Processes

CSE 351 Winter 2021

Instructor:

~~Sam Wolfson~~ Mara Kirdani-Ryan

Teaching Assistants:

Angela Xu
Anirudh Kumar
Catherine Guevara
Dara Stotland
Harrison Bay
Ian Hsiao
Kevin Wang
Mara Kirdani-Ryan
Nick Durand
Sanjana Sridhar

Gentle, Loving Reminders

- Lab 4's due in 10 days!
  - Give some time for thoughts to ferment!
- No lecture on Monday!
  - Maybe find some way for it to feel like a holiday?
  - One of the regrets I hold is working too much in undergrad!
  - I would've saved some time by learning more about me :)

A note on this lecture, and others like it

- I like to do boundary-pushing research!
  - i.e. work that explores radical new territory, without much guidance from prior work
- Necessarily, this means that it's a bit messy!
  - With relatively little prior work to draw from, there's only so much that I can predict
- This'll be better than over the summer, but it won't be perfect!
  - No offhanded references to *neoliberalism* this time :)
  - Please, share feedback! Let me know how this could improve!

Caches, Efficiency, and Normativity

AMAT, Revisited

- *Average Memory Access Time* (AMAT): average time to access memory considering both hits and misses

  AMAT = Hit time + Miss rate × Miss penalty

  (abbreviated AMAT = HT + MR × MP)

- We called this a *cache performance* ***metric***
  - This isn't the only metric we could have used!

Metrics in Computing

- Generally, folks care most about performance
  - Energy-efficiency is more important now since the plateau in 2004/2005
  - This is why we have so many specialized chips nowadays

- Really, this is just **efficiency** – making efficient use of the resources that we have
  - Performance: cycles/instruction, seconds/program
  - Energy efficiency: performance/watt
  - Memory: bytes/program, bytes/data structure
  - Algorithm: Big-O complexity analysis (143)

- Why is there so much focus on efficiency in CS?
- Does this focus make sense to you?
- Is there anything else we should focus on?

Efficiency's a metric!

- What do we do with metrics?
  - We tend to optimize along them!
  - Especially when jobs/funding depend on better performance along some metric
    - See all of Intel under "Moore's Law"
- Sometimes, strange incentives emerge
  - "Minimize the number of bugs on our dashboard"
    - Does it count if we make the bugs invisible?
  - "Make this faster for our demo in a week"
    - Shortcuts might hurt performance at scale
  - "Minimize our average memory access time"
    - What if we add *more* memory accesses that we know will hit?

Metrics and Success

- Success is *defined along metrics*
  - This affects how we measure and optimize

- Let's say that we choose **performance/program** or **performance/program set** (*i.e.*, benchmarks):
  1. Measure existing performance
  2. Come up with a bunch of optimizations that would improve performance
  3. Select a few to build into the "next version"

Metrics and Success

- Success is *defined along metrics*
  - This affects how we measure and optimize

- Let's say that we choose **profit/year** or **stock price**:
  - Success means earning more profit than last year
  - Improvement or optimizations might include:
    - Reduce expenses, cut staff
    - Sell more things or fancier things (*e.g.*, in-app purchases)
    - Make people pay monthly for things they could get for free
    - Increase advertising revenue:


The New York Times

Whistle-Blower Says Facebook 'Chooses Profits Over Safety'

Frances Haugen, a Facebook product manager who left the company in May, revealed that she had provided internal documents to journalists and others.

Metrics and Success

- Success is *defined along metrics*
  - This affects how we measure and optimize

- Let's say that we choose **participation of minoritized folks in computing**:
  - What does success/participation mean (and dangers)?
    - Women? BIPOC? Everyone that's been minoritized?
      - Might optimize for one group at the expense of others
    - Taking intro? Passing intro? Getting a degree? Getting a job?
      - Says nothing about retention or participation/decision-making level

Design Considerations

- Regardless of what we build, the way that we define success shapes the systems we build
  - Choose your metrics carefully!
  - There's more to choose from than performance (*e.g.,* usability, access, simplicity, agency)

- Metrics are a "heading" (in the navigational sense)
  - Best to reevaluate from time to time in case you're off course or your destination changes

Feeling ok so far?

