

CSCI 0150

(also known as CS15)

A Gateway to Computer Science



Please put on your KN95 Masks!!

Andy says “thank you”
in advance :)

Welcome to CS15!!!



CS15 Head TAs: We're here for YOU!



Anastasio

Allie

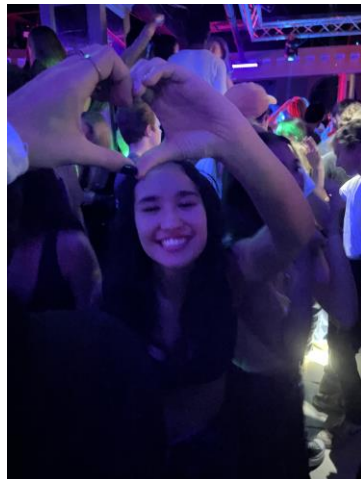


Cannon

Lexi



Sarah



Computer Science

- CS15 is a start to understanding computer science
 - for your own intellectual interest
 - for its enrichment of other fields
 - for its combination of scientific, engineering, art and design concepts and practices, and as a “mode of thought” – “computational thinking”
- IT, or information technology, including CS, is key to the “knowledge economy”
- Omnipresent in a breadth of various applications and fields

Stunning Special Effects



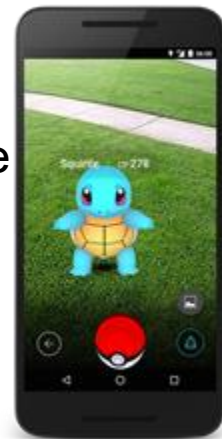
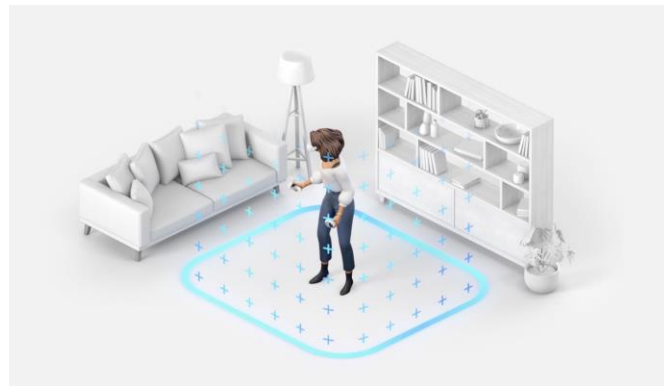
Disney's "Encanto"



Avatar: The Way of Water

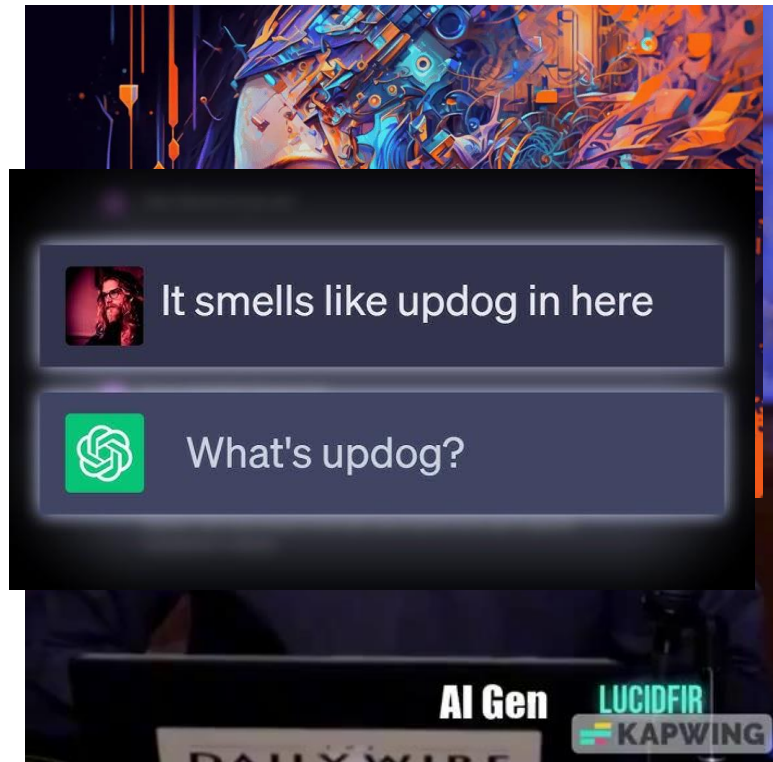
Virtual/Augmented Reality

- Virtual reality uses cutting-edge graphics, best-in-class hardware, and artistically rendered experiences to create computer-simulated realistic 3D worlds
 - consumers can play games, exercise, and interact with others inside the “Metaverse”
 - prodigious amounts of money invested, but not breakthrough yet – GAI has stolen the spotlight
- Augmented reality creates virtual elements “on top of” the real world, blending a digital reality with an existing one
 - Smartphone apps, e.g., Pokémon Go
 - “Smart glasses” and headsets like the Meta Quest 2



Powerful Developments in AI

- Generative AI (GAI) uses large datasets of existing patterns to create new content
 - AI Image generation with DALL-E, Stable Diffusion...
 - Can AI-generated art be copyrighted? What about the artists the AI was trained off of?
 - Text generation that passes the Turing test (ChatGPT)
 - The Turing test measures whether an AI is capable of "thinking" like a human being
 - Realistic voice-generation and Deepfake (facial mimicking) technology
 - Powerful threat in the wrong hands!!



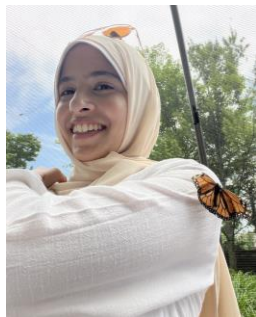
The Internet and Social Networks

- 4.8 billion social media users worldwide, representing
 - 60% of global population
 - 93% of internet users
- Social media ad spending market expected to increase to \$358 billion by 2026
- If someone signed up on a social platform at the age of 16 and they lived to 70, they would spend 5.7 years of their life using social media 🤖



CS15 is Not Just About Learning to Program

- In addition to programming, CS15 also introduces some of the societal context and implications of our field
 - “Socially-Responsible Computing” (SRC) in the Department and in CS15
 - SRC topics in CS15 include AI, privacy, social media, labor practices, and much more!
 - Our STAs:



Faizah Naqvi



Adam Mroueh



Katie Li

Opportunities/Threats of the Digital Age (1/5)

- Machines continue to replace human labor and decision-making
 - machines have increased human productivity while reducing demand for routine, repetitive, and dangerous jobs (factory work, coal mining,...)
 - as middle-skilled, task-intensive jobs disappear, income gap (“income inequality”) widens
 - but as new jobs are being created, old jobs are “upskilled” to be more interesting
 - Impacts not just blue collar jobs (factory work or driving), but x-ray reading, tax advising, bank lending, etc. ...
- GAI (e.g., Chat GPT, Co-Pilot) are affecting highly skilled jobs
- Education is key to economic survival



Considerations:

- Should there be a “robot tax”?
- A “guaranteed minimum income,” also called Universal Basic Income? (Andrew Yang ‘96)
- How to productively co-exist with GAI?
- “What is the future of work?”

