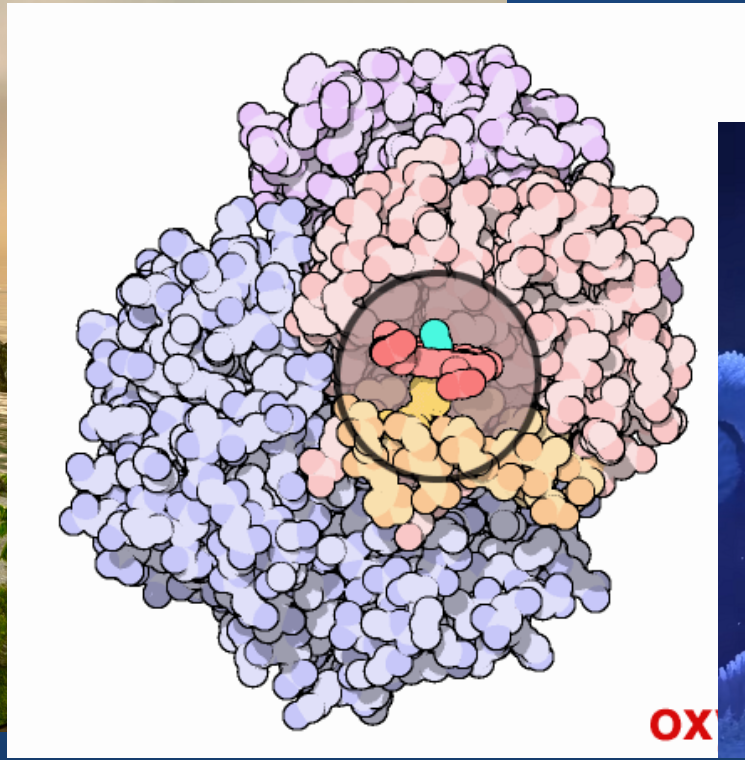
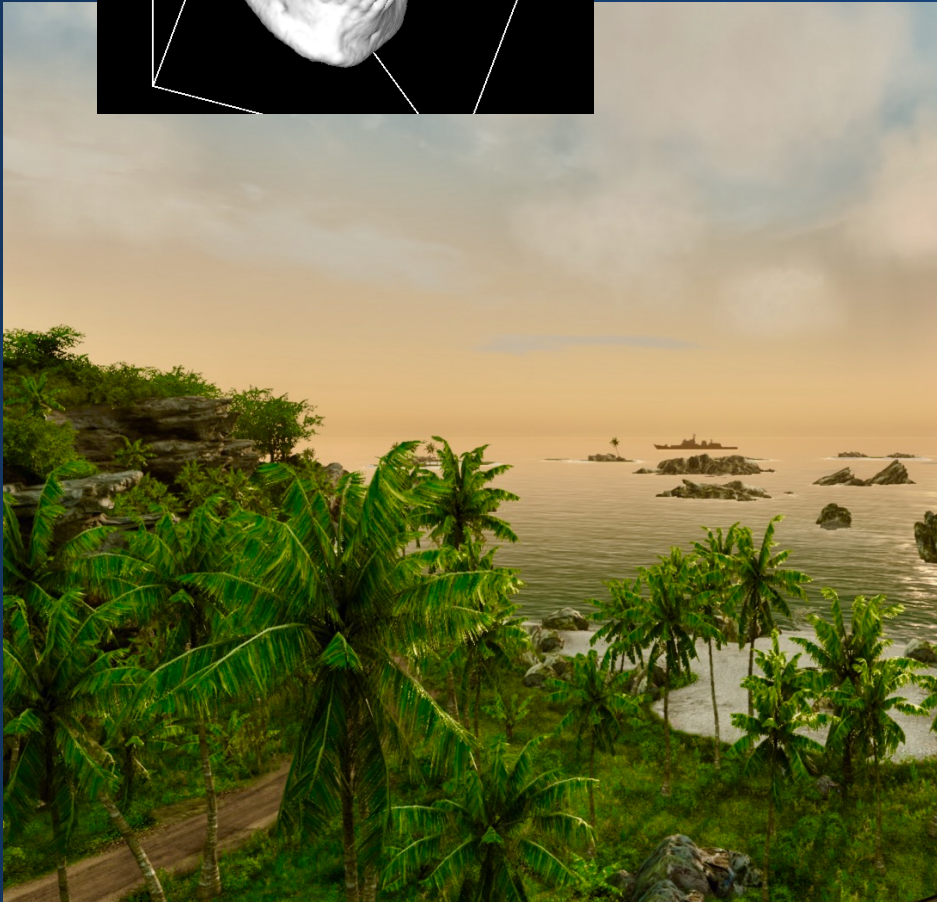
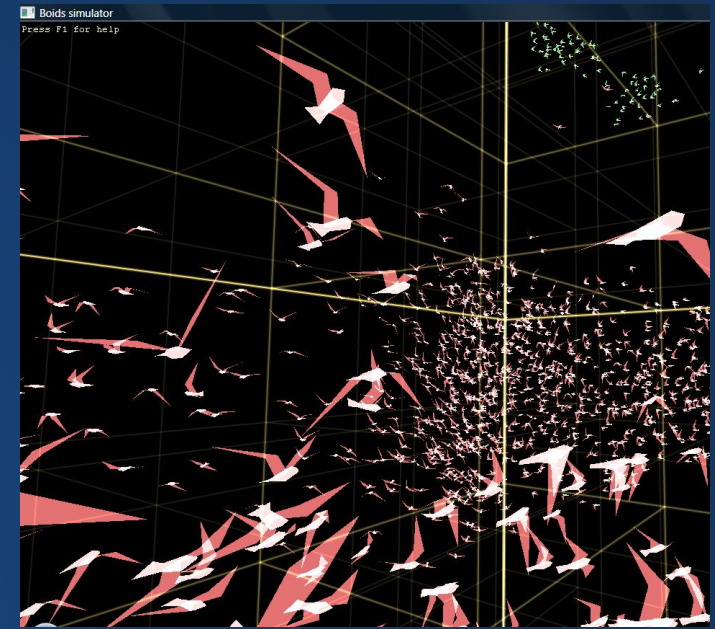


CS324E

Elements of Graphics



Me!

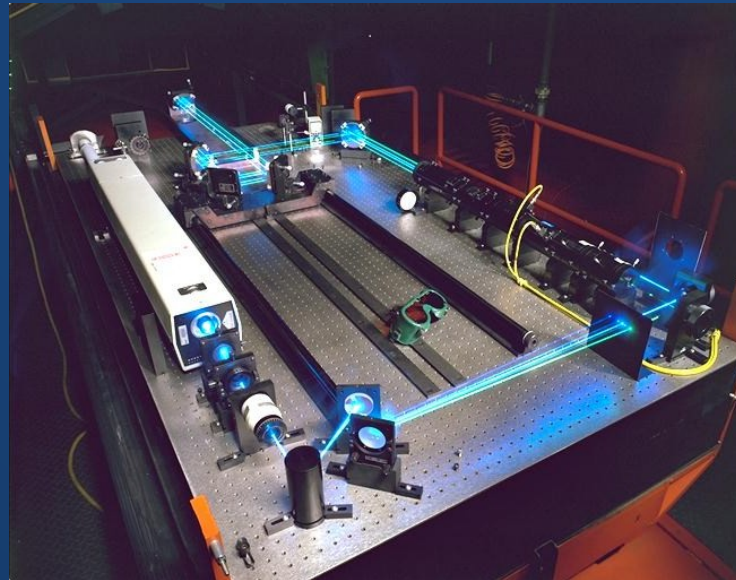
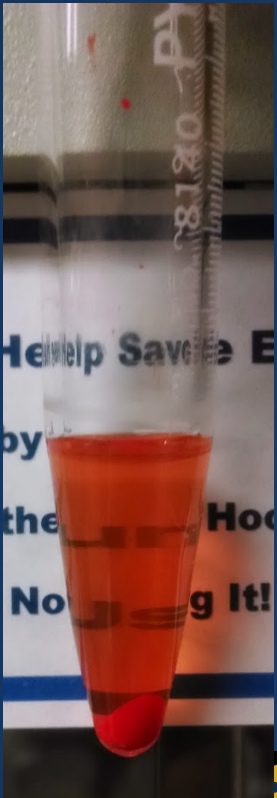
Name: Kevin Song (he/him)

