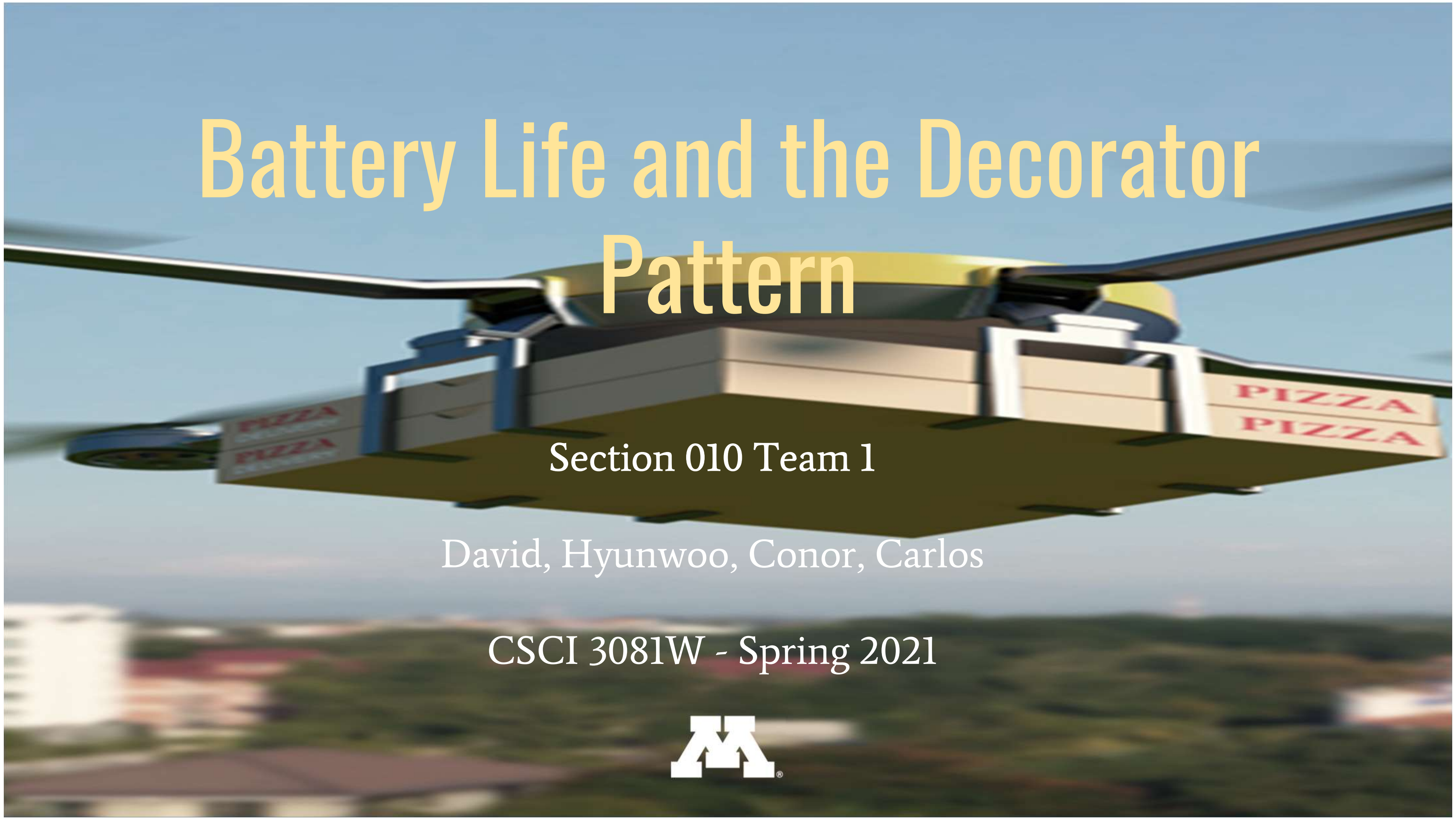


Battery Life and the Decorator Pattern

Section 010 Team 1

David, Hyunwoo, Conor, Carlos

CSCI 3081W - Spring 2021

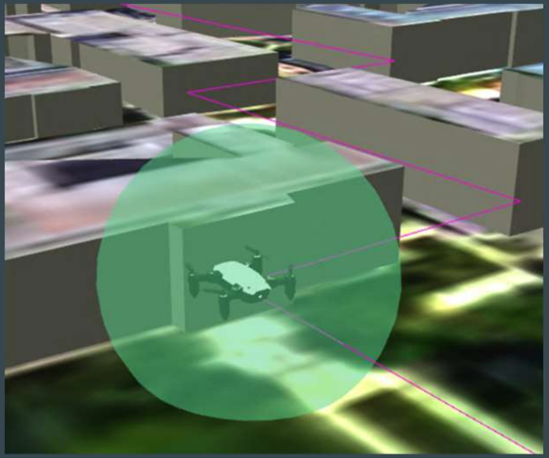
Motivation

Until now, our simulation has dealt with many of the issues involved in scheduling the delivery of packages.
-We have dealt with scheduling issues to allow multiple entities to deliver packages to customers.
-We have implemented a variety of paths to the customer, so that packages can take different routes above and around buildings.
-We have allowed for different kinds of deliverer entities within the simulation, so that we can deliver a package by land or by air.