Opportunities/Threats of the Digital Age (2/5)

- Dangers of yielding too much control to algorithms, some too complex to be understood by most people
 - instability in the stock market due to trading algorithms
 - autonomous vehicles (autopilot on planes, driverless cars and trucks...)
 - nuclear power plants and other infrastructure
 - “bias” in algorithms (facial recognition, mortgage lending, job placement...)
 - hallucination/fabrication by GAI that has zero knowledge of truth, reality – the ultimate BS artist



Opportunities/Threats of the Digital Age (3/5)

- Cyberfraud, Cybercrime, Cyberwarfare

- GAI offers a handful of dangerous tools to scammers— ability to accurately mimic a loved one's voice over the phone, deepfaked videos...
 - A threat to creatives, too. Actors, singers, and visual artists having their work imitated by AI. Big questions of copyright!

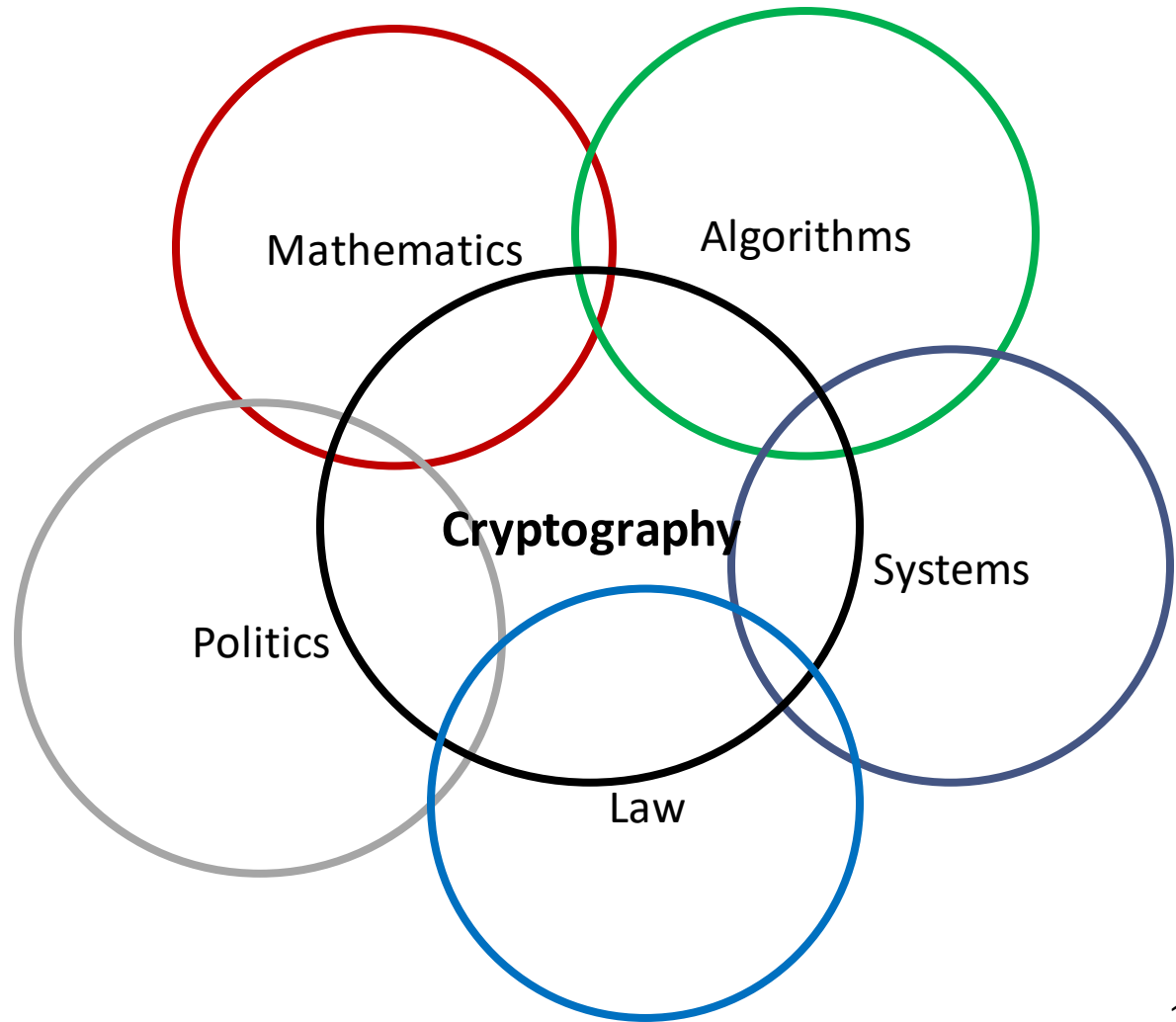
Scammers use AI to mimic voices of loved ones in distress

AI-generated music is going viral. But is it legal?

- culpability of social media in spreading mis- and dis-information thereby causing polarization and distrust (e.g., 2020 election, COVID-19 vaccines, organizing the January 6th insurrection)
- we keep experiencing huge data breaches
 - 200 million Twitter users' emails leaked
 - Digital war: Russia and Ukraine— Russia intruding into Ukrainian institution and reconnaissance networks, Ukraine forms a volunteer "IT Army" to fight with DDoS hacks and data breaches against Russian organizations and services
- offense has the advantage over defense
- schools in Russia, China, North Korea (at least) teach hacking... well beyond amateur hacking
- will the next war be fought by drones, and how can they be controlled?

- Brown is strong in cybersecurity technology and policy

Privacy and Security



Opportunities/Threats of the Digital Age (4/5)

- Big Data

- “data mining,” “machine learning,” “deep learning,” “reinforcement learning”...
 - statistics-based algorithms for detecting patterns, anomalies, etc., in Big Data
 - large language models for GAI: text, images, soon videos
- search engines
- real-time language translation
- facial recognition
 - can identify faces in crowd photos
- gesture recognition for user interfaces
- credit card fraud detection
- crime and terrorism anticipation
- but what about privacy in the age of the “surveillance state”?!?