Email: kcsong@utexas.edu

Office Hours: After Class on M/W/F (tentative)



Undergrad: B.S. Chemical Sciences



Things I've Worked On

Field	Project
Geospatial Imaging	Parallelization of Random Forests for Tree Carbon Estimation
Consumer Medical	Machine learning-based pill identification algorithm
Sociology	Simulation of collaboration and productivity in academia
HPC Systems	Bitflip error tolerance in high-performance parallel computing
Computational Bio	Flexible Protein Docking + Folding
Medical Imaging	Virtual surgery and reconstruction of injured pediatric skulls
Public Policy	Acceleration of agent-based simulation for COVID-19
Compiler Design	Reverse compilation of PTX for custom GPU architectures

Computing lets you work in almost any field!

TA!

Class Format

We'll focus on group exploration and discussion rather than lecture.

Typical Class (75 mins):

- Review of last class (5-15 min)
- Do Two Times:
 - New Material (15-20 min)
 - Hands-on Activity (15 min)

Collaboration is **encouraged** for these hands-on activities!

Hands-on activities will be due at 5pm. Graded on completion.

Speed

This class is going to go *fast*.

Term Type	In-Class Minutes	Calendar Duration
Long Semester	2100	14 weeks
Half-Summer	1875	5 weeks

- Stay on top of the course!
- If you don't quite get what's going on, come to office hours
- Be ready for class
- **Read syllabus and schedule** to answer certain questions
- Ask, read, and answer questions on EdStem

Hands-On Presentations

During the review of the last class's material, I may, time permitting, review one student's hands-on activity from the previous class.

- Will be presented anonymously unless requested otherwise
- Please volunteer and show off your work!
- Share your code with others!
- Use this as a chance for me to debug your code if you weren't quite able to get it working.
- Please let me know you want me to present your work. Send DM on EdStem or attach a comment on Canvas.

Grading Structure

Three primary means of assessment in this class:

- **Being present in class**
- **Hands-on assignments**
- **Projects**

This class is primarily based around discussion, activities, and learning from each other.

I consider attendance to be mandatory.

Attendance measured by index card submission.

You are given four days of absence, no questions asked. For each additional day you are absent, **your final grade is lowered by one letter.**

Miss 8 days and you are guaranteed to fail the course.

Unless...

Attendance Makeup

You can make up unexcused absences by writing an essay discussing an interesting topic covered in the class you missed.

- Paper should:
 - Have your name + EID
 - Have the date of class missed
 - Be 500 words
 - Be double-spaced
- **Tell me** you're submitting this so that I can grade it (send mail or a message on EdStem, don't rely on Canvas)
- Must be submitted within 48 hours of the absence.

Last three days of class cannot be excused either through grace or essay--you need to let me know if you can't make it to those.

Participation

Will be based on the contents of your daily index card submission.

Each index card will ask you for the following:

- Name + EID
- One thing you learned in class
- One question you have about something in class
- Any other questions/comments about that day

Reasonable answers will get credit.

I may also dock participation if you are excessively off-task or disruptive in class, but I hope to never do this.

In-Class Assignments

Short assignments which you will have some time to do in class

Expected time: 20-40 mins per day

Due at 5:30pm after class (?)

Graded on 5/3/1 scale.

Will be due based on material covered in class. Due dates on Canvas will reflect *optimistic* times, but if you miss class, check with Lectures Online (or ask on EdStem).

Please try to work on these during the dedicate in-class periods!

Projects

Extended take-home assignments which require some in-depth thinking.

Main class projects are solo. You may discuss ideas with each other, but you may not share code. Four of these projects.

Final project will have the option of being in groups and require a final presentation.

3 slip days, cannot be used on final project. Projects submitted late without slip day usage incur a 33% earned-grade penalty per day.

Word of advice: the July-August transition is going to be a bit of a mess. I recommend using the slip days for at least some of these projects.

Collaboration, Cheating, and LLMs

Collaboration

This class is heavily collaborative. This makes it easier to accidentally fall on the wrong side of the rules.

In-Class Assignments

- Do not submit identical assignments.
- Otherwise, go nuts!
 - Share ideas with each other.
 - Post your assignments on Discord.
 - Work collaboratively on a cool idea, then each apply your own twist at the end.
- If you use outside code, cite it.
- Please no AI-generated code.

Projects

- Do not submit things that are not your own work (e.g. someone else's project, code you find on the internet, etc.)
- StackExchange snippets may be used as long as they are appropriately cited and referenced within your code.
- Do not look at someone else's code or let them look at yours.

Citing Code

Put a comment that clearly specifies the following:

- What part of this code are we citing? A function? A class? A few lines?
- Where can the original source be found?
- Why do you believe you have the right to use it?

```
1 // This function was written by WiSaGaN on SO
2 // https://stackoverflow.com/a/11237235
3 // Used under https://creativecommons.org/licenses/by-sa/4.0/
4
5 int max1(int a, int b) {
6     if (a > b)
7         return a;
8     else
9         return b;
10 }
```