Let's examine in context…

Caches, in context

- Computing historically has prioritized performance
  - (Mind you, we can pick other metrics going forward)
- *We optimize the common case* for performance
  - A classic! One might even say "foundational"

- For caches, "common case" programs exhibit *locality*
  - We talked about *spatial* and *temporal* locality with code/data
  - Locality defined from existing programs
    - i.e. most existing programs look like

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];

    return sum;
}
```

Caches, Locality, Morality

- Programs exhibit can define good/bad locality
  - *But, we extended this to good/bad code! A value judgement.*
  - Bad code exhibits poor locality, and has poor performance

Bad code, bad locality

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];

    return sum;
}
```

Good code, good locality

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];

    return sum;
}
```

Caches, Locality, Morality

- We're using efficiency to measure success
  - *Performance/program, Average Memory Access Time*
- What optimizations might we choose?
  - Faster processors, faster memory
  - More complexity, in service of more performance
- What might we build?
  - *Specialized, expensive hardware that's a huge source of complexity and bugs, that regularly confuses 351 students?*
  - Caches alone have caused massive HW/SW bugs (meltdown)

Our choice of metric affected what we built!

- We wanted efficiency, so we optimized the common case
- The common case was "locality", so we build caches
- We value-judged programs based on their ability to fit within that structure!

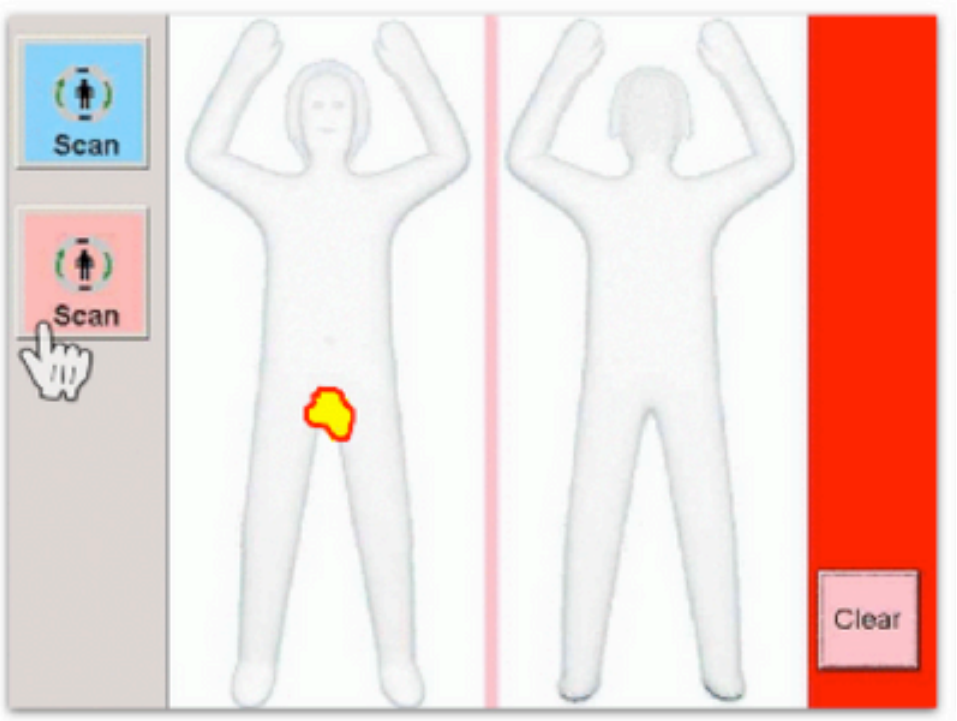
Let's take this idea elsewhere in computing!

Let's take it on a trip!

(Cue airplane noises)


Hudson News
7:10
C10
C10c / C10d

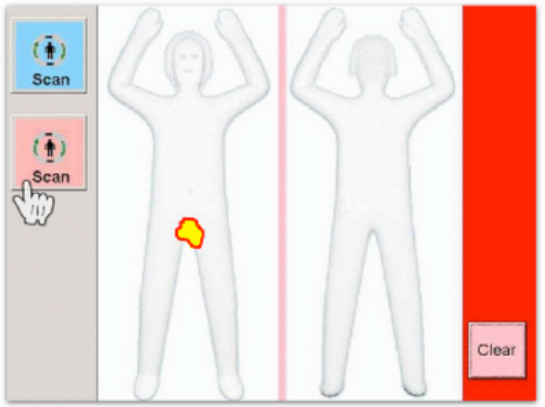

smiths
smiths
smiths



"I have to scan the way that you're presenting"

What's going on here?

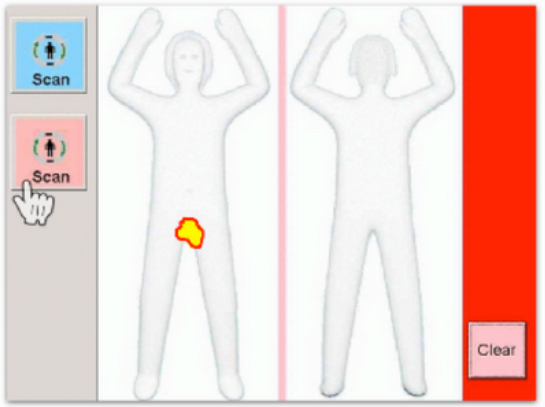
- TSA chooses metrics for government contracts
  - *Efficiency, False negative rate*
- Contractors build systems to succeed along metrics
  - Build machine learning models based on lots of bodies
  - Predictable variation among "common case" bodies
  - Build two models, one for "men", one for "women"
- TSA agents clock travelers, choose a button, clear alarms as necessary



What's going on here?

- I presented more femme that day (i.e. bras)

- I got "pink button'd"
  - My body (happily) is an edge-case
  - I'll take a pat-down over being detained…
