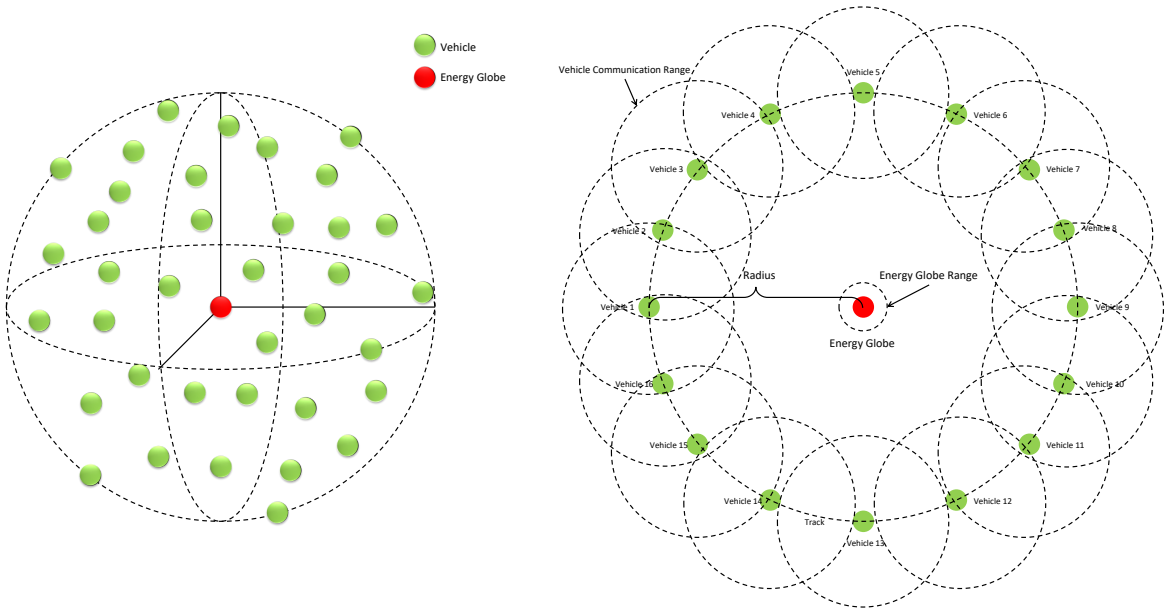
COMP 2310 Assignment 1 Report

Jiahao Zhang U6921098

Sept. 27, 2019

1 Introduction



(a) Ideal Sphere Model 3D

(b) Ideal Sphere Model 2D

Figure 1: Basic Model

The goal for this project is to make Vehicles live as long as many as possible. So in this report an ideal model is given to accomplish this, but within a range of number of Vehicles in a certain time duration. The range is [32, 256] to cover most situations and the running time is 5 minutes to show whether it is stable or not and not too long for saving time.

The ideal model for this report is a sphere model shown as Fig. 1a in 3D. The Energy Globe is located at the center of sphere. All the Vehicles are dynamic located on the spherical surface randomly and evenly.

To make the model more intuitive, the 2D model is used in this report like Fig. 1b. The model maintains a radius that ensures all Vehicles are able to communicate with each other (they must form a network). The radius must not be too small, because in that case there will be too many collisions around the Energy Globe so that no Vehicles can be charged. The radius must also not be too large, because in that case some Vehicles will lose signal and not be able to update the information of the Energy Globe from others.

When a Vehicle wants to get charged, It will go off from its track to the Energy Globe and when it is done, it will come back to the track.

There are two main design purposes for this ideal model:

1. All the Vehicles are positioning around the Energy Globe randomly evenly.
2. All the Vehicles are able to communicate with each other all the time.

2 Analysis & Implement

2.1 Stage A & B

2.1.1 Basic Program Structure

In the design of this project, all the operations are located inside the `Outer_task_loop` after the `Wait_For_Next_Physics_Update` . Because after each `Physics_Update`, the Vehicle need to do something to response to what have changed.

The main program is divided into 4 steps:

1. Check the Energy Globe information

   The positions and the velocities of Energy Globes are the most important information in the system because all the Vehicle can only get charged via them. At the beginning of each loop, the Vehicle detect whether there are some Energy Globe and record all the information.

2. Send message

   To form a network to connect the Vehicles in the space, all Vehicles must send and share all the information it has to others in each loop to keep the information as new as possible.

3. Receive message

   Same as the step 3, but when receiving a message from other Vehicles, the message must be processed first then store part of the message which this Vehicle thought is correct.

4. Set destination and throttle

   In the final step, all the information is updated as new as possible. The Vehicle will consider all the situations, and determine the destination and the throttle.

After these 4 steps, the Vehicle is able to either stay on the track on the surface of the sphere or go to the Energy Globe to get charged.

2.1.2 Message Structure & Central Control

1. Message Structure

   The most basic information are:

   - The information of the Energy Globe including position and the velocity.

     This is for step4 in Sec. 2.1.1 when the Vehicle goes to Energy Globe.

