Course Wrap-Up

CSE 120 Spring 2017

Instructor: Justin Hsia

Teaching Assistants: Anupam Gupta, Braydon Hall, Eugene Oh, Savanna Yee

Administrivia

- Assignments:
  - Project – Code and Video (6/2)
  - CS in Your Future (6/4)
- Final Exam – Thursday 6/8, 9am in SIG 134
  - Scheduled slot is 8:30-10:20, but exam will be 60 minutes
  - Similar to midterm – big ideas & programming sections
  - 10% of course grade
  - TWO sheets of notes (letter, double-sided, hand-written)
- Course Evaluation: https://uw.iasystem.org/survey/178422

Outline

- **What We've Learned**
- Lecture 1 Revisited
- Your Future Beyond CSE120

Source: DragoArt.com

Source: Project Gutenberg

Computational Thinking

- It's all about problem solving
  - How to attack your problem in a way that a computer can help
- Most important idea: abstraction!
  - Detail removal and generalization help us decompose complex problems
  - Use bits to represent *everything* (*i.e.* digitization)
  - Reuse and combine building blocks (algorithms) in ways that hopefully scale well

Building Blocks of Algorithms

- **Sequencing**
  - The application/execution of each step of an algorithm in the order given

```
fill(255);
rectMode(CORNERS);
rect(-r, -r, 0, r);
ellipse(0, -r/2, r, r);
```

- **Selection**
  - Use of conditional to select which instruction to execute next

```
if(mousePressed) {
  fill(0,0,255);
}
```

- **Iteration**
  - Repeat part of algorithm a specified number of times

```
for(int i=20; i<400; i=i+60) {
  line(i,40,i+60,80);
}
```

- **Recursion**
  - Algorithm calls itself to help solve the problem on smaller parts

Programming

- Learned our first programming language
  - Processing (Java syntax)
- Iterative design cycle:
  - The value of a precise specification
  - Design, prototype, implement, and evaluate
  - Testing and debugging
- Coding style and documentation
  - Proper commenting and formatting are essential for maintenance and collaboration

Some Big Ideas

- Computers can only do a small number of things
  - Execute *exactly* what you tell it to
- Computing has physical and theoretical limits
- The Internet is a physical realm
- Data is constantly generated, stored, and analyzed
  - And can be copied and distributed
- Machines can "think" and "learn"?
  - AI & the importance of probability and training sets

Social Context and Impact

- History of computing:
  - Rise of the Internet and access to information
  - Current boom in CS and computing education
- Impacts of computing:
  - Algorithms can have unintended consequences
  - Privacy and security (or lack thereof)
  - Social media influences the way we think and act
  - Automation and the future of labor
- Design matters!
  - Must keep in mind users and user interface

Outline

- What We've Learned
- **Lecture 1 Revisited**
- Your Future Beyond CSE120

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

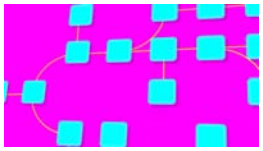
Fitbit
Google Calendar

Intelligent Apps

Siri
Cortana
Alexa

Virtual Assistants

Bitcoin



Blockchain (currency transfers)

Google
Tesla

Autonomous Vehicles

Pokemon Go
Occulus Rift

VR / AR

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

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

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

Outline

- What We've Learned
- Lecture 1 Revisited
- **Your Future Beyond CSE120**

Giving Back to CSE120

- Enjoyed the class? Lots of ways to help out!
  - Feedback: course eval, CS in Your Future, talk to me in OH or via email
  - Examples: Permission to show your work to future classes?
  - Recommendations: CSE120 next offered in Wi18 – tell your friends!

More CS at UW

- CSE 142 + CSE 143: Computer Programming I/II
  - Needed for declaring CS major
- CSE 160: Data Programming
  - Recommended to take 142 first
- CSE 154: Web Programming
  - Must have taken 142, 143, or 160
- CSE/STAT/INFO 110: Intro to Data Science (Wi18)
  - More forthcoming (http://escience.washington.edu/education/undergraduate/)

Social Implications Courses

- Informatics
  - INFO 101: Social Networking Technologies
  - INFO 102: Gender and Information Technology
  - INFO 200: Intellectual Foundations of Informatics
- Human Centered Design & Engineering
  - HCDE 210: Explorations in Human Centered Design
- Sociology
  - SOC 201: Data and Society (Au17)

No More CS at UW or Break

- You are now relatively programming-literate
  - Can automate tasks to make your life easier
  - More aware of possibilities of computing
  - Easier to interact with IT/CS staff at work

- Figure out what will be most useful to you
  - Some languages specific to type of work (*e.g.* R, MATLAB, Ruby on Rails, SQL)
  - Learn on your own via the Internet:

Making the Most of College

- Seek out experiences that lead to new experiences (*i.e.* that pay dividends)
  - Build skills, interests, relationships
  - Meet new people, join interesting clubs, go on adventures
- Don't go it alone – find a friend group for classes
- Take advantage of educational opportunities
  - **Research:** https://www.washington.edu/undergradresearch/students/find/
  - **Student Groups:** ACM, Animation Research Labs, Husky Robotics, WOOF3D, etc.
  - **Classes:** non-major courses, P.E., languages, anything of interest
- Take care of yourself!

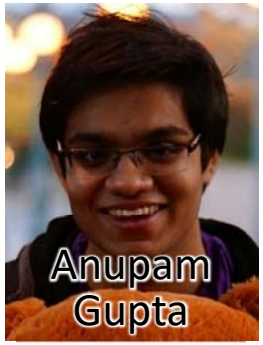
Making the Most of Our Future

- Computing is resurfacing our world
  - Now almost everyone has access to everything, always
  - New technology affects privacy, jobs, safety, beliefs, etc.

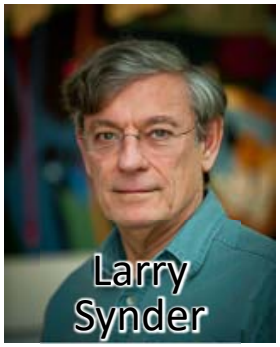
- You now know the most important parts of how it all works!
  - Can bring computing to new fields/jobs/areas
  - Keep these considerations in mind as you use and/or build things

Thanks for a great quarter!

- Huge thanks to your awesome TAs!



- Thanks to course content creators:



- Best of luck in the future!

Ask Me Anything (AMA)


That's all Folks!