  - …but I'd rather not be outed every time I fly…
  - …and, I'd rather not have my body be labeled as a threat
- *Optimizing for the common case at the expense of edge-cases*



Our choice of metric affected what we built!

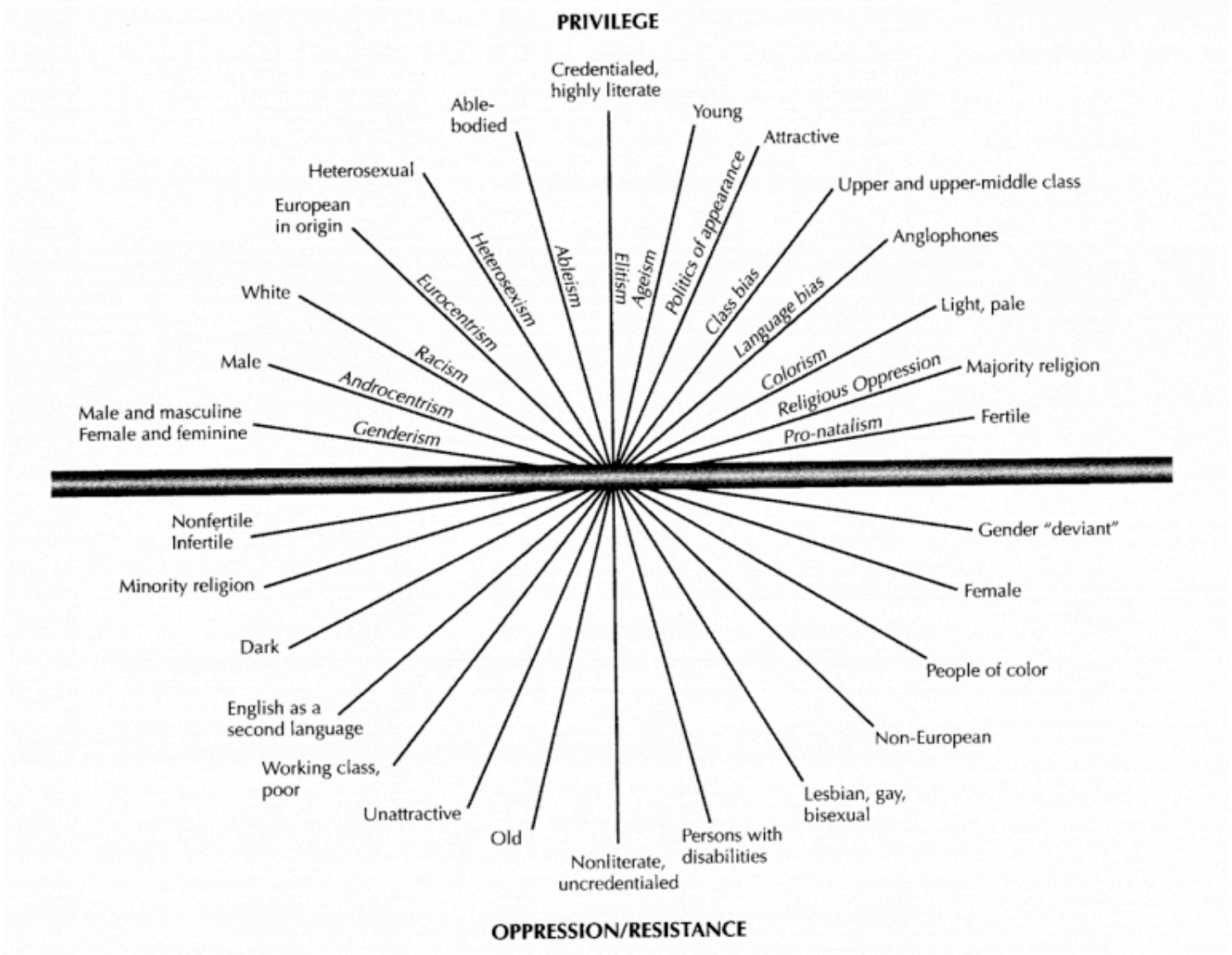
- We wanted efficiency, so we optimized the common case
- The common case was cisgendered bodies, so we built binary models
- We value-judged bodies based on their ability to fit within that structure!

Human Complexity

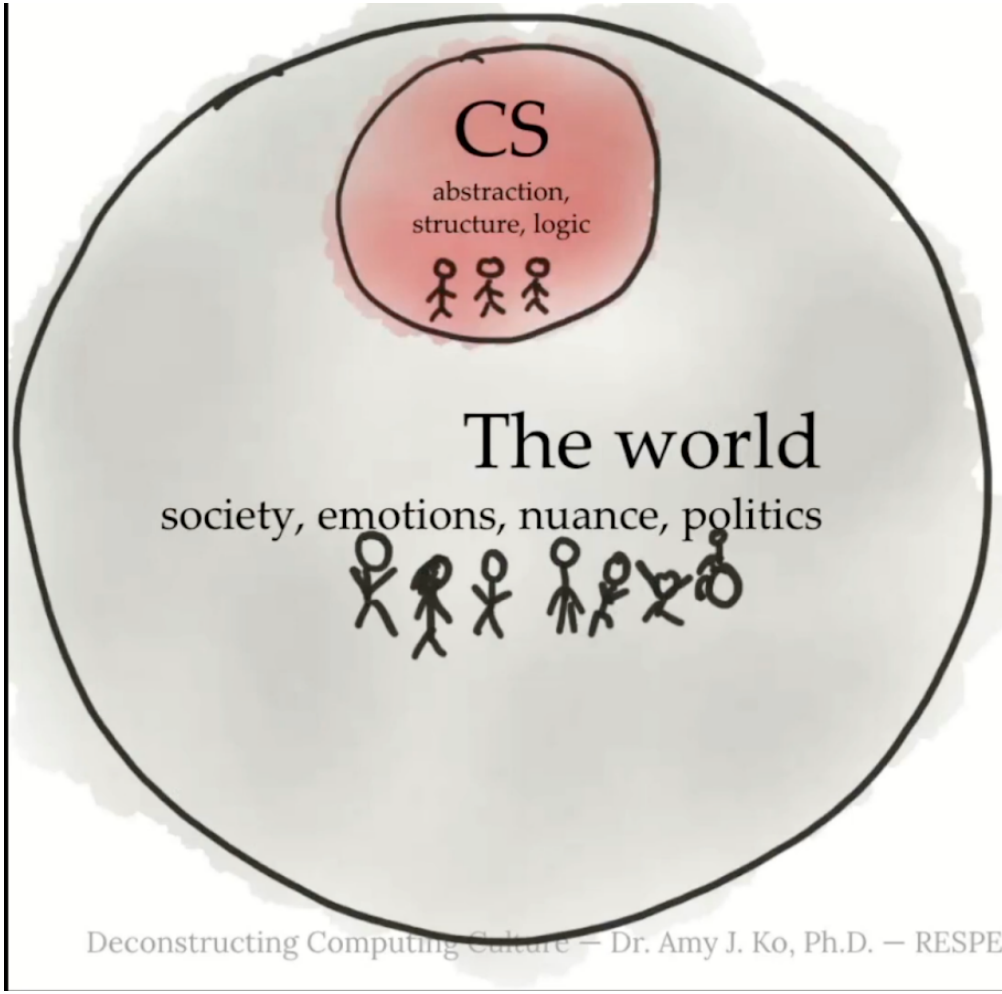
- Humans are incredibly complex and diverse!
  - 5% identify as LGBT (as Gallup polls, which is imperfect)
  - 16% among Gen Z 😎
  - **80%** have some non-normative gender/sexuality/relationship
    - Everyone that feels that gender/sexuality boxes don't fit, everyone with non-normative sexual preferences
  - (Also, this doesn't count anyone questioning, figuring it out, avoiding labels)

- Normative intersections don't leave space for anyone!

Intersections of complexity (1996)



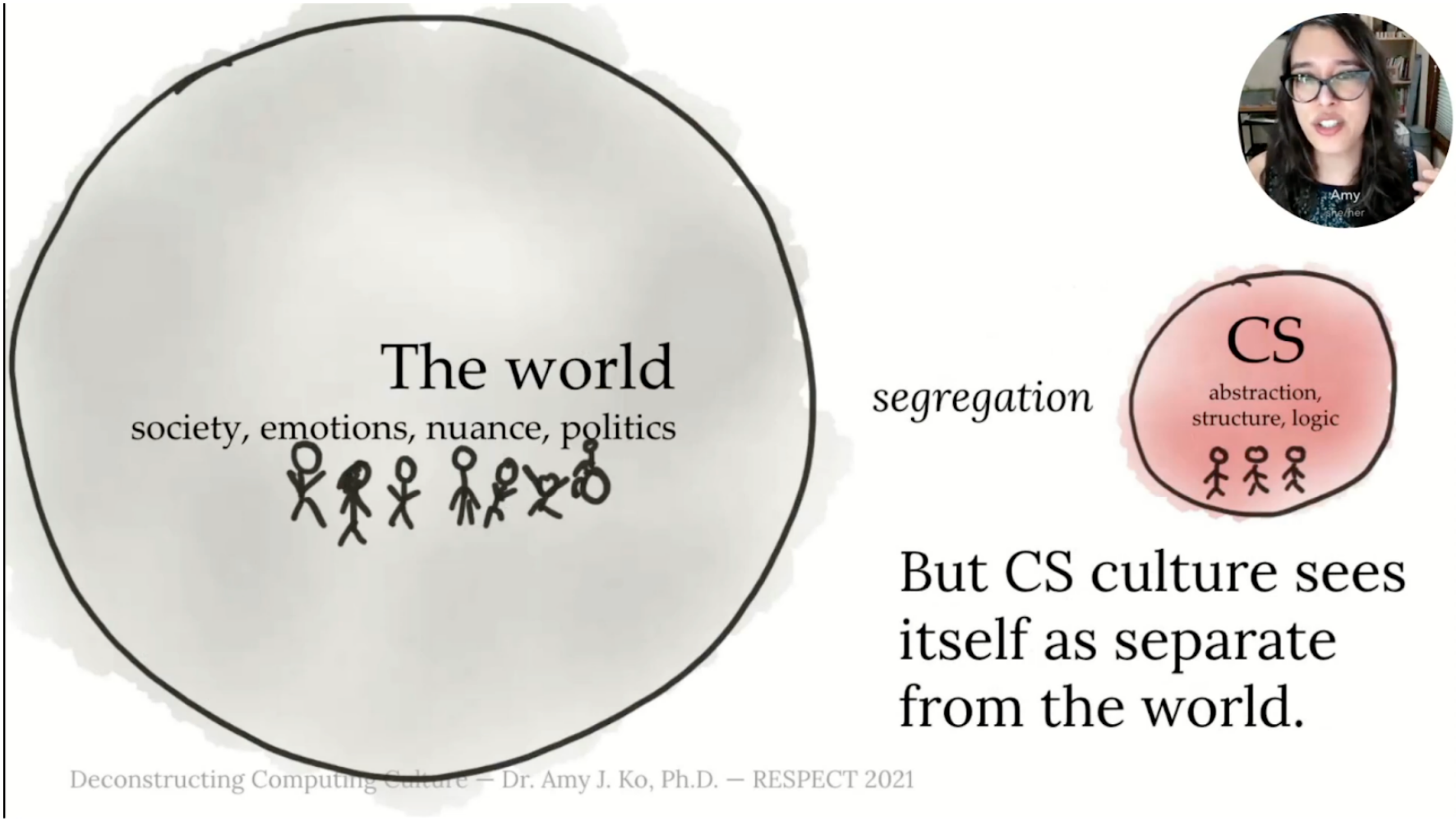
CS is part of the world…



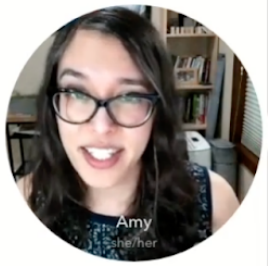
We all know that CS is part of the world, part of that wonderful diversity and complexity.

Deconstructing Computing Culture — Dr. Amy J. Ko, Ph.D. — RESPECT 2021

