2310 ASSIGNMENT 1

KIRAT ALREJA - u7119530

Appendix

1. Problems of the Assignment

Assignment "Gliding in Space" has two major problems that need to be solved: Sharing Globe Location & Charging Methods. Here is a brief overview of my solutions for Stage 1,2 & 3 -

1.1 Sharing Globe Location

As ships move through the orbit, they lose charge. The only way to stay alive is recharging through an energy globe. Thus, it is essential that all ships always have the location of this globe. I used a central coordinator to achieve this for Stage 1, and a fully distributed system for Stage 2 & 3. While the central coordinator was simple, I faced a lot of challenges while implementing the fully distributed system.

The most prominent one was ensuring that the ships can differentiate between outdated & latest messages, so that they can pick the right globe location. After the ships find the location, they should forward it to as many ships as possible. For Stage 3, an additional challenge was dealing with multiple globes, and the uncertainty of their existence. I attempted to solve this by sending the ships to the closest possible globe for recharging.

1.2 Charging Methods

I tried a lot of techniques for charging the ships, and I have explained them in detail throughout the report. However, my three final charging solutions for Stage 1, 2 & 3 respectively are:

1. Grouping Ships – This solution is based on dividing the ships into 4 groups, on positive & negative x-y axis. Each ship gets a unique point in the group, thus forming a uniform structure around the globe. They recharge and immediately return back to their points.
2. Circular Motion – Each ship moves in a circular motion relative to its angle to the globe. Similar to the previous solution, the ships form a uniform structure around the globe. This ensures that they are in constant proximity to the globe and can charge in a coordinated way.
3. Critical Distance – If a ship moves further than the critical distance from the globe, it must turn back to the globe. In case it is within the critical distance and has enough charge, it swarms until it needs to charge. Here, the ships stay in controlled proximity to the globe.

2. Individual Solutions to Stages

The code solution to three stages has been placed in the following branches of the git repo

Stage 1 – "Stage_1_Branch"
Stage 2 – "Stage_2_Branch"
Stage 3 – "Stage_3_Branch"

The main branch has the Stage 3 solution. Note - My Stage 2 solution can hold ships much better than the Stage 3 solution, for reasons explained ahead.

2.1 Stage 1 (Branch on the repo – "Stage_1_Branch")

2.1.1 Sharing Globe Location

Since this stage allowed a central coordinator to be introduced, I started by building an entity that could be shared globally. I implemented the globe location as a shared variable and allowed all the ships to access or update it. As soon as a ship is near the globe, it updates the globe location for everyone else. The advantage to such an implementation is that the shared information is never "outdated" and the entire swarm of ships will always have the correct globe location.

2.1.2 Charging Method

Failed Design

To build the charging method for this stage, I started by looking for a critical charge level – below which a ship would need to travel to the globe & recharge. Since the central coordinator ensures that the globe location is up to date for all ships, I assumed half-charge could work as the critical charge level.

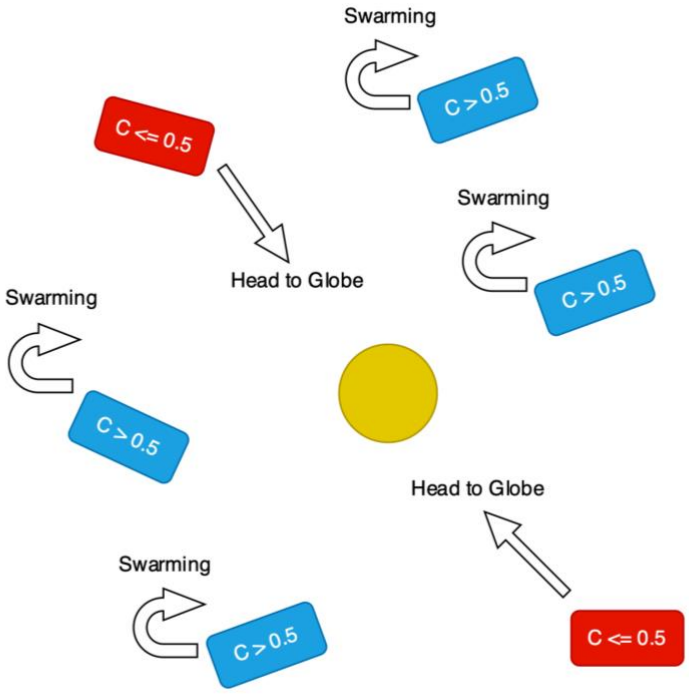

Figure 1 – Simple Critical Charge Method (Rejected)

First, I used a simple charging method (Figure 1) where a ship below the critical charge level would travel to the globe at full throttle. On the other hand, it would switch to the default swarming behavior when above the critical charge level. This method did not work well and led to ships dying quickly. The ships would often travel

far away from the globe when swarming and run out of charge as they throttle to it after reaching the critical charge level.

