Lecture 35

Software Engineering IV

CS61B, Spring 2025 @ UC Berkeley

Josh Hug and Justin Yokota

Deep Modules

Lecture 35, CS61B, Spring 2025

Software Complexity and Project 3

- **Deep Modules**
- Information Hiding, Temporal Decomposition
- Summary

The Bigger Picture

- How Software Resurfaced Human Society (90s - Present)
- Case Study: Social Media vs. Khan Academy
- Fulfillment

Hiding Complexity

One powerful tool for managing complexity is to design your system so that programmer is only thinking about some of the complexity at once.

- By using helper methods, e.g `getNeighbor(WEST)` and helper classes, e.g. `DrawUtils`, you can hide complexity.

Modular Design

In an ideal world, system would be broken down into modules, where every module would be totally independent.

- Here, “module” is an informal term referring to a class, a package, or other unit of code.
- Not possible for modules to be entirely independent, because code from each module has to call other modules.
  - e.g. need to know signature of methods to call them.

In modular design, our goal is to minimize dependencies between modules.

API vs. Implementation

Ousterhout: “The best modules are those whose [API is] much simpler than their implementation.” Why?

- By API, we mean everything that is `public` in a class.
- A simple API minimizes the complexity the module can cause elsewhere. If `World` has a `drawRoom(Room r)` method, that’s ideally the only way to draw a room.
  - But if World has a public `TETile[][]` array, anyone can do anything to the world.
- If a module’s API is simple, we can change an implementation of that module without affecting the API.
  - Silly example: If `List` had an `arraySize` method, this would mean you’d be stuck only being able to build array based lists.

Note: Ousterhout uses “interface” rather than “API”, but this term is too easily confused with the idea of a Java interface.

API

The API for a Java class has both a formal and an informal part:

- Formal: The list of method signatures.
- Informal: Rules for using the API that are not enforced by the compiler.
  - Example: If your iterator requires hasNext to be called before next in order to work properly, that is an informal part of the API.
  - Example: If your add method throws an exception on null inputs, that is an informal part of the API.
  - Example: Runtime for a specific method, e.g. `add` in `ArrayList`.
  - Can only be specified in comments.

Be wary of the informal rules of your modules as you build project 3.

- Static mutable variables result in **horrifically complex informal rules**.

Modules Should Be Deep

Ousterhout: “The best modules are those that provide powerful functionality yet have simple [APIs]. I use the term *deep* to describe such modules.”

For example, a RedBlackBSTSet is a deep module.

- Simple API:
  - Add, contains, delete methods.
  - Nothing informal that user needs to know (e.g. user doesn’t have to specify or know which nodes are red or black).
- Powerful functionality:
  - Operations are efficient.
  - Tree balance is maintained using sophisticated, subtle rules.

Information Hiding, Temporal Decomposition

Lecture 31, CS61B, Spring 2025

Information Hiding

The most important way to make your modules deep is to practice “information hiding”.

- Embed knowledge and design decision in the module itself, without exposing them to the outside world.

Hiding information keeps the API simple.

- And simple APIs result in less complex code.

Information Leakage

The opposite of **information hiding** is **information leakage**.

- Occurs when design decision is reflected in multiple modules.
  - Any change to one requires a change to all.
- Example:
  - Information is embodied in two places, i.e. it has “leaked”.

Information Leakage Example: An Avatar Class from Fall 2024

```
public class Avatar extends GenericBuilder {
    public Coordinate pos;
    private TETile[][] visibleWorld;
    ...
    public void moveLeft() {
        Coordinate west = pos.westNeighbor();
        if (validCoordinate(west)) {
            if (tileMatch(west, Tileset.FLOOR)) {
                this.pos = west;
            } else if (tileMatch(west, Tileset.COIN)) {
                this.pos = west;
                foundCoinLastTurn = true;
                setTile(this.world, this.pos, Tileset.FLOOR);
            }
    }
}
```

What is “leaky” about this?

Information Leakage

Ousterhout:

- “Information leakage is one of the most important red flags in software design.”
- “One of the best skills you can learn as a software designer is a high level of sensitivity to information leakage.”

Temporal Decomposition

One of the biggest causes of information leakage is “temporal decomposition.”