…but sees itself as separate



Separate, and segregated



And it's not just social segregation, but intellectual segregation.

- CS **abstracts**, removing *messy social context*
- CS **neutralizes**, removing *nuanced values and politics*
- CS **normalizes**, erasing *diversity and exceptions*
- CS **automates**, removing *people and their unpredictable decisions*

These foundational CS concepts and values benefit the dominant groups in CS, reducing the burden of understand the complex social world, isolating them from its complexities.

Deconstructing Computing Culture — Dr. Amy J. Ko, Ph.D. — RESPECT 2021

But, caches don't interface with people!

- Yes, but…
  - The values and assumptions in caches are prevalent in CS
  - And, they extend to human-interfacing spaces unquestioned

  - Caches are "neutral", the values underpinning them aren't!
    - But also, google drive caching dead-names folks :/
    - And "conforming" to machine expectations through locality is an achey practice

- *Optimizing for the common case* isn't always so simple
  - Penalizing edge-case performance might mean viewing human diversity/variation as a threat!

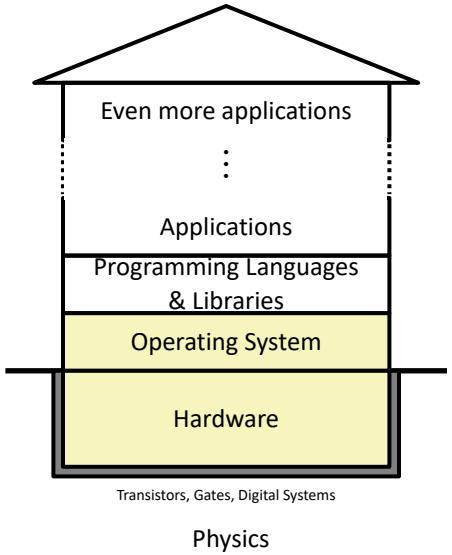
You have incredible power and access!

