Assignment 1 Report

Haoran Song
U6688461
COMP2310

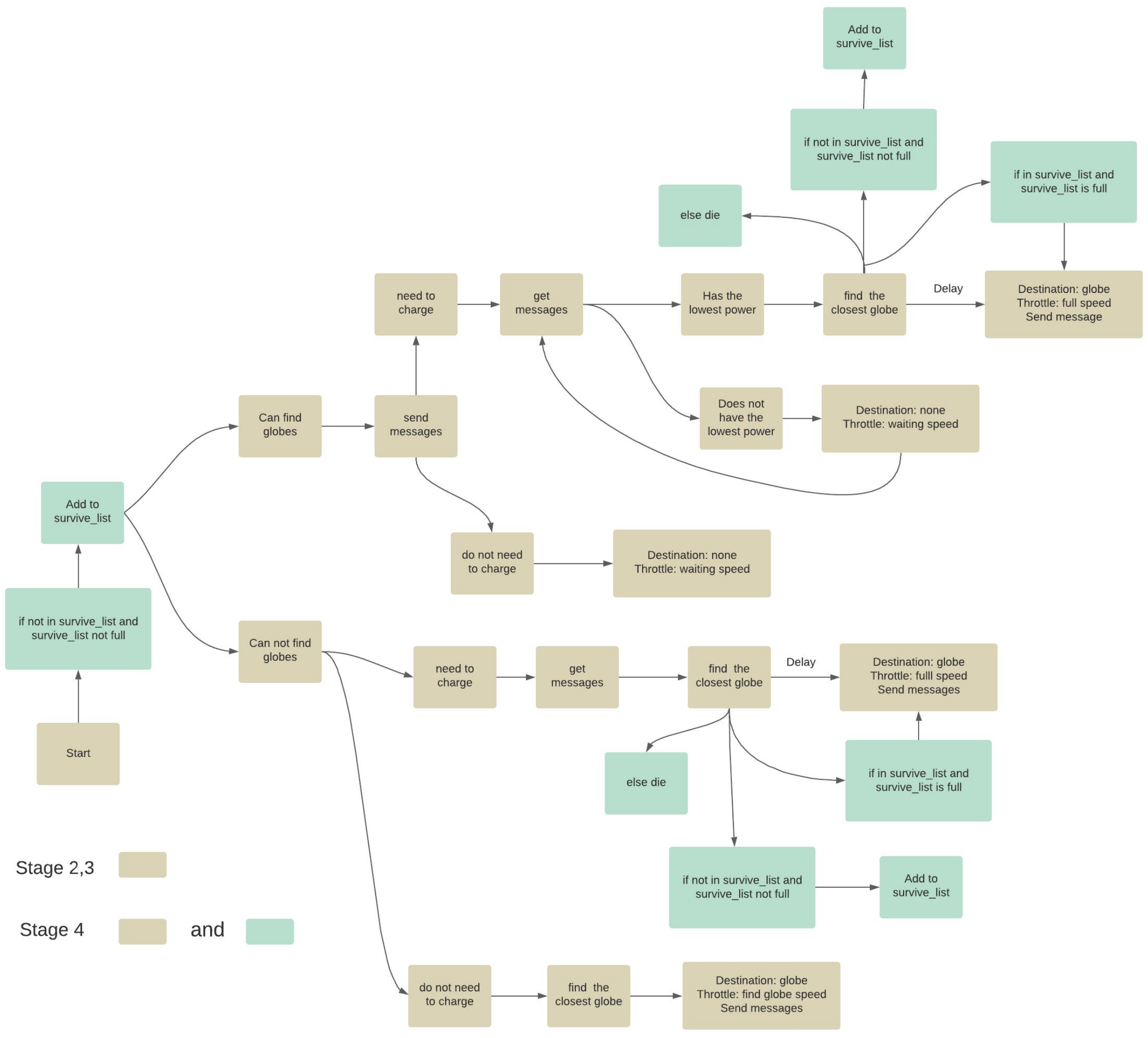
Content

0. Statement

The code for stage2 and stage3 was uploaded to branch stage2,3 and the code for stage4 was uploaded to main branch.

