The Pedagogy of Code

By Prof. Joel Swanson

Programming is quickly becoming an essential skill of the 21st century. The traditional definition of literacy is expanding to incorporate new forms of reading and writing that include markup, scripting, and programming. Having a basic understanding of how code operates, and thereby structures the world in which we live, is a key differentiator in the contemporary workforce. Within higher education, the pedagogy of programming needs to expand beyond computer science, and with it, we need to critically evaluate our methods of instruction and assessment to meet the learning styles and diverse backgrounds of all students. For these reasons, I have chosen to develop this teaching portfolio around a course that I teach frequently, ATLS3000, *Code*.

Code teaches students the fundamentals of programming within a creative studio environment. This means that I teach the fundamental concepts of programming, and students use those concepts towards the production of creative and speculative works. This stands in contrast to the traditional models of teaching programming, which are often more rote and prescriptive. Students within this class are typically Juniors and most are Technology, Arts, & Media (TAM) minor or certificate students from a diverse set of majors with highly variable experiences with programming. Eventually, we would like this course to be accepted as equivalent to Computer Science and Information Science's introductory programming course. This course is required for students who haven't taken an introductory programming course in another department, and it is a pre-requisite for many of the upper division electives within the TAM Program's curriculum.

Overview

This portfolio is divided into four sections. **Background** provides an overview of the course and how it has evolved since I began teaching it several years ago. It also contains a general description of the course as well as overall course goals, course objectives and teaching questions. The **Implementation** section details the specifics of the course, as well as my rationale for the course structure and content. The **Student Work** section contains example projects and evaluation metrics from the course. And lastly, I have written my **Reflections** on the latest offering of this course, Spring 2017, and what I plan to change as I will be teaching it again in a different format in Fall of 2017.

I teach this course regularly as part of my teaching duties. This course has proven difficult for many students who struggle with the technical aspects of coding. I relate with this perspective because programming was difficult for me to learn. However, in retrospect, I don't believe my struggle was because programming is inherently difficult; the challenge of programming has more to do with how it is taught. I take it upon myself to make programming approachable, relevant, and dare I say fun for students who may not view themselves as technically inclined.

I would like to capture the experience of those students who may come into this course feeling apprehensive about programming, yet manage to do well in the course. I feel the message behind this experience is important and compelling for computer science education, and many of its attendant biases and stereotypes.

I want to figure out what works and what doesn't work within my teaching methods. I have had a great deal of success with teaching programming to a wide range of students, but I would like to be better able to articulate which practices and approaches are the most successful. It is my hope that this portfolio could be useful to others teaching introductory programming courses and simultaneously reflects my commitment to teaching.

Background

Catalog Description

ATLS 3000, *Code*
(This course) explores computation as a powerful tool for creative design and expression in a project-based studio environment. Students learn the fundamentals of creative coding, computational thinking, and object-oriented programming. Hands-on topics include generative art and design, interactivity, animation, and visualization.

Course Goals

Stated Course Objectives:

- That you learn the fundamentals of programming, including:
    - how to use basic programming structures like loops, conditional logic, procedures, variables, and event-handlers to design programs
    - how to work with data structures, including objects and arrays, to store and process data
    - how to locate and correct errors in programs
- That you (students) understand and appreciate what it means to "think computationally" and become familiar with how computing can help you solve real world problems
- That you are able to think creatively and conceptually about your work, in whatever discipline or field that may be
- That you become familiar with and appreciate the utility of programming within the creative disciplines
- That you are comfortable using programming as a tool for design and creative expression
- That you produce some amazing creative coding projects that you will be proud to include in your portfolio

My Unstated Goals for my students (not on the syllabus)

- That my students understand the relevance and utility of programming and computational thinking
- That my students gain the ability to continue to learn more about programming in other classes or on their own, and that they gain the confidence that they can learn new programming languages
- That they view coding as a useful tool in their creative skillset
- That programming is demystified, especially for students who may be apprehensive. Make it not scary; dare I say fun!

Historical Background

This course is based on its pre-cursor, *Digital Media 2*, which was designed as a continuation of *Digital Media 1*. *Digital Media 1* was an introduction to basic web design, digital imaging, digital video, and digital animation. *Digital Media 2* focused on interactive scripting and was taught using several different technologies, including JavaScript, Actionscript, PhP, MySQL, and JQuery. While we appreciated keeping the course "technologically agnostic," we found that the student experience varied dramatically based upon the chosen technology. This practice was problematic as students continued in the program because of their varying levels of experience. When we transitioned this course into "*Code*" in 2015 we chose to standardize the course and the technologies taught to ensure a more consistent student experience.

For the past several semesters, I have been teaching the course with p5.js [link to p5.js] a JavaScript library designed to emulate the simplicity and functionality of Processing, the popular Java library. The students appreciate the simplicity and web-based nature of the library, and as an instructor, I appreciate the ability to teach more advanced concepts, such as data structures and object-oriented programming, in a relatively short amount of time. Doing this with other technologies such as JavaScript has proven prohibitively challenging in the past. While there is no perfect technology for teaching programming, I have found p5.js highly effective for my students. Using a web based language also reinforces the technical content taught in the pre-requisite to this course, ATLS 2200, *Web*.

Teaching Questions

- This course uniquely mixes creative projects with technical subject matter. How do I best evaluate the students in these two drastically different areas?
- Programming as a technical skill can be difficult to evaluate since so much of it is learned by looking at code samples. How do I effectively measure a student's technical ability, when I need to simultaneously encourage--and at certain points discourage--using found code samples?
- With the "flipped" nature of the course, how do I ensure that the students are watching, and understanding, the material assigned outside of class?

Implementation

Flipped Classroom

After teaching this course several times, I realized that students struggle the most when they try to implement technical topics from the lectures into their labs and projects. (See schedule at end of this document.) For this reason, I decided to "flip" the classroom, and provide the lecture content outside of the normally scheduled class period. This choice was facilitated by finding a series of videos on YouTube by Daniel Shiffman (NYU). Shiffman has taken his lectures on p5.js and broken them into short, digestable videos that I assign before each class period. This "viewing" activity replaces the traditional "reading" assignments. Before each class, students watch a series of videos, which I try to keep under forty-five minutes per class session, which seems to be the maximum attention span for my students. I then begin the following class by answering any questions that student have from the videos, and showing a few quick examples of projects to reinforce and build upon the technical concepts introduced in the videos. I also provide a list of topics that students should understand after watching the videos to encourage an active engagement with the video content. See sample watch list at the end of this document.

Frontloading Technical Content

In a significant change from previous semesters, for Spring 2017 I "front-loaded" the majority of the technical instruction into the first half of the semester. The result is that by midterm, we have covered 90% of the technical material. This allows us to focus on the design and development of the two major course projects for the remainder of the semester.

Changing the overall pace and tone of the course at the midterm also seems to be welcomed by the students. What was learned in the beginning of the semester can then be applied and built upon during the second half of the semester. This also allows time to address, and readdress, technical topics that prove problematic.

Programming Challenges

For the first half of the semester, I assign a daily programming challenge which requires the student to integrate the recently introduced technical topics within a small creative coding project. For example, if we are learning about nested "for" loops I will have the students create a seamless repeat pattern using loops. These challenges are designed to build upon the technical topics introduced in the videos, and provide the students a way to work through any technical issues within a lab/studio environment. This also introduces the utility of programming as a creative tool.

Pair Programming

Several of the programming challenges are completed in student pairs. Pair programming is a practice that has proven to help students new to programming:

> Pair programming consists of two programmers sharing a single workstation. The programmer at the keyboard is usually called the "driver", the other, also actively involved in the programming task but focusing more on overall direction is the "navigator"; it is expected that the programmers swap roles every few minutes or so.

I have observed that pair programming is effective for beginner programmers as learning the syntax of programming becomes less frustrating with two people working on the code. For example, missing a semi-colon, which can prevent an entire program from running, is easier to spot when there are two students working together. Students also get a chance to work with a range of students with differing technical abilities. For some projects, they will take on more of a teaching role, while in others they will be more of a learner.

The programming challenges are evaluated rather simplistically. Each challenge is worth two points, and I also offer extra credit if students go beyond the requirements to make the solution more technically complex or creatively interesting. I then show exemplary solutions in the following classes. As I choose solutions to show, I am careful to show a broad range of student work that blends technical, conceptual, and creative proficiency. I have found this to be a useful way to reinforce that technical proficiency is only one aspect of this course, which is important since most students tend to value technical proficiency above all else.

Technical Benchmark

In courses that blend technical and creative content, it can be difficult to ensure that students exit the course with a mastery of the technical material. To ensure that students are technically proficient, I designed a technical benchmark exam that students take at the midterm. This exam is a series of short answer questions and a practicum where they have to recreate a provided image using code. This exam isn't designed to be difficult, but simply to ensure that students know the fundamentals necessary for continued success in the Technology, Arts & Media program's curriculum.

I allow students to retake the Technical Benchmark if they don't pass. This reinforces my goal as an instructor to help them learn how to be proficient programmers. As an instructor, I'm not out to critically evaluate their technical ability, but simply to ensure that they have mastered the required course content. Each time a student retakes the Technical Benchmark, I change each question to ensure students aren't memorizing the solutions. A student must pass (75% correct) the benchmark to pass the course, but once passed, it doesn't affect the student's final grade. This again communicates that the benchmark is about a necessary level of technical proficiency, and not about a critical evaluation of technical adeptness (which may lead to certain negative stereotypes around programming and technical subject matter).