What do you want to build?

Who do you want your work to serve?

The Hardware/Software Interface

- Topic Group 3: **Scale & Coherence**
  - Caches, **Processes**, Virtual Memory, Memory Allocation



- How do we maintain logical consistency in the face of more data and more processes?
  - How do we support control flow both within many processes and things external to the computer?
  - How do we support data access, including dynamic requests, across multiple processes?

Reading Review

- Terminology:
  - Exceptional control flow, event handlers
  - Operating system kernel
  - Exceptions: interrupts, traps, faults, aborts
  - Processes: concurrency, context switching, fork-exec model, process ID

- Questions from the Reading?

Leading Up to Processes

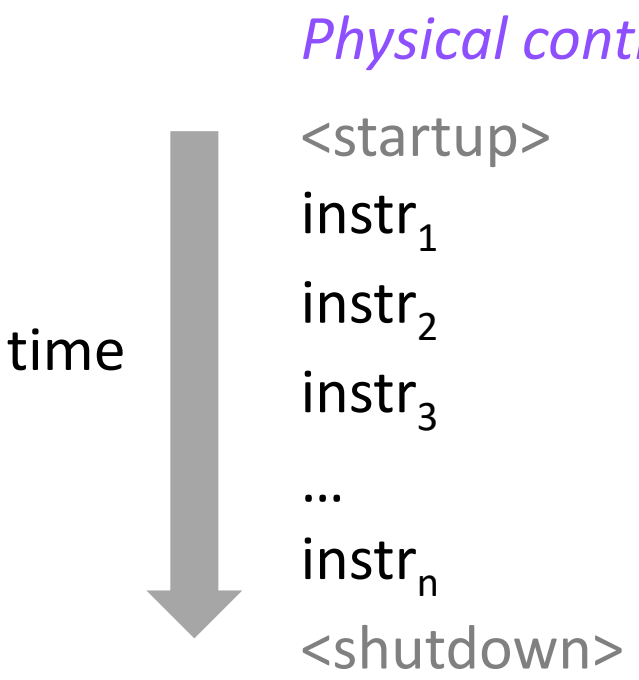
- System Control Flow
  - **Control flow**
  - **Exceptional control flow**
  - Asynchronous exceptions (interrupts)
  - Synchronous exceptions (traps & faults)

Control Flow

- **So far:** we've seen how the flow of control changes as a *single program* executes
- **Reality:** multiple programs running *concurrently*
  - How does control flow across the many components of the system?
  - In particular: More programs running than CPUs