   - The timestamp of getting the Energy Globe information.

     This is for comparing two messages. Because we always want the information of Energy Globe to be as new as possible.

2. Central Control

   If there is a control center to manage and store the message, then this message is always the newest and the most correct. But this center must be mutual exclusive to all the message sender. One of the best advantages is that this message stored in the control center is also almost instant. Which means that if there exists a queuing mechanism, the collisions around the Energy Globe can be solved.

   For example, add a variable called `Available` to the message structure to denote the status about whether there is a Vehicle is charging or not, if a Vehicle want to charge, it needs to check the status of the `Available` to ensure that only one Vehicle can get charged at a time.

2.1.3 Energy Globe Position Estimation

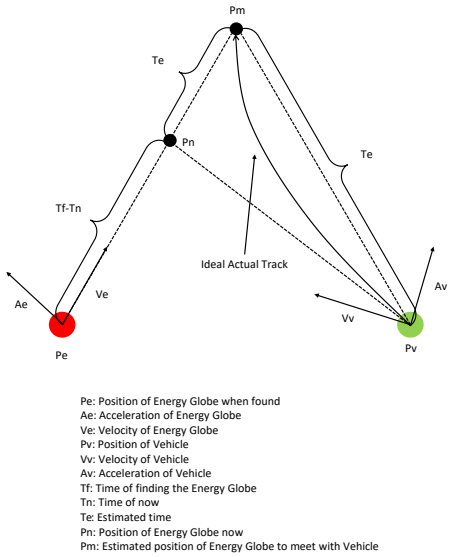


Figure 2: Energy Globe Position Estimation

When a Vehicle wants to know where to charge, we need to find the meet position P_m of the Vehicle and the Energy Globe.

As for the acceleration of the Energy Globe is not provided, we can only assume that it is small enough to be ignored. Ideal model is that the Energy Globe is doing uniform rectilinear motion, and the Vehicle which wants to get charged is doing uniformly accelerated rectilinear motion. And all the variables in this model are not changed in a tiny duration. So we get the equation below,

$$P_m = VeT_{total} + P_e$$

To obtain it, we need to calculate the total time T_{total} first,

$$T_{total} = T_f - T_n + T_e$$

T_f and T_n are given, so all we want is the T_e. To find the T_e, we can form an equation by that the vector from P_v to P_m is constant in the model,

$$P_e + (T_f - T_n + T_e)V_e = V_v T_e + \frac{1}{2}A_v T_e^2 + P_v$$

$$\Delta = (V_v - V_e)^2 - 2A_v[P_v - P_e + (T_n - T_f)V_e]$$

when $\Delta < 0$,

let $T_e = 0$

when $\Delta = 0$,

let $T_e = max(0, \left\| \frac{V_e - V_v}{A_v} \right\|)$

when $\Delta > 0$,

let $T_e = max(0, min(\left\| \frac{V_e - V_v - \Delta}{A_v} \right\|, \left\| \frac{V_e - V_v + \Delta}{A_v} \right\|))$

Because the direction of the Vehicle velocity and acceleration can be some values that makes the Vehicle not able to go to the Energy Globe (In equations, situations like $\Delta < 0$ or $T_e \leq 0$), so in those cases the T_e is set to 0 manually, and the track will be like the Ideal Actual Track in Fig. 2. That is because when moving to the Energy Globe, the Vehicle can get update about the information of Energy Globe. So, it can correct the track.

2.1.4 Charge Determination

We need to determine when a Vehicle needs to get charged. A solution based on current charge and estimate consumption is given below.

In the step 4 at Sec. 2.1.2 the Vehicle will go to the the Energy Globe to charge when all the followings are valid:

1. Local message of the Vehicle contains the information of Energy Globe.
2. Estimated left charge is still greater than the charge danger level. Where the `Left_Charge` is defined as below,

   ```
   1  Left_Charge := Current_Charge - Current_Discharge_Per_Sec * Estimated_Time;
   ```

   The `Charge_Danger_Level` is set manually. If it is too large, it will raise the frequency of charging, making the number of Vehicles charging at a same time too large that this model can not handle. If it is too small, the Vehicle will run out of charge before the get charged. Although we do have a estimation of charge consumption, it is not accrete for most time. In this project, the value of `Charge_Danger_Level` is 0.5 based on some experiments. The `Estimated_Time` is referenced from Eq. **??**.

2.1.5 Radius Determination

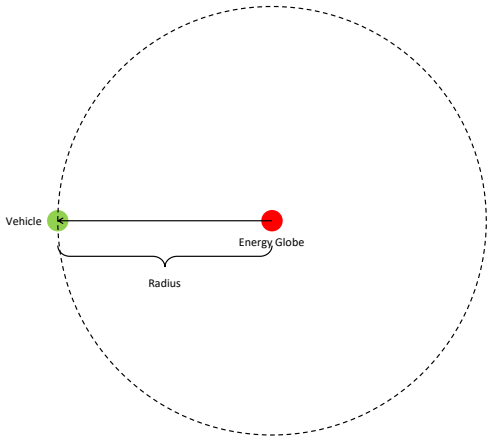


Figure 3: Radius Determination

The most important future of this model is a radius defines how far is the Vehicles and the Energy Globe.

