (mute phones & computers)
- Non-disruptive use okay
  - Stick to side and back seats
  - Stop/move if asked by fellow student

To-Do List

- Explore website thoroughly: http://cs.uw.edu/120
  - Read through the full course policies!!!
- Check that you are registered on Piazza, Canvas, and Poll Everywhere
- Upcoming assignments:
  - Introduction Survey (3/28)
  - Personal Values (4/2)

  available now!

  - Website setup (3/31)
  - Exploring Lightbot (3/29)

  lab tomorrow

Lecture Outline

- Course Introduction
- Course Policies
  - http://courses.cs.washington.edu/courses/cse120/17sp/policies.php
- **Abstraction**

abstract art
↳ not depict anything that's real

abstract description
↳ "levels" of concepts/details

paper abstract
↳ summary, main points

Complexity and Abstraction

- Programming is straightforward, as long as your programs are small
  - *Complexity* is our enemy
  - *Abstraction* is the key to conquering complexity
- Abstraction allows us to build general-purpose artifacts
  - **Detail Removal:** Hide unnecessary details from users and designers
  - **Generalization:** Avoid unnecessary repetitive work
- Learning to reason using the most appropriate abstraction is a key goal of computational thinking

Abstraction: Detail Removal

- "The act or process of leaving out of consideration one or more properties of a complex object so as to attend to others."