“Heart of Stone” starring Gal Gadot (2023)

Opportunities/Threats of the Digital Age (5/5)

- Big data & personal privacy
 - threat to privacy represented by increasing storage of personally identifiable information – is there any real “anonymous data”?!?
 - Google, Facebook, Apple... and their data collection and use of that data – our digital footprint is permanent, and we have no control over how it is used
 - Sun Microsystems’s Scott McNealy – “privacy is dead, get over it!”
 - When Apps are free, **YOU are the product!**
- “Free speech” vs. (appropriate) censorship
 - hate speech, terrorism, violence
 - government-induced censorship (e.g., China, Saudi Arabia, Pakistan,...)
 - are social media content-neutral platforms or publishers, and what laws should apply to them?
- Need an educated government, citizenry
 - can we pass realistic laws to govern behavior?



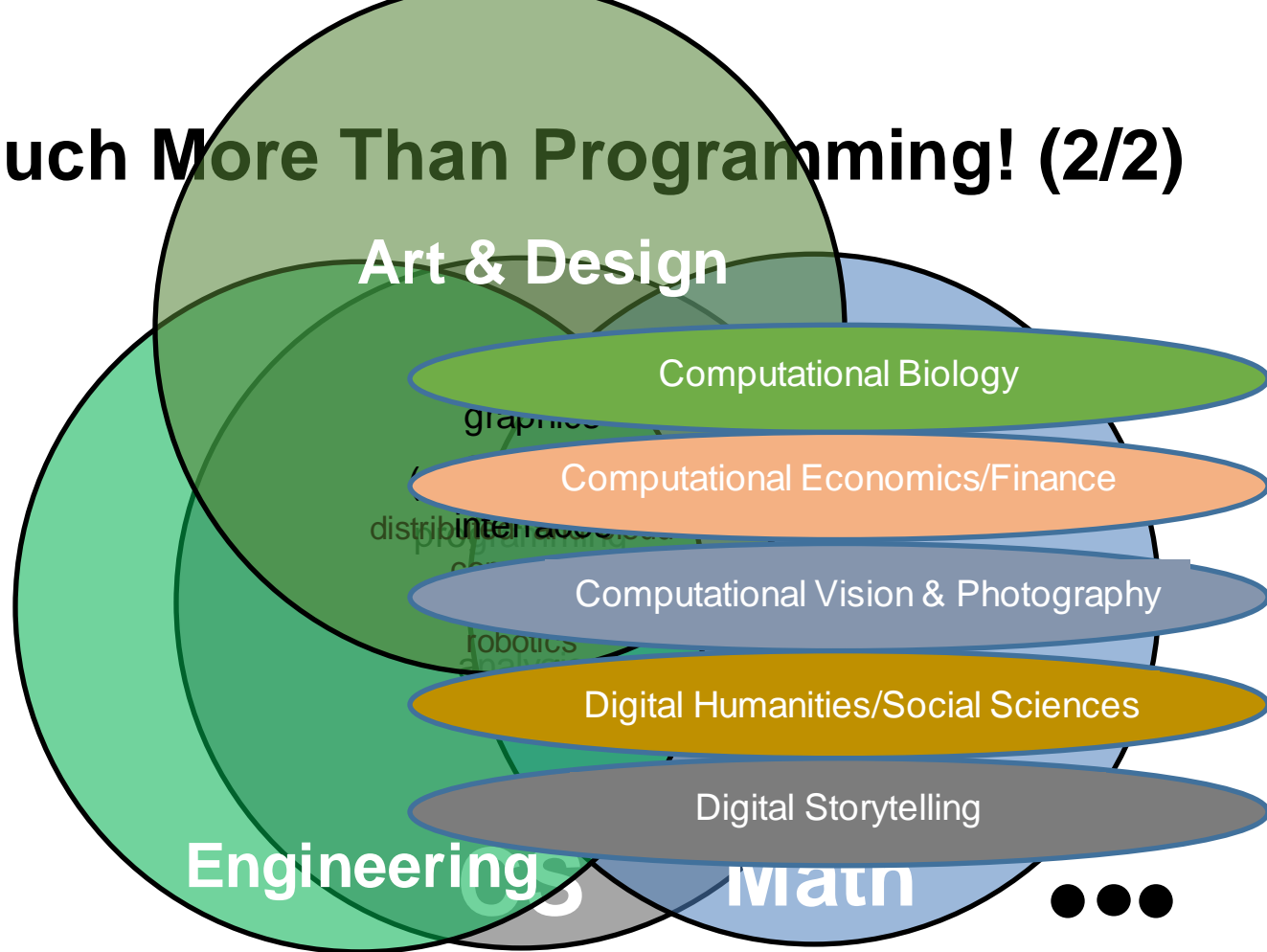
CS: So Much More Than Programming!

- Computers are our only universal machine, through the magic of software...
 - if you can program it, a computer can execute it
- **Programming is a means to an end, much like mathematics is... but they are both also fascinating topics in their own right!**
- Big push to learn how to “code,” but there is no “royal road” to programming or CS – it requires serious, sustained effort

Areas of Research at Brown

- **Algorithms and Theory** (Yu Cheng, Lorenzo De Stefani, Pedro Felzenszwalb, Sorin Istrail, Philip Klein, Tim Nelson, Roberto Tamassia, ...)
- **Artificial Intelligence** (Stephen Bach, Pedro Felzenszwalb, Amy Greenwald, George Konidaris, Michael Littman, Ellie Pavlick, Stefanie Tellex...)
- **Comp Bio** (Sorin Istrail, Sohini Ramachandran, David Laidlaw, Mark Nadel...)
- **Computing Education** (Kathi Fisler, Shriram Krishnamurthi, Alan M Usas, Milda Zizyte)
- **Data Science** (Karianne Bergen, Ugur Cetintemel, David Laidlaw, Ellie Pavlick...)
- **Machine Learning** (Stephen Bach, George Konidaris, Michael Littman, Daniel Ritchie, Ritambhara Singh, Eli Upfal...)
- **Programming Languages** (Kathi Fisler, Shriram Krishnamurthi, Robert Y. Lewis, Tim Nelson, Nikos Vasilakis)
- **Security** (Vasileios Kemerlis, Shriram Krishnamurthi, Anna Lysyanskaya, Steven Reiss, Tarik Moataz...)
- **Visual Computing** (Andy van Dam, John Hughes, David Laidlaw, Barbara Meier, Daniel Ritchie, James Tompkin, Bruce Campbell, Chen Sun...)
- And more... <http://cs.brown.edu/research/areas.html>

CS: So Much More Than Programming! (2/2)



Why Should You Study Computer Science?