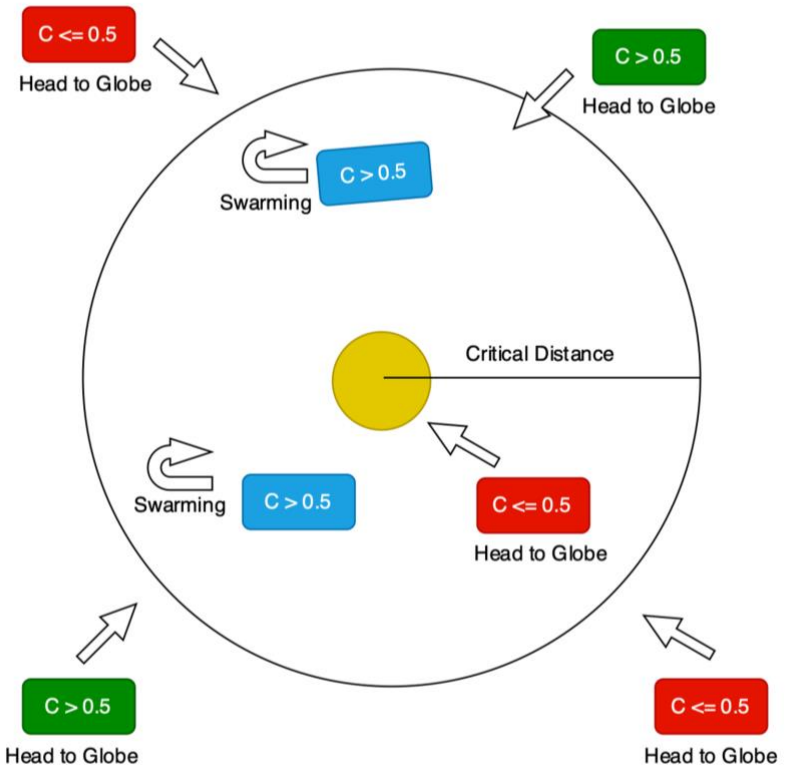

Figure 2 – Critical Ship-To-Globe Distance (Rejected)

This led me to think that the ships need to stay within a certain distance to the globe so that the charging algorithm can ensure they wouldn't run out of charge when travelling. Like the critical charge level, I added a critical ship-to-globe distance level (Figure 2) – surpassing which a swarming ship would travel back towards the globe. This ensured all the ships remained close to the globe. However, the ships were still dying, this time not due to the distance – but the globe being unreachable as multiple ships approached it simultaneously.

Final Design

I took inspiration from the "Cross" example behavior provided with the specification to solve this issue. I decided to divide the swarm into four equal groups, where each ship has a specific point assigned to it, relative to the globe's location. I used the central coordinator to calculate the maximum number of ships in the swarm. The number of ships is shared globally, like the globe location. It works on a simple algorithm where if a ship's vehicle number is higher than the one stored in the shared variable, it would be updated to the higher number.

Assuming the globe's location is an arbitrary point `(x,y,z)` – the groups are calculated as follows

```
Group 1: (x + critical_distance + vehicle_number/1000, y , z)
Group 2: (x, y + critical_distance + vehicle_number/1000, z)
Group 3: (x, y – critical_distance + vehicle_number/1000, z)
Group 4: (x – critical_distance + vehicle_number/1000, y, z)
```

Group 1 and Group 4 send the ships along the globe's positive & negative x-axis, respectively, while Group 2 and Group 3 do the same for the y-axis (Figure 3). To ensure that every ship has a unique location, I added a fraction of its vehicle number to the critical distance. I designed the groups this way so that not only do all the ships have a unique location, but they also uniformly surround the globe.

