

Computer Science Principles

CSE 120, Winter 2020

Instructor: **Teaching Assistants:**

Sam Wolfson Yae Kubota Eunia Lee Erika Wolfe

The biggest tech trends of 2020, according to top experts

In 2020, technologies will move toward the mainstream and begin impacting daily life. The next generation of wireless network, 5G, will begin to take hold, for example, and may work as a catalyst for other things like smart cities and smarter mobile and wearable devices. Augmented reality eyewear, which places digital content in the context of the real world, will begin to appear, and may use fast 5G connections to the cloud to identify people and things for us. The role of AI will increase in business, and the public will become more aware of it.

Next year's tech will appear in the context of a turbulent political scene and perhaps the biggest election in U.S. history, a warming planet, an inefficient healthcare system, and a growing skepticism that tech companies will do no evil. Tech companies (and their investors) will be more aware of the public's expectation that new products solve real, non-trivial, problems

<https://www.fastcompany.com/90374432/here-are-the-top-tech-trends-of-2020-according-to-top-experts>

Who: Course Staff



- ❖ Your Instructor: Sam
 - From Seattle
 - I like: teaching, skiing, hardware hacking
 - I've been a TA for this course twice, and last summer I taught a different CSE course here at UW.



- ❖ TAs:
 - Available during lab, in office hours, and on Piazza
 - An invaluable source of information and assistance
- ❖ Get to know us
 - We are here to help you succeed!

Who: You!

- ❖ 25 students registered
 - Undergrads from *many* different majors (or pre-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
- ❖ Please submit the Pre-Course Survey so we can learn more about you!

Why Study Computer Science?

- ❖ Massive impact on our lives and society as a whole
- ❖ Increasingly useful for *all* fields of study and areas of employment

. painting
. Amish farmer

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 and software people
 - Understanding the computational portion of projects
- ❖ Our goal is to help you make sense of the “Digital Age” that we now all live in

Computing and Society

- ❖ Thumbs up/down if you or someone you know:
 - Work primarily from home
 - Have “stalked” someone online
 - Communicate mostly using 🐾👶 instead of words
 - Have taken extra trips outside to catch that Pokémon (does anyone still play that?)
 - Get the majority of your news from social media (yes, this includes Reddit)
 - Have had their identity or credit card number stolen online
 - Has seen a parent quiet a child by giving them a digital device

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 the Internet works, social implications of computing

What This Course Is NOT

- ❖ Preparation for CSE 142: Computer Programming I
 - This is not *just* a programming course
 - We'll introduce you to the concepts, but we will not expect you to master them
 - 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 and exhausting
 - Intended to be fun, interesting, and reasonable

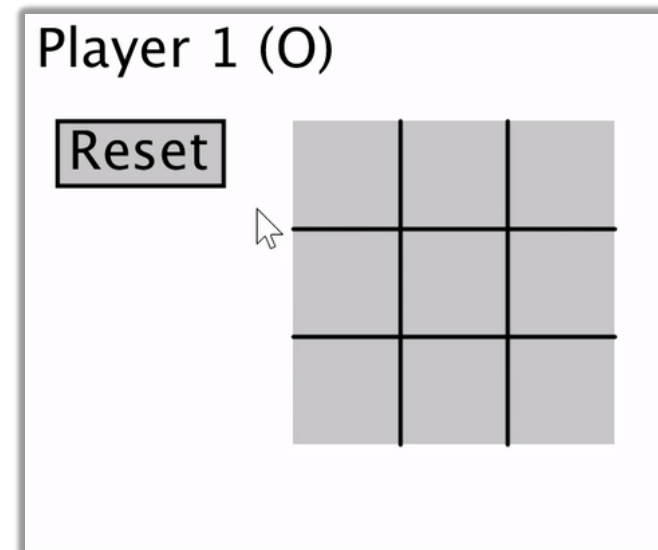
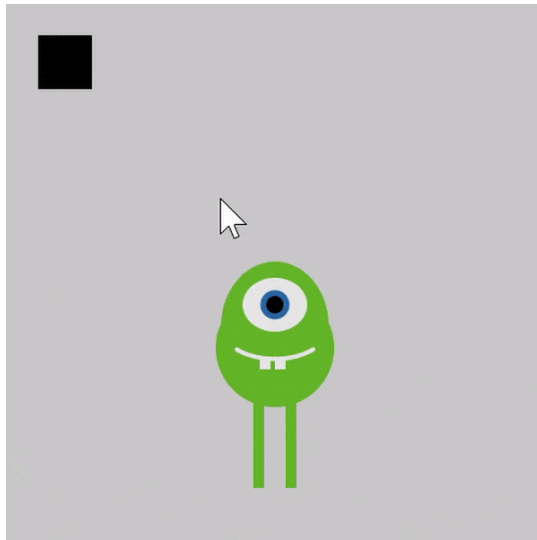
About Programming