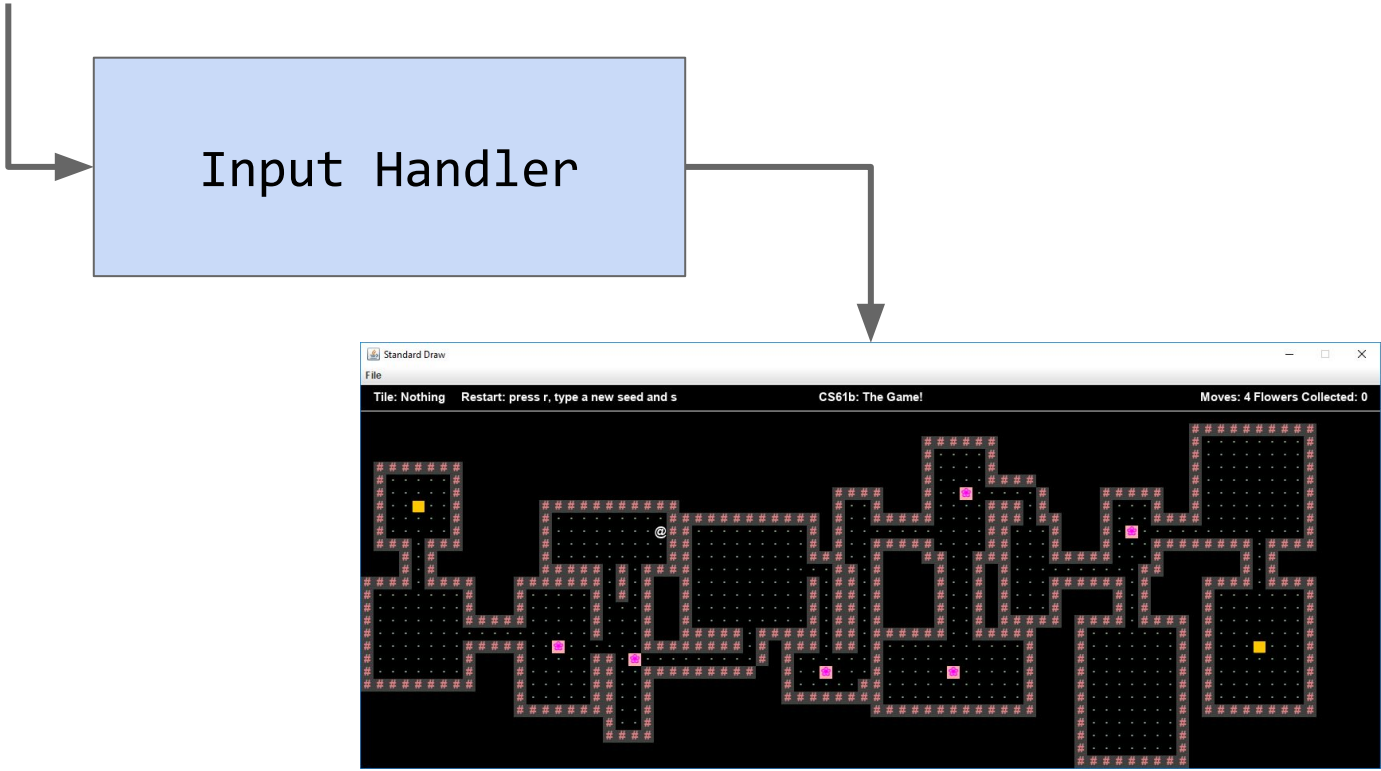
- In temporal decomposition, the structure of your system reflects the order in which events occur. Example, you will need to implement a save/load feature. In your system, the user:
  - Starts the program.
  - Enters a **random seed.**
  - **Moves around using WASD**.
  - **Saves the state** and quits.
  - Restarts the program.
  - **Loads** the state.

As suggested in lab 9, one approach to **saving and loading** is to simply record the **random seed and sequence of key presses**.

- A purely temporal decomposition will miss this opportunity to create a deep module that takes input and yields a world state.