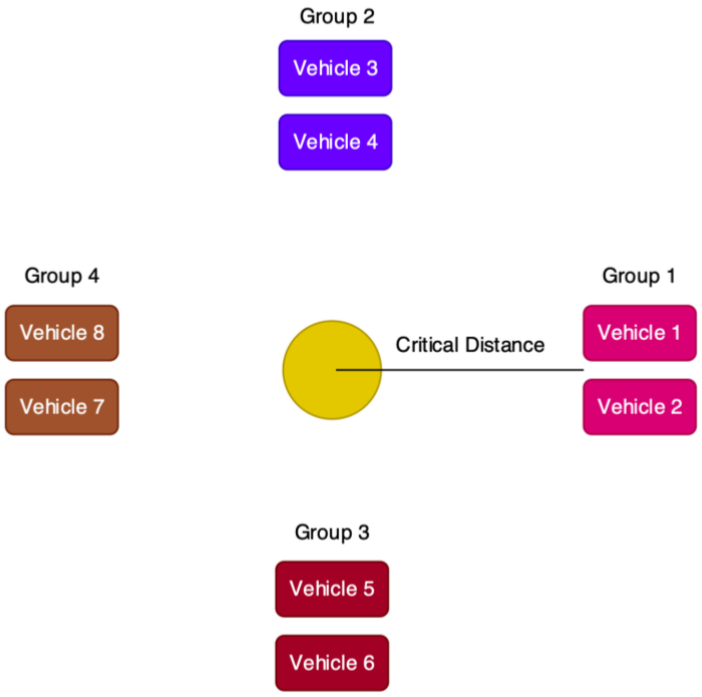

Figure 3 - Grouping Method (Final Stage 1)

When a ship's charge level is lower than the critical charge level, it travels to the globe at full throttle (Figure 4). Otherwise, it would travel to its group location at a third of the throttle. (Figure 5)

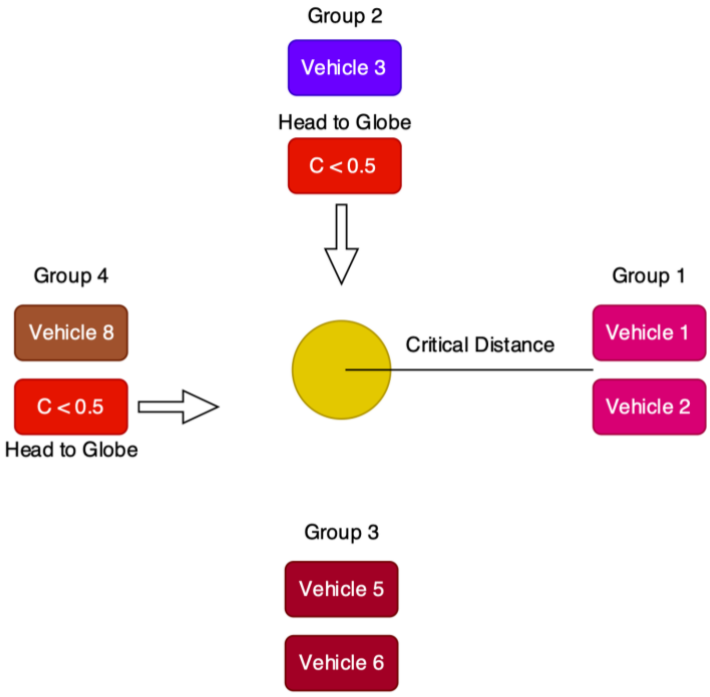

Figure 4 - Grouping Method (Final Stage 1)