To make the Vehicle locates on the surface of sphere when the Vehicle has the information of Energy Globe. Radius vector is defined as below

```
1  Radius_Vector := Radius_Distance * Norm (Position - Vehicle_Message.EG.Position);
```

And the position of the Vehicle is

```
1  Destination := Vehicle_Message.EG.Position + Radius_Vector;
```

`Radius_Distance` is set manually at the beginning. if it is too large, the Vehicles can not be connected to each other making the network unreachable. If it is too small, the probability of collisions will raise. In this project, the value of `Radius_Distance` is 0.3 based on some experiments.

In addition, because at the beginning of the program, all the Vehicles are moving in random, it is better to gather them at the origin point $(0, 0, 0)$, or some of them will be lost.

2.1.6 Radius Optimization With Current Charge

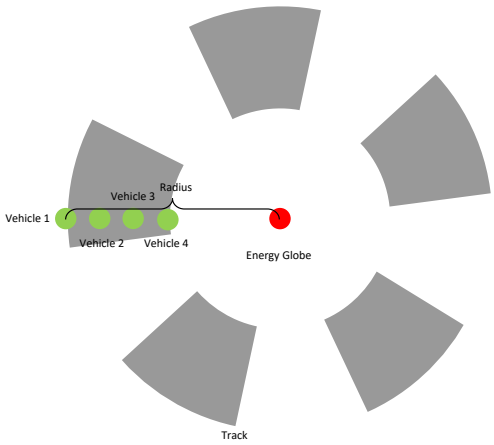


Figure 4: Radius Optimization With Current Charge

To make the Vehicle with lower charge more close to the Energy Globe, so that the Vehicle can go to charge with less distance and fewer time, so that the estimate in Sec. 2.1.3 will be more accurate and the information will be as new as possible. The Radius of the moving Vehicle can be optimized with current charge: lower charge with shorter radius. But the radius can not be too short, or the probability of collisions will be raised. So that there is a cavity in the center of the sphere. And it can not be too small, or the Vehicles run outside of the communication range.

The possible space of all the Vehicles is the gray space in Fig. 4. Vehicle charge in descending order: Vehicle 1, Vehicle 2, Vehicle 3, Vehicle 4.

The radius vector updated as below,

```
1  Radius_Vector = (0.75 + 0.25 * Current_Charge) * Radius_Vector;
```

The 0.75 and 0.25 are set manually and indicate that the possible range of Vehicles is 0.25 times radius.

2.1.7 A Solution To Collision

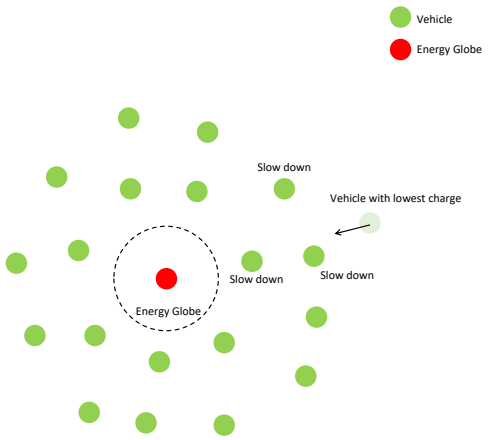


Figure 5: A Solution To Collision

The biggest problem of this model is that Vehicles will sometimes crash at the Energy Globe with low charge, none of them will get charged so most of them will die. In that case, the Vehicle with lower charge must get higher priority.

Add a variable called `Vehicle_Charge` to the message structure to denote the lowest charge locally. As shown in Fig. 5, the Vehicle with lowest charge will slow down all other Vehicles who want to charge and can get message directly from the lowest one.

The most important thing here is that the message is local, none of Vehicles should store a charge from the received message.

2.2 Stage C

2.2.1 Multiple Energy Globes Decision

If there are multiple Energy Globes appearing, moving and disappearing in the space, the Vehicle needs a method to choose the best one from them. Two possible situations when the Vehicles will face more than one Energy Globes is shown below,

1. When `Energy_Globes_Around` return more than one Energy Globes.

   The Vehicle will abandon previous information of Energy Globe and choose and store the nearest Energy Globe. Because position of the Energy Globe can only be estimated for the acceleration is changing and not given. The newer the more precise will the estimation be. So the previous one is abandoned. Go to the closer one is more efficient and charge saving. The nearest is by measuring the distance as below,

   ```
   1  distance := EG.Position + (Clock - EG_Find_Time) * EG.Velocity -
          Vehicle_Position
   ```