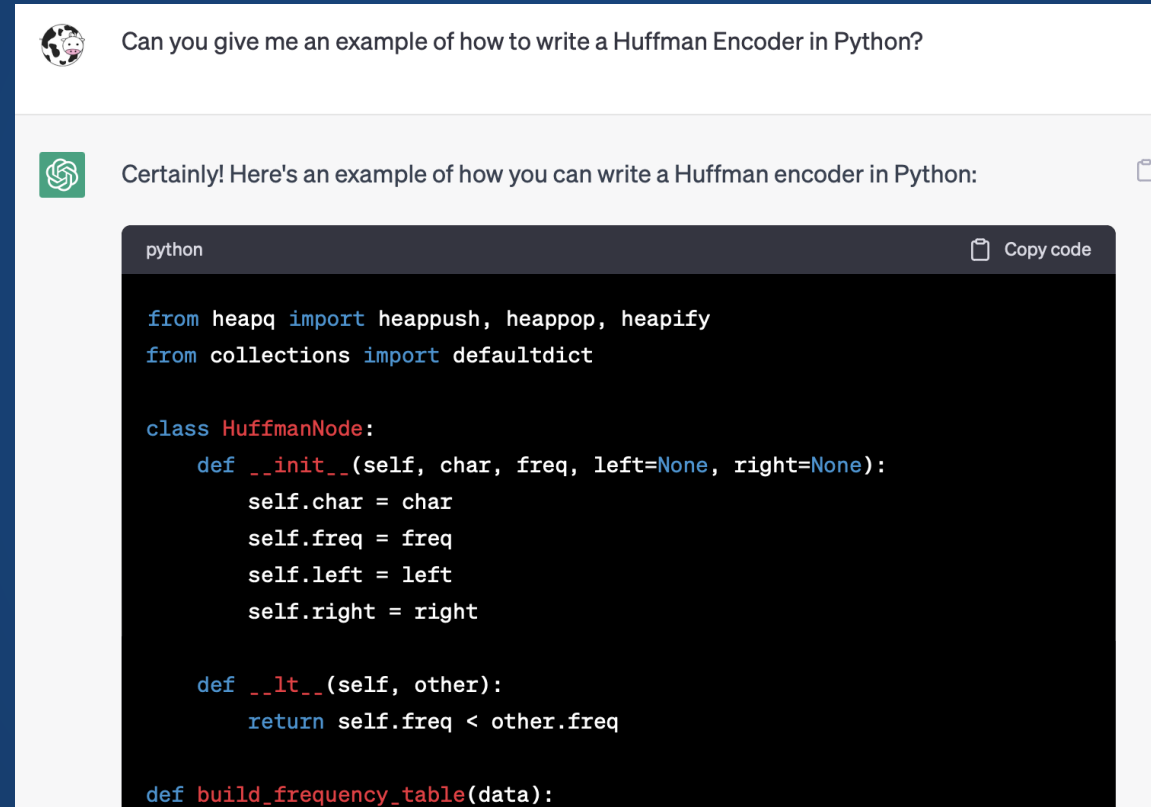

Cheating

If you cheat on an assignment, the minimum consequences are:

- Zero on that assignment
- Minimum 1/3rd letter grade reduction of final grade
- Referral to Student Judicial Services

LLMs (GPT, Bard, and Friends)

Bard, GPT, and friends are amazing tools.



The screenshot shows a chat conversation. The user asks: "Can you give me an example of how to write a Huffman Encoder in Python?". The AI responds: "Certainly! Here's an example of how you can write a Huffman encoder in Python:". Below the response is a code block with Python code for a Huffman encoder. The code includes imports for heapq and collections, a HuffmanNode class with __init__ and __lt__ methods, and a build_frequency_table function.

```
python Copy code  
  
from heapq import heappush, heappop, heapify  
from collections import defaultdict  
  
class HuffmanNode:  
    def __init__(self, char, freq, left=None, right=None):  
        self.char = char  
        self.freq = freq  
        self.left = left  
        self.right = right  
  
    def __lt__(self, other):  
        return self.freq < other.freq  
  
def build_frequency_table(data):
```

Are going to change the way we program computers.

Also come with hazards...

Bard

For example, January has 3 letters, February has 5 letters, March has 5 letters, April has 5 letters, June has 4 letters, July has 4 letters, August has 5 letters, September has 7 letters, October has 7 letters, November has 7 letters, and December has 8 letters.

What does GPT-3.5 think of this?

The statement is accurate in its analysis of the number of letters in each month's name. It correctly notes that the lengths of the month names vary. However, it doesn't provide a clear

Fun fact: it took me an additional 10 or so messages to get GPT to correctly identify the months which were wrong.

Language Models Get Things Wrong!

...so do people.

You need to know enough about the subject to be able to correct the LLM when it goofs.

Alt interpretation: if all you can do is type questions into the LLM and echo its response, you're already obsolete.

The Bigger Danger

CS324E: Elements of Graphics, Summer 2023

Hello, and welcome to Elements of Graphics. This is an accelerated class which teaches the basics of the C++ Programming language, including the basic syntax, the object system, the C++ flavor of polymorphism and inheritance (which is somewhat distinct from the systems found in other languages), the template system, and a few "modern" C++ concepts including iterators, lambdas, smart pointers, and move semantics.

Introduction

Welcome to CS105C! This is an accelerated class which teaches the basics of the C++ Programming language, including the basic syntax, the object system, the C++ flavor of

polym
othe
iterat



🏆 100 - Achievement unlocked
Get Plagiarized By an AI

**How can I tell if you
plagiarized code directly or
used the output of an LLM?**

**How can I tell that you
understand anything about
what's going on in this class?**

So What Do We Do?

You may use any LLM you would like for projects for this class, subject to the following rules:

1. You may not ask the model to solve your project in a one-shot fashion.
2. Any code that is derived from an LLM must be clearly marked within your code with comments.
3. You must attach the logs of the LLM to your assignment (your queries and its responses). This includes any learning queries you make (i.e. even if you do not intend to generate code). For models that support it, **include a link**.

"One-Shot Fashion"

My claim: the day that you no longer need to break down a problem with an LLM is the day this certificate becomes meaningless.

Need to *practice* breaking down problems and building solutions back up.



// Please write a program that has features X, Y, and Z, is nicely commented, no more than 30 lines long, solves problem Q



- // What is the general skeleton of a program that solves problem Q?*
- // How could you add feature X to this program?*
- // How could you add feature Y to this program?*

General rule: it should never look like you're trying to get the entire program in a single response.

General rule: it should never look like you're trying to get the entire program in a single response.

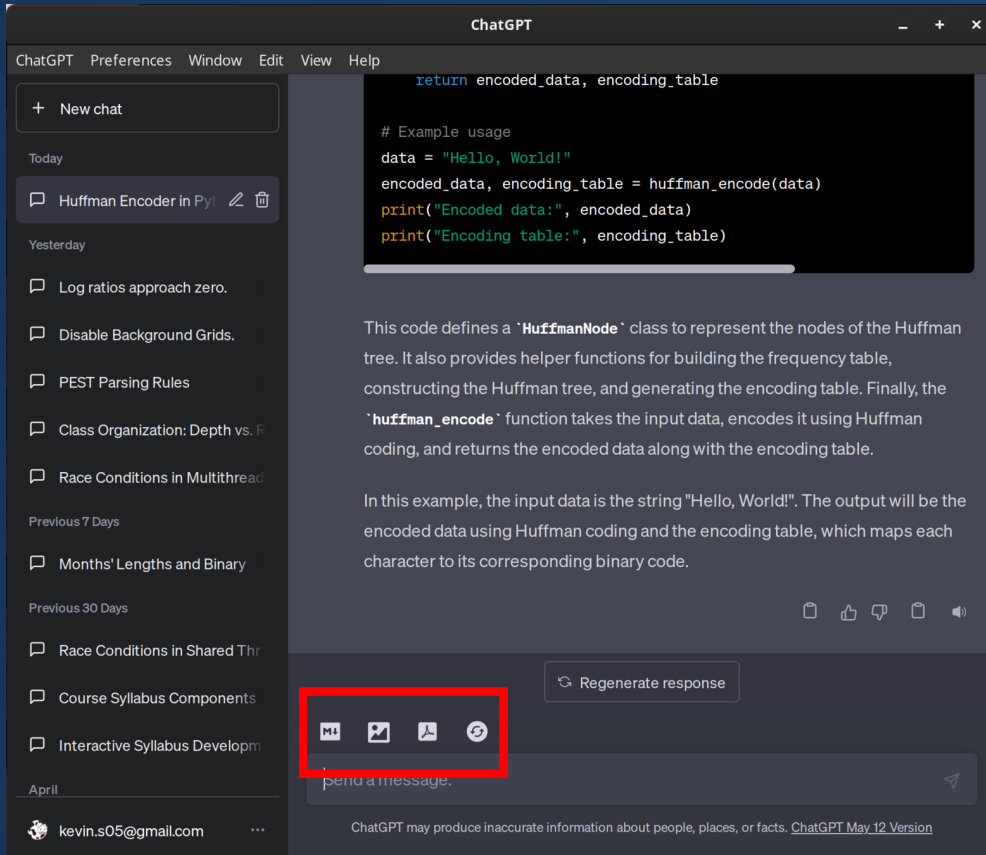
Isn't this awfully fuzzy?