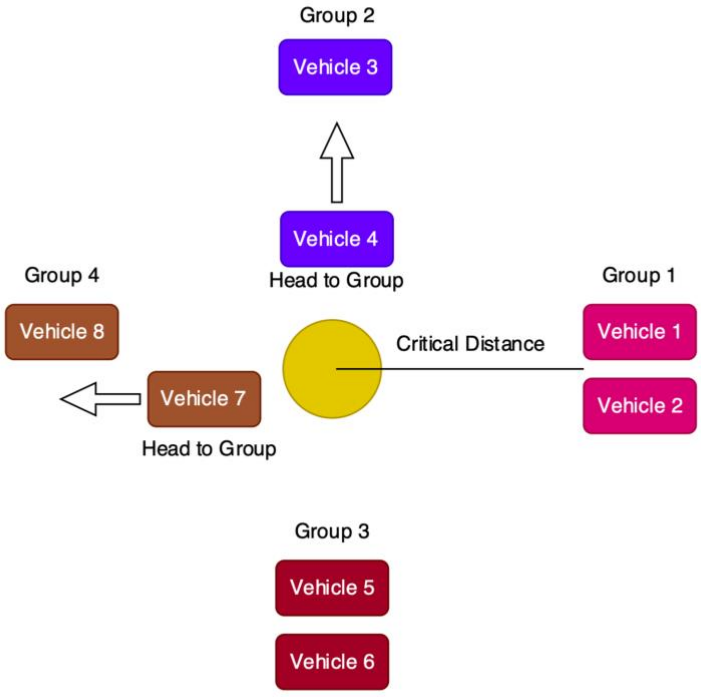

Figure 5 - Grouping Method (Final Stage 1)

Using this charging method, I was able to tackle the distance problem and the globe being unreachable. The ships don't swarm and are always controlled by the algorithm, which reduces uncertainty related to charge & distance levels. In some cases, multiple ships would still try to reach the globe simultaneously, but for the most part – they charge in a coordinated way.

2.1.3 Testing Stage 1

Initial No of Ships	Ships after 15 minutes
30	27
60	60
90	90
150	150
200	199

My final Stage 1 design (Grouping Method) can hold up to 199 ships out of 200, for around 15 minutes. I ran these tests 5 times and took the average number of ships. For reasons I could not figure out, many ships seem to survive perfectly (90, 150) – but in a small swarm of 30 ships, on average 3 seem to die.

2.2 Stage 2 (Branch on the repo – "Stage_2_Branch")

Although a central coordinator solution allows the ships to survive and charge efficiently, shared variables are not ideal for a real-world scenario. Therefore, for Stage 2, I worked on a fully distributed solution – where ships can communicate via local message passing only.

2.2.1 Sharing Globe Location

Failed Design

I started by only passing the globe location through messages. Whenever a ship finds a globe, it sends a message containing the globe location to other ships in range. Ships near the globe can only get the location if they receive it through a message. So, I kept these ships waiting for a message, and when received - they would further send it to all ships in their range (Figure 6). This ensured that when the globe is found, the location reaches as many ships as possible.