We do have batteries and allow for our drones to be idle, but we would like to be able to tell the battery life of the entities as they move through the simulation.

Background

- We want to know when our entities are out of battery, and we want to be able to physically see the difference in the simulation.
- A good way to display this information is to surround our entity with a different colors based on the charge of the battery.

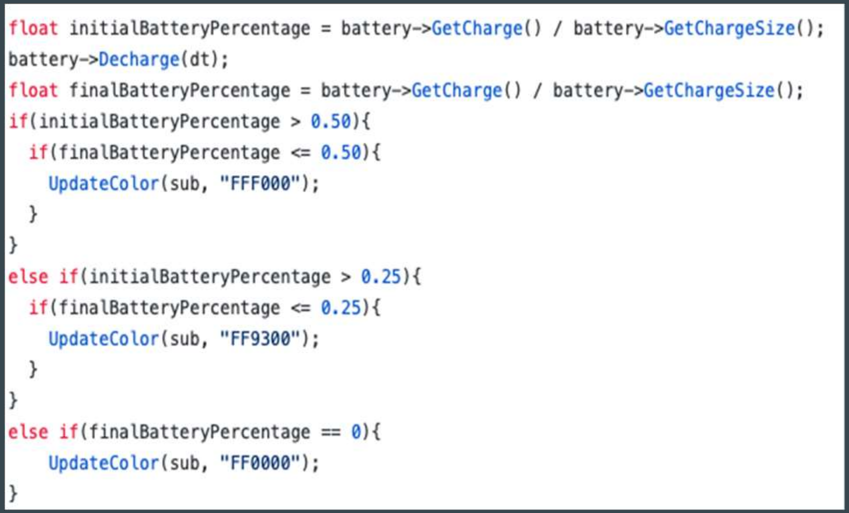


- We do not want to change how our code works, but we want to add extra functionality to just change the appearance of our deliverer entities.
- How would we add this functionality, and could we use the decorator pattern to do so?