- ❖ **programming \neq 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
 - 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 CSE 120

- ❖ We'll 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



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 why it's worth it to learn 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, present, and write* about computing innovations

Lecture Outline

- ❖ Course Introduction
- ❖ **Course Syllabus & Policies**
 - <https://courses.cs.washington.edu/courses/cse120/20wi/syllabus>
- ❖ Abstraction

Communication

- ❖ Website: <http://cs.uw.edu/120>
 - Calendar, schedule, policies, links, assignments, etc.
 - Grade book and assignment submissions via Canvas
 - Note: calendar uses 24-hour time

- ❖ Discussion:
<http://piazza.com/washington/winter2020/cse120/>
 - Ask and answer questions – staff will monitor & 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 3:30 pm on Thursday
 - All readings online, complete “reading check” to prep
- ❖ Labs on Tue, Thu (3 hr)
 - Worksheets and work time with help from TAs
 - First half hour or so of Thursday lab will be spent doing group presentations on the readings
- ❖ Can be a demanding schedule but should be fun! 😊

Monday	Tuesday	Wednesday	Thursday	Friday
Lecture	Lab	Lecture	Lab (Reading)	Lecture

Technology & Gadgets in 120

- ❖ 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

- ❖ We will, however, require you to use computers in sections and some lectures
 - Be disciplined in your usage; don't distract others
 - Lectures for which you'll need a computer are marked with  on the course calendar

Course Components and Grading

- ❖ **Programming Assignments (35%)**
 - Includes a website portfolio of your work
- ❖ **Final Project (20%)**
 - Of your own choosing!
- ❖ **Written Assignments (15%)**
 - Reading checks, field trip report, mini research report
- ❖ **In-Lecture Quizzes (20%)**
 - Every other Friday
 - Double-check your understanding of concepts
- ❖ **EPA: Effort, Participation, and Altruism (10%)**
 - Encourage class-wide learning

To-Do List


- ❖ Bookmark and explore website thoroughly:
<http://cs.uw.edu/120>
 - Read through the full course policies!!!
- ❖ Check that you are registered on **Piazza** and **Canvas**
- ❖ Pre-Course (Introduction) Survey due tomorrow (1/7)
 - More assignments will be introduced in lab tomorrow
- ❖ First reading check due Thursday (1/9)

Lecture Outline

- ❖ Course Introduction
- ❖ Course Policies
- ❖ **Abstraction**

- not physical
- abstract - short summary

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
- 

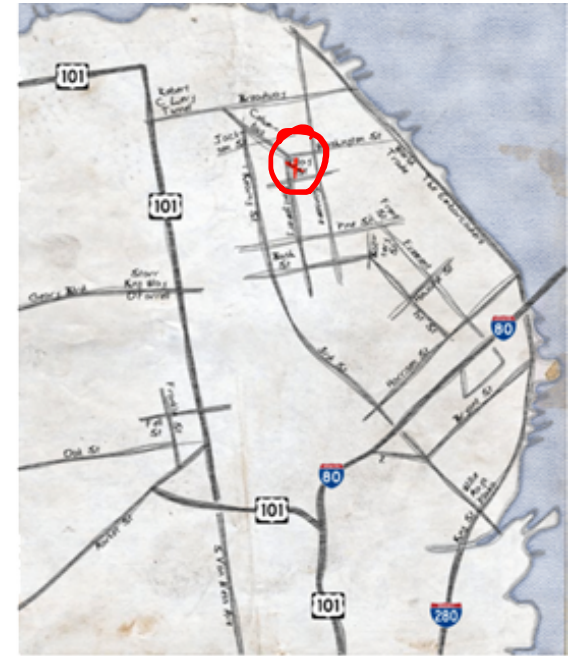
Abstraction: Detail Removal

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

person



Henri Matisse "Naked Blue IV"