Yes. This policy is experimental.

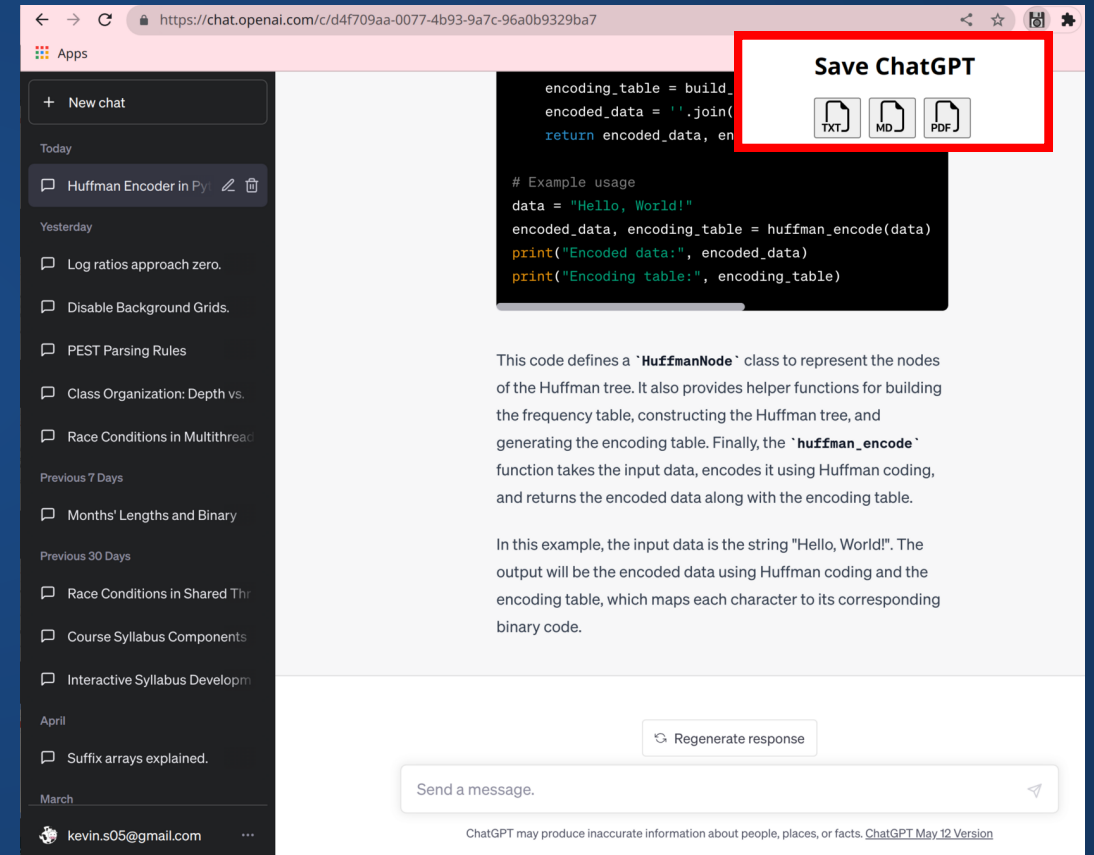
Stick to rules 2 and 3 and I will forgive you if you occasionally cross the line on rule 1.

Tools

GPT Desktop Application NoFWL



SaveGPT Browser Plugin



Please use text formats like txt/markdown when possible!

Golden Rule

If a piece of code was not written by you (or your group), tell me where it came from.

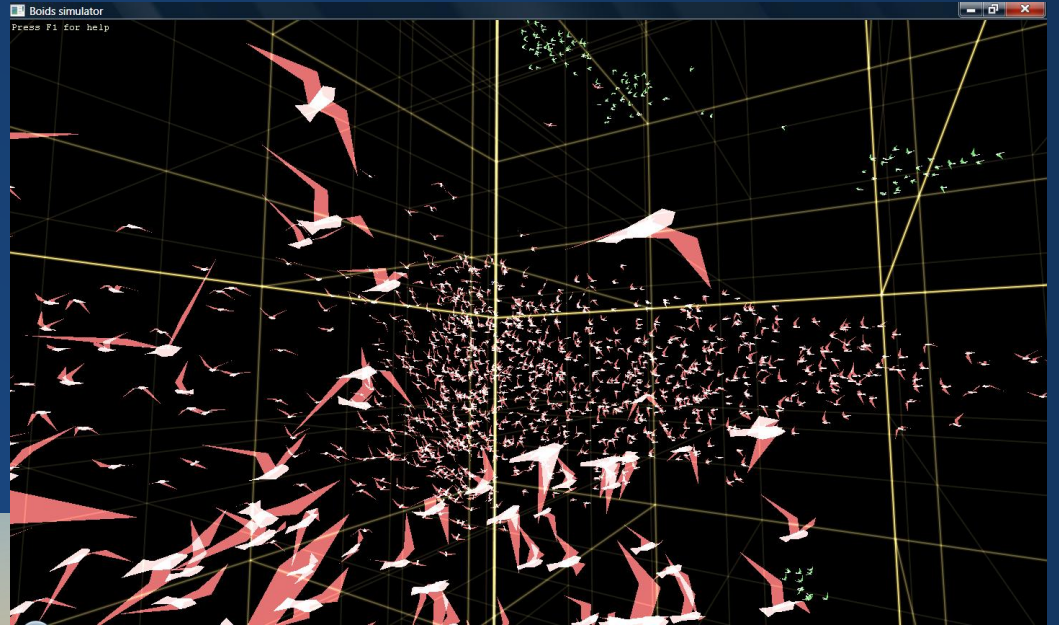
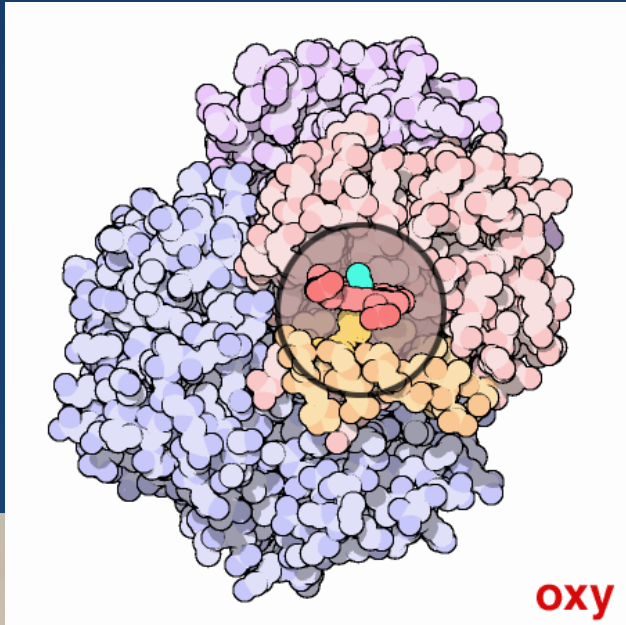
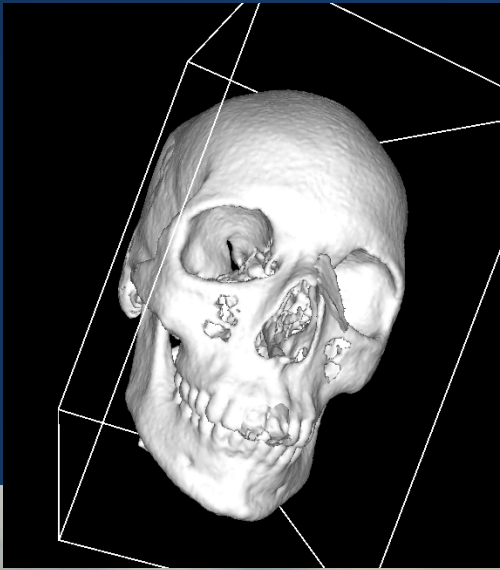
Any questions?