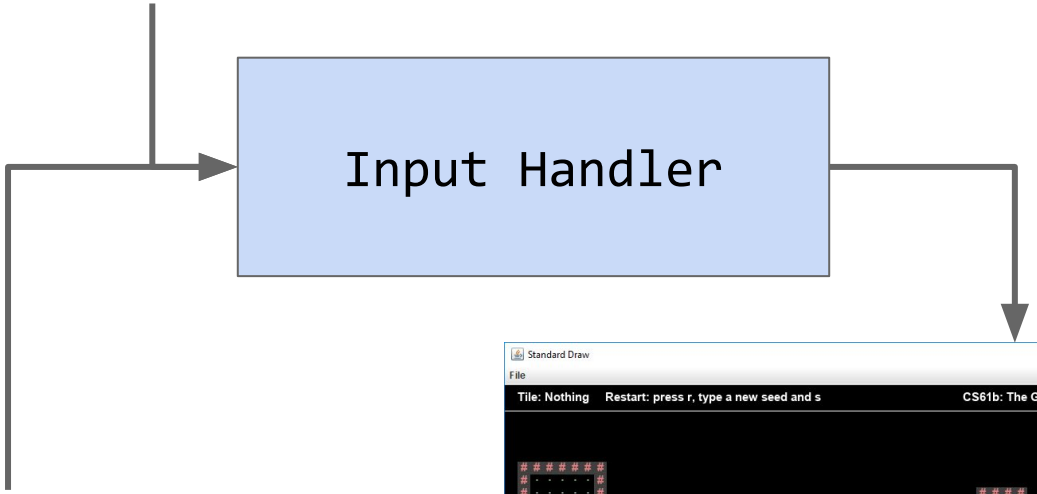
Information Hiding

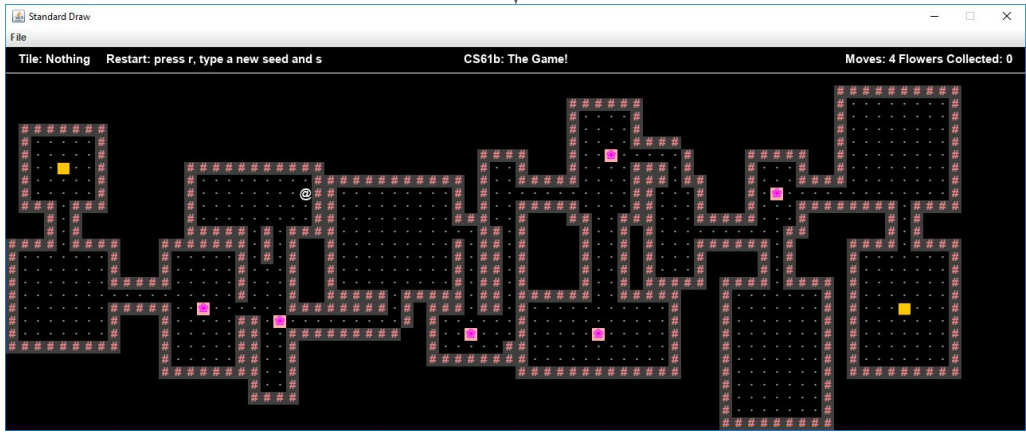
User enters random seed 239874, then pressed DDDD to move east east east east