- *Exceptional* control flow is basic mechanism used for:
  - Transferring control between *processes* and OS
  - Handling *I/O* and *virtual memory* within the OS
  - Implementing multi-process apps like shells and web servers
  - Implementing concurrency

Control Flow

- One processor only does one thing*:
  - From startup to shutdown, a CPU simply reads and executes (interprets) a sequence of instructions, one at a time*
  - This sequence is the CPU's *control flow* (or *flow of control*)



Altering the Control Flow

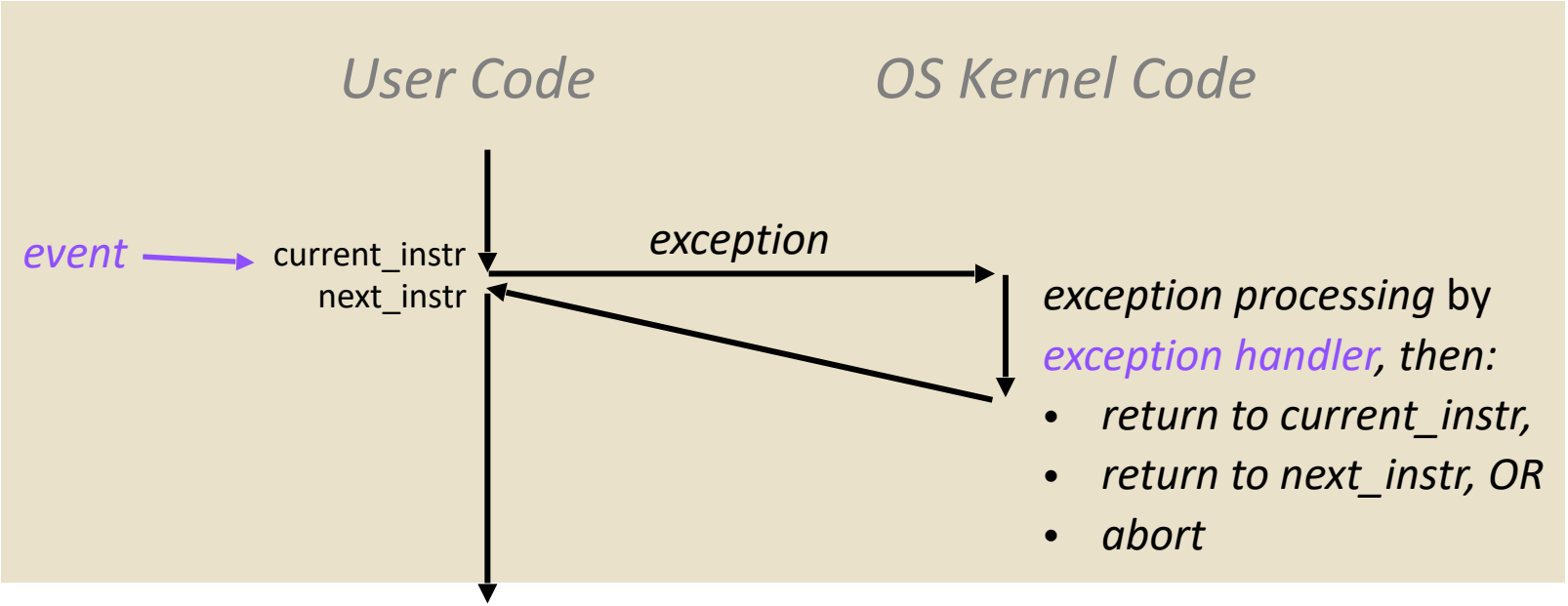
- Up to now, two ways to change control flow:
  - Jumps (conditional and unconditional)
  - Call and return
  - Both react to changes in *program state*
- Processor also needs to react to changes in *system state*
  - Unix/Linux user hits "Ctrl-C" at the keyboard
  - User clicks on a different application's window on the screen
  - Data arrives from a disk or a network adapter
  - Instruction divides by zero
  - System timer expires
- Can jumps and procedure calls achieve this?
  - No – the system needs mechanisms for *"exceptional"* control flow!

Exceptional Control Flow

- Exists at all levels of a computer system
- Low level mechanisms
  - **Exceptions**