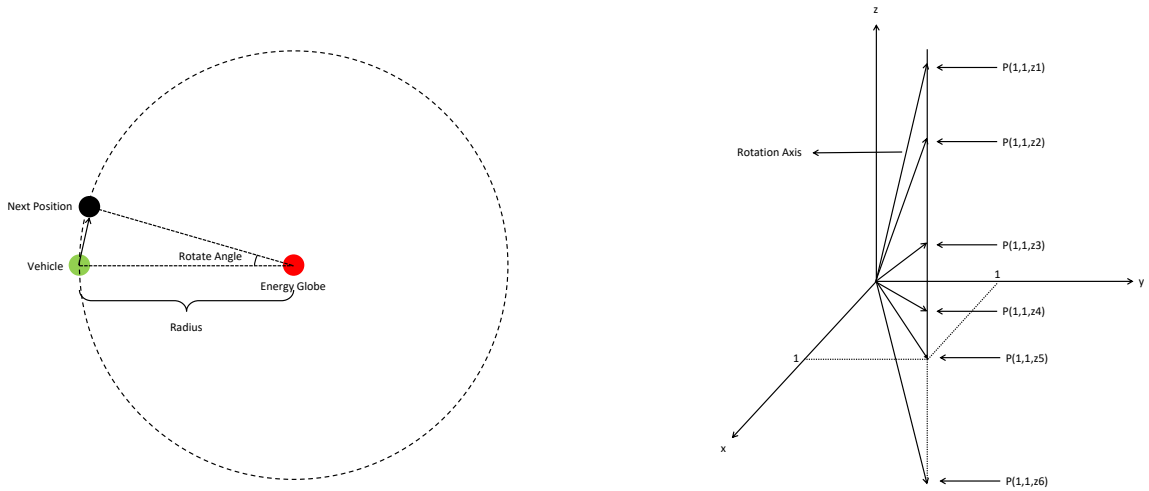
2. When the message from other Vehicles contains different Energy Globes.

   It the intuitive to save the newest one by compare the found time of Energy Globes as same as the situation 1. However, it will cause the Vehicle dilemmas between two Energy Globes and the Vehicle shakes and stays still as

the middle point of two Energy Globes. So there is a expire time mechanism to ensure the Vehicle would insist on his Energy Globe for a tiny while. This variable is called `Vehicle_Message_Expire_Time` and set manually at 1.0 for 1 second. Only when `Clock - Vehicle_Message.EG_Update_Time > Vehicle_Message_Expire_Time` will the Vehicle store the newly received Energy Globe information.

In multiple Energy Globes situation, once two Energy Globes go far away from each other, there would form two spheres. The Vehicles in the middle will determine which one it will follows. If two Energy Globes go near to each other, the two spheres will merge into one.

2.2.2 Optimization With Rotation



(a) Rotation Circle

(b) Rotation Axis

(c) Rotation Track

(d) Rotation Situation

Figure 6: Optimization With Rotation

As in previous model, all the Vehicles is positioned on the surface of the sphere and relatively static to the Energy Globe. If the Energy Globe moving at a high speed, it might just go through the surface via some holes. To fix that all the Vehicle must be moving to cover all the surface. That why Vehicles needs to be moving around the surface of the sphere.

In the Ideal model, the Vehicle moves like in Fig. 6a. In each `Outer_task_loop` the radius vector is rotated by some angle which is set manually at 1.0 for 1 degree. It is needed that in 3D space the rotation radius must be the actual radius. Which means that the rotation radius vector is vertical to the rotation axis vector,

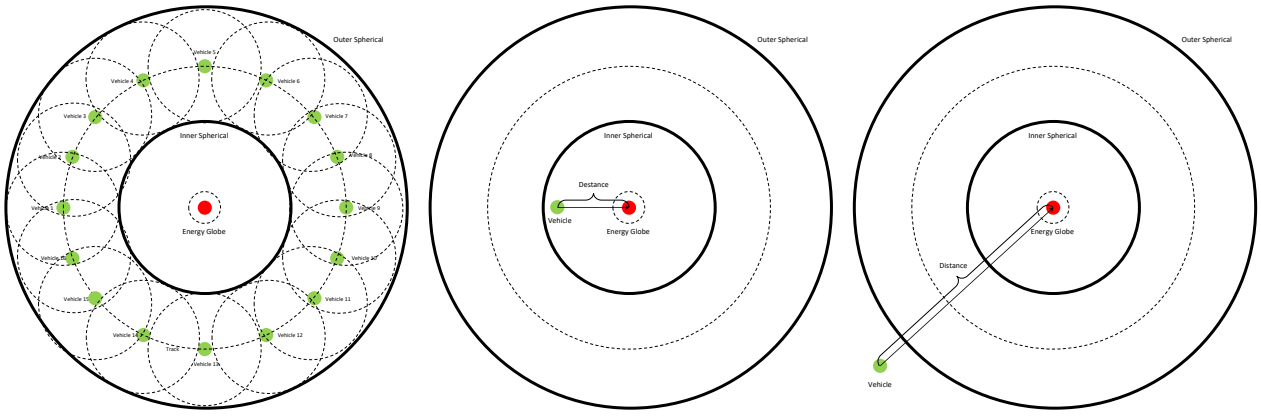
$$R_r \perp R_a$$

To find the R_a,

$$X_r X_a + Y_r Y_a + Z_r Z_a = 0$$

This equation has infinite many solutions for (X_a, Y_a, Z_a). And if it is set randomly, there would be a situation that two Vehicles is moving on a same track, but with opposite rotation direction. To avoid that, I set X_a and Y_a to 1 manually. As Fig. 6b shows, all rotation axis vector is laying in the same two quadrants and coplanar to each other. In that case each track for the specific rotation axis is vertical to the axis as shown in Fig. 6c. So, not only the radius is vertical to the rotation axis, but also that all the Vehicle has similar direction. A possible moving track is shown in Fig. 6d which is seeing from one side of the sphere and ignore the opposite.