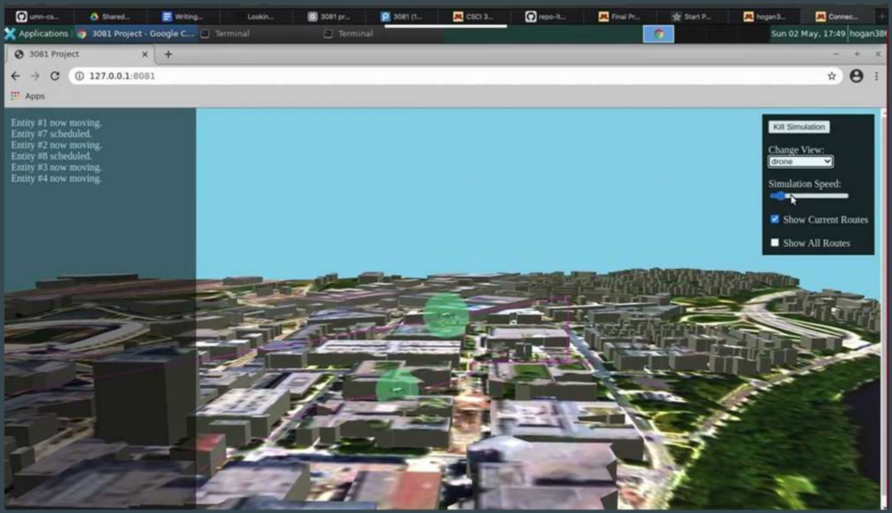
Non-Decorator Pattern Approach

- Initial battery life implementation did not use the decorator pattern
- Move to Points Function
- Pros:
  - Simple implementation, took about twenty minutes to implement
  - Putting this within the move to points function allowed an easy way to tell when the battery life of a drone crosses a threshold.
- Cons:
  - Creates more code within move to points function, violating the single responsibility principle and lowering cohesiveness

Non-Decorator Pattern Approach Code and Video



```
float initialBatteryPercentage = battery->GetCharge() / battery->GetChargeSize();
battery->Decharge(dt);
float finalBatteryPercentage = battery->GetCharge() / battery->GetChargeSize();
if(initialBatteryPercentage > 0.50){
  if(finalBatteryPercentage <= 0.50){
    UpdateColor(sub, "FFF000");
  }
}
else if(initialBatteryPercentage > 0.25){
  if(finalBatteryPercentage <= 0.25){
    UpdateColor(sub, "FF9300");
  }
}
else if(finalBatteryPercentage == 0){
    UpdateColor(sub, "FF0000");
}
```



```
Entity #1 now moving.
Entity #7 scheduled.
Entity #2 now moving.
Entity #8 scheduled.
Entity #3 now moving.
Entity #4 now moving.
```

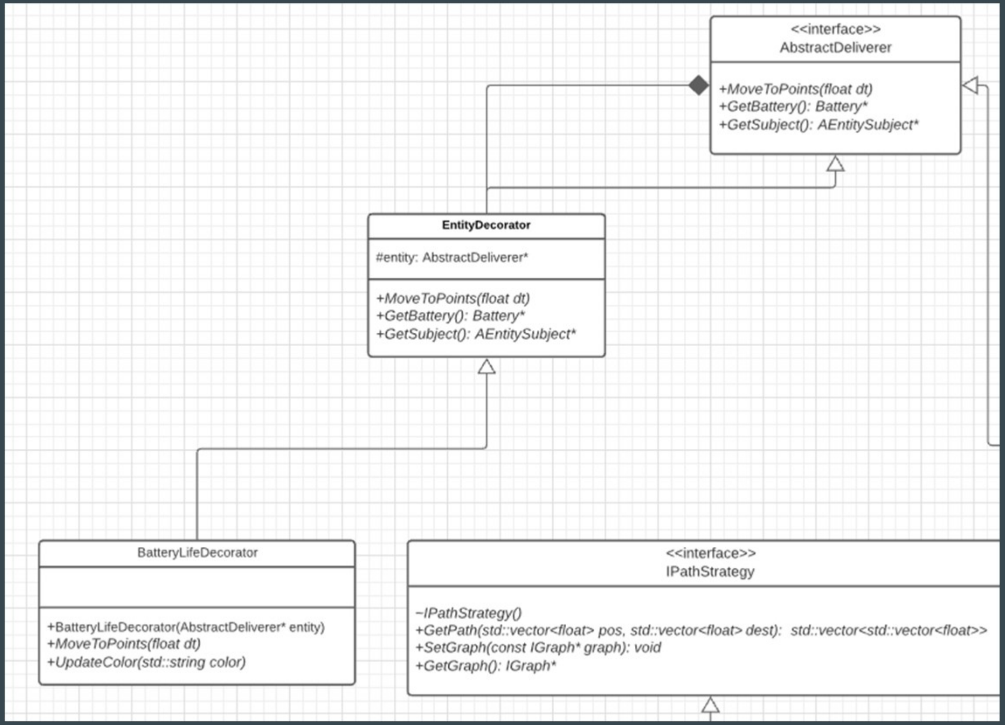
Change View: drone

Simulation Speed:

