Retrospective Essay
Artificial Intelligence
Professor Spector
Mann, Isaiah
12.09.14

My progress with clojure has been slow and painful. Learning not only a new language, but also a new type of programming has proved difficult. Functional programming feels particularly at odds with the object oriented approach of C# and Javascript: which served as my introduction to programming. However, I've become comfortable enough with Clojure to use it in the pucks environment and execute basic mathematical functions. Beyond that, my abilities begin to falter. As for AI concepts, I feel as though I've done a reasonably job of learning them, though I found the textbook challenging to understand. I believe the AI concept I grasp most solidly is state-based control, which I managed to incorporate in multiple instances within my pucks programming (as seen in my portfolio).

In terms of AI concepts, I attempted to engage with the textbook but often times found it difficult to comprehend. The textbook assumed a great deal of pre-existing knowledge which I did not possess. I was able to observe some crossover between it and my Introduction to Discrete Math course, e.g. logic, node based maps, and recursive functions. However, I don't believe I possessed the background necessary properly appreciate and learn from the text.

In terms of my Clojure progress, I started very small: making random text generators and mathematical functions. I spent a great deal of time working with Clojinc, and often got stuck on relatively simple challenges: e.g. shuffling a deck of cards or making a prime number checker. I do believe my slow comprehension of Clojure hampered my in-class engagement.

While the lectures and code demonstrations were interesting, I couldn't always properly follow the Clojure code used. That beings said, one AI concept I engaged well with was search, owing mostly to my collaboration with fellow student, Grusha Prasad. Collaborating with Grusha on the Missionaries and Cannibals logic problem was perhaps one of my most successful moments in marrying the AI concepts and Clojure. We employed the search function from class and successfully designed a fitness function to solve Missionaries and Cannibals in 11 steps.

I would accredit this success to a minor breakthrough on my part in the basic functionality and syntax of Clojure, and for scope of vision on Grusha's part. This combination of ambition and epiphany proved a perfect mixture for solving this logic problem. To elaborate

on our collaborative process, Grusha and I worked side by side to write the code. We communicated verbally, workshopping ideas and debating the best modes of implementation. While a majority of our work was done in person, we also sent code back and forth through email, describing our intentions and struggles.

We went on to demo the Missionaries and Cannibals solution in class, followed by an intelligent user agent from the Pucks environment. This puck made use of :memory (a hash map stored within it) to reason about the locations of hazards in the world, based off the use of beacons (which stored absolute world positions in their ID's). In addition to this, it also featured a system of hierarchical control that prioritized its necessary survival functions, from avoiding destruction to finding food. We borrowed the basic hierarchical control from Grusha's prior work with pucks, and worked in tandem to develop more advanced navigation. My personal contributions were implementing how the agent stored beacon ID's in memory, and compared them to currently sensed ones, as well as fine tuned reasoning about when to rotate away from obstacles. Once again though, a great deal of our code was written in tandem, working to develop a puck with a strong sense of hierarchical control.

I continued to use this concept in building my final project for the semester, known as "Puck Wars." This project was partially inspired by Michael Forrest's project: simulating a game of Quidditch, and also by the description of the final for a prior iteration of the class: using AI concepts to design players for a game of Capture the Flag. Taking these concepts, and the feature of shooting pucks --which was introduced in a recent update to the environment-- I set about creating a battle simulation.

My initial goal was designing agents that had a priority in addition to their own survival: to chase and attack the agents they identified as enemies. A great deal of my work with the puck agents has centered around rotation, as this is the main way to control a puck's movement. While acceleration is also tunable, I found the greatest success in designing intelligent agents centered around acceleration. I borrowed a great deal of code from multiple agents in the core pucks environment. One of my greatest sources of inspiration was the swarmer's code for following a group of pucks by averaging their position and setting its rotation to follow.

I co-opted this code, as I'd seen others do to avoid stones, and expanded it to evade torpedoes and other dead agents. This helped to keep the pucks alive by lessening the frequency with which the took collision damage. I also borrowed the "shooter" agent code to form the foundation for the "grunt" agents. These pucks are the main ingredient in the "Puck

Wars” simulations. I coded two other agents, the “gunner” and the “medic” pucks. For the “gunner” puck, I struggled to find a way to stabilize it against the recoil from firing torpedoes. In order to solve this problem, I chose to make it a stationary puck. However, this prevented it from being able to draw energy from vents. Instead, I implemented it through a “nursery” spanner; each time a gunner dies, the nursery spawns a new one soon afterward. While this is the implementation I hoped for, I believe it’s detrimental to optimization. The environment doesn’t unspawn dead “gunner” pucks the same way it does for agents with the “:mobile” key enabled.

The “medic” puck uses a great deal of the same hierarchical control as the “grunt.” However, instead of borrowing from the “shooter” agent, it draws inspiration from the “vent,” and gives energy to “grunt” and other “medic” pucks if it considers them allies. I also contemplated the feature of it granting energy to “gunner” pucks, but this seemed pointless, seeing as they remain constantly present, owing to the nursery’s functionality. Therefore, the medics only focus on mobile agents.

It proved challenging to spawn a large amount of mobile pucks in a world. One of my fixes was to assign each “medic” and “grunt” puck to a squad. This squad assignment can be seen in the body color of the agents, especially in the latter worlds where there are more squads. The concept is for pucks to only warm with their designated squad. I see room for further implementation of this squad based feature, chiefly in the creation of an “officer” puck.

The concept for this “officer” puck is roughly equivalent to a hive mind. The “officer” would issue orders to the other pucks in the squad, using “transfer” proposals it would dictate the state of the pucks in its squad. The pucks would prioritize avoiding obstacles, attacking enemies, or gathering energy based on what the “officer” proposed. The version of “Puck Wars” in my portfolio lacks this functionality, but there is some observable implementation the squads. The ability to assign different numbers of pucks to a squad allows for a variety of different simulations.

Another way in which I did explored an array of possibilities was hand building multiple worlds, six of them in total. The first was small and fenced in by a number of stones. In the second design, I took the same layout and expanded to double the size. This allowed for a great deal more maneuverability among mobile agents. From here, I built an even more unrestricting map. My next two iterations of worlds have no obstacles apart from a series of vents on each side, where the mobile pucks are initially created. This layout of vents promotes agent survival while not majorely inhibiting movement. It is in these worlds that I

saw the most interesting mobile agent behavior. For my final world, I set more of an asymmetrical conflict in motion. I spawned a very large force of blue agents with no vents, “medics,” or “gunners.” On the opposing side, I created a much smaller force of red agents, but supported by “medics,” “gunners,” and “vents.” Interestingly enough, these mismatched forces seem to have a roughly even chance of eliminating the opposition.

If I were to expand upon this project, I’d want to greatly improve the pucks navigational skills, spatial reasoning, and tactical movement. However, for an initial test with an open-ended environment, I’m quite satisfied with the result. Overall, “pucks” and by extension this course have proved an interesting initial foray into Artificial Intelligence.