- For fun and intellectual excitement
- Really exciting era is just beginning
 - CS still a young discipline, computers just starting to act intelligently
- Fundamental “mode of thought”
- Increasingly important component of all other fields
- Plenty of exciting and impactful jobs in established companies, start-ups, research labs, and academia



Welcome To CS15!



“May the odds be ever in your favor.”

Welcome to CS15 in Salomon 101!

- We encourage you to download the PowerPoint slide deck before lecture and bring your laptop – lets you see clearly and annotate
 - <http://cs.brown.edu/courses/cs015/>
- We record and give you web access to every lecture
 - for review
 - in case you must miss a synchronous lecture
 - PowerPoint slides come with associated recording via Panopto



CS Department's Hardware

- The Sunlab: 80 PCs running Debian Linux
- All CS15 work will be done locally on your own computer
 - using **GitHub** to acquire and store code
 - **IntelliJ IDEA** – integrated development environment (IDE) we will use to write, compile, and run code
 - **Loaner laptops available** -- Office of Information Technology (OIT), Financial aid offered for eligible purchases (BCMS, Financial Aid Office)
- Working From Home setup materials and instructional video released after lecture
 - come to TA hours for help setting up this software!



HTAs Lexi and Sarah
working locally

CS15 is All That

- **NOT** a course about video games or game design
- Uses games as a domain to teach Object-Oriented Programming (OOP)
 - most common current programming methodology
 - Brown was earliest to switch to Java for intro courses over two decades ago
 - still a dominant web programming language (e.g., Google's Android)
- Teaches fundamental *problem-solving skills* useful in all disciplines
- Provides introduction to computer science concepts
- Is **intense**, but **fun**, especially with interactive graphics



Who is CS15 For?

- Students with varying levels of programming experience, including *NONE!*
 - however, CS15 still requires a **serious** commitment
- Most students have **little or no** programming experience, including the TAs and HTAs when they took the course!
 - let's visualize this!
- Prospective CS concentrators, who will go on to CS200 course
- This is **not a weeding-out course**, but it is time-consuming
 - don't worry!! We expect lots of confusion in the beginning. All 64 TAs are here to help you through that initial confusion!

CS15 isn't About Getting the Correct “Answer”

- It's about the ***process***, not just the final product!
 - design
 - planning efficient, effective designs for program structure
 - investing upfront (e.g., reviewing materials) saves time in long run
 - implementation
 - coding incrementally
 - debugging
 - diagnosing and fixing bugs/errors in code effectively

Diversity & Inclusion in CS15

- CS15 welcomes all, helps you succeed, and aims to build **community**. These additional groups are also here for you:
- Mosaic+ mosaic.plus.brown@gmail.com
 - *“created to foster Community, inspire Innovation, and provide opportunity to underrepresented minority students.”*
- Women In Computer Science (WiCS) wics@lists.cs.brown.edu
 - *“formed by female undergraduate students at Brown in the late 1980s, The goal of WiCS is to increase the participation of women in the field of Computer Science.”*
- Women in Science & Engineering (WiSE) WiSE@brown.edu
 - *“to encourage women who study in all science and engineering fields, by building a community of like-minded scholars that provides peer support on their journey to becoming successful scientists at Brown and beyond.”*
- Our own CS15 mentorship program!
 - more on this later



Why Java



- Supports interactive OOP
- Simple, clean, and beginner-friendly
- Allows platform-independence: write once, run everywhere (in principle)
- One of the most prevalent languages in industry today, e.g., Android, web servers (others include C, C++, C#, Python, Ruby, etc.)
- Note: *not* the same as JavaScript, a less purely object-oriented language used commonly in web applications
- OOP is one of several programming paradigms – CS17 uses ReasonML and Racket for “functional programming”
- CS200 uses both OOP and functional programming

Course Mechanics (1/4)

- **No** quizzes or exams!
 - no exam time pressure
 - no “grading on a curve”: you do the work, you get the grade you deserve! Thus A is by far the most common grade
- 9 Assignments
 - programming assignments, some of which have a design component
 - from brief homework to Tetris and beyond!
 - choose from a selection of final projects, or create your own “indy” project
 - all programs must meet a baseline level of functionality to receive credit, lots of room for “bells and whistles” for fun and extra credit
 - **all programs must be handed in with baseline functionality by end-of-semester!**

Course Mechanics (2/4)

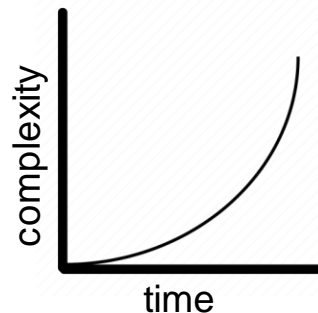
- Assignments are graded on a hand-in schedule
 - most assignments have early, on-time, and late hand-in
 - early hand-in: 2% increase to your grade
 - late hand-in: 8% decrease from your grade
 - **all assignments must be handed in before the end of the course**
- Weekly discussion/lab sections
 - graded on mini-assignments and participation
 - 12% of your final grade
- Code Debriefs
 - chance to explain code to a TA—Great skill!
 - meet for two of the last four assignments
 - 8% of final grade
- TopHat questions during lecture
 - interactive questions to improve engagement and comprehension
 - 5% of final grade



Course Mechanics (3/4)

- Keys to success

- increase in program size and complexity throughout the semester
- you can't procrastinate and then cram, unlike in some other courses
- **start early, start today, start yesterday!!!**
- other courses don't teach you to tackle programs of this complexity



- TA Hours

- 59 UTAs and 5 Head TAs
- **180+ TA hours** of personalized help per week!!!
 - more than in any other course!
 - everyone struggles sooner or later, including the TAs – part of the learning process
 - we strongly encourage you all to go to TA hours and get to know the TAs - it is integral to the course (and NOT a sign of weakness!)

