Task: Task 19

Title: Spike Extension Report

Author: Sam Huffer, 101633177

Instructions

Author's Note

I do not recall it being made clear to me that the extensions were to be discussed in separate reports to the base work of a spike. Whether this just wasn't mentioned until a week or two or it was and I forgot, where I have done extensions to a task, I have discussed them in the same report as the base work for the task, in some cases alternating between adding a base feature and discussing it, then an extension feature, then back to a base feature, such that they're entangled. Due to this, I'll be pretty much referring back to those spike reports and summarising which extensions I did and how I went about them. For more detail on what I did for each task's extensions, please look at the relevant spike report.

Task 4: Goal-Oriented Behaviours and SGI

The extensions available for Task 4 were:

- Create an object-oriented version of the code and discuss the pros and cons.
- Create a console-based turn-based RPG.
- Complete the GOAP spike that looks at using a plan to overcome the limits of simple goal insistence.

For this task, I limited myself to converting the code to an object-oriented version of the code. I did a diagram of each class created for this, and briefly discussed the advantages of code being divided into objects that can be passed and manipulated, of inheritance and polymorphism, and the tiresome drawback of referring to methods or field within that class by self.X.

Task 7: Tactical Analysis with PlanetWars

The extensions available for Task 7 were:

- Consider what extra information could be analysed and exploited in decision making.
- Explore the implications of and ways to exploit the fog of war view of the simulation environment.
- Discuss what asymmetrical game maps create in terms of game bias, game balance, and tactical opportunity.

For this task, I discussed (but didn't implement any code for) each point raised. For the additional possible information, I noted that the number of ships in a fleet and which planets are or aren't visible to the enemy due to the fog of war would be possible pieces of information to analyse.

For the implications of the fog of war, I noted the possibility of building up an attack force just outside the opponent's awareness and then attacking.

On the question of asymmetrical maps, I noted the disparity in advantage afforded to each player on an asymmetrical map, and argued this was shown with one of the maps I collected data on in this task.

Task 10: Tactical Steering

The extensions available for Task 10 were:

- Agents avoiding obstacles.
- Multiple hunters, and prey agents intelligently pick which hiding spot associated with an obstacle to use.
- Prey agents consider more than proximity to a hiding spot when picking a hiding spot, also accounting for its proximity to the hunters and if travelling to a spot will result in the prey crossing a hunter's line of sight.
- Hunters chasing prey they can see, and eating them on contact, including the ability to respawn prey.

I attempted each of the extensions presented. For the obstacle avoidance extension, I couldn't quite get feeler-lines-based object avoidance to work, so I adapted the idea and used the agent's position and a point in front of them to check if something was within an avoidance radius, then calculate the force needed to avoid the obstacle if something was detected to be too close.

For the additional considerations when picking hiding spots for the prey, had each hunter generate one hiding spot per obstacle, which were all ranked based on both distance between prey and hiding spot and distance between hiding spot and hunter. If travelling to a hiding spot would mean crossing a hunter's line of sight, that hiding spot went to the bottom of the rankings.

For the hunters chasing visible prey and eating them on contact, I implemented the necessary changes to have hunters pursue prey agents if they crossed their line of sight, destroying them if the distance between them was small enough, and spawning more prey with the press of a key.

Task 11: Emergent Group Behaviour

The extensions available for Task 11 were:

- What happens when agents can't overlap?
- Add a predator agent for all other agents to avoid.
- Add walls for agents to avoid. What is necessary to get the pool of agents circling?
- Create different types of agents and see what behaviour emerges between groups.

I attempted each extension except for the multiple types of agent groups; I figured that would be somewhat time consuming, and I had other homework that needed attending to.

From the previous task, I retained one of the hunters, having the program generate it automatically, and made sure that the prey agents all fled it if it was within their flee range.

I implemented the walls, and got the agents to avoid them, as well as not spawn outside in the padding between the walls and the edge of the screen. However I didn't manage to get the movement values to a point where the prey agents were circling around the simulation space.

For the separation of agents, I adapted the obstacle-avoidance code instead of using the suggested logic from the lecture slides, as it proved more effective at separating agents from each other. Reducing the separation multiplier let prey agents move over each other, but when it was high

and the predator came past, the prey agents crammed as tightly as their separation would allow into a corner and shook in place until it left.

Task 14: Agent Marksmanship

The extensions available for Task 14 were:

- Include rate of fire and effective range in the decision-making process, i.e. make a shotgun.
- Make explosive projectiles.
- Make the target dodge slow projectiles or take cover.

I attempted each of the extensions for this task. With the rate of fire and effective range, I implemented that not only with the shotgun I created for the extension (which fired five pellets randomly offset from where it was trying to shoot), but with each other weapon type already implemented for the base requirements of the task. (The rifle's effective range encompasses the entire simulation space, so though it mightn't look like it's not restricted by an effective range, it does have one in effect.

I tweaked the rocket and hand grenade weapon types to explode on contact with an obstacle, agent, or (for the grenade) a target signifying it landing on the ground. For both explosives, I had them deal damage to agents over the duration of the explosion, rather than all in one hit as with the hand gun, rifle, and shotgun.

For the target dodging slow projectiles, I adapted the obstacle-avoidance code into a method for avoiding projectiles, making it move perpendicular to the vector it would take if moving towards the projectile. If the obstacles are enabled, the target agent will also seek a hiding spot to flee the shooter.

Task 15: Soldier on Patrol

The extensions available for Task 15 were:

- More high-level modes
- More attacking modes
- More agents
- Different regions of the map (recharge, hiding location, etc.)

I did not extend this task beyond the base requirements as part of this task, owing to needing to attend to responsibilities to my capstone project and other homework. However, looking at it now, in Task 16: Goal-Oriented Action Planning, I did implement high-level modes for the soldier: getting food when hungry, getting ammo when out of ammo, and resuming an attack on the target if it spots it on the way to either of those resource stations; and different regions of the map: a location for swapping empty weapons for full weapons, and a location for eating food.

Task 16: Goal-Oriented Action Planning

For this task, we were required to think of our own extensions. I chose not to do further extensions for this task due to time constraints, the same as with Task 15, but while doing this task I did end up implementing some of the extensions for Task 15 by accident (see above).

Task 18: Navigation with Graphs

We were required to think of our own extensions for this task as well. I chose to get the soldier's weapons shooting the fugitives within the soldier's awareness and its weapon's range, with the prediction calculations being based on the target fugitive's path and where the soldier predicts the fugitive will be on that path at a given time. I decided to forgo additional extensions for this task, as they would fall within what I have in mind for my custom project anyway, this task needs to be submitted, and I have an assessment for another unit to attend to as well.