1. Major Design

(0) Logic diagram



Add to survive_list
if not in survive_list and survive_list not full
if in survive_list and survive_list is full
else die
need to charge
get messages
Has the lowest power
find the closest globe
Delay
Destination: globe Throttle: full speed Send message
Does not have the lowest power
Destination: none Throttle: waiting speed
Can find globes
send messages
Add to survive_list
do not need to charge
Destination: none Throttle: waiting speed
if not in survive_list and survive_list not full
Can not find globes
need to charge
get messages
find the closest globe
Delay
Destination: globe Throttle: fulll speed Send messages
Start
else die
if in survive_list and survive_list is full
Stage 2,3
if not in survive_list and survive_list not full
Add to survive_list
Stage 4
and
do not need to charge
find the closest globe
Destination: globe Throttle: find globe speed Send messages

Image1. concept graph

(1) Update and transfer of messages

For each vehicle, I designed the following information, which can be passed among vehicles:

Name	Use
Globe	Record the discovered globes information
Globe_num	Record the number of globes found
Get_Time	Record the time when the information was last updated
V_charge	Record the power of the vehicle
V_id	Record the vehicle number
survive_list	Record the list of surviving vehicles

Table1. Messages table

(2) Basic ideas

First of all, we stipulate that when the power of the vehicle is less than 0.5, it will be regarded as needs to be charged. I divided the states of vehicles into four status, and I think we need to give priority to whether vehicles can find the globe because it represents whether the vehicle can survive without external information. And then, we consider whether it needs to be charged at the current moment.

Status 1: The globe can be found and vehicle does not need to be charged
This is the state we most want to see. We don't need to worry about if such vehicles will disappear because of no Current_Charge for the time being, they have enough power and they can find the globe by themselves. And in order to help others, they also need to send the location information of the latest globes they observe for others vehicles to reference.

State 2: The globe can be found and vehicle needs to be charged
This group of vehicles can find the globes themselves, but they need to consider in which order they should be charged. Power is the measurement standard, vehicles need to receive the latest information around them and compare the Current_Charge of surrounding vehicles with their own. The one with the lowest Current_Charge has the change to charge and set the throttle to full speed. Other vehicles need to wait for themselves to become the vehicles with the lowest power.

State 3: The globe cannot be found, but vehicle does not need to be charged
Since this kind of vehicles cannot discover globes by themselves, they need to update their own information through other vehicles' information, and also need to

do a message transmission to transmit the latest information they received. They have enough power temporarily, so we stipulate that they move in a slow speed to the direction of the globes, and try to find globes by themselves. We try to reduce the number of vehicles that cannot find the globe to avoid the sudden disappearance of vehicles making the chain of information broken. In other words, to avoid a situation where a part of the isolated group cannot find the globes and cannot receive the information.

State 4: The globe cannot be found and vehicle needs to be charged
This is the most urgent situation. They update their information through the information of other vehicles and send out the latest information. The inability to find the globe also means that the vehicles are far away from the globe, and the consumption on the road may cause them to disappear on the way to the globe. Therefore, we stipulate that such vehicles should start immediately to find the globe and set the throttle at full speed.

(3) Charging and anti-collision design

As the number of vehicles increases, multiple vehicles will arrive at the same globe at the same time waiting to be charged, which will cause the vehicles to collide and disappear. Therefore, we set a delay. Before setting the throttle to full speed and charging, the vehicle needs to wait for a short time. The delay time is determined by the vehicle's current power and the power consumption per unit time.

$$D\text{elay} = \frac{0.01 * \text{Current\_Charge}}{\text{Current\_Discharge\_Per\_Sec}}$$

Image2. Calculate delay

(4) Design for multiple globes

When there are multiple globes, the globe which the vehicle goes must be selective, because we need to avoid that the vehicle chooses the wrong destination and causes the vehicle to travel too long at full speed and run out of power. Therefore, before departure, vehicle compares the globes information in the latest information it has obtained, and chooses the nearest globe as the destination. Due to the concurrency between tasks, the process of calculating the distance is likely to be interrupted. In order to obtain instant and accurate distance information, I added a protected function to calculate the distance between the vehicle and the globe.

(5) Design for swarm shrinks

In the transmitted information, a new survive_list is added, and finally the vehicles in the survival_list can survive. Each vehicle tries to add itself to the survive_list, representing that it wants to survive, and sends this request. At the same time, all vehicles are also receiving the latest survive_list information. If the survive_list is not full, that is, it has not reached Target_No_of_Elements, which means that the vehicle still have a chance to survive. Then, the vehicle can add its own information to the survive_list if it isn't in the list. When the survive_list is full, the vehicle checks whether it exists in the survive_list. If it does not exist, it cannot go to the globes to charge and eventually die.