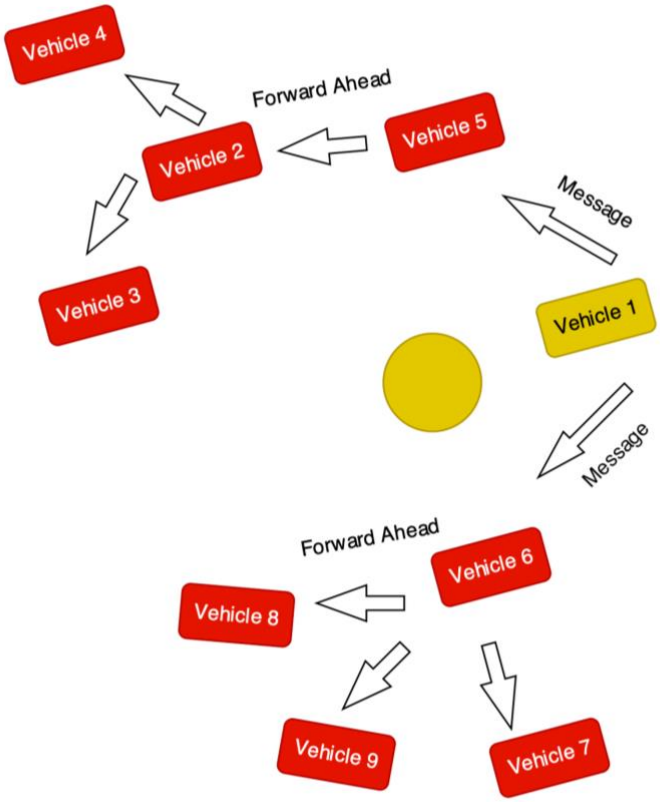

Figure 6 - Message Passing

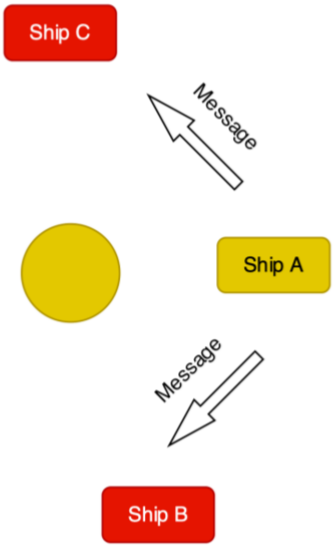

Figure 7 - Outdated Message Issue Example

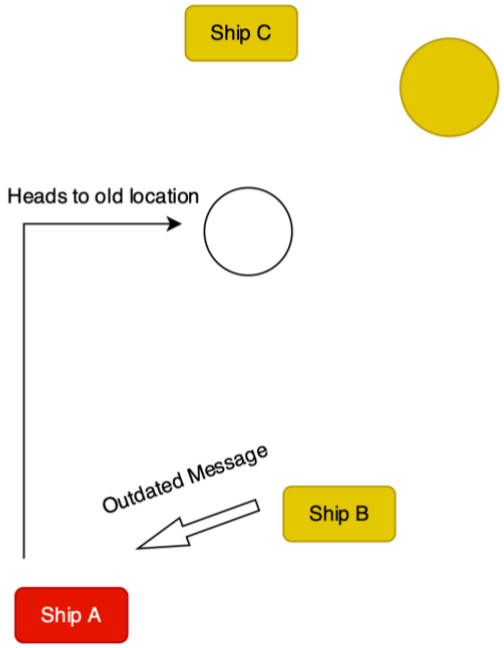

Figure 8 - Outdated Message Issue Example

However, I discovered that this method fails miserably since messages can be outdated. For example, ship A finds the globe and passes it to ship B & ship C (Figure 7). Now, if ship A is no longer near the globe, it would receive messages from nearby ships. Ship B & ship C might no longer have the latest globe location, and would pass the old location to ship A. This leads to ships heading for the wrong locations and dying (Figure 8). Even if a ship is sending a message with the latest globe location, there is no way to compare the outdated & latest locations to choose between them.

Final Design

To judge if the received message is outdated or not, I decided to pass the local time along with the globe location. When a ship receives a message, it calculates the difference between its local time and the received time. Through trial and error, I discovered that a time difference of 0.5 acts as a critical time to judge if the message is outdated or not. Thus, if the difference calculated is less than the critical time, the ship would use the globe location instantly (Figure 9) – as it is ensured to be the latest.

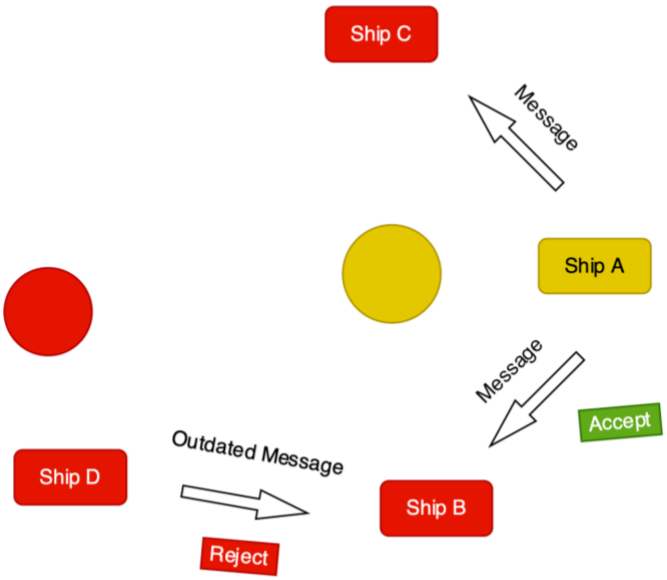

Figure 9 - Accept only latest messages

On the other hand, if the time difference is greater than the critical time, the ship would keep receiving messages until it gets the latest message. If all the received messages are outdated, the ship will use the first message as received. However, I did not encounter this case throughout my testing. When the receiving ship finally gets the latest globe location, it passes it further to ships in range. Using this algorithm, I solved the outdated location problem and selected the ideal location amongst multiple messages.

2.2.2 Charging Method

Failed Design

For charging the fully distributed swarm, I tried to extend the grouping algorithm from Stage 1. However, since there was no shared variable, I had to pass ship vehicle numbers via messages. All ships have their local maximum number of ships, updated according to the received vehicle numbers. If the received vehicle number is greater than the local ship's vehicle number, it will update the maximum number of ships to the received number.