Additional details in syllabus

What is Graphics?

Computer Graphics creates images through computing

- Rendering
- Simulation
- Modeling
- Artist Tools
- Games
- Data Visualization



Questions

- **How can we describe visual elements to the computer?**
- **How can we control where these elements go?**
- **How can we control how these elements move?**
- **How can we do all this without making the code overwhelmingly complex?**

Tools of the Trade

Processing

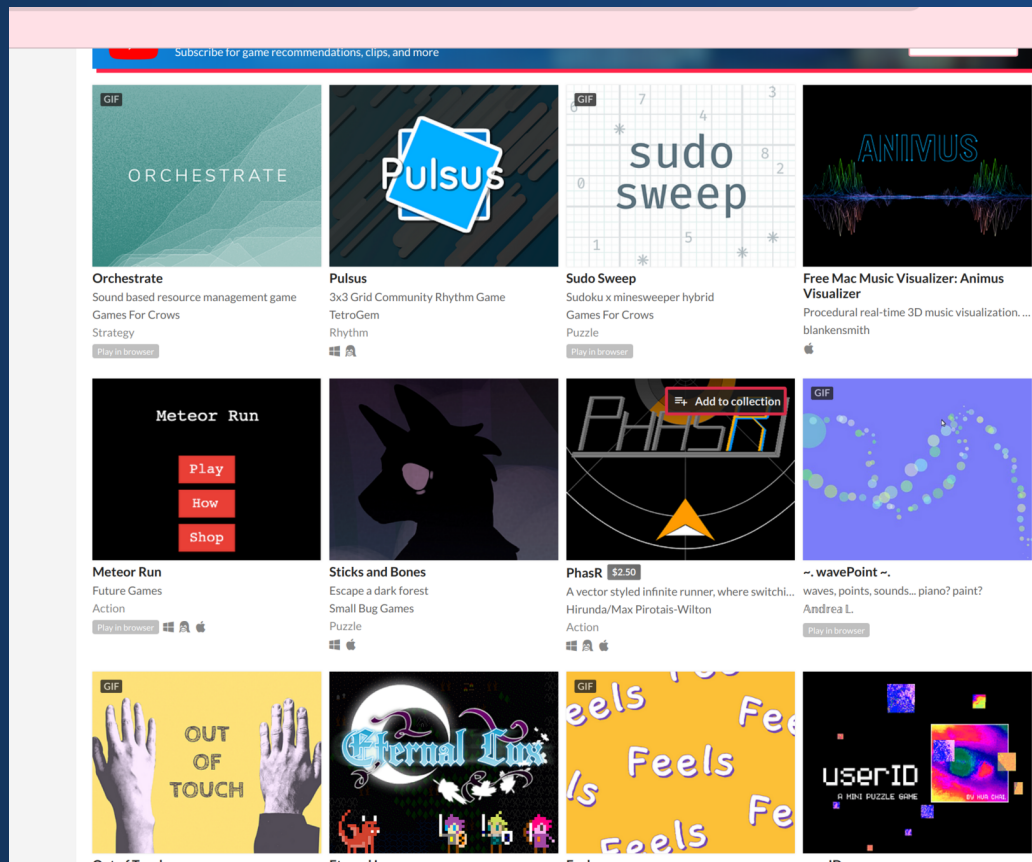


- A Java-based language for visualization
- Designed for non-programmers
- All documentation can be found at <https://processing.org/>

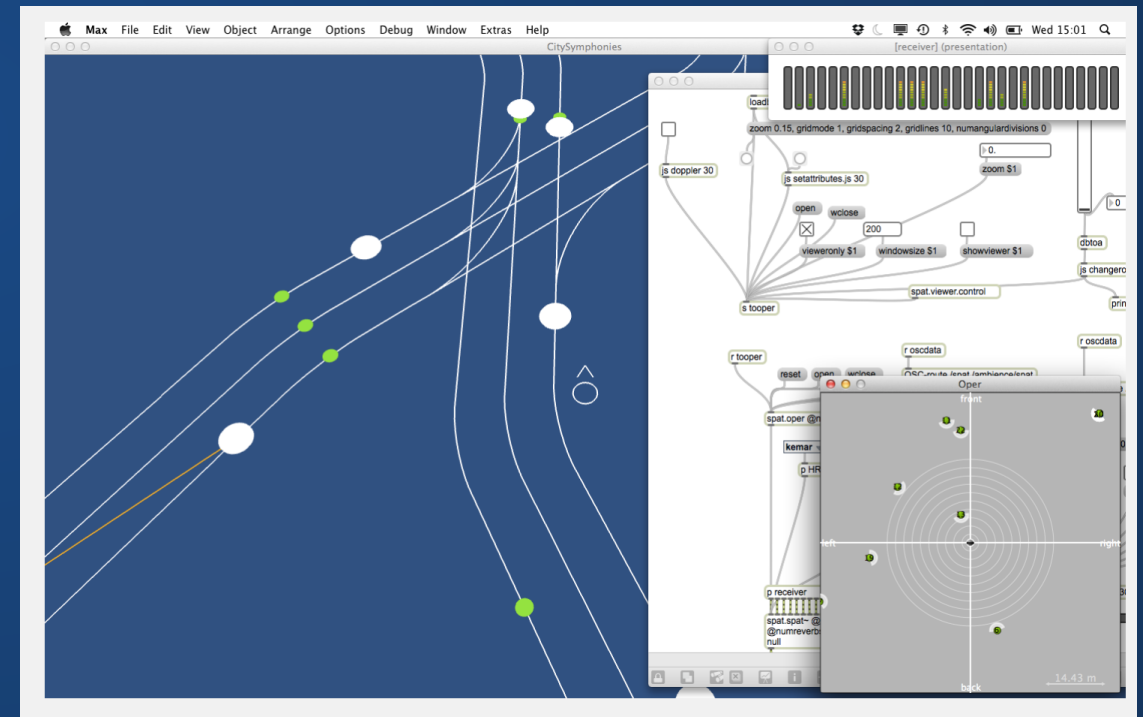
CS labs have computers which can run these. If you want an account, see the syllabus.

Well, what can we make with it?

Games (itch.io page for "Made with Processing")



A fairly impressive traffic simulation, complete with sound



Processing is designed to be easy to learn, even for someone who doesn't know how to program. You will not need to learn all of Java at once.

There are other language interfaces to Processing, such as Processing.py (Python) and p5.js (JavaScript). **Please do not use these for this class:** it's a huge pain to have assignments turned in which use the wrong language.

For basic syntax questions ("what symbols do I need to write down for a for-loop/if-statement/function"), consider either [Syntax Cheatsheets](#) or asking an LLM.

Processing docs are really good!

Hello World

Important syntactic differences from Python:

1. Curly braces instead of whitespace to mark function bodies
2. Return type in front of function name
3. Variable type in front of variable
4. Semicolons after each line

```
1 void setup(){
2     // This is a comment!
3     int win_edge = 600;
4     size(win_edge, win_edge);
5 }
6
7 void draw(){
8     ellipse(250,200,200,200);
9     rect(250, 200, 150, 100);
10 }
```


Hands-On: Welcome to Processing

1. Install Processing on your laptop
2. Create your own `void setup()` and `void draw()` functions.
Look on the Processing website to see some of the available calls
3. Write a function which doubles the number given to it, and call the function on a variable.
4. Use `println()` to print your variable to the console.
5. Visit the Processing reference through your IDE's help menu.