Find Technical Benchmark examples at the end of this document.

Code Journals

I want to know how my students are feeling about programming throughout the semester. Learning how to program can be a very frustrating endeavor, and I need some mechanism to allow students to express and communicate their progress in the course. To that end, I assign weekly Code Journal Entries, which are questions asking for qualitative reactions to their experiences in the course. This allows me to reach out to students who are struggling, respond to specific technical matters that seem to be problematic, and adjust the general pace of the class as needed. Students often complain that these journals feel like busy work, but as a pedagogical tool I have found them to be invaluable.

At the start of the semester I make each a student a Google Doc, which is shared between me and the student. Each week they respond to the Journal Entry by 5 p.m. on Sunday, which allows me and my lab tutor the time to read through the journals before class on Monday.

In terms of evaluation, the journals are worth 5% of the student's overall grade. Students receive a grade at the midterm and again at the end of the course. Checking the journals at midterm ensures that students are completing the journals weekly, and not waiting until the end of the semester to complete the entries retroactively.

Projects

My goals for the major projects are two-fold:

1. show that the student understands and can integrate the technical content into a creative project
2. show that the student can approach a project from a creative and unique perspective.

The project prompts are constantly evolving as I continue to teach the course, but the current projects are described below:

Project 1: Game Remix

Measuring "creativity" is a difficult if not impossible venture, especially if the project is open ended. Curiously, I have found the television cooking show "Chopped" to offer a unique solution. For each cooking challenge, the contestants must integrate a series of "mystery ingredients" which are presented to them immediately before the allotted cooking time begins. A series of celebrity chefs then evaluate the resulting dishes based on presentation, taste, and how creatively the cook integrated the mystery ingredient. Cooks can add any other ingredients they want, but the mystery ingredients must be significantly present in the solution.

In term of measuring creativity and technical skill, requiring all the cooks to work with the same mystery ingredients makes it easier to critically evaluate the level of creativity that went into the solution. For the first project, I adapted this concept into what I call a "Game Remix." Students are required to design a basic video game (think simple 2D arcade style game) that utilizes certain “ingredients” or media assets that I provide. For example, I give them a list of images, sound files, and other requirements that they must integrate into their solution. The specific project prompt is:

> Use p5.js and the required media assets outlined below to create a video game remix.
>
> - Your game must include:
>
> - A famous painting of your choosing
> - One or more sounds from this folder
> - One or more characters from the images in this folder
>
> - You can edit, rework, and remix the required assets, but they must be present in some way, and you will be evaluated on how well you integrate the assets together (think “Chopped”).
> - The goal of this project is to challenge you creatively, conceptually, and technically!
> - The project will be turned in as an online game.

I also have students develop a Project Plan, which is a one-page description of their project and goals for the assignment. This helps me to guide the student in case their project plan is overly ambitious, or too simplistic technically. The Project Plan is worth one point out of the total fifteen points for the project.

This semester I allowed the students to integrate the comments from the project critiques to improve their project and overall grade. A concern with this practice is that students might not work as hard on the project initially, since they know they can improve the project after the critique. But allowing students to do revisions is consistent with my goal to have them produce the best possible work.

Project 2: Data Visualization

Using programming to input, filter, and visualize large data sets is one of the most effective ways to show students the power of computing and computational thinking. There is something almost magical about writing a program that inputs a massive Excel file and visualizes it in a meaningful and compelling way. Data visualization teaches students about data formats, data structures, and key visualization methods. For these reasons, I frequently include a data visualization assignment as the final project.

The specific project prompt:
Using p5.js, create a data visualization that explores how the design of data can affect its meaning.
Objectives:

- Use data visualization to demonstrate a non-obvious insight gleaned from the data, or to make a particular point
- Use p5.js to visualize large sets of data
- Explore how design can be used as a persuasive tool
- Learn about the various ways data is stored and accessed (e.g. CSV, JSON, XML, API’s)

- Become familiar with the various approaches in information design and visualization

The challenge with the project is two-fold: finding the data to work with, and visualizing the found data. Often times finding robust, clean data is more frustrating than the actual visualization. While I have considered standardizing the data sets to prevent this frustration (i.e. giving all the students the same data set to visualize) I find that struggling to find the data is an important learning experience.

Course Materials

I do my best to keep ancillary course costs at a minimum, so while not required, I recommend Lauren McCarthy's *Getting Started with P5.js: Making Interactive Graphics in JavaScript and Processing*. For difficult concepts, an explanation and examples from a different person or perspective can be valuable. I will also make copies of certain chapters for concepts that in the past have proven difficult for students.

An ancillary goal of the course is to show students how relevant programming can be in a diverse range of creative fields. Towards that goal, I do weekly "Show and Tells" where I show exemplars who use programming in creative ways in various disciplines, practices, and careers. This list of people I show is constantly growing, and changing, as I do my best to keep this list up to date and relevant to my students' interests.

This course requires that I use a learning management software other than Desire 2 Learn (D2L), which is the current software provided by the University. D2L has a built-in security measure which prevents code from being turned in through the drop box mechanism to prevent the uploading of computer viruses. I inquired with D2L tech support, and they did not have an integrated solution, so I am currently using a third-party solution, Chalkup.co. While not as robust as D2L, Chalkup provides everything I need for the course, including a calendar, assignment upload, online grades, and discussion forums. The other benefit is that Chalkup is integrated with G-mail (Google Mail), so students can use their G-mail enabled University of Colorado e-mail accounts, meaning they don't need to create an additional account, which is something I try to avoid.

Lab Tutor/Assistant

As is common across campus, this last semester I was asked to increase my enrollment size from eighteen students to thirty+ students. With the increased number of students, I wanted to ensure that I had the time and resources necessary to keep a high level of contact with each student. Using a Lab Tutor / Assistant was crucial in successfully enlarging the class. The tutor's duties included:

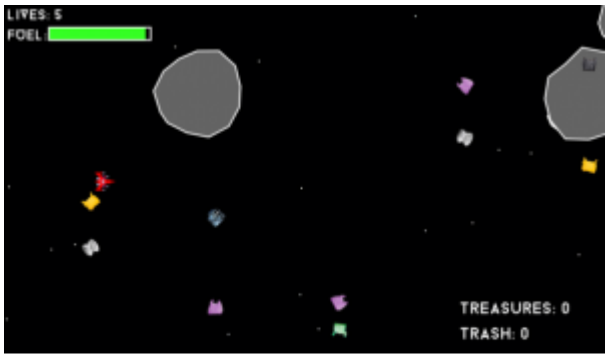
- Read and check-in the weekly Code Journal entries and alert me to any students who were feeling overwhelmed—or conversely were not feeling challenged within the course. The Lab Tutor also let me know what topics seemed to be causing problems for the students, and what topics the students were curious to investigate further.

- Be in class for the Programming Challenges to help students. Having two people available to help students proved to be necessary for the larger class size.
- Check the weekly Programming Challenges for completeness.

This past semester, McKenzie Weller was my Lab Tutor/Assistant and she was ideal. I will be working with her again in the fall in the same capacity.

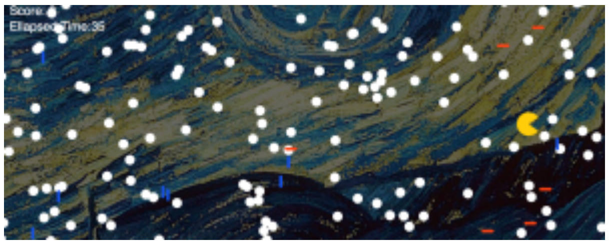
Student Work

Game Remix



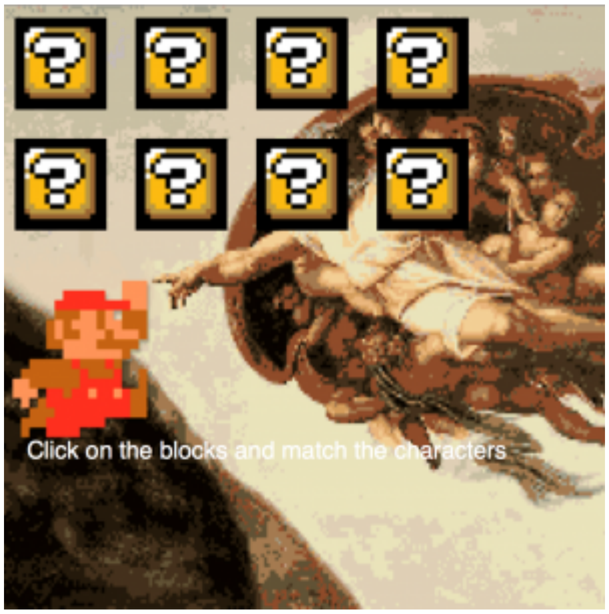
Sample Project 1 for Game Remix (links to game)

This project was impressive technically, aesthetically, and conceptually. All the required assets were integrated to creative a cohesive narrative within the game, and the overall design and development reflect a high level of technical ability.



