Post-Reflection

As a student of an unrelated field, being sculpture, I had no prior knowledge of coding even believing this practice to be completely unattainable for my exposure to coding was that of difficulty beyond my skill level. Yet, I have always found interest in the computational arts thus leading me to step out of my comfort zone and reattempt a dream, once lost in busied workloads, to broaden my skillset and utilize resources at Concordia. The night before my devised plan of sitting in on the first day of the course was filled with uncertainty, even hours before I arrived at the classroom, but I am proud of my decision. Acting almost likewise of the butter-fly effect, the weeks that followed admission into CART-253 led to extraordinary fulfilment in opportunity, new friendships and renowned intrigue of what the computational arts represents. I must include that my use of terminology might not be accurate, I learn by creating my own “language” or means of understanding, whether effective or not. The early beginnings of my coding journey were as expected, a humorous example being the fact that it took me approximately two weeks to learn how to efficiently create and link my projects to the moodle submission. I took initiative in learning outside of class deliberately to excel due to my embryonic skill level. Besides the tutorials on the p5js website, youtube and such, I would like to sincerely thank Pippin Barr for the interactive and engaging presentations, that without would have left me stranded. The first project I made, the night after the second class;https://consumedlocusts.github.io/cart253/topics/tests/testP5/
was using mouseX, mouseY, an appreciatively pivotal “function” in my learning path. Thankfulness of Sabine Rosenberg’s patience and assistance is an underestimate, the help I received on the weekly basis led to an extreme upgrade from simple ellipse drawing to using for() loops and applying character index (counters) in the first art-jam project;
https://consumedlocusts.github.io/cart253/topics/billy-ball/
from “how can I draw with text?” to “how can I access each letter in a string, and “change” them, one by one?”. This project was a foundational grasp of everything, from how to use global vs local variables, naming, calling, working with images/pixels in general (most used feature in all of my projects is images/videos, preloading properly, when to loadPixels/access pixel data to read RGB values), complex grid and “string” optimization related functions (textAscent() + textDescent() = approximately full height of fonts, floor: rounding down how many characters fit across the canvas, making this number into an integer, rows, columns how they fit total number of character positions (technically cells but being the first project, weird names I chose) in the grid), the for loop and array fundamentals (starts the counter at 0 keeps going as long as “counter” is less than my grid, increases by 1 each iteration. As the loop counts upward, column index (h index in the grid) calculates the column by taking the remainder when the counter is divided by the number of columns so it wraps back to 0 after each row, then the row index (v in the grid) calculates the row by dividing the counter by the number of columns and rounding down to get how many full rows have been completed, thus h (x) and v(y) are converted into a pair of grid coordinates, making mapping of every character to/from its correct position(left-to-right and top-to-bottom) accessible for my plans), and finally what the project was purposed to be; ASCII-fied interactive string-image (pixel index in the image as if the image were a 1D array of pixels, each pixel in the array is stored as four consecutive numbers representing the red, green, blue, and alpha channels(RGBA, but ignoring Alpha in this case), then computing the “brightness” by averaging the RGB values,producing a single grayscale value=selecting the character whose visual density best corresponds to the pixel’s brightness level). This quick transition, although producing a rough

outcome, is when I realised that I enjoyed the writing of the code, its story and the learning journey of each line typed and committed(complex or not)more than the final result.
In between the “how else can I apply this (without repeating themes)” creative purgatory, I slowed down and focused on each topic with the corresponding challenge, a favourite; https://consumedlocusts.github.io/cart253/topics/randomhogevent/
highlights the first time using/experimenting sound, with the great help of our TA Kamyar, being a success.
Leading to my first playable game, using hand-drawn, original sprite GIFs and reused code from previous challenges I heavily modified the assigned frogfrogfrog (by Pippin Barr) to resemble the aesthetics of an old FPS (first person shooter) game;https://consumedlocusts.github.io/cart253/topics/frog.mod.jam/ This was inspired by 1993’s DOOM, my “alien” creatures parallel to the enemies of that classic game. Revisiting an in-class follow up of “game states” by Sabine Rosenberg accommodated the majority of the game’s user-friendly foundations (states regard the current condition of a collection of variables that will change, in this case its the visual scene changes, once an action or task is complete, current state === next state). Likewise of the previous learned methods, I used for loops to process and draw the many elements (aliens) without repeating code manually. Each loop moves through an array of aliens (8) one alien at a time, allowing the hidden “special” alien (that is concealed under a dark mask), to be differentiated (larger) amongst the “decoy” aliens who are “activated” in the following scene (after this alien is “scanned”). These technical aspects would later on shape my final variation jam, that with the help of many sample codes, was able to be translated to my vision. Titled after my github username “locust.host”, I am finally able to apply everything absorbed from the course into a triad (quad) of heavily stylized playthroughs. It is built on everything I struggled with earlier in the semester; pixel reading, text animation, sound analysis, for loop indexing, classes, state progression, and particle systems;
https://consumedlocusts.github.io/cart253/topics/locust.host/. Each “storyline” abstractly relies on different techniques, such as ASCII conversion, drifting particles shaped into sentences, audio-reactive waves(made a song for this feature), brightness-mapped pixel systems, and masked scenes that reveal content through interaction. Additionally, and most important, I also learned to manage complexity, which became crucial when working across four interconnected scripts. The highlighted “state” within all of these scripts is found as the particle “de-attractor” mountain image, using a base code;https://openprocessing.org/sketch/2308471 a hidden p5 graphics canvas (integral to using particles in ALL of the states, is p5.graphics) loads the mountain image and reads its pixel data(brightness, then silhouettes the image), these new coordinates become destinations for the free particles in the “swarm”. The particles are then matched with these points and begin moving toward them using lerp. As each particle reaches its assigned spot, the shape of the mountain slowly appears, yet the particles dismember (hence “de-attractor”) once the mouse is hovering/in contact with them. The image is not drawn directly but is reconstructed (then deconstructed again) through many particles following pixel data.Tthis is repeated in the state following but instead of “free particles” it is directly animated, and no longer contains interactive particles (the locust image cannot be distorted). Now I know that with intense time and practice I can create experiences that merge narrative, interactivity, sound, and visual complexity. I look forward to developing this skill further and continuing to integrate computational thinking into my art. In short, before this course, I would not have believed that I could build something like “locust.host”.

I have many concepts to further understand deeply, such as object-oriented programming(classes, explored heavily in locust.host, a whole essay on its own), cleaner code structuring, and optimizing beautiful large particle systems. However, the existing foundation allows me to approach these challenges without fear. I am excited by the idea that code can support the interactive installations I want to build in sculpture, especially projects that combine physical structures with projection, sensors, and responsive visuals.