2.2.3 Automatic Radius Adaptation



(a) Communication Range **(b)** Inner Spherical **(c)** Outer Spherical

Figure 7: Automatic Radius Adaptation

Because the number of Vehicles will decline after reach the `Target_No_of_Elements`. So as suggested in Sec. 2.1.5, the radius must response to the change. In the ideal model shown in Fig. 9 outside the outer sphere or insider the inner sphere, Vehicles are not able to communicate with others. Three situations below defines how the model works:

1. If the `Distance` is so small that the inner sphere almost do not exist.

   In this scenario the Distance needs to be scaled up and Vehicles can only act like Fig. 11. So when one Vehicle goes outside the outer sphere, it will come back towards the Energy Globe and corrects the `Distance` with the distance when it lost communication.

2. If the `Distance` is so large that Vehicle can not sending or receiving message form others.

   In this scenario the Distance needs to be scaled down. Because there is no communications at all, so no matter how large the `Distance` is initialized, it will be corrected down to a smaller one.

3. If the `Distance` is fitting.

   In this scenario, if we make the Vehicle goes a little bit farther than the out sphere after it charged, the whole `Distance` will be balanced dynamically. Because when it goes to get charged, it will get inside the inner sphere, so the `Distance` will be corrected to be smaller. Then when it finished charging, it will get outside the outer sphere, so the `Distance` will be corrected to be larger, and after two steps, the`Distance` should be stable.

Although this model can correct it radius length itself, there are still two variables need to be set manually,

1. The initial value of `Distance` .

   Same as Sec. 2.1.5.

2. The `Track_Correction_Rate` denotes how much a correction should be.

   The smaller the less impact will be apply to the `Distance`. But if the value is too large, at the vary beginning it would make some disasters to the model, because no Vehicle needs to be charged at that moment, so only Fig. 11 will happen, making the radius bigger and bigger. After a few simple tries, the value is set at 0.001.

To accomplish what have explained, each Vehicle needs to go a little bit farther than it used to do after charging, the following equation is used for that,

```
1  Actual_Radius_Vector := 1.5 * Rotated_Radius_Vector;
```

In the way for the Vehicle just charged, it will go 1.5 times farther.

2.3 Stage D

2.3.1 Consensus Algorithm[1]

Because there is no control center to make the decision, all the decisions are made during the message passing. In this project, a consensus algorithm is designed to solve this problem. The basic principle is shown as below.

1. The message and the Vehicle both store an array of type `Vehicle_No` in length `Target_No_of_Elements` and the last update time of the array `Vehicle_List_Update_Time`.
2. All the Vehicles are always sending and receiving messages.
3. At the beginning, all the Vehicles put its Vehicle number into the array as the first element.
4. If the Vehicle receiving a message that contains different message (the array or the last update time), it will store the one of them using the rule below,

   If two arrays contain difference number of Vehicle numbers, choose the the one contains more.

   If two arrays contain same number of Vehicle numbers, choose the one with earlier last update time.

5. For each Vehicle, check the array, include the Vehicle number if it is not in the array and there is still space.
6. `Confirm_Time_Inteval` seconds after last update time, if the array is fulfilled and the Vehicle number is not included. The Vehicle with that Vehicle number is not allowed to go to the Energy Globe. It is manually set at 1 for 1 second.

The reason for the principle 4 is that we want the array to be fulfilled as soon as possible. So the longer the better. As it is assumed that the sphere of influence of newer message is smaller, it is wiser to choose the older message when two array shares same length. All above is based on assumption that the time stamp generation is mutual exclusive, there could not be two same time stamp.

As for the principle 6, the value of `Confirm_Time_Inteval` is set manually based on some experiments. Also in this model, the Vehicle whose number is not in the array would keep going as it used to do, except that it would not to go charge any

[1] https://en.wikipedia.org/wiki/Consensus_decision-making

more. Which means this Vehicle is still in network, before the battery runs out, there is still change form it to go charge if the array is updated again. So the `Confirm_Time_Inteval` is only a safe measure, the value is not needed to be vary large.