Sample Project 2 for Game Remix (links to game)

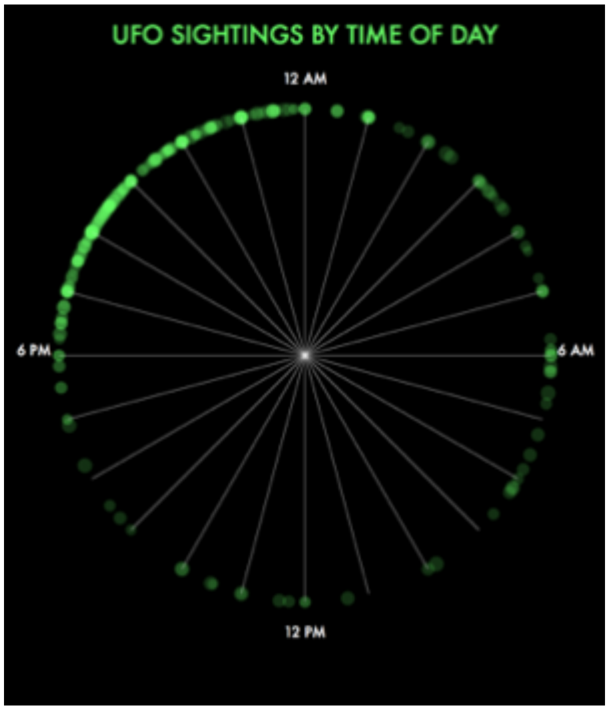
The student who created this project initially wanted to create a game similar to the traditional "Pac-Man" arcade game. However, the student soon realized that despite its perceived simplicity (when compared to contemporary video games) programming all the interactions involved in the original "Pac Man" is no small task. The student was able to successfully scale the project technically, and yet still produce an enjoyable game experience. The ability to scale a project is something I emphasize in the course, and I was pleased with how the student approached and successfully completed this project.



Sample Project 3 for Game Remix (links to game)

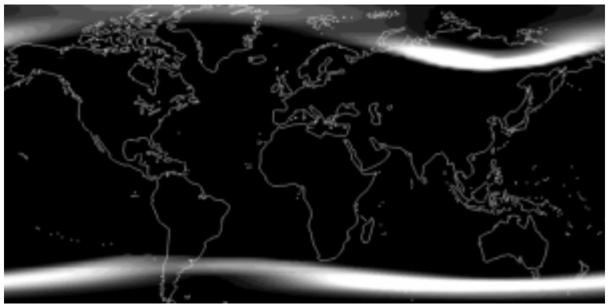
This project was creative, technically sophisticated, and humorous. Card Matching games require a high level of technical sophistication. Additionally the student was also able to create a playful and engaging narrative by integrating the Super Mario Brother sprites with Michelangelo's *Creation of Man*.

Data Visualization



Sample Project 1 for Data Visualization (links to visualization)

This project was successful due to the unique data set and corresponding visualization. The student found an interesting data set, UFO sightings, and thought it would be interesting to pull out the time stamp from the dataset. The method of visualization uses the 24-hour clock to clearly visualize the data. The overall aesthetic resonates with a paranormal aesthetic, without being campy or distracting.



Sample Project 2 for Data Visualization (links to visualization)

While visually simple, this project takes a dataset of the occurrences of the Aurora Borealis, then maps that data onto a map of the world. The data set is simply a list of integers, which meant the student had to devise a method for converting those numbers into a visual pattern. I was particularly proud of this student who had made significant progress throughout the course. This student didn't have significant previous programming experience, but was extremely dedicated and ended up doing very well in the course.

Shooting Stars Over The Years
September 1994

Sample Project 3 for Data Visualization (links to visualization)

For this project, the student found a historical data set that contained the latitudinal and longitudinal coordinates for shooting stars. The student was able to successfully work with an oddly formatted data set to create a visually minimal yet technically sophisticated project.

Reflections

Technical Benchmarks

In Spring 2017, I had more students struggle to pass the Technical Benchmark than in the past. Roughly a third of the students passed the benchmark on the first attempt. Admittedly, the benchmark was more robust, and therefore more difficult, but I still feel the benchmark was appropriately rigorous and I don't intend to significantly change the structure or content. By the end of the semester, I developed three different versions of the benchmark to allow all the students the opportunity to take and retake sufficiently different examinations. Seeing what the students struggle with was extremely useful as it will allow me to spend more time with those subjects in the future. Specifically, students seemed to struggle with the technical topics of *Objects* and *Constructors*, two key features of object-orientated programming which are crucial to this class.

Projects

Game Remix
In general I was pleased with the quality of work that the students turned in for this project, however some students simply modified existing code samples (that either I provided in class or they found independently online). I don't mind when students use existing code—looking at code is a great way to learn—however, I wish they would have challenged themselves more technically to build these examples into more robust solutions. I was also expecting more originality with the given media assets. From speaking with students, many felt so overwhelmed

by technical issues, they weren't able to think through the creative aspects such as aesthetics, narrative, and game play. I am rethinking ways to address these issues moving forward.

Data Visualization
In an informal poll, most students preferred the Data Visualization project to the Game Remix. Perhaps this was due to the aforementioned "magic" of data visualization, but in many ways this project is also more straightforward technically, as successful solutions can avoid many of the trickier technical topics such as objects and interactions of objects. Many students had to change their plans mid-project, as they couldn't find appropriate datasets, or the data sets that they found weren't accessible or formatted properly, but even in that frustration they learned the important lesson that we often need to adapt to the realities of deadlines and available resources.

Code Journals

I will be the first to admit that the Code Journals become tedious for the students to write, as well as for me or the class assistants to read, but I still find the journals to be an indispensable way for me to gain a sense of how the students are doing in the course. Especially as the enrollment of this class continues to grow, I need some mechanism to keep track of the students, how they are doing, and when then are struggling. I also find that the journals encourage students to be reflective about the process of learning programming. In the future, I may decrease the number of journal entries, but I will continue to use them in this course.

Growing the course for Fall 2017

For Fall 2017, I was asked to turn this course into a large/lecture format course. The lecture enrollment will be up to 60 students, and each lab (3 total) will have 20 students. The labs will be run by tutors/lab assistants. This new format comes with new challenges, in particular how to run effective critiques, and ensure that students receive meaningful feedback on their work. My current plan is to turn the two projects into one major project (most likely a data visualization project) but make it more of an iterative experience with specific benchmarks and due dates throughout the second half of the semester. I will then sit in on all the project critiques, which will take place during the labs, to ensure that all students receive detailed feedback. I will also need to develop a series of supplemental lectures since I will have all the students in a large lecture hall once a week. Although frustrating to retool a format that now feels comfortable, I look forward to the new challenges of teaching this course to a larger number of students.

ATLS3000, CODE

Mondays and Wednesdays
11:00 a.m.–12:50 p.m.
ATLS1B31

Instructor: Joel Swanson Joel.Swanson@Colorado.edu
Office Hours: Mondays and Wednesdays 1–2 p.m. ATLSL101 (or by appointment)

Learning Assistant: Kenzie Weller: Mckenzie.Weller@Colorado.EDU
Help Hours: Tuesdays 2–4, ATLS231 (TAM Suite)

COURSE DESCRIPTION
(This course) explores computation as a powerful tool for creative design and expression in a project-based studio environment. Students learn the fundamentals of creative coding, computational thinking, and object-oriented programming. Hands-on topics include generative art and design, interactivity, animation, and visualization.

PRE-REQUISITES
Officially, there are no pre-requisites for this course. However it is strongly recommended that you have taken ATLS2100, *Image* and ATLS2300 *Web*. All of your labs and projects will be "turned-in" as websites, so you need to be comfortable posting content to your personal website for this class.

A basic understanding of algebra, geometry and trigonometry is required for this course. (e.g. Pythagorean Theorem, SoCahToa, Radians, Unit Circles, etc.). Here is a refresher if you are feeling rusty:
https://www.youtube.com/watch?v=X5uFqpypDy4

COURSE OBJECTIVES
- That you understand and appreciate what it means to "think computationally"
- That you become familiar with programming as a creative tool
- That you understand the fundamental structures and syntax of programming in p5.js / JavaScript
  - How to use basic programming structures like loops, conditional logic, variables, and event-handlers
  - How to work with data structures, including objects and arrays, to process data
  - How to locate and correct errors in programs
- That you will produce some amazing creative coding projects that you will be proud to include in your portfolio!

WHAT WE WILL DO
This class is a mix of technical instruction (both inside and outside the classroom), readings, viewings, lectures, workdays, and critiques. The course is structured around two major projects, however, in order to make the best projects possible, we have a lot of technical content to cover first. If you look at the schedule, you will notice that the bulk of the technical instruction is front-loaded into the first half of the semester. The latter half of the semester will be devoted to working on projects and critiques.

SHOW & TELL
Throughout the semester I will introduce you to the work of several designers, artists, and creative coders who use programming in a wide range of disciplines, practices, and careers. The goal of this is to show you how useful (some would say necessary) programming is as a creative skill in today's professional landscape.

CREATIVE CODING PRESENTATION
Later in the semester each of you will be doing a research presentation on a creative programmer of your choosing. More details on this project will be given as we approach the second half of the semester.

REQUIRED TEXTS
None, but I will provide various readings throughout the semester as PDFS. I encourage you to print them out and keep them in a folder or binder as a way to organize the readings from the class. Viewings and readings are to be completed BEFORE the day that they are listed on the schedule.

If you want a book as a supplement, I highly recommend this one: Getting Started with p5.js, Lauren McCarthy

COURSE WEBSITE
For this class, we will be using https://www.chalkup.co. I will send you access codes after the first class period. On this site you will find the course syllabus, schedule, assignments, discussion forum, resources, and grades. This is also where you will turn in your assignments.

PAIR/GROUP PROGRAMMING
For the first half of the semester, we will be learning how to program. On a typical day, you will come to class having watched a series of videos that I previously assigned. I will also provide a list of concepts you should watch for in the videos. You will then come to class, and in randomly assigned pairs or groups, collaboratively work on a "Programming Challenge" that utilizes the concepts from the tutorials. You will work together on one computer, but will switch "drivers/navigator roles" at some point during the challenge. Your solution will be due before the next class period.

