Joby eVTOL Challenge

Adrian Deardorff - 80deardorff@gmail.com - 5/11/2022

Here is my thought process:

- I need a vehicle class that all vehicles are derived from. This will keep functionality uniform across all instances of vehicles of different companies.
- I need to keep data that is total among the company.
    - Maybe I use one class per company with static data?
    - ~~Maybe use a template class to not have to duplicate code~~
    - After many hours of digging, there is a design pattern called Factory Method Design Pattern. Seems promising to create vehicles of each company to be created at runtime randomly. Can use a nice template as shown above for the company classes to keep data consistent across companies.
- Need a way of randomizing company vehicles 6 - 20. 1 - 5 will be an instance of each company vehicle - otherwise, what is the point of having the class if it won't get used.
    - Should have a loop create vehicles generated by factory method up to MAX_NUM_VEHICLES.
    - Vehicle companies can be put in enum in order to get total number of companies and make for easy random number generation
- Need a charging queue for vehicles awaiting charge
    - Create a charger class that holds a static shared queue amongst all charger instances.
    - Charger instances can be stored in a vector at runtime in the main and have a bool for isCharging along with pointer data for the vehicle they are charging.

Plan of Attack:

The following picture below shows the three different states an eVTOL can be in. It can either be flying, waiting to charge, or on one of the 3 chargers. There is no other state at which the vehicle can live; this really simplifies the design. In this case, I plan to use a a vector for the flying state and the waiting to charge state. Then, for the chargers, I will create a charger class with its own Vehicle data member. The vector for the waiting to charge state can live statically in the charger class….I think this makes sense to keep it together.

With all of this in mind, the eVTOLs I can represent with shared smart pointers. ~~No eVTOL will be in more than one place at a time so a unique pointer would allow me pass the plane around like a hot potato from one state to the next as needed.~~ (EDIT: decided to change to a shared pointer because moving a unique ptr around in and out of vectors caused a ton of headaches as there is no copying of unique pointers.)

For the timing, pointers will be very efficient so I don't see a reason for starting multiple threads in order to run through each vector and manipulate the data of each object through its pointer. I will test the timing of this and modify my approach if this doesn't work out. Timely completion on a single thread is my only concern at this point. However, when considering a more simple design with an acceptable speed of completion versus a complicated design that runs slightly faster, I'll take the simple approach if I can help it. (EDIT: Found this does work and can run through all objects fairly easy within 1 to 2 ms.

I will show my ability of multi-threading by having the battery charge in its own thread. This way the main simulation only has to pass the vehicle to the charger queue. The charging of the vehicle's battery will be handled at the charger in the newly created thread.