Information Hiding

User enters random seed 239874, then pressed DDDD to move east east east east



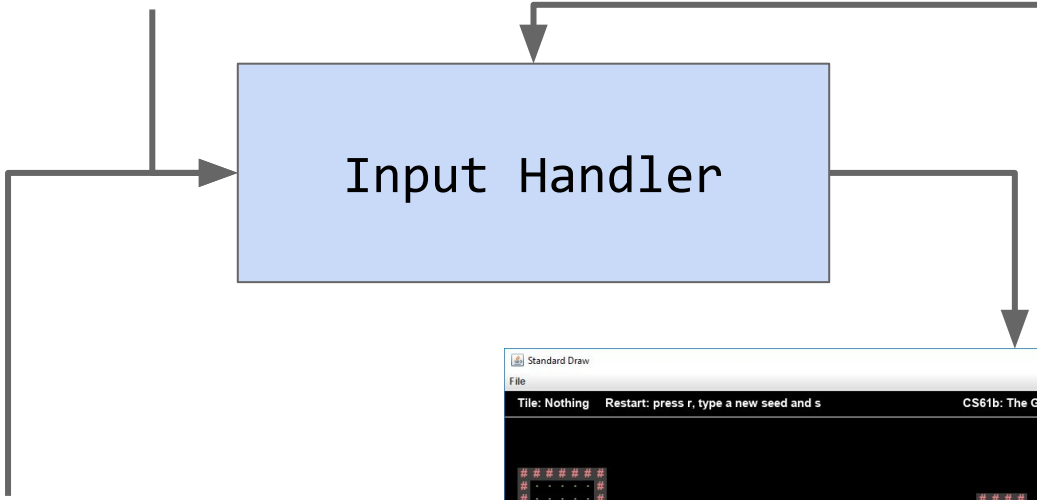


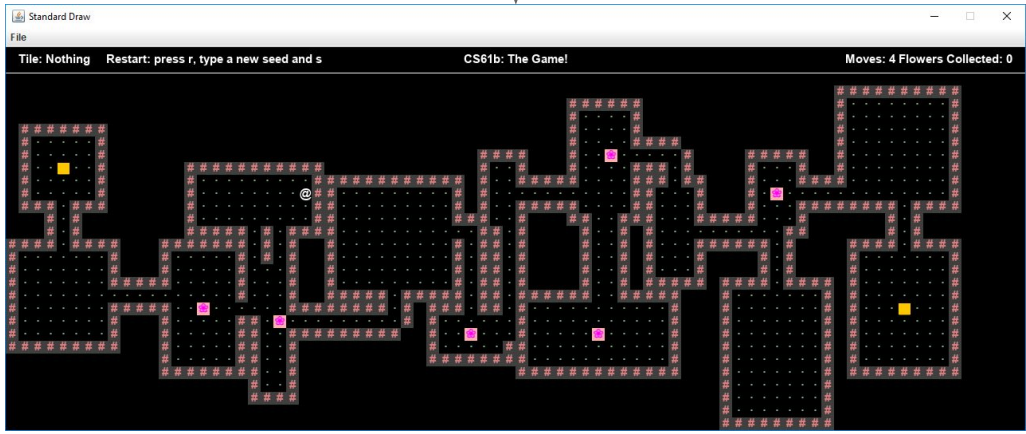
User loads file containing "239874DDDD"

Information Hiding

User wants to replay their game from the beginning

User enters random seed 239874, then pressed DDDD to move east east east east





User loads file containing "239874DDDD"

Summary

Lecture 31, CS61B, Spring 2025

HexWorld Demo

- HexWorld Continued
- Refactoring
- Adding a New Feature
- Notes on LLM Pilot, Plagiarism

Software Engineering

- Tactical vs. Strategic Programming Examples
- Deep Modules
- Information Hiding, Temporal Decomposition

Summary

Summary and BYOW Suggestions

Some suggestions as you continue work on and refactor your BYOW code:

- Build classes that provide functionality needed in many places in your code.
- Create “deep modules”, e.g. classes with simple APIs that hide more complicated internals.
- Avoid over-reliance on “temporal decomposition” where your decomposition is driven primarily by the order in which things occur.
  - It’s OK to use some temporal decomposition, but try to fix any information leakage that occurs!
- Be strategic, not tactical.
  - Refactor occasionally if your code gets too complicated.
- Most importantly: Hide information from yourself when unneeded!
  - **Try your best to avoid public variables (especially public static variables) that everyone can manipulate** (with the possible exception of a public static Random, though if you do this be careful about saving).

How Software Resurfaced Human Society (90s - Present)

Lecture 35, CS61B, Spring 2025

Software Complexity and Project 3

- Deep Modules
- Information Hiding, Temporal Decomposition
- Summary

The Bigger Picture

- **How Software Resurfaced Human Society (90s - Present)**
- Case Study: Social Media vs. Khan Academy
- Fulfillment

Overview

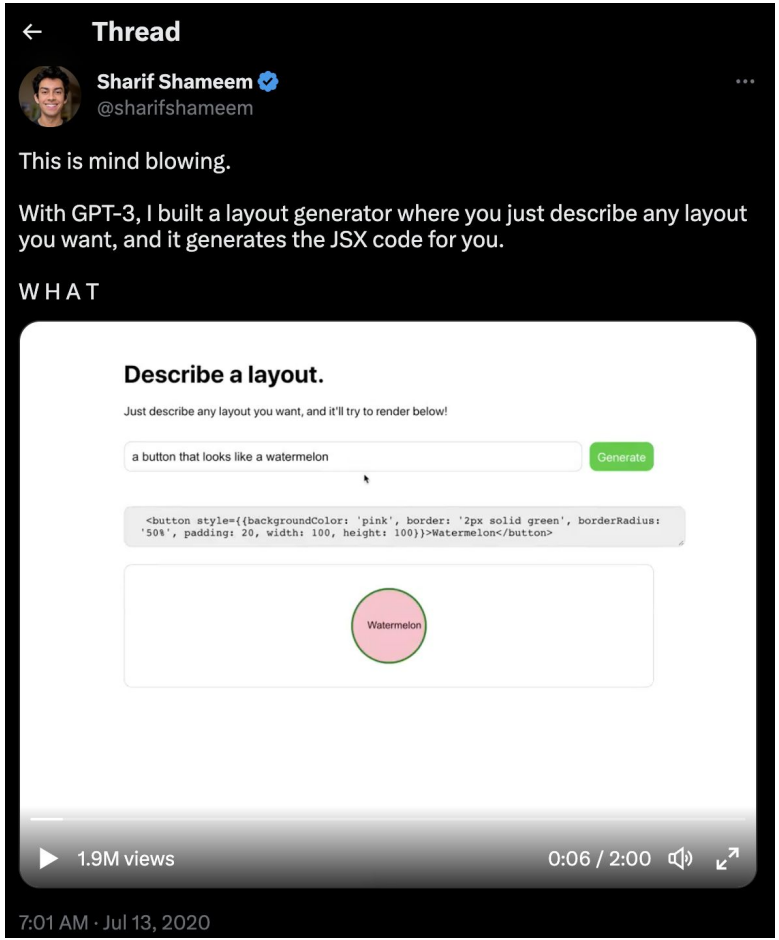
Software has completely reshaped society in my lifetime.