As you can see, it is important that you come to class having watched and digested the tutorials so that you and your partner(s) can successfully complete the challenge.

http://www.realsearchgroup.org/pairlearning/videos/pairprogramming_students.mov

ONLINE PORTFOLIO
At the end of the semester you will turn in a portfoilo website that is hosted on creative.colorado.edu. This will include your major projects, pair programing challenges, and other assignments. **Make sure you save copies of all your work.**

If for some reason you don't have a creative.colorado.edu let me know so we can create you an account. More information about creative.colorado.edu server accounts: http://tam.colorado.edu/itresources.html

CODE JOURNALS
Throughout the course, you will keep a journal of what you are experiencing and learning. I will create a Google Doc for each of you that you will use as your Code Journal.

Write about ideas, thoughts, and questions you have as you progress throughout the semester. I expect around 250 words, per week, per student. Please make sure to do some journaling before each Sunday at 5pm. I will read these and make sure that my class plans reflect your questions, ideas, and curiosities. But also feel free to keep on writing after that.

Here are some topics to journal about. (You don't have to use these, but they might be helpful)

- How are the programming challenges going? Are you well prepared? Is your partner? Are you working well with your partner(s)?

- What are the major ideas or techniques that you are working on?
- Is there anything that we have covered that is confusing?
- How are you feeling about programming in general? Is it fun? Is it frustrating? Why?
- Think of this like a sketchbook. What project ideas do you have? (feel free to include sketches or images in your journal!)
- Document your progress on your major projects
- Make sure to label and date each entry!

In addition to these general questions, I will also suggest writing themes for specific weeks.

The final entry will be a longer summary of what you learned in the course. You should identify two to three things that you have learned in this course, select evidence to demonstrate your learning, and write two-three pages about what you've learned, how the evidence shows it, and what you'd like to learn next.

TECHNICAL BENCHMARK
Towards the middle of the semester we will have a technical benchmark exam. This will be an in-class exam that will test your general knowledge of programming as well as the syntax of p5.js. This exam isn't factored into your final grade, however you must pass this exam in order to pass the course. You can retake the exam until you pass it, up until the final week of classes.

GRADES

Category	*Points*
Project 1 (Game Remix)	25
Project 2 (Data Visualization)	25
Programming Challenges	~30* (two points each)
Creative Coding Presentation	5
Online Portfolio	5
Code Journals	10
Technical Benchmark:	0**
Total	100

*you can earn extra points for going over and above the requirements of the pair programming challenges
**you will not pass the course if you do not successfully pass the benchmark

PROJECT EVALUATION RUBRICS
All major projects will be evaluated according to the following categories. Detailed rubrics will be provided for each project, but my general method for evaluating creative projects is:

- Technical (33%)
- Design/Aesthetic (33%)
- Creative/Conceptual (33%)
- Project Plan (1%)

GRADING SCALE
A = excellent work
B = above average work
C= average or competent work
D = below average work
F = unsatisfactory work

In order to counteract grade inflation, I do not give out A's easily. If you turn in all your work on time (and if it is satisfactorily completed), and if you attend class and participate, you are ensured a C. A's and B's are for students who excel beyond average and competent work.

LATE WORK

Late work will not be accepted! If you have not finished a project, you will need to show what you have to avoid getting a "0" for the project. Make sure to backup your work! Lost and/or corrupted work is not an acceptable excuse for late work.

COURSE CONTENT

In this class I reserve the right to show a broad range of course materials, some of which may be offensive to some people. It is not my goal to intentionally offend anyway, but should you feel offended by something you have seen or heard, I would appreciate you staying to be part of a dialogue as I welcome your perspective. If you feel that you cannot stay, feel free to excuse yourself from the classroom as discretely as possible.

IN CLASS WORK/OUT OF CLASS WORK

I want to use in-class time as effectively as possible. Using class-time for technical demos is a waste of your time, therefore the bulk of the technical instruction will happen outside of class via assigned readings, viewings, and tutorials (i.e. this is a "flipped" classroom). This will allow you time to code in class, and get help if you get stuck.

TECHNOLOGY

If you have one, bring your laptop! The software we will be using is free and easy to install, so working on your own laptop will make things much easier for you.

If you plan on using a laptop from the laptop cart, please fill out this form: http://goo.gl/DfXjsX

P5.JS CLUB http://www.creativejs.club/

Meets every other week, on the 1st and 3rd wednesday of each month. This club is not "help hours" and is not curriculum-based but is focused on gathering together students who are excited about creative coding with p5.js as well as other JavaScript libraries.

KENZIE

Since this is larger class, Kenzie is going to be helping out with various aspects of the class. She will be holding walk-in help hours, and will be available to help answer questions about your code on the discussion forum.

HELP!

It is completely normal to need help in this class! We will often get bugs in our code that can be hard for us to find and having someone else look at it can really be helpful. Helping others to debug their code is also a great way to improve your programming skills. When you have a problem with your code, post your question to the discussion topic for the given Programming Challenge or Project (e.g. PC1 Help) and don't forget to post your code. I will also give extra credit to students who actively help others on the discussion boards. Kenzie and I will be monitoring your questions on the Discussion Boards.

E-MAIL

You must use your Colorado.edu E-mail account for this course. Please check your E-mail and the class website regularly. I will notify you of all class cancellations and scheduling changes via the class website and/or e-mail. It is my goal to respond to all E-mails within 24 hours. If I fail to reply within 24 hours, feel free to resend. Please put ATLS3000 in the subject line.

Also, do your best to send me E-mail from your Colorado.edu account. I am not allowed to respond to non Colorado.edu E-mail accounts. For the sake of efficiency (and sanity), I only check class-related E-mail during certain times during the day, typically in the late afternoon.

ATTENDANCE

Attendance will be taken in this class and can negatively affect your final grade. You are allowed three absences* after which your final grade will be lowered by a letter grade for each additional absence. Note that repeated and/or significant tardiness will be considered as absences.

**All absences fall under these three absences (e.g. minor illnesses and injuries, oversleeping, vacations, job interviews, ski-days, family obligations and situations, etc.) so it isn't wise to use them all at the start of the semester. Exceptions will be made for religious holidays, severe illnesses, and prolonged family emergencies.*

CODE AND PLAGIARISM

In this class, we will use other people's code as a way to learn, and you are encouraged to look at and play with online examples and tutorials. If you use someone else's code in a project or lab, you are required to give credit to the author of that code in the comments. **Turning in someone else's code as your own is considered plagiarism and will be dealt with according to the University's plagiarism policies.**

General guidelines:

- No more than 20% of your lab or project should be code snippets from other sources.
- You must give a citation in the code comments for all code from other sources.
- You must be able to explain in detail all the code that you use from other sources.
- If in doubt, ask me!

DIGITAL DISTRACTIONS

Part of learning how to be an adult is learning how to manage various digital distractions such as texting, E-mailing, and using social media. I am not going to prohibit these activities because you need to learn how to integrate these tools into your life, and prohibitions will not help you learn how to manage these activities professionally. However, during class-time, I will ask that you refrain from texting, checking your E-mail, and using social media, as it creates a distraction for you, and more problematically, for your classmates. In the event that these activities become an issue, I reserve the right to amend this policy.

CRITIQUES

I highly value critiques and take them seriously. Even if your project is not completed, it is required that you come to class to offer feedback on your classmates' projects. It will negatively affect your project grade if you are not present for critiques.

PARTICIPATION

It is my goal to make this the best possible experience for you that I can. In order to achieve this goal, I have outlined the expectations that I have for you, and what expectations you can have for me:

My expectations of you:

- That you attend class regularly and arrive on time
- That you are "present" in class, which means:
  - you have completed all the required reading, viewings, exercises, and assignments before class begins
  - That you are present physically and mentally
  - That you ask questions and actively particpate in the class

- That you respect your classmates and the learning environment, and avoid causing unnecessary distractions
- That you are committed to your education and learning experience

What you can expect from me:

- I will do my best to make this class compelling, challenging, and relevant.
- I will take full advantage of being with each other in a physical space (e.g. we will use the class-time in the most efficient way possible).
- I will be fully present and highly prepared.
- I will be available to answer your questions either in person on online.
- I will know your names and will be flexible and responsive to your questions and interests
- I will begin each class on time

ACCOMMODATION FOR DISABILITIES

If you qualify for accommodations because of a disability, please submit to your professor a letter from Disability Services in a timely manner (for exam accommodations provide your letter at least one week prior to the exam) so that your needs can be addressed. Disability Services determines accommodations based on documented disabilities. Contact Disability Services at 303-492-8671 or by e-mail at dsinfo@colorado.edu. If you have a temporary medical condition or injury, see Temporary Injuries guidelines under the Quick Links at the Disability Services website and discuss your needs with your professor.

RELIGIOUS OBSERVANCES

Campus policy regarding religious observances requires that faculty make every effort to deal reasonably and fairly with all students who, because of religious obligations, have conflicts with scheduled exams, assignments or required attendance. In this class, (*insert your procedures here*).
See the campus policy regarding religious observances for full details.