For this class, you should always have your code within `setup()` and `draw()` functions.

More Processing

Processing

Data Types

```
1 // var-type name = value ;
2 int x = 7;
3
4 /*
5 return-type name(arg-type arg-name*) {
6
7 }
8 */
9
10 int addTwo(int x){
11     return x + 2;
12 }
```

- boolean
- byte
- char
- int
- float
- color

Example Program

Code inside `setup()` runs once

Code inside `draw()` runs in a continuous loop

Combined, these act a little like `main()`

```
1 void setup(){
2   size(600, 600);
3 }
4
5 void draw(){
6   ellipse(250,200,200,200);
7   rect(250, 200, 150, 100);
8 }
```

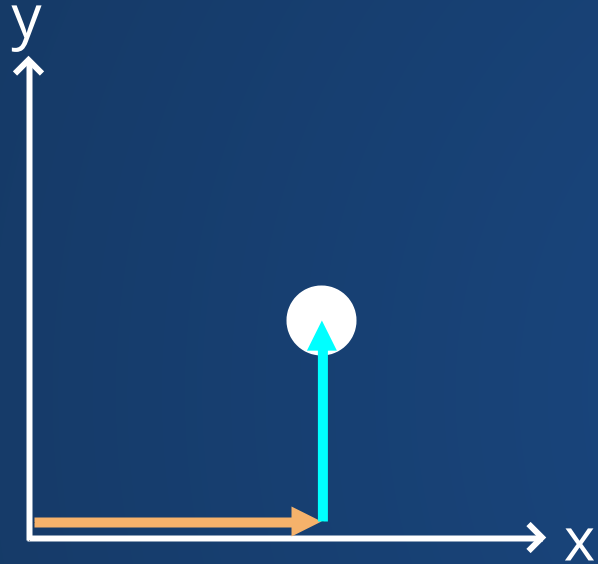
Aside: Variable Scope

- Variables declared within a block are local to that block.
- Global variables are declared outside of all blocks.
- Useful visual: curly braces prevent outward spread of variable visibility.

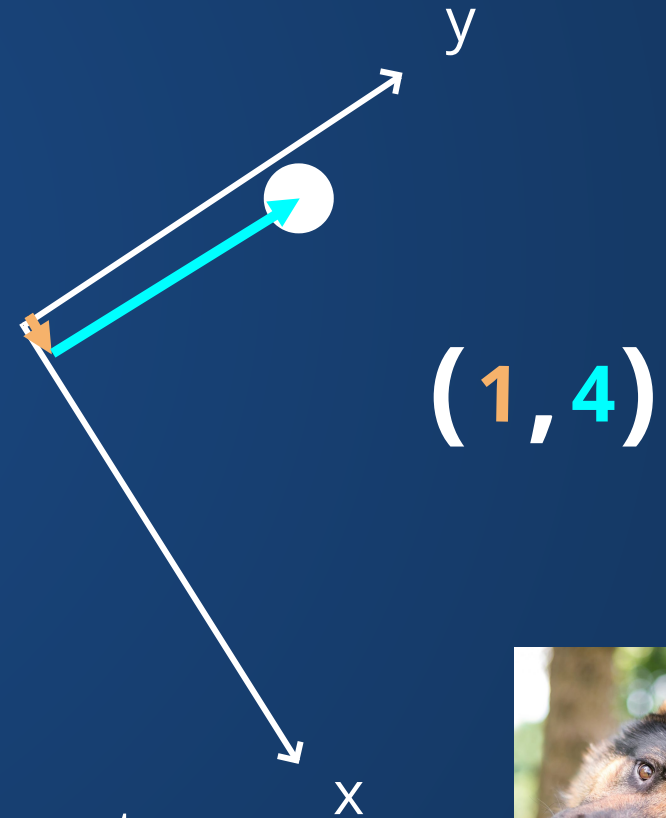
```
1 int x = 0;
2
3 void setup()
4 {
5     x = 2;
6     int y = 7;
7 }
8
9 void draw()
10 {
11     x++;    // Okay, x is visible here
12     y += 2; // Not okay, y is out of scope
13 }
```

Coordinate Systems

How do we write down location?



$(3, 2)$

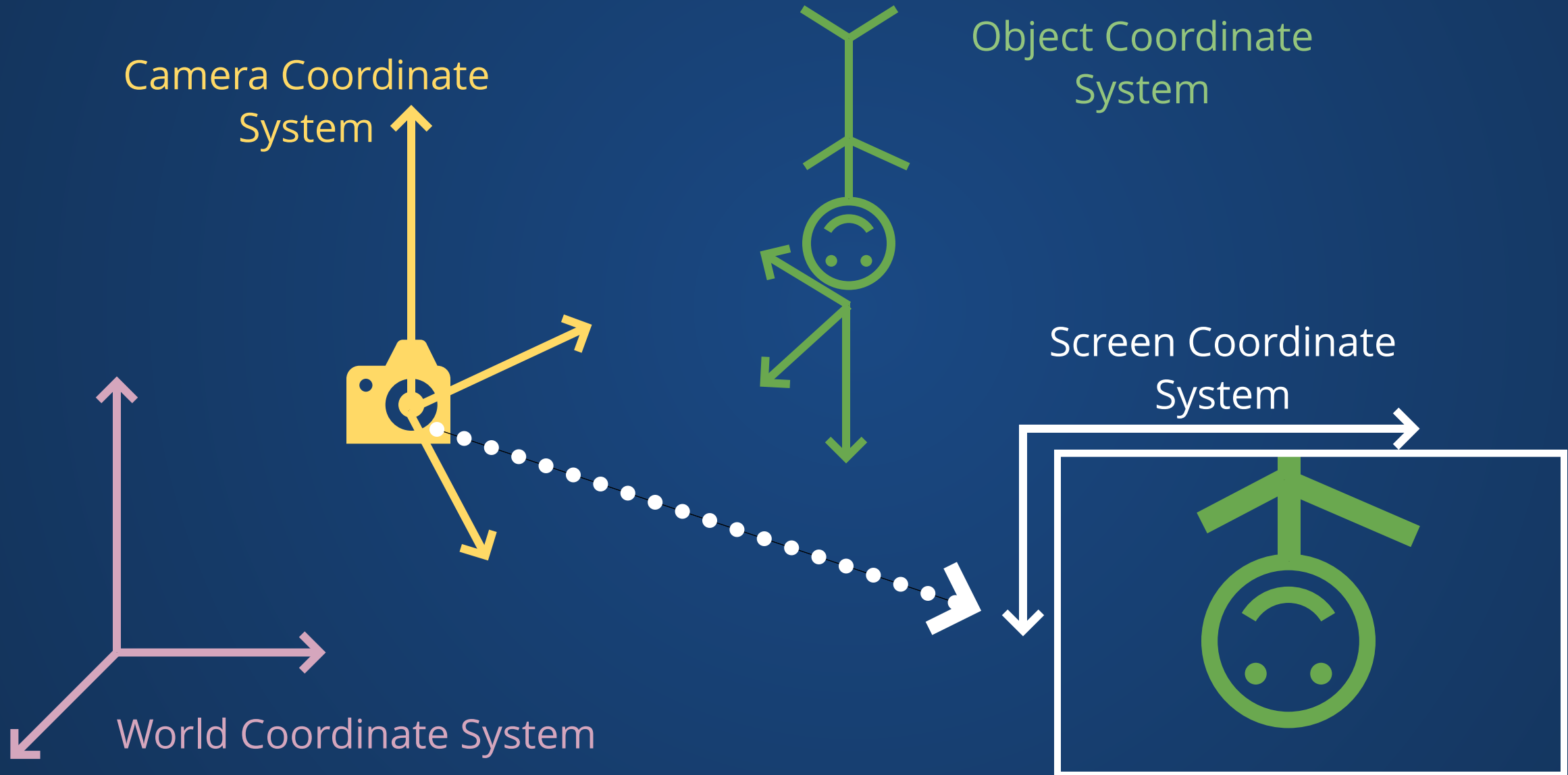


$(1, 4)$

The same point can have different coordinates in different coordinate systems!