3 Result

3.1 Screenshot

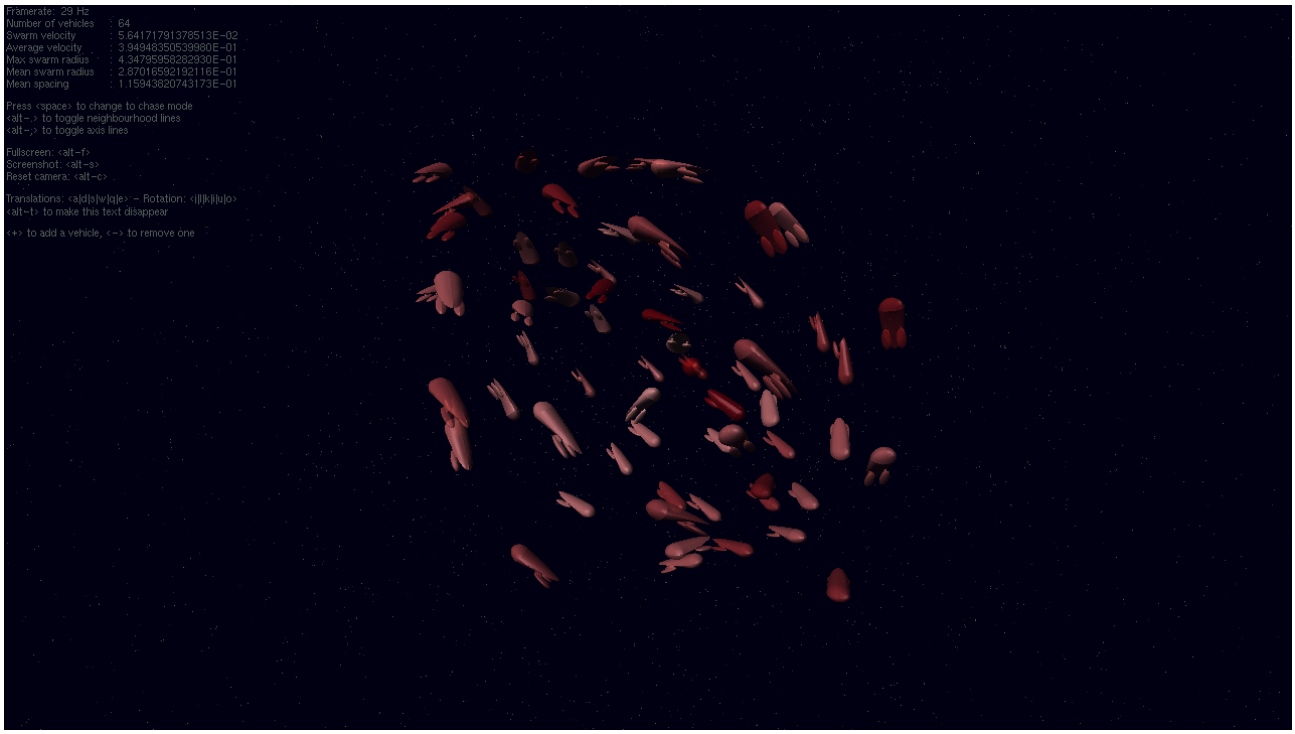


Figure 8: Screenshot

3.2 Testing[2]

The each test case will run a with only two different variables: the initial number of Vehicles and the target number of Vehicles. It will run 5 minutes and 5 times. For Every 10 seconds, the program will report how many Vehicles are left. Then record the average of the result. The average frame rate is also recorded but not vary precisely.

3.2.1 Stage A & B

Using `Single_Globe_In_Orbit`.

Table 1: Stage A&B

Initial Number	Target Number	Duration	Test times	Average Result	Survival Rate	Average Frame Rate
32	32	5 min	5	31.8	0.99375	30 Hz
64	64	5 min	5	63.4	0.990625	28 Hz

[2]The test is running on a laptop with intel i7 processor at 3.43GHz with 16GB memory. The performance might be lower than actual because I ran part of the test concurrently.

Initial Number	Target Number	Duration	Test times	Average Result	Survival Rate	Average Frame Rate
128	128	5 min	5	126.8	0.990625	20 Hz
256	256	5 min	5	233	0.910156	9 Hz