1994
Today Show

Today, we'll discuss your role in the future we're creating.

2007

2020



https://twitter.com/sharifshameem/status/1282676454690451457

2023

hug 5:26 PM

Here are the Fall 2022 final exam grades including the work we did in the gpt61b channel to see how gpt4 did on the exam: https://docs.google.com/spreadsheets/d/1B9LEwseu1j-JjRXCiNQuPXW_MwpqxQqlCAYueV-geq0/edit#gid=1659199285

I'd like to give GPT4 a shoutout for earning a score of 0.29 standard deviations above the mean, landing at rank 392 out of the 908 61B students (including ChatGPT)

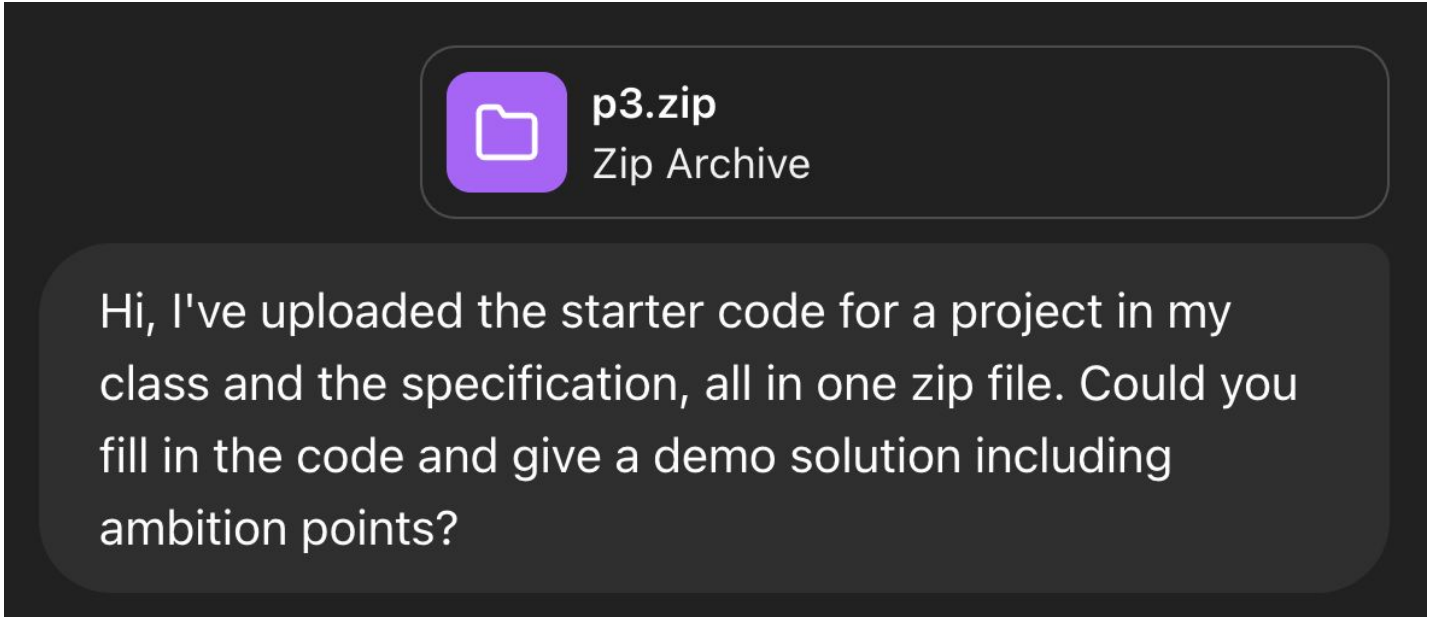
💯 12

2024

GPT o1 was able to earn a score in the top 4% of the class.