Course Mechanics (4/4)

- CS15 thrives on your feedback
- Questions *highly* encouraged during lecture! And we will add TopHat questions next week...
- We provide a lot of written material; YOU are responsible for digesting all of it



Partner Project

- Doodle Jump
 - last year's feedback form had high support for partner projects
 - complete collaboration allowed between partners—can discuss and share code
 - Collaboration is an invaluable skill in the world of CS



Alternatives to CS15 (1/2)

For Concentrators & Non-concentrators:

- CS17 (fall semester) – John Hughes
 - also, no prior experience required
 - multiple programming paradigms
 - multiple programming languages
 - Racket, ReasonML
 - mastery, not mystery → no magic
 - focus on problem-solving skills/strategies
 - emphasis on *abstraction* and *scale*
 - integrate programming with analysis of algorithms
 - multiple application areas (AI, databases, etc.)
 - pair programming for labs and projects
 - final exam
 - for more information on other CS courses:
<http://cs.brown.edu/degrees/undergrad/whatcourse/>



Summary of CS15/17 Choice

- Both will adequately prepare you for upper-level courses
- No prior experience needed for either, similar work loads
- Different material covered
 - CS15 – Object-Oriented Programming, CS17 – Functional Programming
 - CS15 is more practice-oriented, CS17 is more foundations-oriented
 - CS15 celebrates magic, while CS17 emphasizes no magic
 - CS15 has **little** reliance on TA support code, but uses JavaFX extensively
 - All CS15 programs are interactive: use GUIs and 2D shapes
- Less pair programming and collaboration on projects than CS17
 - but **no** tests
- CS15: program interactive games; skits in lecture
- Pick based on your taste and what appeals to you – you can't go wrong!
- Both CS15 and CS17 feed into CS200 in spring semester
 - CS15 will get quick 2-week intro to functional programming, CS17 to OOP and Java

Alternatives to CS15 (2/2)

For Concentrators & Non-concentrators:

- CS0111 (Fall + Spring) – Kathi Fisler
 - no prior experience required
 - the first in a sequence that spreads the foundational concepts over three courses rather than two
 - *“allow more time to combine CS with other studies & mastering the fundamentals”*
 - functional programming and imperative programming
 - learn Pyret and Python
 - integrates programming with data science and discussion of use of digital information
 - less intensive workload
 - option to do extra work at the end for students who want to feed directly into CS200

Collaboration (1/6)

- Brown's Academic Code
 - *“Academic achievement is evaluated on the basis of work that a **student produces independently**. A student who obtains credit for work, words, or ideas that are not the products of his or her own effort is **dishonest and in violation** of Brown's Academic Code. Such dishonesty undermines the integrity of academic standards of the University. Infringement of the Academic Code entails penalties ranging from reprimand to **suspension, dismissal, or expulsion** from the University.”*

Collaboration (2/6)

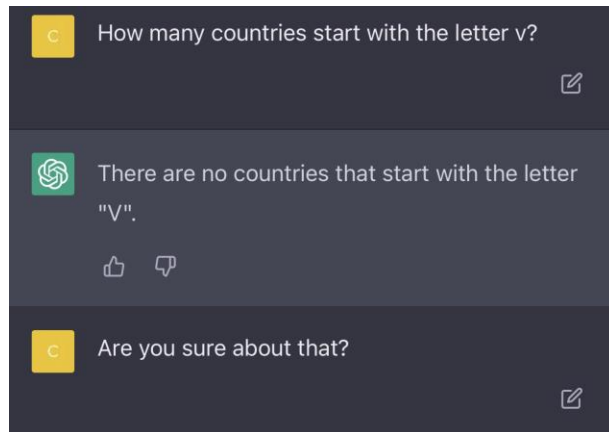
- CS15 Collaboration Guidelines
 - Lectures
 - **always** allowed to review and discuss with your peers!
 - Mini-assignments
 - collaboration and discussion are **allowed and indeed encouraged**
 - Labs / Sections
 - collaboration and discussion are **allowed and again encouraged**

Collaboration (3/6)

- Collaboration on Programming Assignments
 - first month of semester, **no** collaboration on assignment programming or debugging
 - rest of semester, more flexible rules around debugging with peers
 - the partner programming assignment (Doodle Jump) is designed and implemented with a partner
 - demonstrate your understanding of your project in a **Code Debrief!**
 - 8-minute conversation with a TA about your implementation
 - not intended to be scary — a great opportunity to practice talking through your code!
 - must read and sign detailed [CS15 Collaboration Policy](#) during first assignment--includes guidelines on collaboration with peers **and** AI

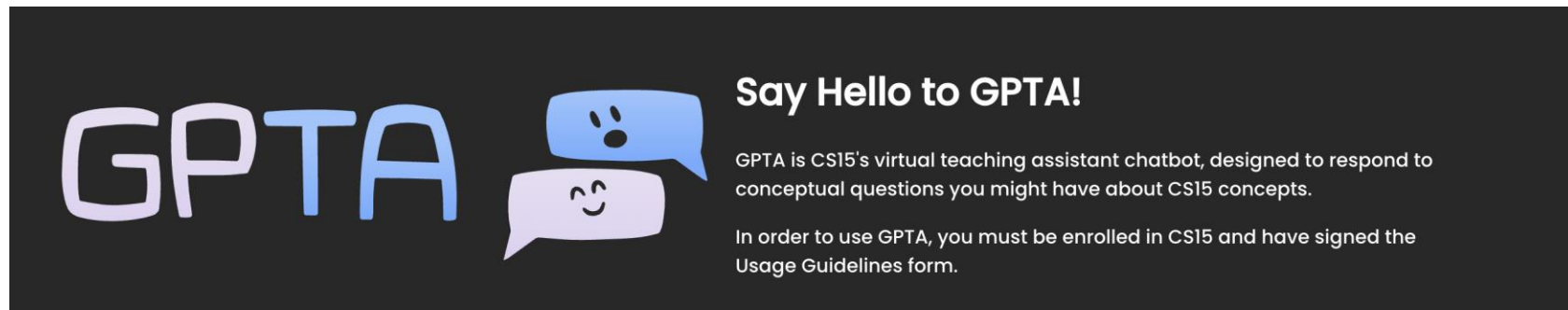
Generative AI (ChatGPT, Co-Pilot,...) (1/2)

- How many of you have used GAI?
- Unexpected by even the pros, exponential improvement this past year
- False confidence: GAI's tendency to present *everything* confidently as fact
 - GAI gurus are working on: guardrails, revealing sources, ingesting curated training data, etc.
- Like any tech, can be used for good or bad, learn how to have a productive co-existence
- MatLab and Wolfram Alpha can do the entire ugrad math curriculum
 - learn the fundamentals so you have a better sense of what it is doing, and what you can and can't trust
 - same way you can't rely on your peers in this course
- Your parents are paying you to learn, not just to get a good grade – need to have mastered the fundamentals for future courses, jobs, etc.
 - if you cheat, you cheat yourself