CLASSROOM BEHAVIOR

Students and faculty each have responsibility for maintaining an appropriate learning environment. Those who fail to adhere to such behavioral standards may be subject to discipline. Professional courtesy and sensitivity are especially important with respect to individuals and topics dealing with differences of race, color, culture, religion, creed, politics, veteran's status, sexual orientation, gender, gender identity and gender expression, age, disability, and nationalities. Class rosters are provided to the instructor with the student's legal name. I will gladly honor your request to address you by an alternate name or gender pronoun. Please advise me of this preference early in the semester so that I may make appropriate changes to my records. For more information, see the policies on classroom behavior and the student code.

SEXUAL MISCONDUCT, DISCRIMINATION, HARASSMENT AND/OR RELATED RETALIATION

The University of Colorado Boulder (CU Boulder) is committed to maintaining a positive learning, working, and living environment. CU Boulder will not tolerate acts of sexual misconduct, discrimination, harassment or related retaliation against or by any employee or student. CU's Sexual Misconduct Policy prohibits sexual assault, sexual exploitation, sexual harassment, intimate partner abuse (dating or domestic violence), stalking or related retaliation. CU Boulder's Discrimination and Harassment Policy prohibits discrimination, harassment or related retaliation based on race, color, national origin, sex, pregnancy, age, disability, creed, religion, sexual orientation, gender identity, gender expression, veteran status, political affiliation or political philosophy. Individuals who believe they have been subject to misconduct under either policy should contact the Office of Institutional Equity and Compliance (OIEC) at 303-492-2127. Information about the OIEC, the above referenced policies, and the campus resources available to assist individuals regarding sexual misconduct, discrimination, harassment or related retaliation can be found at the OIEC website.

HONOR CODE
All students enrolled in a University of Colorado Boulder course are responsible for knowing and adhering to the academic integrity policy of the institution. Violations of the policy may include: plagiarism, cheating, fabrication, lying, bribery, threat, unauthorized access, clicker fraud, resubmission, and aiding academic dishonesty. All incidents of academic misconduct will be reported to the Honor Code Council(honor@colorado.edu; 303-735-2273). Students who are found responsible for violating the academic integrity policy will be subject to nonacademic sanctions from the Honor Code Council as well as academic sanctions from the faculty member. Additional information regarding the academic integrity policy can be found at honorcode.colorado.edu.

By enrolling you signify your awareness and understanding of the policies contained within this syllabus and your agreement to conduct yourself accordingly.

ATLS3000, CODE
Schedule

Mondays and Wednesdays
11:00 a.m.–12:50 p.m.
ATLS1B31

Instructor: Joel Swanson Joel.Swanson@Colorado.edu
Office Hours: Mondays and Wednesdays 2–3 p.m. ATLSL101 (or by appointment)

Learning Assistant: Kenzie Weller: Mckenzie.Weller@Colorado.EDU
Help Hours: Tuesdays 2–4, ATLS231 (TAM Suite)

	Monday	Wednesday
1	January 16 Martin Luther King Jr. Day No Classes	January 18 **Introduction** Syllabus
2	January 23 **Journal Entry 01:** After reading the Wing article, in your own words, define "Computational Thinking" **Show and Tell:** Experimental Jetset **Computational Thinking** **Read:** *Computational Thinking*, Jeannette Wing **Watch:** Chapter 1 **PC01:** Imitation (pairs)	January 25 Sol Lewitt: Wall Drawing, Boston Museum 1971 **Generative Art** **Show and Tell:** Casey Reas **Watch:** Chapter 2 **PC02:** Generative Logo (solo)
3	January 30 **Journal Entry 02:** How did pair programming go for you this last week? Were you prepared? Was your partner? Did you work well together? Why or why not? **Algorithms** **Watch:** Chapter 3 **PC03:** Buttons (pairs)	February 1 **Watch:** Chapter 4 | Examples **Watch:** Algorithms **Show and Tell:** Francis Bitioni / http://studiobitonti.com **Show and Tell**: Daniel Temkin: Stripe Modulator **Review:** PC02 Solutions: Kailey | Emma | Dana | Jacob **PCO4:** Pattern
4	February 6 **Journal Entry 03:** How are things going for you? Anything you would like to review in class? Is there anything you would like to learn? **Review:** PC03 Solutions: Hector & Edmond | Meredith | Ian M. **Show and Tell:** Radar Boy **Modularity & Reusability** **Watch:** Chapter 5 | Examples **PC05:** Robot	February 8 **Watch:** Chapter 6A **Abstraction & Objects** **Examples:** Chapter 6A **Review:** PC05 Robot & Robot Rewritten as Object **PC06:** Microbes **Extra Walk-In Help Hours:** Thursday, 10.09.2017 10 am–12 pm ATLSL101 (Joel's Office) I'll be available to review anything, explain something, help you get more practice, whatever!
5	February 13 **Journal Entry 04:** What is one thing you would like to learn how to do with code/programming? How many lines of code? **Show and Tell**: Reza Ali Supershapes 2D 3D **Review:** PC04 Solutions: Taylor | Kailey **Decompose** **Watch:** Chapter 6B | Examples **PPC07:** Screensaver	February 15 **Watch:** Chapter 6C | Examples **Review:** PC05 Solutions: Ian | Emma **Versioning** **PPC08:** Game Someone else's take on Objects Handout on Objects from *Getting Started with p5.js*

6	February 20 **Journal Entry 05:** What seems to keep tripping you up? Semicolons? Matching code blocks/Curly braces? Getting used to looking things up in the Reference? What could you do to help fix this in the future? Quiz (not graded, but do your best!) **Review:** PC06 Solution: Pedro and Edmond **Show and Tell:** Justin Gitlin | Cacheflowe **Various Topics:** Translation, Rotation, Push & Pop | Examples **PC09:** Orbits	February 21 **Demo Day:** Figure out how to do something new in p5.js. This could be a new function, a new code structure, or a new technique. Then, prepare a demo sketch to teach someone else. Here are some videos to get you started. **Show and Tell:** Puddle Builder **Review:** PC07 Valerie | PC08 Kailey, Jillian, and Sophia **Quiz Review:** Results **Demos:** SpaceShooter | Sprite **Project 1 Assigned** No Programming Challenge today! Take a breather, but start thinking about Project 1.
7	February 27 **Journal Entry 06:** What are your initial ideas for Project 1? What do you think will be the big challenges? **Show and Tell:** Holger Lippmann **Watch: Chapter A: Fun with Math** Polar Coordinates, Perlin Noise 1, Perlin Noise 2, and Recursion | Examples **PC10:** Aquarium http://genekogan.com/code/p5js-perlin-noise/	March 1 **Show and Tell:** Ryoji Ikeda | Test Pattern **Chapter B:** More Fun with Math: Sin/Cosine and Forces | Examples **PC10 Continued:** Aquarium 2.0
8	March 6 **Review PC09:** Ali Sophie **Chapter C:** Text (read through the link and play with the examples) | Examples **Show and Tell:** Peter Cho *Wordscapes* video documentation Reza Ali *Misshapenness* Nanika *For All Seasons* Camille Utterback *Text Rain* **PC11:** Word Design **Project 1 Plans Due** (turn in as .PDF to Chalkup) **Technical Benchmark Review**	March 8 **Demo:** SpaceShooter **Benchmark Exam** **Midterm Check-in** for Code Journals and Attendance
9	March 13 **Journal Entry 07:** How did the technical benchmark go? What topics or questions tripped you up? Is there anything you would like to review in class? **Demo:** SpaceShooter (finish up) **Review Technical Benchmark** Practicum Solution **Optional Videos:** Chapter 7A: HTML/CSS/DOM | Examples ***Creative Coder Research Video Project Assigned*** **Workday**	March 15 **Review:** PC 11 Solutions: Hector | Wade **Project 1 Help Hours:** Tuesday, March 21 1–4:00 ATLASL101 **Optional Videos:** Chapter 7B: More HTML/CSS/DOM **Examples:** Timer | BasicPong | GradeGraph **Show and Tell:** Jason Nelson secrettechnology.com **Workday**
10	March 20 **Journal Entry 08:** How is Project 1 going? Are you having any problems or challenges? What needs to be improved? **Optional Technical Benchmark Retake** (If you passed the Benchmark, or if you wish to retake it at a later date, you can work on your Project 1, but I won't be offering any future in-class opportunities to retake the Benchmark) **Workday**	March 22 **How to Critique** **Project 1 Critiques | Feedback Form** Project 2 Assigned ***Homework over Spring Break:*** *Don't forget to get started on your **Creative Coder Video** (due April 10).* *Do **Journal Entry 9,** ideally sometime today.* ***Project 1, Version 2** is due April 3.*
11	March 27 Spring Break / No Classes	March 29 Spring Break / No Classes
12	April 3	April 5

	Journal Entry 09: What did learn from Project 1? What would you do differently next time? **Project 1, Version 2 Due** (make sure that you post both Version 1 and Version 2!) **Show and Tell:** Aaron Koblin Basic Text Visualization | 8A Examples **PC12:** Graph	**Project 2 Links** **Show and Tell:** David Mccandless **Watch:** Associative Arrays **Show and Tell:** Stephanie Posavek **Show and Tell:** Understanding Shakespeare | nand.io **Show and Tell:** The Generative Gatsby **Show and Tell:** Ben Fry **Text Visualization** | Examples **PC13**: Text Visualization
13	April 10 **Journal Entry 10:** How is Project 2 coming along? Are you having any problems finding data sources? **Creative Coder Videos Due and Viewings**	April 12 **Watch**: Chapter 9A | 9A Examples Resubmit a lab: April 26 Things due on May 7: ● Final Project ● Portfolio with working links to: ○ All Programming Challenges ○ Project 1 ○ Project 2 (and documentation) ○ Link to Creative Coder Video ○ Final Version of Code Journals ● Technical Benchmark (Last chance to retake is April 26 in office hours) Creative Coder Video Links **Project 2 Project Plans Due** **Show and Tell:** Feltron.com
14	April 17 **Journal Entry 11:** Update your progress on your Project 2 **Show and Tell:** Jer Thorpe http://www.kellymonico.com/filter/Data-Visualization/Bitches-n-Hoes **Watch**: Chapter 9B | 9B Examples Workday	April 19 **Show and Tell:** Black Cube **FCQ's** Workday
15	April 24 **Project 2 Critiques** Mandel-Van de Pas *Programming Challenge resubmission is due on Wednesday April 26!*	April 26 **Project 2 Critiques** Bachrach–Livingston Resubmit PC to Chalkup *Last chance to retake the Benchmark in Office Hours! (1–2 p.m ATLSL101.)*
16	May 1 Workday with Kenzie (Joel Out) (integrate feedback from final critiques)	May 3 Workday with Kenzie (Joel Out) (integrate feedback from final critiques) **Journal Entry 12:** What did you learn in this class? Due Sunday May 7
F	Final Exam Sunday, May 7 10:00pm We will not be meeting during this Final Exam period, but this is the "due date" for your: ● **Final Project** ● **Online Portfolios** (all programming challenges, projects and a working link to your Creative Coder Video must be present and working to receive full credit) ● **Code Journals**, including Journal Entry 12.	