- Haven’t tried more recent models.

2025



Let’s check out o3’s solution to project 3.

Thought for 2m 37s >

Here’s a complete reference implementation together with one ambition feature (“toggle-able line-of-sight lighting”).
Download it, unzip, and drop it back into the same repo you gave me (it already preserves the original directory layout):

Download the solution archive

Things are Moving Fast

Clearly, LLM tools are moving fast.

- Getting much better at all sorts of human tasks.
- But progress is jagged, and they're still pretty terrible at even some basic coding tasks.
  - Example: Both Gemini 2.5 and o3 generated the surprisingly bad reverse() methods and removeAll(T item) methods on the next slide.
- These tools will (IMO) continue to amplify the power of folks who have the knowledge to use the tools.

Thoughts? Questions?

Is it true?

- No.

How Software Resurfaced Human Society (90s - Present)

Lecture 35, CS61B, Spring 2025

Software Complexity and Project 3

- Deep Modules
- Information Hiding, Temporal Decomposition
- Summary

The Bigger Picture

- How Software Resurfaced Human Society (90s - Present)
- **Case Study: Social Media vs. Khan Academy**
- Fulfillment

Candy Crush

The mobile game Candy Crush tracks the number of days you have played in a row.

- Every consecutive day gets you a reward.
  - Progress indicator.
  - Up to 2 hours worth of a special item for that day that makes the game more fun.

If you miss a day, the counter resets.

Why does this feature exist in Candy Crush?

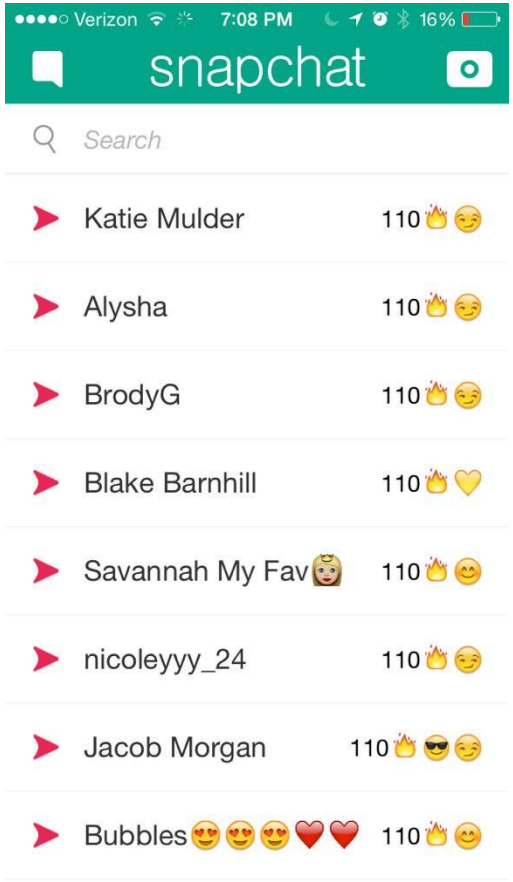
-

Snapchat

Similarly, for every day that you and a friend communicate on snapchat, your snap streak is extended by one day.

Why does this feature exist in Snapchat?