Generative AI (ChatGPT, Co-Pilot,...) (2/2)

- CS15 provides a curated version of ChatGPT for answering questions, giving coding examples, but not allowing assignment-specific questions
 - A 24/7, virtual TA! (affectionately named GPTA)
 - we log your sessions
 - all programs are checked for code similarities against ChatGPT and other CS15 submissions



Collaboration (4/6)

- MOSS (Measure of Software Similarity)

- Stanford-hosted AI software used to detect plagiarism – it signals undue similarity and we hand-check the code
- used across industries in multi-million dollar lawsuits to protect intellectual property
- every year, MOSS finds multiple collaboration violations (we check multiple years!)
- last year, over a dozen cases, all found guilty by UH's official Academic Code Committee
- punishments typically reprimand, possibly directed NC; always parental notification
- MOSS is *very good* at what it does – don't even think of trying to outwit it! (which is more work than doing the assignment!)
- we also check the web

If ever in doubt about what is allowed, ask a TA!

Better to NC an assignment or even the course than being accused (and likely convicted)!

Note: we have a Regret Policy

Collaboration (5/6)

The issue of collaboration in intro CS courses has been in the news in past years:

Possible cheating uncovered in popular
Harvard computer class

By Travis Andersen and Brian MacQuarrie Globe Staff May 05, 2017

The Boston Globe

As Computer Coding Classes Swell, So
Does Cheating

By JESS BIDGOOD and • MAY 29, 2017

The
New York
Times

**ChatGPT Can Get Good Grades.
What Should Educators Do about
It?**

AI can generate clear, concise text—but people still need to learn how to write

By Lauren Leffer on August 25, 2023

SCIENTIFIC
AMERICAN®

Collaboration (6/6)

- Illegal collaboration is **not** worth the risk
 - start early and get help when you need it! Lots of resources available to help you succeed in this course



What is Programming?

Aspects of Programming, Computer Languages, Objects and Object-Oriented Programming



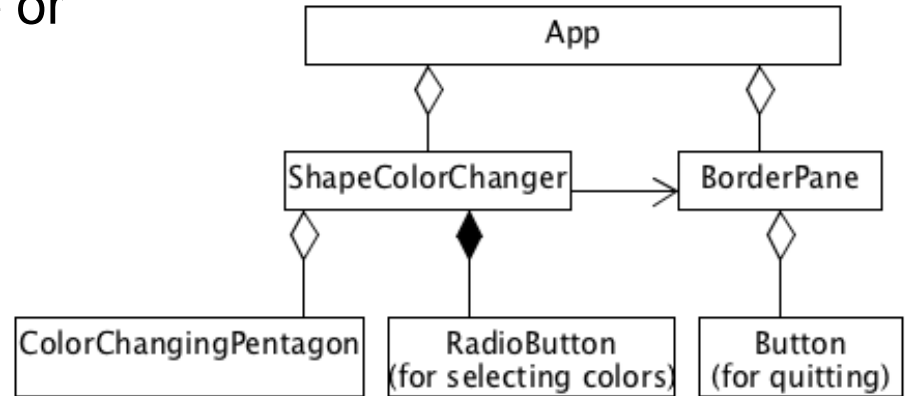
Many Aspects of Programming

- Programming is **controlling**
 - computer does exactly what you tell it to do – literal minded idiot savant
- Programming is **problem solving**
 - always trying to make the computer do something useful
 - e.g., finding an optimal travel route
 - methodology is applicable to other fields
- Programming is **creative**
 - must find the best solution out of many possibilities
- Programming is **modeling**
 - describe **salient** (relevant) properties and behaviors of a system of components (objects)
- Programming is **abstraction**
 - identify important features without getting lost in detail
- Programming is **concrete**
 - must provide detailed instructions to complete task
- Programming is a **craft**
 - a bit like architecture, engineering – disciplined and creative craft for building artifacts



What's a Program? (1/3)

- Model of complex system
 - model: simplified representation of important features of something, either tangible or abstract
 - system: collection of collaborating components



What's a Program? (2/3)

- Sequences of instructions expressed in specific programming language
 - syntax: grammatical rules for writing instructions
 - semantics: meaning/interpretation of instruction

What's a Program? (3/3)

- Instructions written (programmed/coded) by programmer
 - coded in a specific programming language
 - *programming languages* allow you to express yourself precisely unlike *natural (human) language* that thrives on “shading”, nuance, ambiguity, implicit context...
 - algorithms are 100% literal, cannot have ambiguities
- Real world examples
 - Banner, email, video games, smartphone and its apps; embedded computers in appliances and vehicles, ATMs...A Tesla is a display-driven computer on wheels
- Executed by computer by carrying out individual instructions

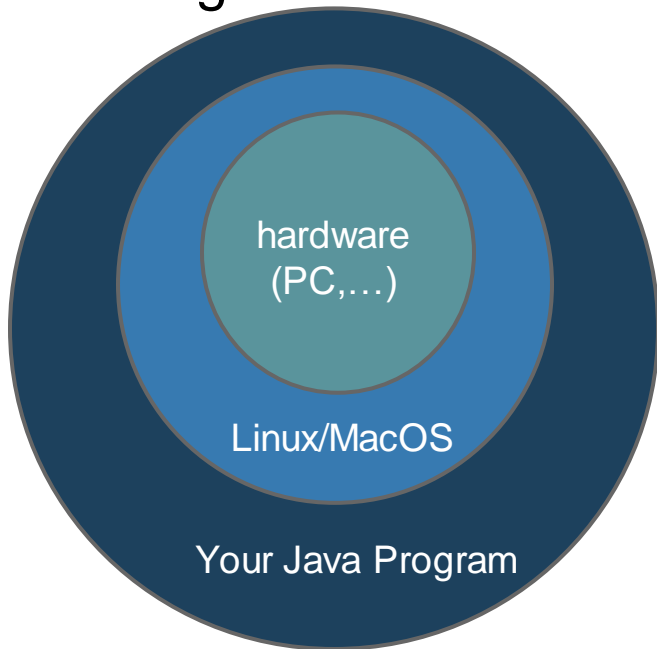
Java Programs



- CS15 uses *Java*
 - Java was developed by Sun Microsystems (absorbed by Oracle)
 - the Sunlab was named for the desktop computers that it held for over a decade
 - it is meant to run on many “platforms” without change, from desktop to cell phones
 - platform independence
 - but Java isn’t sufficient by itself: many layers of software in a modern computer