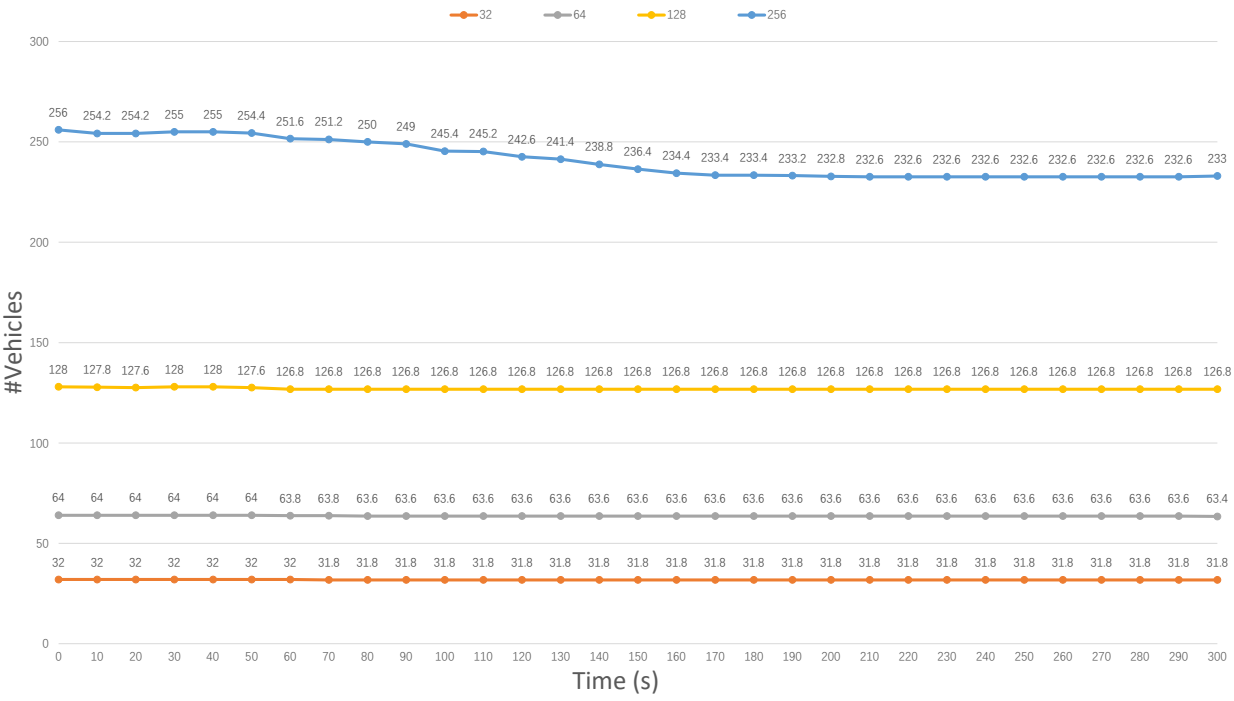


Figure 9: Stage A&B

For the stage A&B, the average survival rate is all above 90% for 4 cases. And get stable before 5 minutes. The model is a little bit unfit if there are around 256 Vehicles. By observation, the Vehicle died because too many collisions happen around the Energy Globe.

To avoid that, there should be a better strategy to solve the collision problem or if the `Distance` is larger at the beginning will alleviate this situation in some way.

3.2.2 Stage C

Using `Random_Globes_In_Orbits`.

Table 2: Stage C

Initial Number	Target Number	Duration	Test times	Average Result	Survival Rate	Average Frame Rate
32	32	5 min	5	26.8	0.8375	30 Hz
64	64	5 min	5	60.2	0.940625	28 Hz
128	128	5 min	5	109.2	0.853125	20 Hz
256	256	5 min	5	208.2	0.813281	9 Hz

