Christopher Bradford

Scott Kannawin

Project 4

Throughout the semester we have been developing an AI ship to traverse through Space Settlers, this time the game took a turn making it a team based game, capture the flag. With that we had to implement a multi agent cooperative system between our ships. To accomplish such goals we had to split responsibilities between the ships, focusing on an offensive strategy since we were in the competitive ladder. We started the game with 3 ships and aimed for ending with 6. The ships were given the following tasks

* The first ship and second ship go for the flag
* The third and fourth ships will gather resources
* The fifth ship will defend our flag, gathering resources otherwise
* The sixth ship will hunt down and harass enemy ships

We decided to have our first set of ships have duplicate functions in order to ensure that we accomplished our main objectives as quickly as possible- that is, capturing flags and gathering resources to capture flags quicker. However, we did take measures to ensure that the ships ended up not chasing identical goals. With the flag runners, we had both race for the flag. Whichever one reached it first would then go to deposit it, while the other would stay near both enemy alcoves in order to retake the flag as quickly as possible once it was deposited and respawned. Similarly, we kept track of what each resource gathering ship was going after, ensuring that no two ships were trying to mine the same asteroid. The ship focused on defending the flag prioritized enemy ships that had the flag, then prioritized enemy ships and bases on our half of the map. If it still had no targets, it would start gathering resources following the same behavior as ships 3 and 4. Ship 6 had the simplest code, where it would simply try to find the nearest enemy ship or base and destroy it.

For planning, we decided that the most effective route would be to plan our purchases and the asteroids needed to get them. Once we decided which purchase would be our goal, we used a iterative deepening depth first search through a graph made of the mineable asteroids, searching for a path that would result in having enough of each resource for the purchase. We replanned every frame, since this form of planning was low-impact and we wanted to be sure we were as responsive as possible.

(Note: The planning described above is the planning implemented in the code submitted for the on-time competitive ladder. We also implemented planning more comprehensively in our late submission. In that, planning controlled all of the ships behaviors, from gathering resources to chasing the flag. That version used the following PDDL-like functions:

* buyShip()
* buyBase()
* mineAsteroid(ship, asteroid)
* pickupFlag(ship)
* returnToBase(ship)
* standby(ship)

with a variety of constraints such as whether ship had a flag, the number of ships, and more. Unfortunately, the tree was large enough to cause timeout issues, even when we limited ourselves to 4 ships, and scaling it back resulted in suboptimal behavior that had a very low rate of capturing flags.)

While ships are being built, it is also determined that if a base can be built that only the flag carrying ship will be able to buy said base. This is done because the flag carrier is already traveling the optimal path to the enemy’s flag, and when a base gets dropped in its path it will inherently shorten the path and optimize point gain. The key to our strategy was to defend our own flag while giving our flag carrier ship the shortest path it could take to get to the flag.

In the early stages of the project we had put just one ship up on the ladder to see how it did by just going for the flag, as it turns out that little ship singlehandedly won the ladder most days. Because of those results found on the early ladders, it was then determined that the flag carrier did not need any additional defense in place, as the potential ship defending it could be put to better use elsewhere.

To implement our multi-agent system, we developed an Actions class to generate targets to move to for each of our 4 behaviors: flag capturing, resource gathering, defending, and harassing. Whenever a ship was created, we decided which behavior it would carry out for the rest of the game. This was to ensure that there would be no loss of efficiency from changing tasks suddenly, possibly abandoning a near-dead enemy ship, forgetting to deposit a large amount of resources, or simply wasting fuel by turning around unnecessarily. Additionally, since we made sure to always have a ship near the enemy flag alcoves, a large benefit of dynamically switching behaviors (grabbing the flag as quickly as possible) wouldn’t be as relevant. The only exception to this system was if a ship accidentally picked up the flag. In that case, they would use the flag capturing behavior, regardless of what behavior they normally had. This made sure the flag was deposited as quickly as possible, instead of being dragged around the map while a ship hunted its enemies while carrying a flag it accidentally picked up.

There were many more things we implemented in this project from previous findings as well. The traversal method we used was a the same A\* mechanics as before, with a small twist. When we implemented A\* into Space Settlers originally, we went with a roadmap method that had each mineable asteroid or beacon as a node on our dynamic traversal graph. In capture the flag, that no longer worked, because there are much fewer mineable asteroids/beacons to obtain at a time. Instead, we made a graph with 64 hand chosen nodes spread around the map to best facilitate pathing finding.

Along with our traversal method, our genetic algorithm from Project 3 found an optimal max velocity to travel. Using that max velocity allowed us to minimize collisions with asteroids because of momentum shifts, while still traveling as fast as possible. This was especially useful while collecting the flag. Finally, since we had chosen the competitive ladder the entire semester, we had used much of the same logic through the year to determine who and when to shoot. When we had created those algorithms it had worked out very well for us and allowed us to win many nights in the ladder.