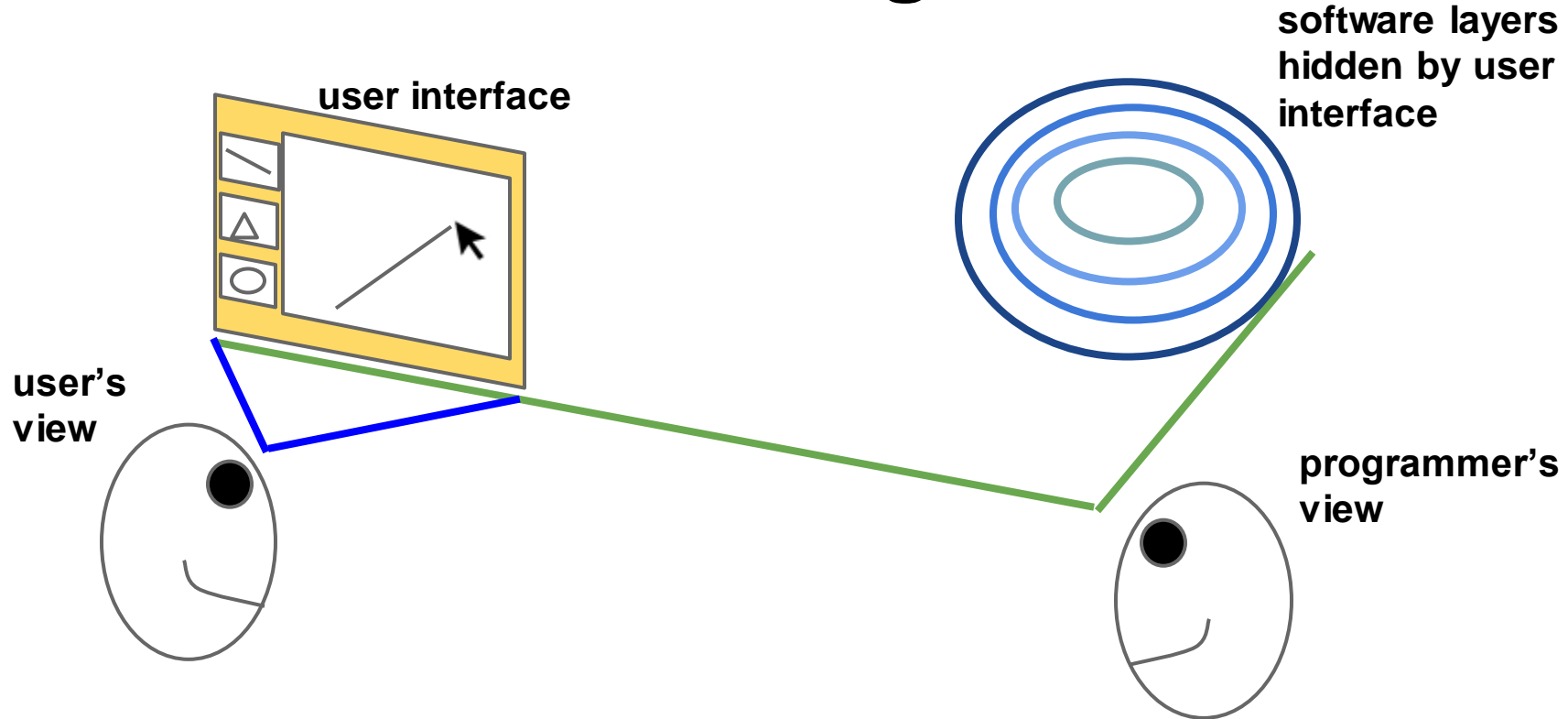
The Computer Onion

- Layers of Software
 - cover hardware like an onion covers its core
 - make it easier to use computers
 - organized into libraries and programs



In CS15, we only deal with the outermost layer

Two Views of a Program



Programming Languages (1/2)

- Machine language – computer's native language
 - sequence of zeroes and ones (binary)
 - different computers understand different sequences
 - too hard for humans to understand (01010001...)
- Assembly language
 - symbolic but still one-to-one with machine language
 - still hard for humans to understand:
 - `ADD.L d0, d2`
 - assembly language taught in CS33

Programming Languages (2/2)


- High-level languages
 - FORTRAN, C, C++, Java, C#, Python, JavaScript, Scheme, Racket, Pyret, ML, OCaml, etc.
 - high level: each instruction is composed of many low-level instructions
 - closer to English and high school algebra

```
hypotenuse = Math.sqrt(leg1 * leg1 + leg2 * leg2);
```
 - much easier to read and understand than Assembly language
 - unlike machine and assembly language, it is machine-independent

Running Compiled Programs (1/2)

- In CS15, code in a high-level language, Java
- But each type of computer only “understands” its own machine language (zeroes and ones)
- Thus must translate from Java to machine language
 - a team of experts programmed a translator, called a “**compiler**”, which translates the entirety of a Java program to an *executable file* in the computer’s native machine language

Running Compiled Programs (2/2)

- Two-step process to translate from Java to machine language:
 - compilation: your program  executable
 - execution: run executable
 - machine executes your program by “running” each machine language instruction in the executable file
 - not quite this simple “underneath the covers” – “Java bytecode” is an intermediate language, a kind of abstract machine code

Object-Oriented Programming (1/2)

- OOP: the dominant way to program, yet it is over 50 years old! (Simula '67 and Smalltalk '72 were the first OOPLs)
 - Dr. Alan Kay received ACM's Turing Award, the “Nobel Prize of Computing,” in 2003 for Smalltalk, the first complete dynamic OOPL
- OOP was slow to catch on, but since mid-90's it's been the dominant programming paradigm
 - but it isn't the only useful programming paradigm...functional programming is increasingly common, as is hybrid programming combining OOP and FP
- CS17 and 19 teach functional programming in
 - Racket, ReasonML, and Pyret
- CS200 will also teach some functional programming – you won't miss out

Object-Oriented Programming (2/2)

- OOP emphasizes objects, which often reflect real-life objects
 - have both properties and capabilities
 - i.e., they can perform tasks: “they know how to...”
- Look around you... name that object!