Examples of Coordinate Systems

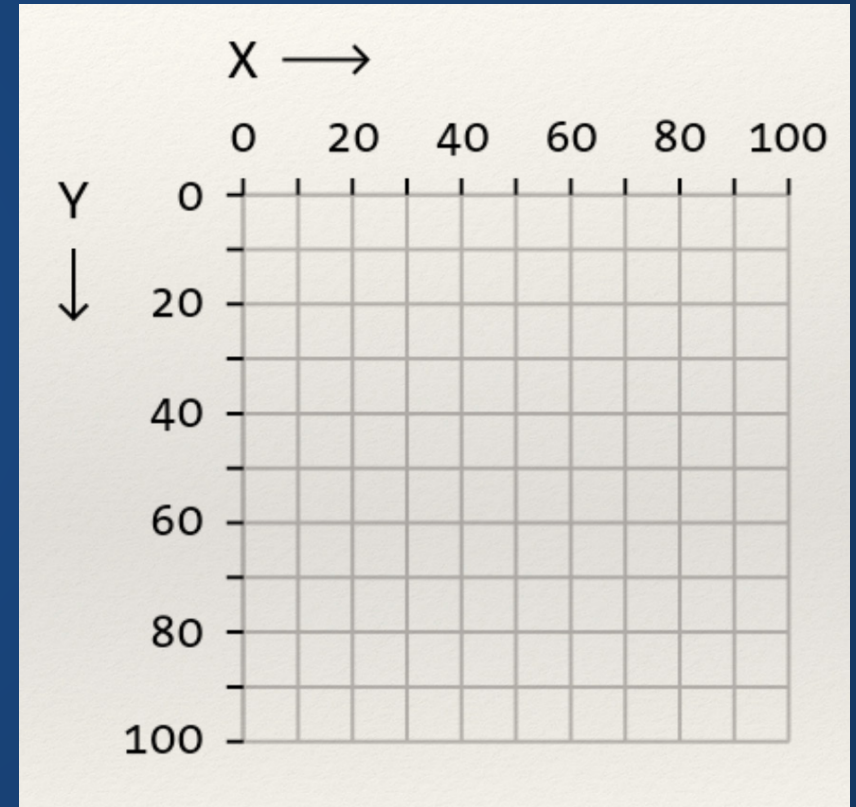


Coordinate Systems

- Coordinate systems define the "space" of the scene.
- Tell us how to interpret data about the points (e.g. "3,5") as an actual location.
- Converting between coordinate systems can require some work!
- You will sometimes hear these referred to as "spaces", e.g. "world space" or "screen space". You can think of "X space" as meaning "within the X coordinate system."

Screen Coordinate System

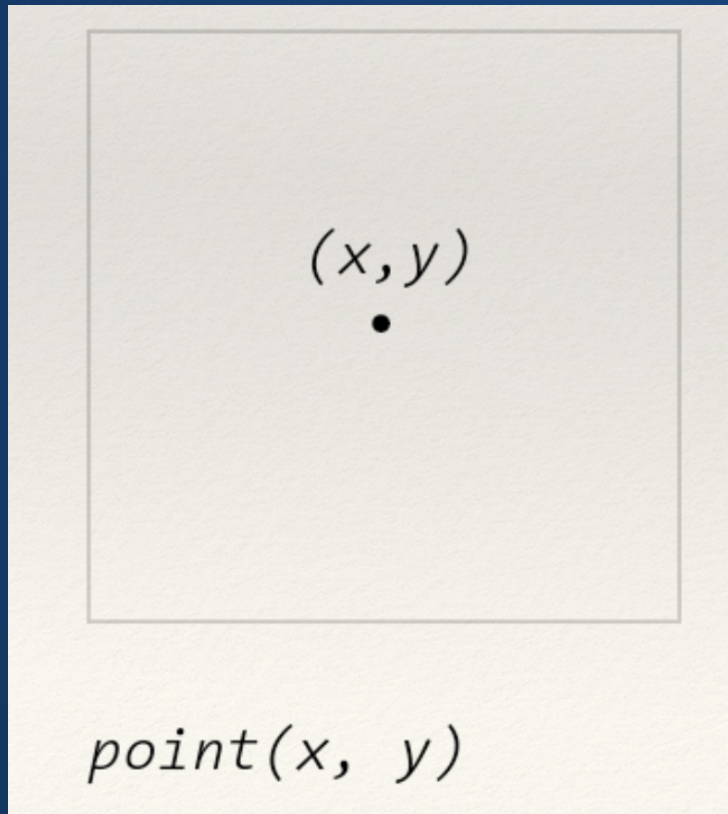
- A 2D, pixel-based coordinate system.
- Based on the size/resolution of the screen or window.
- Pixel position defined using (x,y) coordinates.



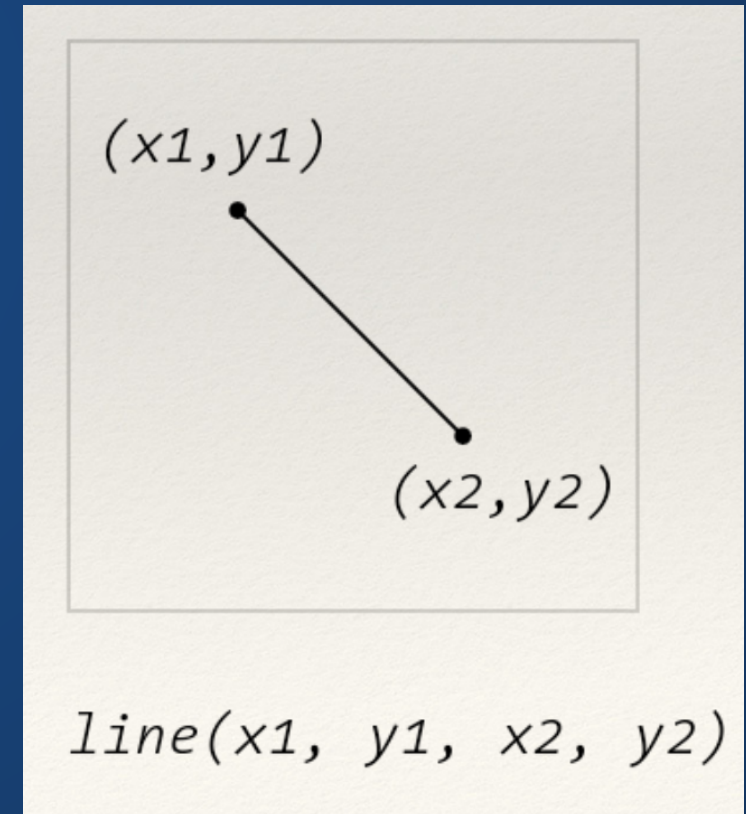
Like text!