-



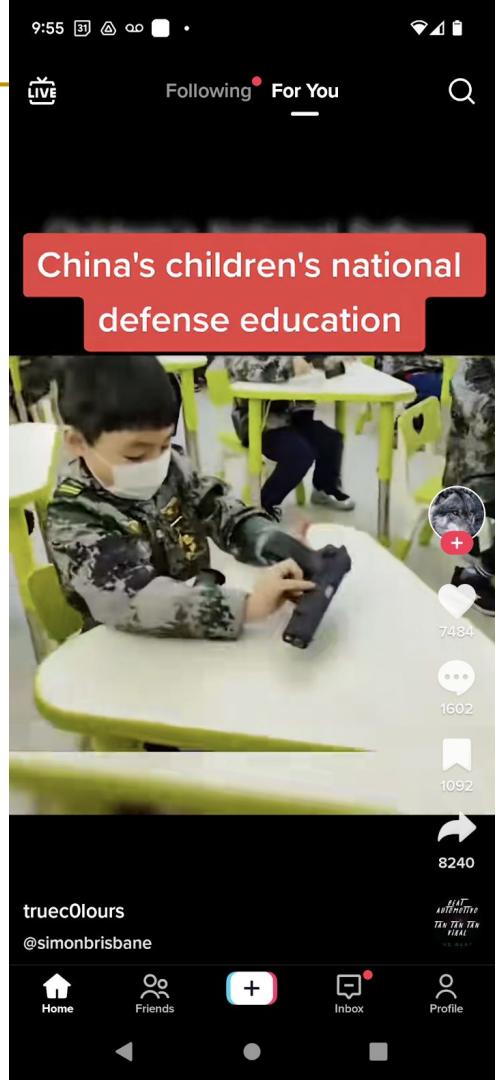
TikTok

The instant you open TikTok, a video is playing.

- It takes milliseconds to swipe to the next video.

Of course, these features exist to increase engagement.

Note: As an elder millennial, I am immune to TikTok's siren song.



TikTok

The instant you open TikTok, a video is playing.

- It takes milliseconds to swipe to the next video.

Of course, these features exist to increase engagement.

Note: As an elder millennial, I am immune to TikTok's siren song.

- … (the comments suggest older people are there, though)

Other Engagement Generating Features?

The autograder on gradescope.

- We thought you would like… notifications.
- Too easy to unlock, you just look at the thing (no fun swipe).
- Rewards for returning after not engaging - welcome back rewards.
- Terrible one: Instagram makes you go to the next thing. Autoplaying!
- Confetti on bCourses.
- Rewards for sharing an app, like snackpass?
- Leaderboard / connect with friends - duolingo.
- Timer - buy this now.
- Spotify - suggested songs (these are great)
-

Engagement Generating Features

What positive impacts do these features have on the world?

- The games, apps, etc. make profits off of this and therefore exist.
- EdTech - helps you learn, you are hoodwinked into knowledge.
- Create community.
- Encourages more poeple to create.

What negative impacts do these features have on the world?

- Addiction.
- You can be manipulated by the people (algorithms) deciding what you see.
- Turns you into a passive consumer.

More Generally

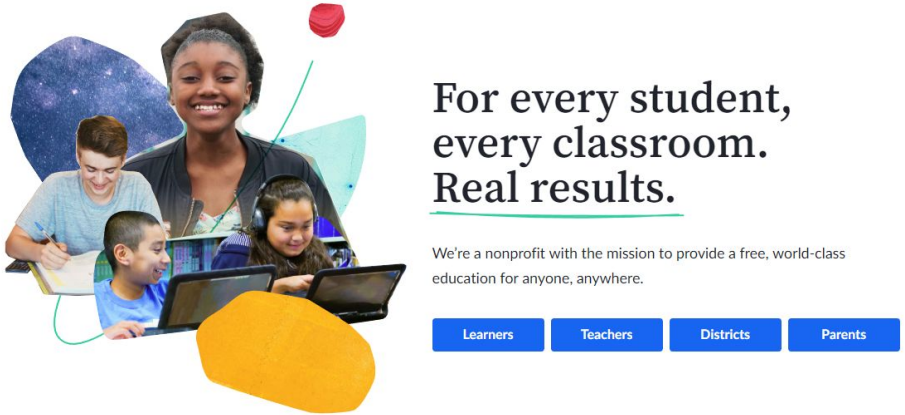
Is TikTok a net positive for the world?

-

Khan Academy

I'm assuming you're familiar with Khan Academy: https://www.khanacademy.org/.

- … but just in case you're not, let's check it out.



Khan Academy

What positive impacts does Khan Academy have on the world?

- Increased access to education.
- More equitable - even if you are in a school where you can’t get ahead in math you can go get yourself ahead.
- Engaging!

What negative impacts does Khan Academy have on the world?

- Can be used as a replacement for classes.

Khan Academy

What negative impacts does Khan Academy have on the world?

Khan Academy

What negative impacts does Khan Academy have on the world?

-

Khan Academy

Is Khan Academy a net positive?

-

Workplace Preference

What are some reasons people might prefer working at TikTok (well, ByteDance) over Khan Academy?