OOP as Modeling (1/3)

- In OOP, model program as collection of cooperating objects
 - program behavior determined by group interactions
 - group interactions determined by individual objects
- In OOP, objects are considered *anthropomorphic*
 - each is “smart” in its specialty
 - e.g., bed can make itself, door can open itself, menu can let selections be picked
 - but each must be told when to perform actions by another object – so objects must cooperate to accomplish task

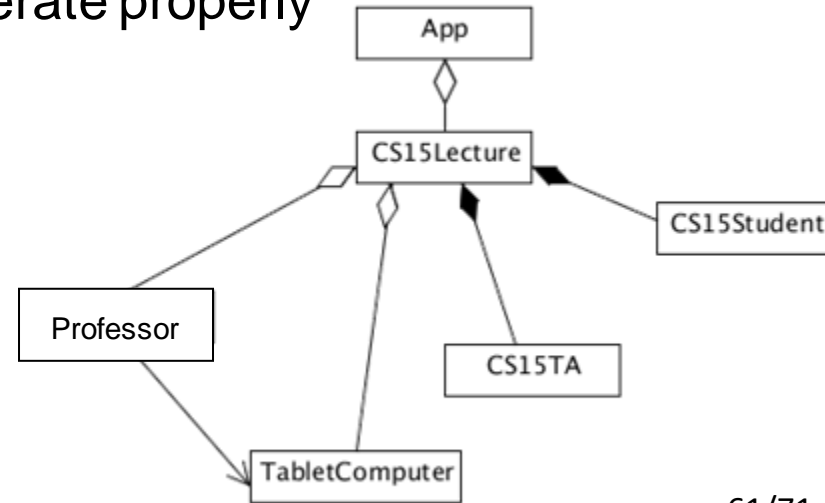
OOP as Modeling (2/3)



- Each object represents an *abstraction*
 - a “black box”: hides details you do not care about
 - allows you as the programmer to control programs’ complexity – only think about primary features

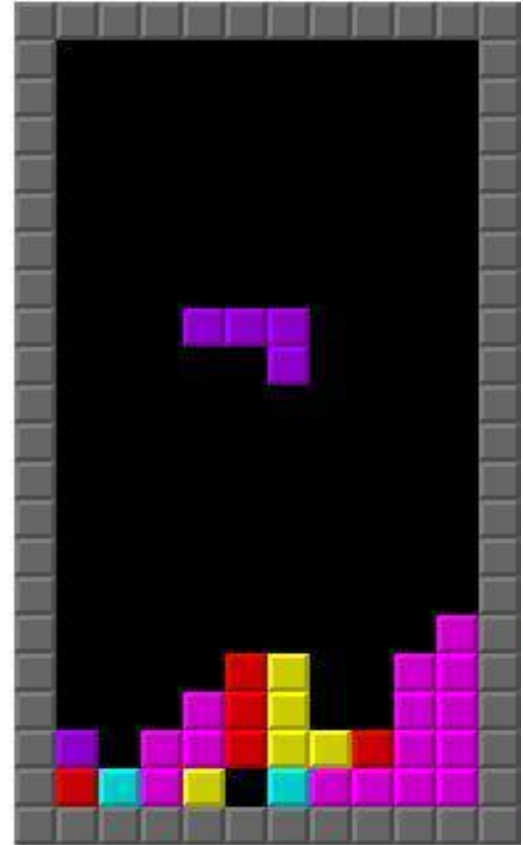
OOP as Modeling (3/3)

- So, write programs by modeling the problem as system of *collaborating components*
 - you determine what the building blocks are
 - put them together so they cooperate properly
 - like building with smart Legos, which are pre-defined, some of which you design!
 - containment/association diagrams, like the one shown here, are a great way to help model your program!



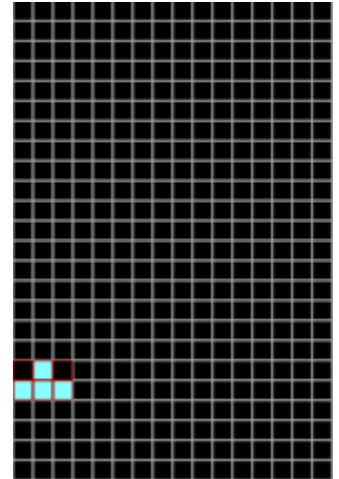
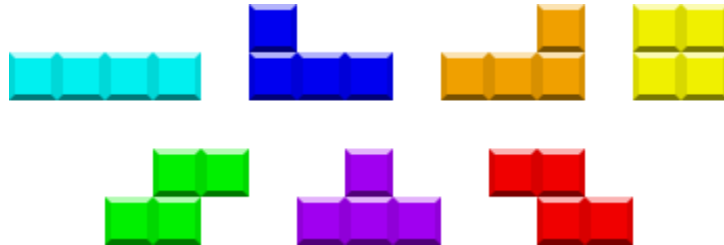
Example: Tetris (1/3)

- What are the game's objects?
- What properties do they have?
- What do those objects know how to do?



Example: Tetris (2/3)

- What are the game's objects?
 - piece, board
- **Properties:** What attributes and components do they have?
 - piece
 - orientation
 - position
 - shape
 - color
 - tiles
 - board
 - size
 - rows



Example: Tetris (3/3)

- **Capabilities:** What do those objects know how to do?
 - piece
 - be created
 - fall
 - rotate
 - stop at collision
 - board
 - be created
 - remove rows
 - check for end of game

Announcements (1/2)

- **NO ADMITTANCE TO TUESDAY'S LECTURE WITHOUT A KN95**
- If you are even considering taking the course, register or add it to your primary cart on C@B by this weekend
 - you will not get course emails unless you do this!
- Lab/Section form released today (9/7) – instructions sent via email
 - Updated lab assignments will be sent out by **Sunday on Ed** (9/11)
 - first lab occurs Tuesday-Thursday (9/12-9/14)
- We will send an email with instructions to set up for CS15
 - set up IntelliJ code editor **before your lab time**
 - set up Top Hat lecture quiz software **before Tuesday's lecture**
 - join Ed online Q&A forum as soon as you can

Announcements (2/2)

- RISD and other non-Brown undergrads please speak to Andy and HTAs after class
- Head TA (HTA) Hours this weekend – come if you have any questions about CS15 or just want to say hello!
 - Friday 3-5pm in CIT 210
 - Sunday 6-8pm in CIT 210
- Check the [website](#) and mark your calendar for Code-Alongs!!
 - Code-Alongs let you code examples alongside TAs and help give you a head start on projects 😊
 - They are led by TAs **most weeks** on with another date and time throughout the week (not this week) – see the course website!
- Follow us on [TikTok](#) @cs15.brown
- Check the course website at <http://www.cs.brown.edu/courses/cs015> and your email daily
- If you are undecided about which CS intro course to take, this document is a good reference:
 - <https://cs.brown.edu/degrees/undergrad/whatcourse/>

Hope you're excited for a great semester!