Ariel Malka

Chapter 1

Video Links:

1.1 Introduction (12:05)

- What is P5.js?
- libraries
- Java isn't JavaScript
- Processing vs. p5.js

1.2 Basics of Drawing (15:11)

- Canvas: HTML element that we draw into
- Functions: what are they?
- Arguments
- Coordinates in p5.js
- x and y locations on a grid
    - y axis is inverted!
- Basic drawing functions
    - `rect(x,y,width,height)`
    - `ellipse(x,y,width,height)`
    - `line(x1,y1,x2,y2)`
    - `point(x,y)`
- Reference on the p5.js website

1.3 Color (15:18)

- Basic color functions:
    - `background()`
    - `stroke()`
    - `fill()`
- color modes
- comments in your code
    - Single line comment: `//`
    - Multiple line comment: `/* … */`

1.4 How to Upload Your Sketch (13:46)

- Local vs. remote servers
- FTP Client
- How to save/export your sketch
- How to upload (and view) your sketches

Remember to put your name (and your partner's name), date, and the name of your lab/project at the top of your code! Everyone must turn in a copy of the lab/project when working in a group.

Technical Benchmark

ATLS3000 Code / Spring 2017

Name ______________________________ Date ____________

1. What will the following code print out to the console?

```
var apples = 45;
var bananas = 10;
var oranges = 3;
var x = apples - bananas * oranges;
Var x = 15;
print("the value of x = " +  x);
```

2. What color will the following code draw to the background? (Give the general color name, e.g. “pink”)

```
var myNumber = 37;

if(myNumber > 75){
       background(255,0,0);
} else if (myNumber > 50){
       background(0,255,0);
} else if (myNumber > 25){
       background(0,0,255);
} else {
       background(255);
}
```

3. Circle all the *arguments* in the following code:

```
var yloc = width/2;
fill(255,0,0);
rect(x, yloc, 50, 50);
print("y = " + yloc);
```

4. What will the following code print to the console?

```
for(var i = 0; i < 5; i++){
 println(i*10);
}
```

5. What symbol(s) do we use to create an array?

6. What is an object?

7. What is a *constructor function* and what does it do?

8. What does the `this` keyword do inside a constructor function?

9. What is the difference between the `setup()` and `draw()` functions inside p5.js?

10. In the code below, circle the **global variables**, including each time they are used.

```
var apples;
var pineapples;

function setup(){
	createCanvas(500,500);
	apples = 100;
	var bananas = 90;
}
function draw(){
	apples = apples - bananas;
	var oranges = 10;
}
```

11. What is the syntax for making a multi-line comment?

12. p5.js is a “library” of JavaScript. What does that mean? Give an example of another JavaScript library.

13. Give **three** examples of *system variables*.

    1.
    2.
    3.

14. What is wrong with the following code?

```
var things = 10;
if(things = 100){
 println("you have 100 things!");
}
```

15. What will the following code print out to the console?

```
var count = 32;
var Count = 33;
println("count = " + count);
```

16. Circle the **allowable** variable names:

```
var one_Count;
var 1count;
var _count
var my name;
var *score;
var myScore;
var width;
```

17. What is a two-dimensional array and when are they useful?

18. What is a *boolean* value?

19 . What value(s) will the following code produce? Be specific!

```
random(5.5,10);
```

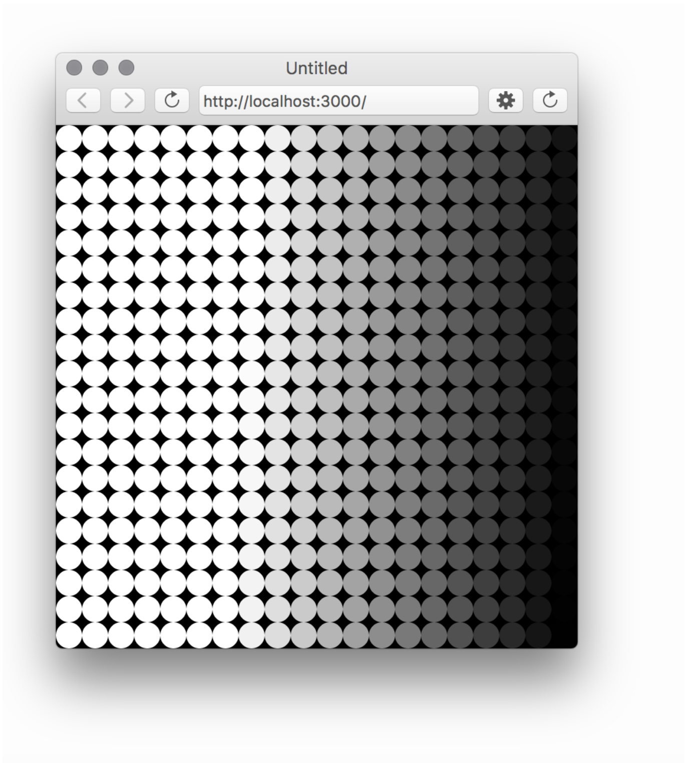
20. What is an infinite loop?

21-25. **Correct** the errors in the Constructor function below.

```
function Cell(radius) {
  this.x = random(width);
  this.y = random(height);
  this.r =radius;
  this.c = color(rgb(255, 0, 0));

  this.display = function() {
    fill(this.c);
    ellipse(this.x, this.y, this.r, this.r);
  }

  This.move =  function() {
    this.x = random(-3, 3);
    this.y = random(-3, 3);
  }
}
```



Practicum: 5 points
Recreate the visual above using p5.js.

- You can use the p5.js reference: https://p5js.org/reference/
- There are many different ways to solve this, but your solution must use loops (i.e. do not "hard code" the values of each ellipse)
- You don't need to use objects for this solution.
- You can accomplish this in less than 20 lines of code.
- Give this your best attempt! I will give partial credit.
- E-mail your completed code (**copy and paste** the code directly into the message of the e-mail. Don't email it as an attachment or it will get filtered) to me at joel.swanson@colorado.edu

Extra:
22. What is “computational thinking?”
23: while loops
24: function calls
25: object methods
26: What is an infinte loop?
What is the name of the mathematical equation that we use to find the distance between two points in space?

Technical Benchmark Version 2

ATLS3000 Code / Spring 2017

Name ______________________________ Date ____________

1. What will the following code print out to the console?

```
var cats = 10;
var dogs = 22;
var gerbils = 2;
var x = dogs - cats / gerbils;
print("x = " +  x);
```

2. What color will the following code draw to the background? (Give the general color name, e.g. “pink”)

```
var myNumber = 200;

if(myNumber > 100){
       background(255,0,0);
} else if (myNumber > 200){
       background(0,0,255);
} else if (myNumber > 300){
       background(0,255,0);
} else {
       background(255);
}
```

3. Circle all the *arguments* in the following code:

```
var speed = 2;
fill(255,0,0);
rect(x, y, 50, 50);
x += speed;
```

4. What will the following code print to the console?

```
for(var x = 50; x > 0; x-=10){
 print(x*5);
}
```

5. What is an *array*?

6. What is the relationship between an *object* and its *constructor*?

7. How is the `noise()` function different from the `random()` function?

8. What is wrong with this code?

```
for(var y = 0; y >= 0; y++){
 print("y = " + y);
}
```

9. What does the `noLoop()` function do?

10. In the code below, circle the **global variables**, including each time they are used.

```
var score;

function setup(){
       createCanvas(500,500);
       score = 0;
       var players = 2;
}
function draw(){
       score++;
       var Score = 3;
       print("score");
}
```

11. What is the output of the following line of code:

```
var lastName = "Bieber";
var firstName = "Justin";
var name = firstName + " " + lastName;
print(name.charAt(7));
```

12. Give **two** p5.js functions that *return* a value:

1.
2.

13. What is wrong with the following code?

```
var score = 10;
if(score = 100){
 println("you win!");
}
```

14. Give **two** examples of *system variables*.

1.
2.