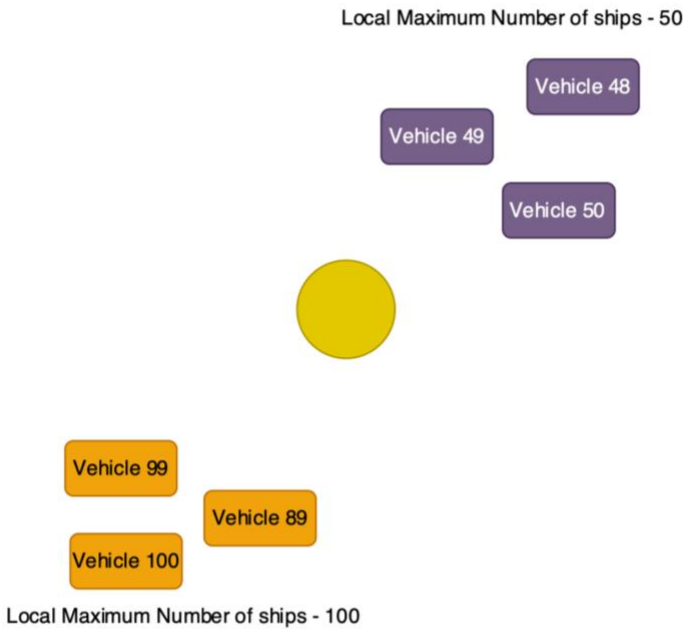

Figure 10 - Why Grouping Method Fails in Stage 2

However, I found that all ships had a different local maximum number of ships—for example, ship A with vehicle number 100 and ship B with vehicle number 50. There is no assurance that ship B would ever receive a message from ship A or any ship with a vehicle number greater than 50. As a result, ship B would assume that there are only 50 ships in the swarm (Figure 10). Thus, there is a lot of uncertainty with calculating the maximum number of ships via message passing, which breaks the foundation of the grouping algorithm. The ships kept moving to the wrong groups and were not spread around the globe effectively.

Final Design

The ships must stay in a specific formation around the globe for the message passing algorithm to build an effective "network." If they swarm, there would be uncertainty if all ships would receive the globe location in time or not. I took inspiration from the example videos given with the spec and decided to implement each ship moving on a circular path relative to the globe.

For every individual ship, I used the angle to the globe to calculate the new point for the circular path (Figure 11). Let the globe's location be an arbitrary point `(x,y,z)`, then the new point would be calculated as – `(x + radius * cos (angle_to_globe), y + radius * sin (angle_to_globe), z).` Using trial & error, I found that a radius value of 0.05 keeps the ships close enough to pass messages effectively. For Stage 1, I used a critical charge value of 0.5.

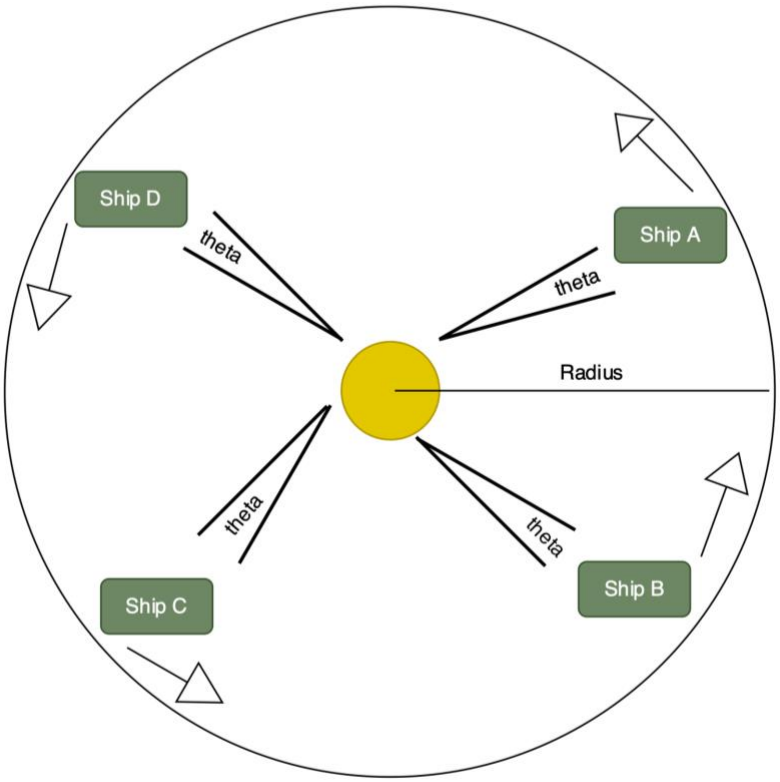

Figure 11 - Circular Motion (Stage 2 Final)