☑ Show Current Routes

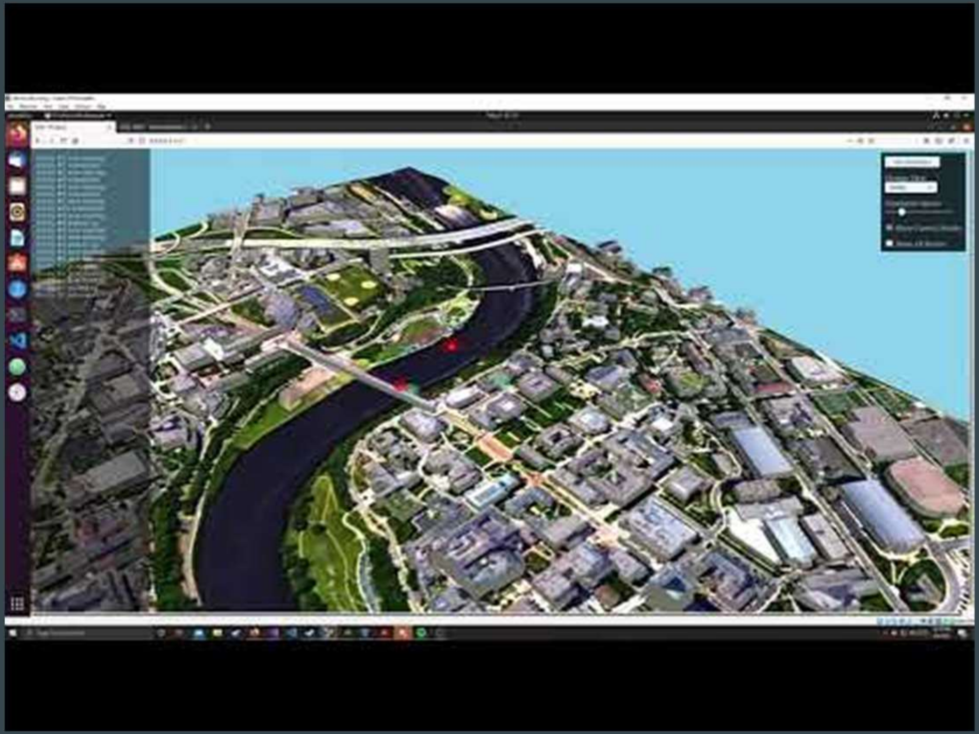
☐ Show All Routes

Decorator Approach Details/UML



<<interface>>
AbstractDeliverer
+MoveToPoints(float dt)
+GetBattery(): Battery*
+GetSubject(): AEntitySubject*
EntityDecorator
#entity: AbstractDeliverer*
+MoveToPoints(float dt)
+GetBattery(): Battery*
+GetSubject(): AEntitySubject*
BatteryLifeDecorator
+BatteryLifeDecorator(AbstractDeliverer* entity)
+MoveToPoints(float dt)
+UpdateColor(std::string color)
<<interface>>
IPathStrategy
~IPathStrategy()
+GetPath(std::vector<float> pos, std::vector<float> dest): std::vector<std::vector<float>>
+SetGraph(const IGraph* graph): void
+GetGraph(): IGraph*

Demonstration



Results and Conclusions

- The drone/robot now change color based on the battery charge
  - This is only if we are using the BatteryLifeDecorator
- Adding the decorator pattern was difficult
  - If only adding one feature and no other feature ever, then it’s not worth it
- Now that it is added it will be easier to add visual or other similar features
  - If we are adding multiple features that are similar in the future then using the pattern is worth it

Drone Pool/CSV Reader

- We also created a Drone Pool and a CSV Reader in our presentation
- Different attributes were given in the drone_models.csv file

Attributes included:

- model #
- mass
- max speed
- base acceleration
- weight capacity
- base battery capacity

- However, due to time constraints and presentation limitations, we left more discussion out of our presentation.

Thank you for watching!

References

- https://www.digitaltrends.com/cool-tech/drone-deliveries-step-closer-in-uk/
- https://www.dofactory.com/net/facade-design-pattern/
- https://www.dofactory.com/net/abstract-factory-design-pattern/
- https://www.dofactory.com/net/observer-design-pattern/
- https://www.dofactory.com/net/strategy-design-pattern/