Haolei Ma 40316304

Professor Pippin Barr

CART 253-A

2 December 2025

Post-Reflection

When I first walked into CART 253, my relationship with programming felt awkward—almost like meeting someone new and not knowing what to say. And when I realized my classmates could effortlessly answer questions in class and speak in terms I couldn't understand, I felt they probably didn't even need to take this course. I had touched JavaScript before, and I've also seen some p5.js projects and examples. but I didn’t really understand how any of it worked. I could copy bits of code, change the color or size of something, and pretend I knew what I was doing. But if anything broke—which it usually did—I had no idea how to fix it. Looking back now, that feels like a completely different person.

One of the biggest changes this course gave me is the ability to read code with confidence. At the start of the semester, terms like variables, loops, or conditions felt abstract, even after repeatedly watching the videos in the learning materials, I still couldn't fully grasp it. Now they feel like actual tools I can use to build something. For example, I used to think “if” statements were just like a magic trigger. Now I understand how logic flows, how programs make decisions, and how I can structure interactions around them. The same happened with loops. Before, repetition in code felt frustrating— “Why does this run so many times?”—but now I actually use loops intentionally to generate visual patterns, animations, and entire systems.

p5.js has also transformed from an incomprehensible machine into a medium I can actually shape. In the past, I didn’t understand why we needed both “setup()” and “draw()”. Now they feel almost natural, like the inhale and exhale of a sketch—setup builds the space, draw makes it move. I’m finally building things from scratch instead of assembling pieces from tutorials, and this has made me feel more like an author of my code instead of a visitor.

Before this course, I relied mostly on visual intuition from my background in design and media arts. Perhaps because I primarily focused on graphic design in the past, my process was mainly about composition, color, storytelling, and

atmosphere. But creative coding introduced something I had never really used before: rules. Instead of drawing the final image, I learned to build a system that could generate the image. Even small interactive elements felt exciting. The moment I mapped “mouseX” and “mouseY” to visuals, suddenly the work wasn’t static anymore. Adding randomness also changed my thinking—it showed me that I could let the computer participate in the creative process. I started making sketches that could shift, react, and behave differently every time they run.

But I won’t pretend everything is easy now. I still get overwhelmed when programs get bigger or when I try to organize everything into functions. And I know I still need more practice with objects and arrays. I can feel how powerful they are—especially for systems like particles, scenes, or autonomous agents—but I’m not fully confident yet. Instead of discouraging me, this gap motivates me. I want to create more complex systems, and I know I’ll need these tools to do that.

Right now, I can already do things I couldn’t do before: I can create movement, behaviors, patterns, feedback loops, and interactions. I can use logic to shape aesthetics. I can design relationships between elements instead of just designing elements themselves. I’ve started to think like someone who uses computation as part of their artistic vocabulary—not just someone who “uses a program to draw things,” but someone who builds the rules behind the drawing.

One of the biggest changes for me is that I actually feel closer to calling myself a creative coder. At the beginning of the course, I saw creative coders as people who were somehow “born good at this,” people who could read abstract logic the same way I read visual composition. Now I see that creative coding is more about experimentation than perfection.

My understanding of creative code has expanded too. I used to think code restricted creativity because it was so structured and logical. But now I see the structure itself as a kind of expressive material. A loop can create rhythm. A conditional can create behavior. Randomness can create surprise. All these technical things actually support artistic intuition instead of blocking it.

Looking forward, I want to build more complex generative systems—things that feel alive, things that evolve over time, things that interact with each other. I want to learn how to use classes to create swarms, ecosystems, or objects that behave with their own logic. I’m also excited to explore other directions like sound, sensors, or even machine learning.

But more than anything, I want to bring more conceptual intention into my creative code. Early in the semester, I was focused on making things that simply looked interesting. Now I’m thinking about what computational art can say—how code can express ideas about perception, interaction, structure, and control.

This course changed my relationship with programming in a way I didn’t expect. I went from someone who copied code without understanding it to someone who can write, modify, troubleshoot, and imagine with code. I learned the logic behind creative computation, but I also learned how to bring my artistic thinking into that logic.

I know I still have a lot to learn, but I finally feel like creative coding is something I can grow into—not something I’m standing outside of. I’m starting to feel like this role belongs to me. So, I’m excited to see where this path will take me next. And this is only my first semester in this field of study; I believe the upcoming courses will present more challenges and difficulties. But at the same time, it must be quite interesting. The true joy of programming lies not merely in completing a perfect piece of work, but in the continuous exploration and self-transcendence that make the process itself so much more interesting, I guess that's how it is.