However, in this stage, I discovered that the critical charge value needs to be higher. If a ship has an outdated globe location, it can still receive the new location on the way. So, the critical charge value needs to be higher to account for the cases where a ship might need to move to a new location – that might be further away. Thus, I decided to use a critical charge value of 0.7. Using this algorithm, I was able to keep the ships in a structured formation around the globe – to charge and share information as needed.

2.2.3 Testing Stage 2

Initial No of Ships	Ships after 15 minutes
30	30
60	60
90	83
150	136
200	139

According to an average of 5 tests, my final Stage 2 solution can hold up to 139 ships out of 200, for around 15 minutes. The circular motion for charging seems to be very effective for around 150 ships but degrades drastically above that. I noticed that as the number of ships increased, there were cases when the globe became unreachable, due to which the ships could not charge.

A possible fix for the same could be dynamically increasing the radius, depending on the number of ships - to reduce clutter at the globe. However, since my approach to calculating the number of ships did not work (as explained above), I could not implement this.

2.3 Stage 3 (Branch on the repo – "Stage_3_Branch")

In this stage, multiple globes in orbit can disappear and reappear at any time. Also, as given, two globes would always exist.

2.3.1 Sharing Globe Location

As a ship can be near multiple globes, I decided to calculate the distance of each of these globes to the ship. Then, locally, the ship uses its closest globe location. However, the local closest globe might not be the same for the other ships in range. To tackle this issue, I decided to pass every globe as a separate message (Figure 12).

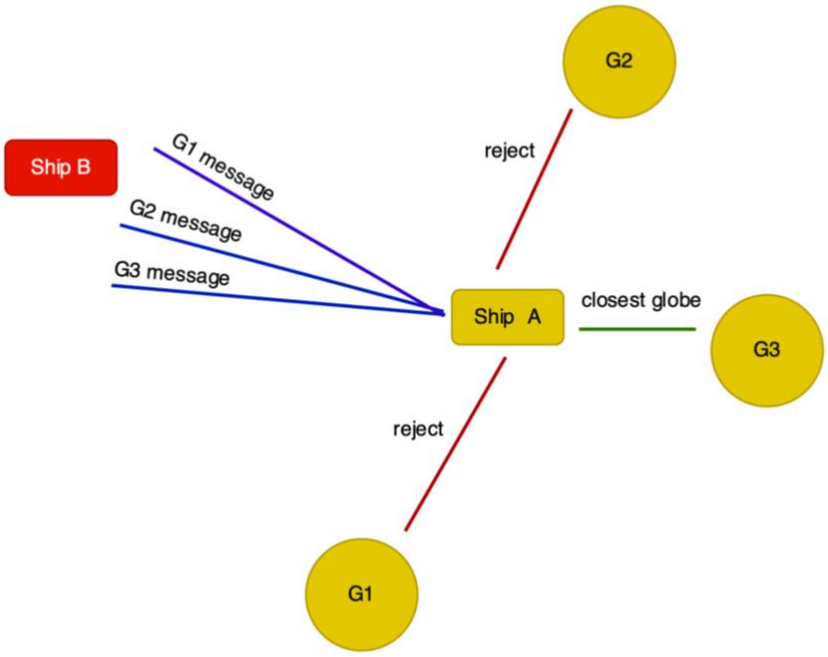

Figure 12 - Multiple Globes Message Passing

In stage 2, the receiving ship chose the most recent message among multiple messages. For this stage, ideally, the ship should pick a message considering the time

and the globe distance. I tried to implement a heuristic that takes both these values and ranks all the received messages accordingly.

For example, message A has a globe distance of 0.3 and a time difference of 0.2, while message B has a globe distance of 0.1 and a time difference of 0.4. Even though the message A globe is further away, it is more likely to be the latest globe when compared to message B globe. If the ship chooses message B globe based only on the distance, the globe's location may be outdated by then. Therefore, a heuristic is required to select the ideal message.