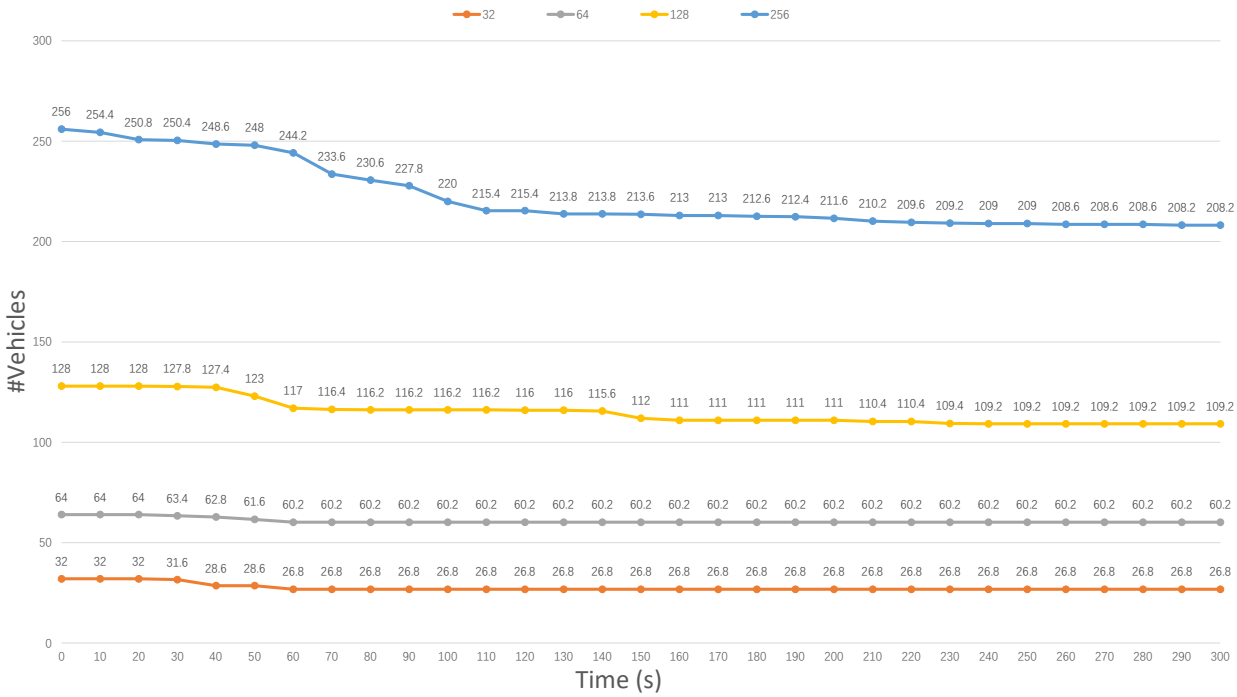


Figure 10: Stage C

For stage C, the average survival rate is better at 64 than other initial numbers. Although the result is not as good as stage A&B, but still get stable within 5 minutes. By observation, the death at 32 Vehicles is due to that the Vehicles around the Energy Globe can not form a network because the radius is way too large for the number of Vehicles. And the reason for more than 64 Vehicles is that if a Energy Globe suddenly disappears, the Vehicles whose following and on the way to charge will meet at the position of the Energy Globe as it used to be. Making the communication with other spheres unreachable. The Vehicles will go to some where in a small group and use up there battery.

To avoid that, there should be a mechanism to solve the disappearance of Energy Glove. For the 32 Vehicles situation, scaling down the `Distance` or scaling up the `Track_Correction_Rate` might help.

3.2.3 Stage D

Using `Random_Globes_In_Orbits`.

Table 3: Stage D

Initial Number	Target Number	Duration	Test times	Average Result	Survival Rate	Average Frame Rate
32	42	5 min	5	24	0.571429	30 Hz
64	42	5 min	5	37.4	0.890476	28 Hz
128	100	5 min	5	83.4	0.834	20 Hz
256	150	5 min	5	133	0.886667	9 Hz

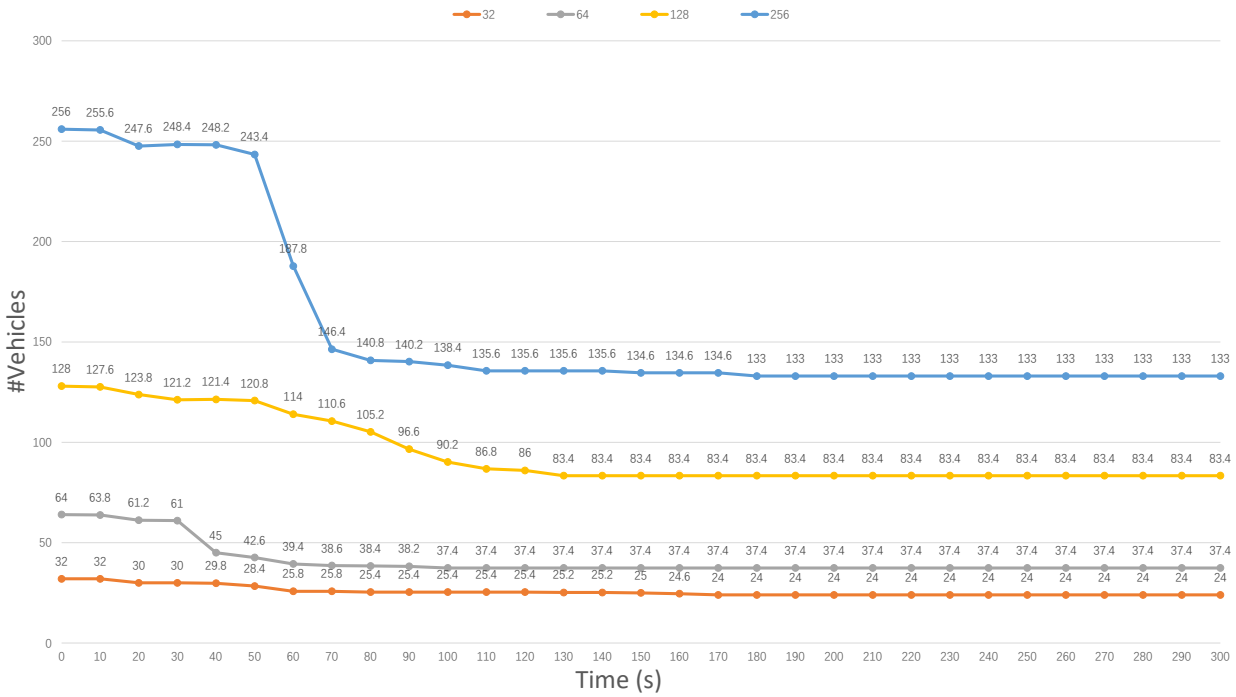


Figure 11: Stage D

For stage D, the consensus algorithm actually performs accurately(can be verified using `Single_Globe_In_Orbit`), the low performance for the result is due to that there will be death of Vehicles before they reach a consensus for the same reason in stage C.

To avoid that, there should be a method to adjust the array dynamically with the information of liveness of every Vehicles.

4 Appendix

Discussed with,

- U6919043
- U6921112
- U6921252
- U6921163
- U6920262
- U6920122
- U6920829