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.

- Computer Science

Tags:
- 2016-17 MTLV
- Portfolio Keywords
- art
- code
- creative projects
- p5js
- programming
- technology

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) 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.

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).

**Technical Benchmark 1**  
**Technical Benchmark 2**  
**Technical Benchmark 3**

### 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:*

This project was impressive technically, aesthetically, and conceptually. All the required assets were integrated to create 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:

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:

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:**

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:**

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.

Sample Project 3 for Data 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—but 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.