Due to the lack of assignment time, I couldn't finish this heuristic solution and decided to pick a globe similar to the Stage 2 solution.

2.3.2 Charging Method

Failed Design

The uncertainty that the globes can disappear and reappear poses a challenge for charging methods for Stage 1 & Stage 2. My solutions to both these stages were based on the foundation that the ships form a structure around the globe. When I used the Stage 2 solution here, the ships would create a structure around the globe as expected. However, when the globe disappeared, they would move to the next closest globe. For example, assume there is globe A and globe B.

The ships around globe A would start moving to globe B as soon as globe A disappeared. Now, when all the ships are sufficiently charged, they would reform the structure around globe B. On the other hand, if the globe B ships are charging, and the globe A ships move to globe B for charging simultaneously, the globe would be unreachable. Thus, the ships would end up dying.

Final Design

Since using a structure didn't seem like a good idea, I decided to try a simple critical distance-based algorithm, one of my rejected designs for Stage 1. Each ship is constrained with a critical globe distance, going further than what would send the ship back towards the globe. If the ship is sufficiently charged and within the critical distance, it swarms – to avoid cluttering the globe (Figure 13).

On the other hand, if the ship is sufficiently charged and outside the critical distance, it would head to the globe at 0.8 throttle instead of full throttle. Whenever the charge goes below the critical charge, the ship heads to the globe at full throttle, regardless of distance constraints. Like Stage 2, I decided to use a critical charge value of 0.7 instead of 0.5 to account for the uncertainty of globes.

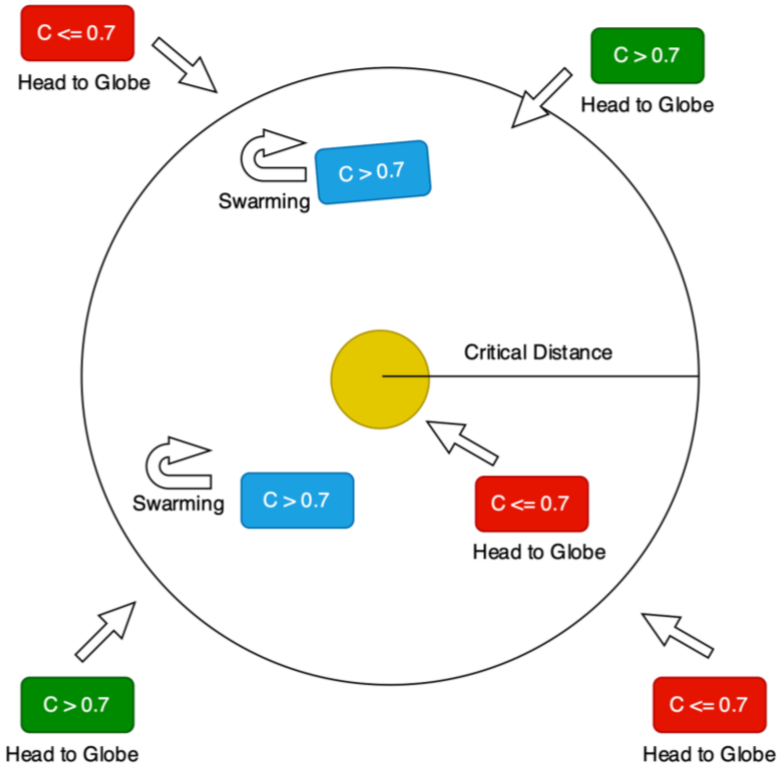

Figure 13 - Multiple Globes Charging

Since the ships swarm instead of staying at a relative globe distance, they are not dependent on a single globe. Instead, the closest globe would keep changing with every physics update since they move randomly. When a globe disappears, the ships would not clutter at another globe but spread out to all other globes in range. While it is still possible that all swarming ships head for a single globe in some instances, there are better chances of them surviving here when compared to a structure-based solution.

2.3.3 Testing Stage 3

Initial No of Ships	Ships after 15 minutes
30	30
60	44
90	63
150	61
200	65

As expected, this solution does not perform well compared to the Stage 2 & Stage 1 solutions. The message receiving ships do not head for the ideal globe, but the most recent one – even if it's the farthest away. Out of 200 ships, only 65 survived after 15 minutes. Surprisingly, the number is quite similar to when tested with 90 ships. To improve this solution, the heuristic message method can be implemented (as discussed above).