Project ANVIL - Spec 3.0

Aerodynamic Navigational Vehicles for Instantaneous Locomotion

Figure 1. Drone Delivery

October 31, 2020

Authors: CSCI 3081 Staff

Iteration 3: Analysis and Optimization

Note: This document is subject to change at any time depending on the business needs. We will send out a notification if there are any changes along with the documented revision.

Revision History:

- 3.0 - 11-24-2020 - CSCI 3081 Staff - ANVIL 3.0 specification release.

1 Project Description and Overview

In this iteration you will release your Drone Delivery System. This should include high-quality code and documentation (See McConnell, Chapter 32). In addition, you will be responsible for adding one more feature of your choosing. There will be one final deliverable for this project. Below is a project timeline:

Dates & Deadlines	**Item**	**Description**
12/11/2020 **(Due: Friday 11:55pm)** **Canvas / GitHub**	Final Deliverable: Iteration 3	All requirements.

Document Organization:

2 Iteration Three: ANVIL Simulation Specification

The purpose of Iteration Three is to extend the drone delivery system and develop a useful optimization or feature. Your group is to decide on a feature to implement, implement it, and present your results (as a final presentation for the course). Below is a list of possible extensions to choose from. If you would like to implement your own extension, you may. We will ask all teams to report their proposed work and we will give feedback or suggestions. Please review the suggested extensions below (these can be modified to fit the needs of each team):

We can help technically if you need to modify the UI.

Extension	**Description**	**Relevant Patterns (suggestions)**
Drone Pool Strategy	When a package is scheduled, it is important to pick the right drone. Modify the system to allow for several different drone selection strategies. ● For example, consider the following characteristics about a drone: closest, most capacity, available, battery life, fastest. ● It would be interesting to make these more complex by combining the composite pattern to create composite strategies.	Strategy, Factory, Composite References: https://www.geeksforgeeks.org/composite-pattern-cpp/ https://www.geeksforgeeks.org/strategy-pattern-set-1/
Entity Color Decorator	Use the decorator pattern to add color to any entity based on some feature (e.g. battery life, distance to destination, weight, etc…).	Decorator Reference: https://www.geeksforgeeks.org/decorator-pattern/?ref=lbp
Porch Thieves	Create a new entity that disguises itself as a different type of entity. When a package is close enough, the porch thief steals the package and takes it home.	Decorator

Collision Avoidance	Make sure drones do not hit each other. If drones are sharing the same graph, they should find alternate paths. If drones must be on the same path, they should use the vertical dimension.	Strategy
New Scenario	Create a new scenario. Verify that all functionality works in this new world, and add new actions (e.g. trolly car carrying passengers in San Francisco). Create new entity types that are relevant to the new location.	Factory
Customers that Move	Make the system truly dynamic using more efficient shortest path algorithms (e.g. A* with euclidean distance heuristic). Allow the drone to deliver to moving customers.	Strategy, Observer
Create your own Weighted Graph	Add a weighted graph that indicates some other quality (e.g. congestion). Reference network graph examples. Implement and analyze multiple heuristics for A*.	Strategy
Transit System	Modify the drone delivery system into a bus transit system that delivers passengers rather than packages.	Adapter Reference: https://www.bogotobogo.com/DesignPatterns/adapter.php
Drone Physics	Develop a better physics model that more realistically models a drone (e.g. the drone slows down when turning corners, etc…). Add more forces including wind patterns and gravity.	Strategy
Drone Statistics	Collect drone delivery statistics based on the simulation (e.g. delivery time, packages delivered, distance travelled). Store these in a CSV file or other mechanism (e.g. database). Analyze the data and share how to better optimize the system.	Singleton Reference: https://gameprogrammingpatterns.com/singleton.html
Remote Control	Add a way to control drones / customers / packages using the Command Pattern. Perhaps create another interface (web based or console) for moving drones, scheduling packages, creating new entities. (e.g. keyboard arrow keys / WASD)	Command Reference: https://gameprogrammingpatterns.com/command.html

Model Animations	Our models have different animations (for example a customer can fall apart or wave, and a drone can stop moving). Add a way to control model animations based on actions done in the simulation.	Decorator
Recharge Stations / Drones	Add a way for drones to gain battery life. They can either go to a recharge station or be given battery power from a recharge drone. A recharge drone is one that delivers power to a drone.	Factory
Internet Emulation	Break packages up into many pieces (packets) and allow a large number of drones to carry them to their destination over different paths (perhaps based on route congestion). ● Store the calculated route information at each node in the graph to avoid recalculation. For example, each node would store a map of destination (as key) to the next node (as value). The tables are built whenever a route is calculated. ● The packets will store the destination (not the route) and will be routed through the nodes based on the information at the nodes. ● Each customer will reassemble the package once the packets have arrived.	Composite, Flyweight, Strategy References: https://www.geeksforgeeks.org/flyweight-design-pattern/ https://gameprogrammingpatterns.com/flyweight.html
Multiple Delivery Systems	Combine multiple delivery systems into one delivery system.	Adapter, Facade, Composite
Other	Choose your own feature to implement. Please check with the instructional team to verify that your extension is adequate or feasible through email, csci3081f20@umn.edu, or on Piazza.	

3 Evaluation Criteria

Iteration Three is evaluated as a group project (***with the exception of the Final Presentation, which is an individual grade***). **If you do not contribute to the development of the project, you will receive 0 points for this iteration.**

Below is the point breakdown:

Design

- 40 points - Updated Doxygen Documentation & UML
- 10 points - Short description of feature or extension for the project on Doxygen mainpage **(Be sure to add this on the mainpage).**

Development (No automated feedback)

- 40 points - Code Quality based on the **Release** branch.
  - Self Documenting Code - (McConnell, Chapter 32)
    - Good class / method naming, comments, documentation in general.
    - Follows coding style - (e.g. Google Style Guide)
  - Robustness
    - Good memory management
    - Considers input edge cases
  - Patterns are used correctly and consistently.
  - Good use of C++ Classes / Enumerations
    - Uses big three
- 10 points - Development Process
  - Correct use of branches, issues, and pull requests.

A 5 - 10 Minute Final Presentation (This is not part of the iteration 3 grade - this will be graded separately at 5% of your overall grade in the class)

This is a group presentation, but each person is graded individually based on participation within the group. We will give you a breakdown of how this will be graded later. A spreadsheet of scheduled time will be provided soon.

- Introduction and Motivation
  - Persuasive arguments
  - Relevance to the big picture
- Related work
  - How does the feature fit into existing work?
- Implementation
  - Describe what you did (e.g. algorithms / patterns used).
  - Describe development process
- Results
  - Show a video / show graphs / describe numerical charts
- Conclusions
  - What did you learn?
  - What would have you done differently?
  - Does your approach work?