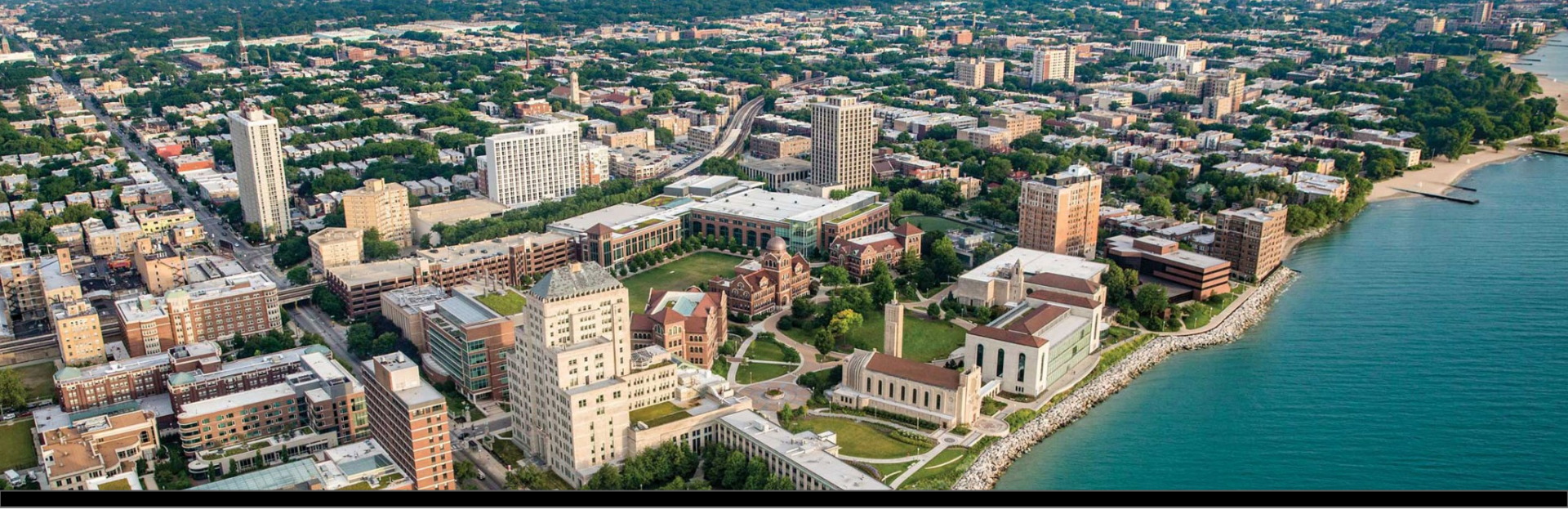

COMP 125:

Week 14, Lecture 21

Agenda

1. Attendnce
2. Review, Day 1: “Splash” Screens and Click Counter
3. Day 2: Adding Objects
4. Day 3: The Keyboard and Paddle
5. Day 4: Arrays of Bricks
6. Day 5: All Together Now

Attendance

Click Counter

The design!

- Score the number of clicks
- End the game with a double-click
- To start:
  - **Score should be zero**
  - **Have a page that gives a welcome and start instructions**
  - **Let's start the game with a click if it isn't already started**
- When player clicks
  - **If the game isn't running, start it**
  - **If the game is running, add a point to the score**
  - **Display the running score**
- When the player double-clicks
  - **End the game**
  - **Show the score**

Objects

- Add to the types of variables
  - Int, string, etc
  - Now, you can make your own!
- Class: Name of the object
- Parameters: Attributes of the class
- Methods: Things the class can do
- Example: [p5 Vector]

Syntax

```
p5.Vector([x], [y], [z])
```

Parameters

x	Number: x component of the vector.
y	Number: y component of the vector.
z	Number: z component of the vector.

Methods

toString

Returns a string representation of a vector.

Calling `toString()` is useful for printing vectors to the console while debugging.

Objects

- Why?
  - Reusability
  - Flexibility
  - Tidies the logic of code
- Object-Oriented Programming (OOP)
- Why now?
  - Up until now, “functional programming” (function-based)
  - Now, we need more flexibility, and we need parts of our code to interact
  - ---> Objects!

Making a New Class

```
class Frog {
  constructor(x, y, size) {
    // This code runs once when an instance is created.
    this.x = x;
    this.y = y;
    this.size = size;
  }

  show() {
    // This code runs once when myFrog.show() is called.
    textAlign(CENTER, CENTER);
    textSize(this.size);
    text('🐸', this.x, this.y);
  }

  hop() {
    // This code runs once when myFrog.hop() is called.
    this.x += random(-10, 10);
    this.y += random(-10, 10);
  }
}
```

Classes are **declared** using the keyword “class Name”

“Constructor” makes a new **instance** of a class

“**this**.variable” makes class variables, values associated with each instance of a class

Methods are declared as functions and interact with the variables of the instance

Making a New Class

```
class Frog {
  constructor(x, y, size) {
    // This code runs once when an instance is created.
    this.x = x;
    this.y = y;
    this.size = size;
  }

  show() {
    // This code runs once when myFrog.show() is called.
    textAlign(CENTER, CENTER);
    textSize(this.size);
    text('🐸', this.x, this.y);
  }

  hop() {
    // This code runs once when myFrog.hop() is called.
    this.x += random(-10, 10);
    this.y += random(-10, 10);
  }
}
```

```
// Declare a frog variable.
let fifi;

function setup() {
  createCanvas(100, 100);

  // Assign the frog variable a new Frog object.
  fifi = new Frog(50, 50, 20);

  describe('A frog face drawn on a blue background.');
}

function draw() {
  background('cornflowerblue');

  // Show the frog.
  fifi.show();
}
```

Classes and Brick Breaker

- What elements do we need to have be repeatable, interactable, and easy to move?
  - Bricks
  - Paddle
  - Ball
- We can start with the paddle!
- Class must:
  - Make a rectangle
  - Move it around with the left-right motion of the mouse
  - Keep it at the lower edge of the canvas
  - Bonus: make it a different color when the mouse is held