Maps for directions

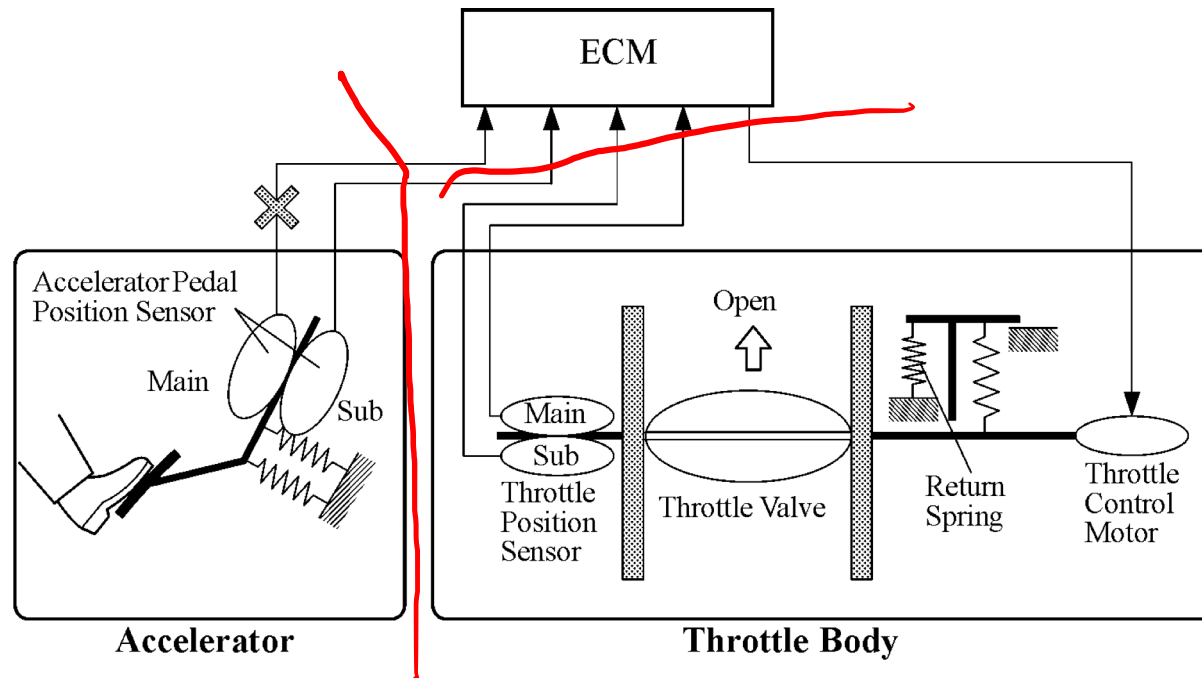
Abstraction: Detail Removal

- ❖ 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)
 - Electric cars



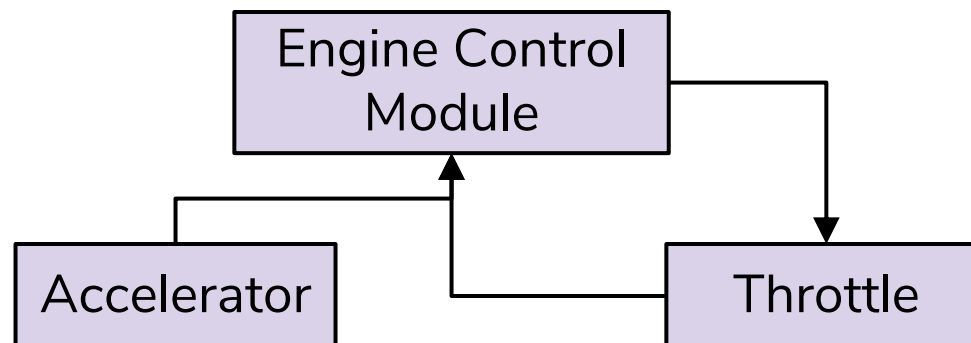
Abstraction: Detail Removal

- ❖ 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 and belts
- ❖ Single recipe for <fruit> cheesecake
- ❖ Feeding animals on a farm
 - To feed <animal>, put <animal> food in <animal> dish
- ❖ Your brain makes generalizations every day, without you even consciously realizing it!

Audience Responses

- ❖ Other examples of **detail removal**:

onesie - less clothing to deal with

IKEA furniture

- ❖ Other examples of **generalization**:

clothing comes in S/M/L

categorize food groups

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?
 - I certainly don't understand every detail (I don't know if anyone does...)
 - How many of you can use a computer?
 - Thanks to abstraction!!!