15. What will the following code print out to the console?

```
var score = 100;
var Score = 200;
var SCORE = 300;
print("Score = " + score);
```

16. Circle the **allowable** variable names:

```
var player1_score;
var 1score;
var _score;
var player score;
var *score;
var playerScore;
var new;
```

17. What does *concatenate* mean?

18. What is the difference between the `floor()` and `round()` functions?

19 . What does the ++ symbol do?

20. What will the following code print:

```
var i = 47;
if (i < 50 && i > 0) {
  if (i == 45 || i == 46 || i == 47) {
    print("cheese");
  } else {
    print("pickles");
  }
} else {
    print(“peppers”);
}
```

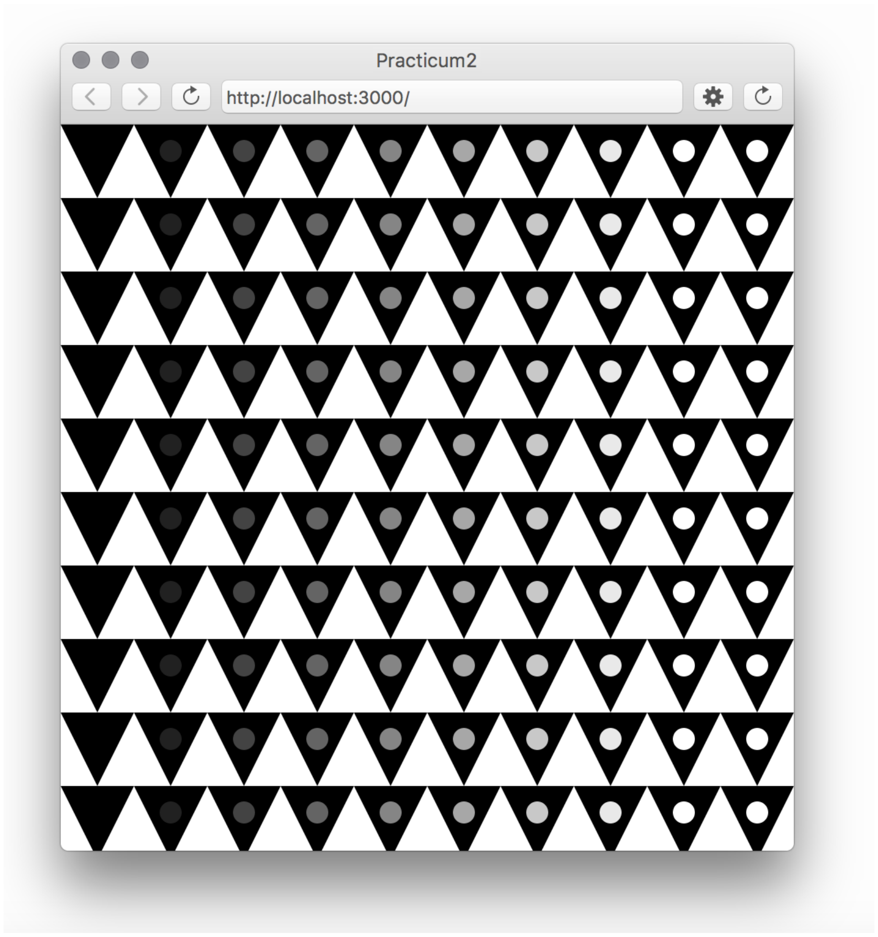
21-25. Correct the **five** errors in the Constructor function below.

```
function Orb(x, y) {
  this.x = x;
  this.y = y;
  this.r = random(50);
  speed = 2;

  this.display = function() {
    fill(this.color);
    ellipse(this.x, this.y, this.r, this.r);
 }

  function update() {
    this.x += this.speed;
    if(this.x = width){
        x = 0;
  }
}
```

Practicum:
5 points



Recreate the visual above using p5.js.

- You can use the p5.js reference: https://p5js.org/reference/
- There are different ways to solve this, but your solution must use loops (i.e. do not "hard code" the values of each shape)
- You don't need to use objects for this solution.
- You can accomplish this in less than 20 lines of code.
- Give this your best attempt! I will give partial credit.
- E-mail your completed code (**copy and paste** the code directly into the message of the e-mail. Don't email it as an attachment or it will get filtered) to me at joel.swanson@colorado.edu

Technical Benchmark Version 3

ATLS3000 Code / Spring 2017

Name ______________________________ Date ______________

1. What will the following code print out to the console?

```
var hearts = 5;
var diamonds = 7;
var spades = 2;
var clubs = 13;
var cards = (diamonds - hearts) * spades;
print("cards = " +  cards);
```

2. What color will the following code draw to the background? (Give the general color name, e.g. “pink”)

```
var myNumber = 30;

  if (myNumber < 10) {
    background(0, 255, 0);
  } else if (myNumber < 20) {
    background(0, 0, 255);
  } else if (myNumber < 30) {
    background(255, 0, 0);
  } else {
    background(0);
  }
```

3. Circle all the *arguments* in the following code:

```
var x = 0;
var speed = 2;
x += speed;
fill(255,0,0);
ellipse(x, y, 50, 50);
```

4. What will the following code print to the console?

```
for(var i = 0; i < 5; i++){
 print(i*5);
}
```

5. What is an *array*?

6. What is the relationship between an *object* and its *constructor*?

7. What does the `this` keyword do?

8. What is wrong with this code?

```
for(var x = 0; x >= 0; x++){
 console.log(x);
}
```

9. What is the syntax for a multiline comment?

10. In the code below, circle the **global variables**, including each time they are used.

```
var score = 0;
var total = 100;

function setup(){
       createCanvas(500,500);
       score = 1;
       var players = 2;
}
function draw(){
       score++;
       var Score = 3;
       print(“score”);
}
```

11. What is the output of the following line of code:

```
var firstName = "Oprah";
var lastName = "Winfrey";
var name = firstName + " " + lastName;
console.log(name.charAt(9));
```

12. Give **two** p5.js functions that *return* a value:

1.
2.

13. What is wrong with the following code?

```
var score = 10;
if(score = 100){
 print("you win!");
}
```

14. Give **two** examples of *system variables*.

1.
2.

15. What will the following code print out to the console?

```
var score = 100;
var Score = 200;
var SCORE = 300;
console.log("SCORE = " + score);
```

16. Circle the **allowable** variable names:

```
var variable_one
var variableOne;
var 1variable;
var _variableOne;
var variable one;
var +variableOne;
var var;
```

17. What does the `translate()` function do?

18. What does the `splice()` function do?

19 . What is a variable?

20. What will the following code print:

```
var i = 5;
if (i < 10 && i > 0) {
  if (i == 3 || i == 4 || i == 5) {
    console.log("mustard");
  } else {
    console.log("ketchup");
  }
}
```

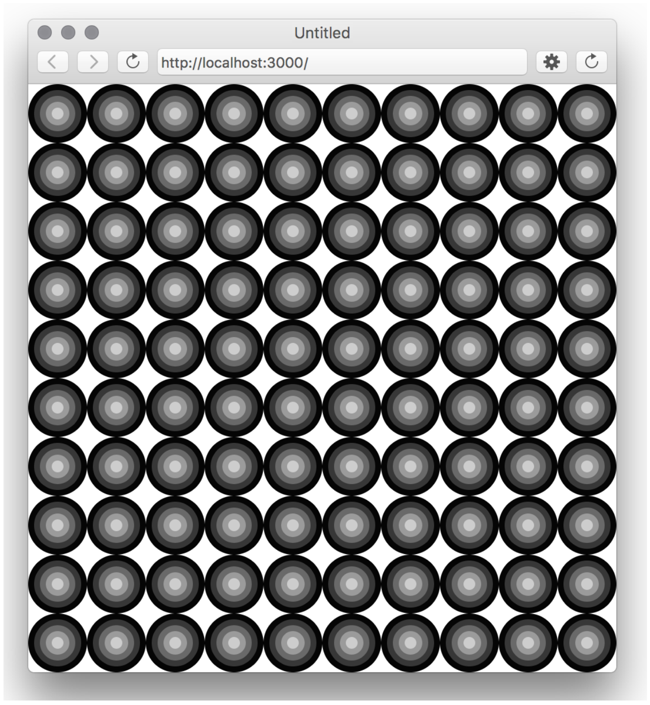
21-25. Correct the **five** errors in the Constructor function below.

```
    function 1Orb(x, y) {
      this.x = x;
      this.y = y;
      this.r = random(50);
      speed = 2;

      this.display = function() {
        fill(255,255,100,50);
        ellipse(this.x, this.y, this.r, this.r);
     }

      function update() {
        this.x += this.speed;
        if(this.x = width){
            x = 0;
        }
      }
    }
  }
```

Practicum:
5 points



Recreate the visual above using p5.js.

- You can use the p5.js reference: https://p5js.org/reference/
- There are different ways to solve this, but your solution must use loops (i.e. do not "hard code" the values of each shape)
- You don't need to use objects for this solution.
- You can accomplish this in less than 20 lines of code.
- Give this your best attempt! I will give partial credit.
- E-mail your completed code (**copy and paste** the code directly into the message of the e-mail. Don't email it as an attachment or it will get filtered) to me at joel.swanson@colorado.edu

Project 1
Video Game Remix

Use p5.js and the required media assets outlined below to create a video game remix.