Defining Geometry

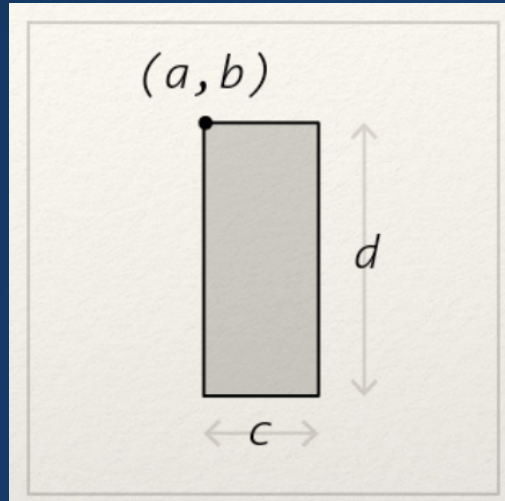
Specify a pixel within the window



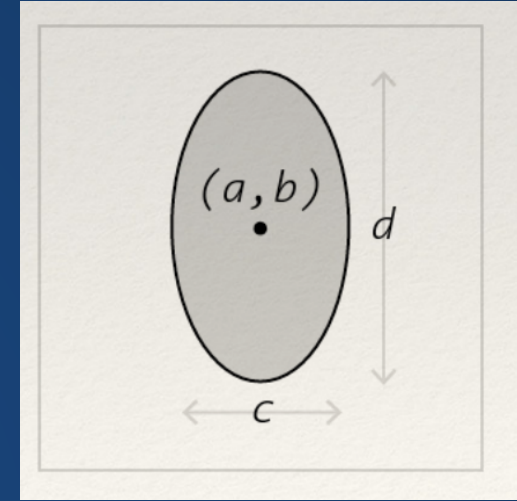
Define a line between (x_1, y_1) and (x_2, y_2)



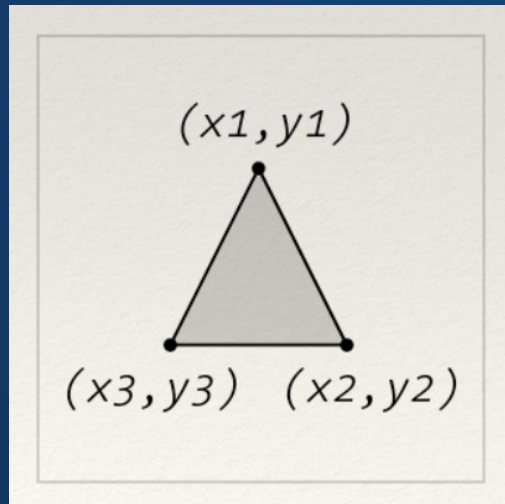
`rect(a,b,c,d)`



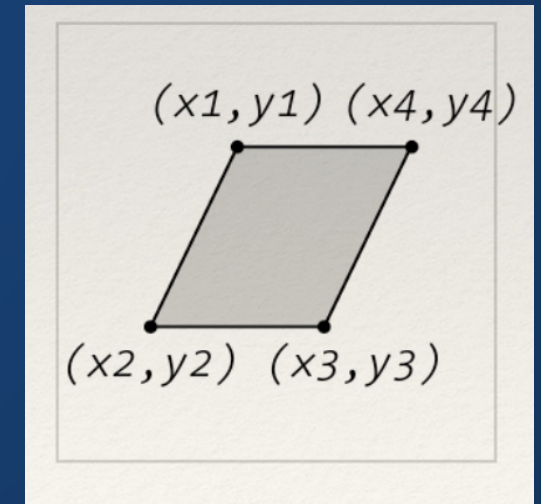
`ellipse(a,b,c,d)`



`triangle(x1,y1,x2,y2,x3,y3)`

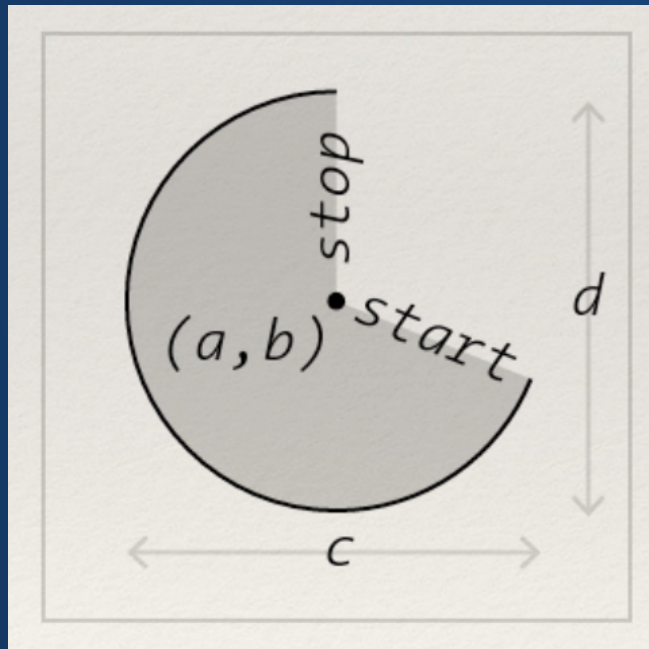


`quad(x1,y1,x2,y2,x3,y3,x4,y4)`

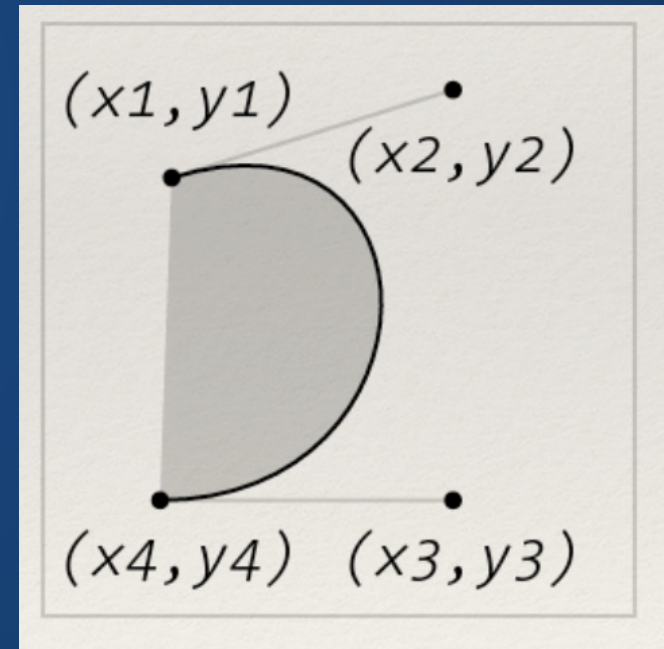


Curves

`arc(a,b,c,d,start,stop)`



`bezier(x1,y1,x2,y2,x3,y3,x4,y4)`



Hands-On: Creating Geometry

1. Create a Processing sketch
2. Use the `point`, `line`, `rect`, `ellipse`, `triangle`, and `quad` methods at least two times each
3. Create at least one shape with the `arc` method
4. Create at least one shape with the `bezier` method
5. Answer the following question in a comment: what makes `bezier` challenging to work directly with in code?

```
/* Java lets you write block  
comments like this */
```

Index Cards!

We will usually do this towards the end of class.

Write the following information:

1. Your name and EID.
2. One thing that you learned from class today. You are allowed to say "nothing" if you didn't learn anything.
3. One question you have about something covered in class today. You *may not* respond "nothing".
4. (Optional) Any other comments/questions/thoughts about today's class.