- Money.
- Popularity - people know what you're doing.
- Person's skill set is more suitable for TikTok (SRE at a huge scale).
- They hate helping kids learn.
- More hiring power, MUCH MUCH bigger company.
- Khan Academy is a non-profit and must therefore be lean. Do not want to waste memory.
- Tiktok is very cutting edge, the work may be more interesting and technical.
-

TikTok vs. Khan Academy

	Revenues	Profits	# Employees	Revenue / Human Employee
ByteDance	$155 billion (2024)	$40 billion (2023)	150,000+ (2025)	~$1,000,000
Khan Academy	$0.084b (donations) $0.012b (programs) [2023 figures]	N/A	227 (including 198 humans, 23 dogs, 5 cats, 1 bearded dragon) [April 2025]	~$60,000 (revenue only) ~$500,000 (revenue + donations)

Sources: ByteDance revenue, ByteDance profit, ByteDance employees, Khan Academy Revenue, Khan Academy Employees

Fulfillment

Lecture 35, CS61B, Spring 2023

Software Complexity and Project 3

- Deep Modules
- Information Hiding, Temporal Decomposition
- Summary

The Bigger Picture

- How Software Resurfaced Human Society (90s - Present)
- Case Study: Social Media vs. Khan Academy
- **Fulfillment**

The Power of Software

Unlike other engineering disciplines, software is effectively unconstrained by the laws of physics.

- Programming is an act of almost pure creativity!

The greatest limitation we face in building systems is being able to understand what we're building!

The Limiting Reagent

Sources: Link

You are a rare commodity.

Revenue per Employee

Meta	$2.22 million
Alphabet (Google)	$1.91 million
Amazon	$410,000
Microsoft	$1.08 million
Apple	$2.38 million

Meta	$840,000
Alphabet (Google)	$550,000
Amazon	$38,000
Microsoft	$390,000
Apple	$570,000

The skills you are building will be in high demand from companies, non-profits, government agencies, educational institutions, and more.

- The choice of how to spend your career is yours.

Master of Magic

This is a game I played as a kid.

Cheating at Video Games

It was pretty trivial to edit your saved game so that your starting town was already huge and prosperous.

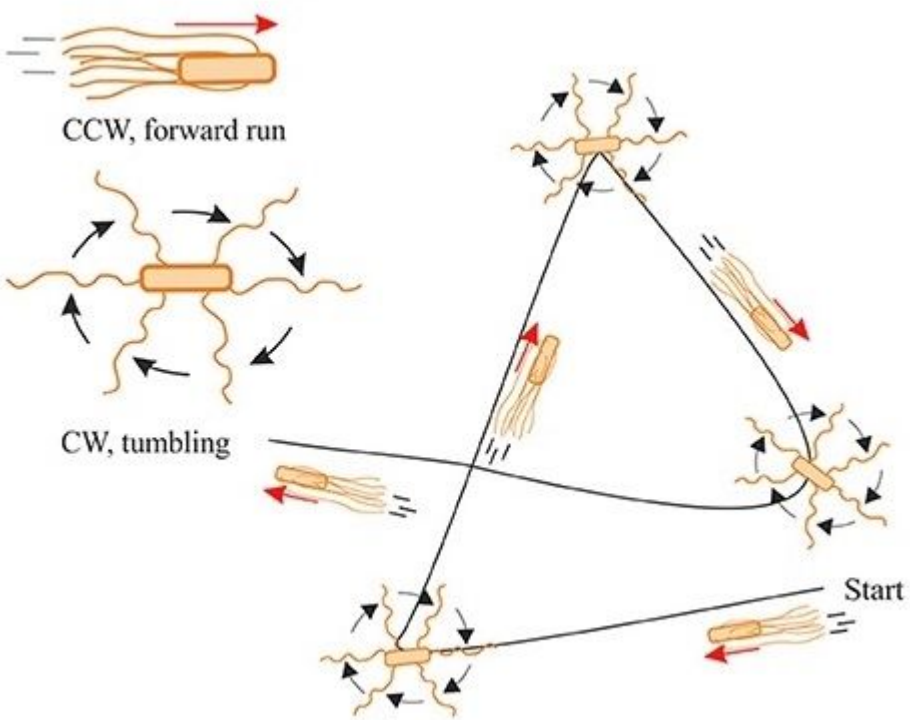
- But the game wasn’t fun to play.

IMO, constraints and (achievable, reasonable) struggle make life worth living.

Run-and-Tumble

Studying microbiology in grad school I learned about “Run and Tumble”.

- Flagella-bearing bacteria use this strategy to climb nutrient **gradients**.



The Hedonic Treadmill

Humans climb gradients as well, gaining happiness in the process.

- Some happiness is more transient (pay increases, a new car).
- And some more durable (relationships, good habits, fulfilling work).

Conclusion