- Your game must include:
  - A famous painting of your choosing.
  - One or more sounds from this folder.
  - One or more characters from the images in this folder.
- You can edit, rework, and remix the required assets, but they must be present in some way, and you will be evaluated on how well you integrate the assets together (think "Chopped").
- The goal of this project is to challenge you creatively, conceptually, and technically!
- The project will be turned in as an online game.
- Due March 22
  - Create a Project Plans for Project 1. Project Plans should contain an overall description of your project, sketches/images, as well as a technical synopsis of how you are going approach the project. Turn in your project plan as a .pdf

Tips:

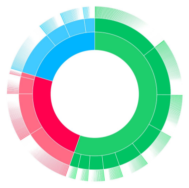
- Start out with simple goals technically. You can always make it more complex as you go along, but biting off something overly ambitious technically can lead to lots of frustration.
  - Games that seems simple, but are actually quite complicated:
    - Mazes
    - Side scrolling games
    - Platform games
- VERSION! Build this thing in steps and save versions as you successfully accomplish each step.
- Get the fundamental structures of the games working first, then spend time on the aesthetic components.
- Kenzie and I are here to help you! We are happy to meet with you to give you feedback on your overall idea, get you started in the right direction technically, whatever!

Steps:

1. Design and describe your game.
2. Break the game down into OBJECTS
3. Figure out what PROPERTIES and FUNCTIONS each object needs to have
4. Figure out how those objects interact and write the interactions as functions
5. Keep iterating to make things more robust!
6. Test, test, test!

Rubric:
25 total possible points
Project Plans are worth 1 point

	Exceeds Expectations (7–8 points)	**Meets Expectations (5–6 points)**	**Approaches Expectations (4–3 points)**	**Fails to meet Expectations (0-2 points)**
Conceptual	The project is unique, conceptually compelling, and addresses the assignment in a creative way. The aesthetic, technical, and conceptual aspects have a strong relationships, and are compelling. The required assets are used creatively and uniquely and make a cohesive narrative.	The project addresses the prompt creatively, but could be more unique. The aesthetic, technical, and conceptual aspects have a strong relationships. All the required assets are present, but could be used in a more interesting and compelling way.	The project isn't particularly creative, but still addressed the prompt sufficiently. The aesthetic, technical, and conceptual aspects could have a stronger relationship. Most of the required assets are present, but aren't used in an interesting way.	The project doesn't address the prompt, and the concept lacks creativity. The aesthetic, technical, and conceptual aspects have no relationship. All the required assets are not present.
Design	The overall design is outstanding, professional and connect compellingly to the technical and conceptual aspects of the project.	The overall design is cohesive and compelling, but could be executed more professionally.	The overall design reflects effort, but could be stronger and more visually compelling.	The overall design is sloppy, lacks visual interest, and reflects little or no effort.
Technical	The project utilizes the technical concepts presented in class in a robust way, and includes more concepts not presented in class. The code is commented and is concisely and efficiently written.. The project is technically robust with no bugs or glitches.	The project utilizes most of the technical concepts from the class. The code is commented, and well structured, but could be more efficient. The project is sophisticated technically with a few minor bugs or glitches.	The project only uses the most basic technical concepts presented in class. The code is commented, but could be written in a better and more efficient way. The project is technically simplistic and with significant bugs and glitches.	The project doesn't utilize the technical concepts presented in class. The code is written in a confusing/problematic way. Lacks comments The project is extremely simple technically with serious bugs and glitches.



Project 2: **Data Visualization**

Using p5.js, create a data visualization that explores how the design of data can affect its meaning.

Objectives

- Use data visualization to demonstrate a non-obvious insight gleaned from the data, or to make a particular point.
- Use p5.js to visualize large sets of data
- Explore how design can be used as a persuasive tool
- Learn about the various ways data is stored and accessed (e.g. CSV, JSON, XML, API's)
- Become familiar with the various approaches in information design and visualization

Project Plan: Due April 12

- A **link** to your data set or data source
- A **general description** of your visualization
- A **list of goals/outcomes** (what do you want to "say" about your data?)
- **Sketches/mockups** of your visualization
- Description of the **technical aspects** of your project
- Any **challenges or obstacles** that you foresee

Details

- Your final project can in any media (online, interactive, print, 3d print, physical object) but it must use p5.js as the primary development tool (i.e. not Photoshop).
- Your final project should utilize code to accomplish what would otherwise be tedious, cumbersome, or even impossible (think **data visualization** not **infographic**).
- You may use any dataset you wish for the assignment, but finding a good dataset can be difficult. Pick something that interests you, but be flexible as you search for data.

Project 2 Rubric: 25 total possible points / Project Plans are worth 1 point

	Exceeds Expectations (7–8 points)	**Meets Expectations (5–6 points)**	**Approaches Expectations (4–3 points)**	**Fails to meet Expectations (0-2 points)**
Conceptual	The project is unique, conceptually compelling, and addresses the assignment in a creative way. The aesthetic, technical, and conceptual aspects have a strong relationships, and are compelling. Data is integrated proficiently and creatively.	The project addresses the prompt creatively, but could be more unique. The aesthetic, technical, and conceptual aspects have a strong relationships.	The project isn't particularly creative, but still addressed the prompt sufficiently. The aesthetic, technical, and conceptual aspects could have a stronger relationship.	The project doesn't address the prompt, and the concept lacks creativity. The aesthetic, technical, and conceptual aspects have no relationship. Data isn't integrated proficiently or creatively
Design	The overall design is outstanding, professional and connect compellingly to the technical and conceptual aspects of the project.	The overall design is cohesive and compelling, but could be executed more professionally.	The overall design reflects effort, but could be stronger and more visually compelling.	The overall design is sloppy, lacks visual interest, and reflects little or no effort.
Technical	The project utilizes the technical concepts presented in class in a robust way, and includes more concepts not presented in class. The code is commented and is concisely and efficiently written.. The project is technically robust with no bugs or glitches.	The project utilizes most of the technical concepts from the class. The code is commented, and well structured, but could be more efficient. The project is sophisticated technically with a few minor bugs or glitches.	The project only uses the most basic technical concepts presented in class. The code is commented, but could be written in a better and more efficient way. The project is technically simplistic and with significant bugs and glitches.	The project doesn't utilize the technical concepts presented in class. The code is written in a confusing or problematic way. Lacks comments The project is extremely simple technically with serious bugs and glitches.

Inspiration:

- http://flowingdata.com
- http://www.informationisbeautiful.net
- Data vis example: https://www.youtube.com/watch?v=tEczkhfLwqM

Some Possible Datasets:

- OpenWeatherMap - http://openweathermap.org/API
- NYTimes - http://developer.nytimes.com/

- http://open-notify.org
- https://thecountedapi.com
- The Guardian - http://www.theguardian.com/open-platform
- flickr - https://www.flickr.com/services/api/
- MTA - http://web.mta.info/developers/developer-data-terms.html#data
- Foursquare - https://developer.foursquare.com/
- Sunlight Foundation - http://sunlightfoundation.com/api/
- http://en.wikipedia.org/wiki/List_of_open_APIs
- https://gist.github.com/afeld/4952991
- https://github.com/dariusk/corpora/tree/master/data
- This is a newsletter created by Jeremy Singer-Vine: https://twitter.com/jsvine
- https://www.dataquest.io/blog/free-datasets-for-projects/
- http://www.idvbook.com/teaching-aid/data-sets/
- http://tinyletter.com/data-is-plural/archive
  http://web.stanford.edu/class/cs46n/datasets.htm
- http://rs.io/100-interesting-data-sets-for-statistics/
- https://perso.telecom-paristech.fr/~eagan/class/igr204/datasets
- https://www.quora.com/What-are-some-interesting-public-datasets-to-visualize
- http://flowingdata.com/2015/12/22/10-best-data-visualization-projects-of-2015/
- https://www.springboard.com/blog/free-public-data-sets-data-science-project/
- Good API: https://developers.themoviedb.org/3/movies
- http://data.un.org/
- https://data.nasa.gov
- https://www.cdc.gov/nchs/data_access/NIST Mugshot Identification Database (MID)
- **Colleges and economic mobility.** A team of economists studying “the equality of opportunity” has published new research identifying which colleges “help the most children climb the income ladder.” For their analysis, the researchers combined federal tax records and data from the Department of Education. California State University–Los Angeles was one of the greatest engines of mobility; nearly 1 in 10 students enrolled there began in the bottom 20% of income but reached the top 20% by their early thirties. You can download the findings, which include similar statistics for more than 2,000 schools, as a series of spreadsheets. **Related:** “Some Colleges Have More Students From the Top 1 Percent Than the Bottom 60. Find Yours,” from the *New York Times*.
- **Three centuries of UK macroeconomic data.** The Bank of England publishes a spreadsheet of historical economic data going back, in some cases, to the late 1600s. The country’s GDP in 1700 was £11.7 billion in 2013 prices. That’s about 1/157th the size of the UK’s GDP in 2015. And in November 1694, monthly short-term interest rates were roughly 6%. [h/t Ian Greenleigh]
- **TV talk.** The GDELT Project and the Internet Archive have collaborated to make the latter's Television News Archive more powerfully searchable. Their new tool, announced in December, lets you search across “more than 5.7 billion words from over 150 distinct stations spanning July 2009 to present” at a sentence-by-sentence level. The results are downloadable as CSV or JSON files. **Previously:** The Political TV Ad Archive (Feb. 2, 2016).
- **European trees.** EU-Forest is a new dataset that, according to its authors, “extends by almost one order of magnitude the publicly available information on European tree species distribution.” The new project merges and harmonizes data from 21 national forest surveys and two related databases. In all, EU-Forest includes more than 580,000 observations of more than 200 species in 1km-by-1km square plots of land, and is available in both tabular and geospatial file formats. **Previously:** American tree maps (Dec. 23, 2015) and NYC street trees (Nov. 16, 2016).