2. Testy Runs and Analyze

I have tested the program in many aspects: first, the basic function of the test program, that is, under normal circumstances, how many vehicles can survive in 10 minutes. Secondly, I tested the program's response to special situations, that is, reducing or increasing the number of vehicles during operation. Finally, the operation of swarm shrinks was tested.

(1) Basic running

Globe Number	Vehicles Number	Run Time	Survival Number
1	64	10 minutes	64
	128		128
	256		~215

Table2. Single globe basic test

Globe Number	Vehicles Number	Run Time	Survival Number
Random	64	10 minutes	64
	128		128
	256		~250

Table3. Random globes basic test

It can be seen from the above test data that the program runs well under normal conditions. When there is only one globe, the number of vehicles can be stabilized at around 210. When there are multiple globes, it can be stabilized at around 250. In addition, the disappearance and increase of globes did not affect the survival of vehicles.

The main reason for the disappearance of vehicles is that the vehicles are accumulated near the globe, some disappear because of no power, and some disappear because of collision with each other.

Possible solutions:

1) We can reduce the collision of vehicles near the globe by increasing the differentiation of delay to make vehicles reach the globe at as different times as possible. However, after testing, if the waiting time is extended to a large extent, the vehicles will accumulate in a range far away from the globe, which will still cause the collision of the vehicles; if the waiting time is increased slightly, it will not be very effective.

2) I tried to keep the vehicles away from the globe as soon as possible to avoid collisions, but I found that the charged vehicles were not the cause of the accident, but the accumulation of uncharged vehicles. So I tried to increase the standard that needs to be charged, that is, if the Current_charge is less than 0.4, it needs to be charged (originally 0.5). The same solution is not feasible, because it will cause more vehicles to disappear in the waiting time.

In summary, the program can run stably, but there are still collisions near the globes that need to be resolved. When the number of vehicles is large and the space near the globes is small, some collisions cannot be avoided. In subsequent research, I hope to try to classify the power of vehicles in more detail, so that the vehicles that need to be charged can move in a larger range near the globes, thereby avoiding collisions.

(2) Vehicle disappearance/appearance

In this test, we used "-" and "+" to increase and decrease the number of vehicles during operation. It can be concluded that the reduction or increment will not affect the normal running of the program.

(3) swarm shrinks

Vehicles Number	Target_No_of_Elements	Survive_list confirm time	swarm shrinks completes time
64	10	~30 secs	~5 mins 30 secs
128	42	~3 mins	~6 mins

Table4. Swarm shrinks test

The code runs well. And according to the test data, it can be seen that the greater the number of vehicles and the greater the number of Target_No_of_Elements, the greater the number of information transfers required, and the longer the time required to determine the survive_list. In order to ensure the consistency of the survive_list, that is, no multiple copies of the survive_list are allowed, we have to spend more time to make sure that all vehicles have obtained the latest survive_list, and judge whether they can survive based on it.

The program also has some problems: in order to ensure the consistency of survive_list, the program stipulates that when the survive_list in the local information of the task is full, it can be executed to determine whether the vehicle can survive. And this series of actions are not atomic, which means that they can be interrupted in the middle. That is to say, the local message of the task was originally full, but it can be overwritten by the latest message and become not full. It leads to the fact that basically all the information transmitted in the environment is full, that is, only when all vehicles agree on a certain survive_list, swarm shrinks can be started, which leads to too long information transmission time and the time that complete swarm shrinks is also very long.

Possible solution: write swarm shrinks operations as “atomic operations”.

3. Conclusion

In summary, this program can complete the stable operation of more than 200 vehicles under normal conditions, increase or decrease of vehicles will not affect the operation, and the program also complete the process of swarm shrinking. The logic of the program is relatively clear: according to different measures for vehicles delivered in different situations. In terms of safety, a delay is set to prevent the collision of vehicles, and a judgment condition is set to ensure the consistency of the survive_list. The disadvantage of this procedure is that there is still a situation where vehicles accumulate near the globe and complete swarm shrinks needs a long running time. Although some solutions have been proposed, the implementation has not yet been completed. I hope that I can try and complete it in the future.