missing person's features

missing trees, buildings

missing unnecessary small streets

Henri Matisse *"Naked Blue IV"*

San Francisco

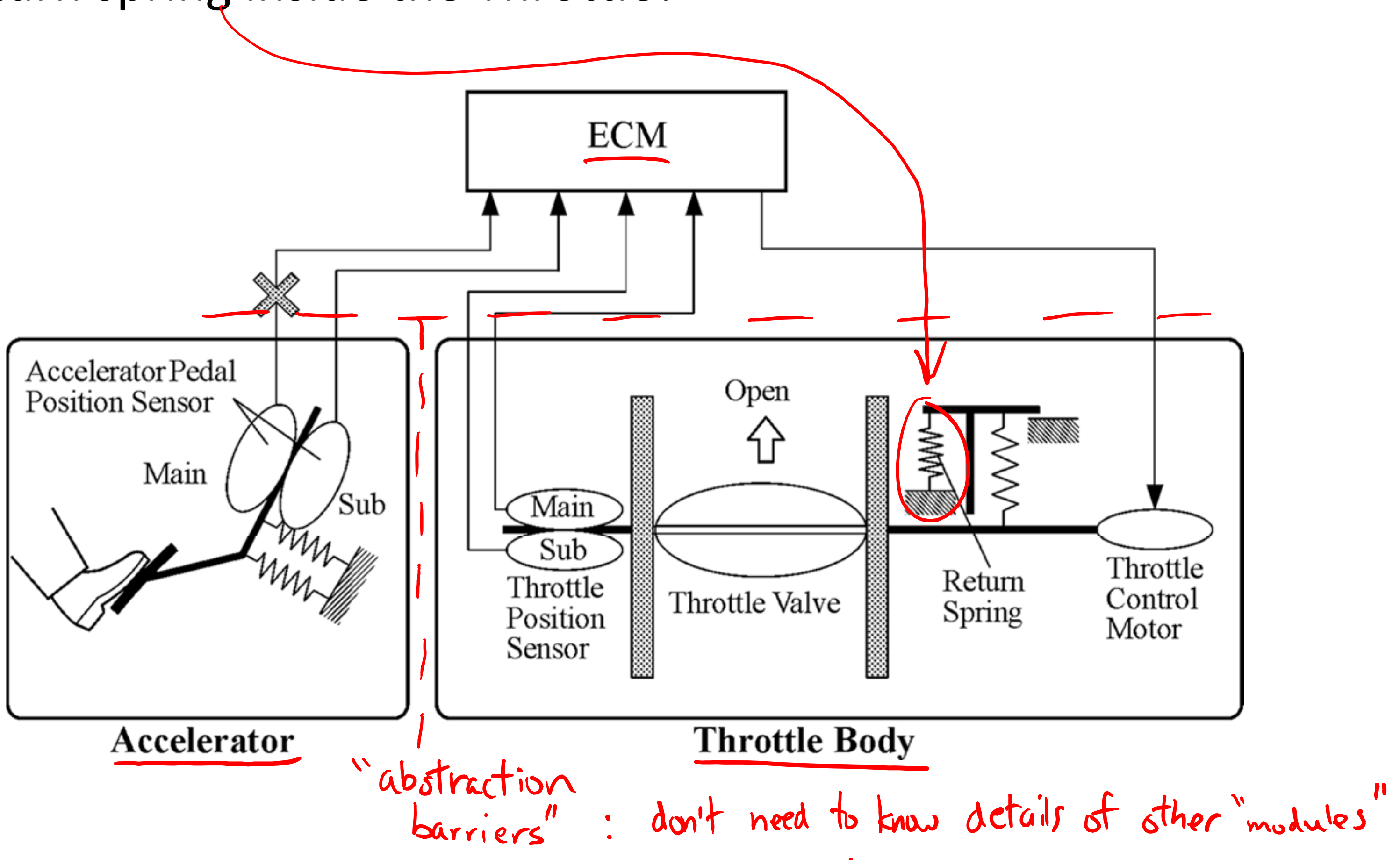
Maps for directions

Abstraction: Detail Removal

- Detail removal example:
  - Modern user interface: Right pedal is "accelerate", left is "decelerate"
  - Even as underlying technology has changed, this abstraction has not!
    - Computer controlled fuel injection
    - Anti-lock brakes (ABS)

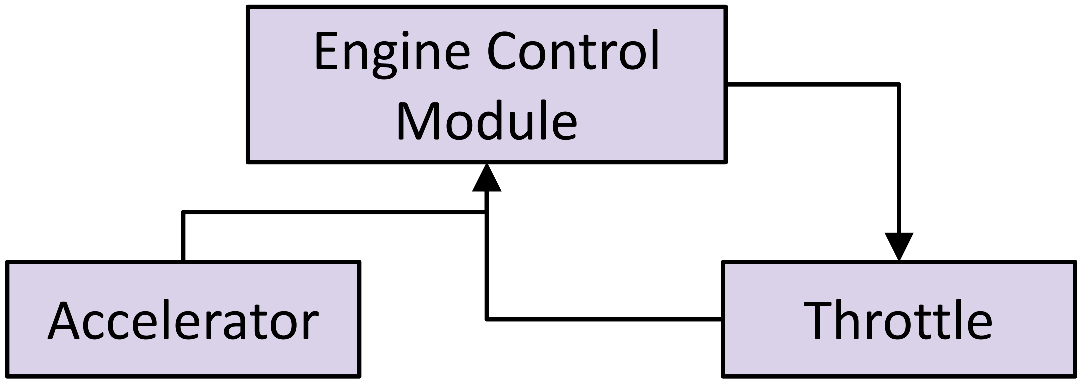
Abstraction: Detail Removal

- Detail removal example:
  - Hide unnecessary details from other designers
    - *e.g.* Engine Control Module (ECM) designer doesn't care about the return spring inside the Throttle!



Abstraction: Detail Removal

- Detail removal example:
  - Hide unnecessary details from other designers
    - *e.g.* Engine Control Module (ECM) designer doesn't care about the return spring inside the Throttle!
  - Nice to be able to think of a system as a hierarchy of well defined "chunks" with precise functionality
    - In CS, we say that we have a **separation of concerns**



Abstraction: Generalization

- "The process of formulating general concepts by abstracting common properties of instances."

- Extensible shower rods
- Adjustable hats
- Single recipe for <fruit> cheesecake
- Feeding animals on a farm
  - To feed <animal>, put <animal> food in <animal> dish

Summary

- Abstraction is one of the most important challenges in computer science
  - How do you identify the right abstraction you need (block to build) to solve your problem?

- Think about computers:
  - How many of you actually know how a computer works?
  - How many of you can use a computer?
    - Thanks to abstraction!!!