    - Change in processor's control flow in response to a system event (*i.e.*, change in system state, user-generated interrupt)
    - Implemented using a combination of hardware and OS software
- Higher level mechanisms
  - **Process context switch**
    - Implemented by OS software and hardware timer
  - **Signals**
    - Implemented by OS software
    - We won't cover these – see CSE451 and EE/CSE474

Exceptions (Review)

- An *exception* is transfer of control to the operating system (OS) kernel in response to some *event* (*i.e.*, change in processor state)
  - Kernel is the memory-resident part of the OS
  - Examples: division by 0, page fault, I/O request completes, Ctrl-C

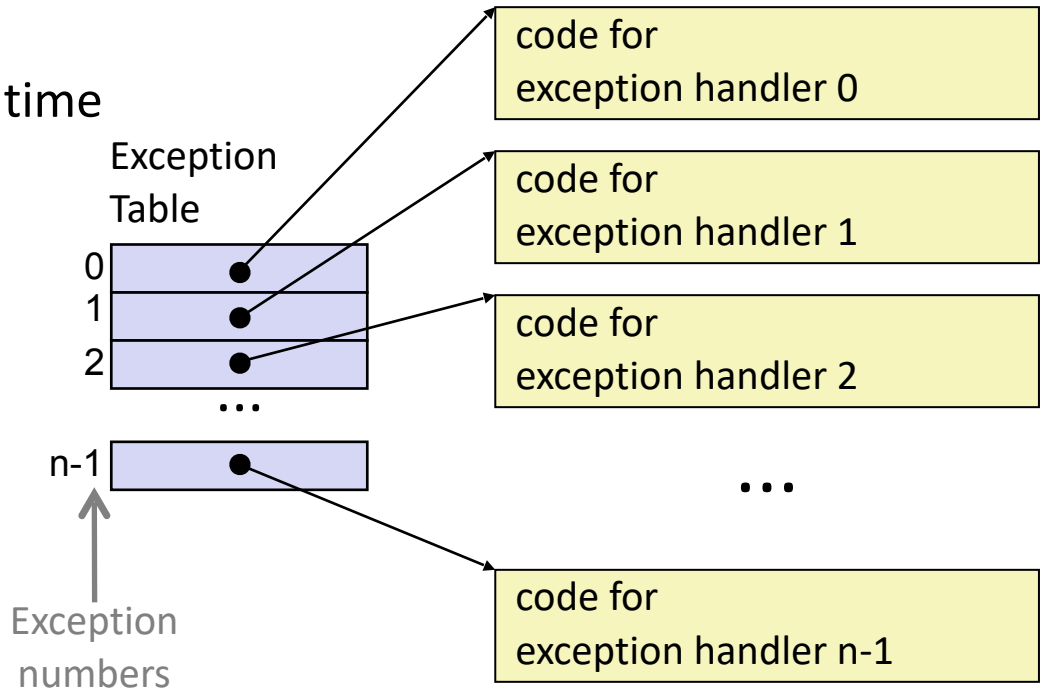


- *How does the system know where to jump to in the OS?*

Exception Table

This is extra (non-testable) material

- A jump table for exceptions (also called *Interrupt Vector Table*)
  - Each type of event has a unique exception number k
  - k = index into exception table (a.k.a interrupt vector)
  - Handler k is called each time exception k occurs



Exception Table (Excerpt)

This is extra (non-testable) material

Exception Number	*Description*	*Exception Class*
0	Divide error	Fault
13	General protection fault	Fault
14	Page fault	Fault
18	Machine check	Abort
32-255	OS-defined	Interrupt or trap