Team number: 010-30

Member names and x500s:

Erik DeVries Smith devri212

Newton Smith smit9160

Drew Osmundson osmun049

Kyle Bekken bekke035

What the project is about

The whole project is about a package delivery drone simulation. The delivery drones will path to a package called robot and deliver it to the desired location using one of three pathing strategies that have been implemented.

To run the simulation

First in a Linux terminal in the project directory run

$make clean

This is done to ensure there are no leftover files from the previous run.

The next step is

$make -j

This will compile the project, so the server is ready to start.

To then start the server run (./build/web-app <port> <web folder>) to run locally use

$./build/bin/transit_service 8081 apps/transit_service/web/

This will start the server and the scheduler is ready to be accessed.

To add a drone to the simulation in a browser, navigate to

http://127.0.0.1:8081/schedule.html

This is where each new trip can be scheduled and the user is prompted to give a name, a pathing strategy, the number of passengers, and the pickup and drop off location.

To then view what is being simulated in a different browser tab, navigate to

http://127.0.0.1:8081/

This is where the drones can be seen flying to pick up the robots to add more drones, navigate back to http://127.0.0.1:8081/schedule.html to make a new trip.

What does the simulation do

The simulation allows the user to schedule delivery trips for their robots by entering the robot's current location, destination, and what flight pattern the drone should use to get to the destination. The simulation also allows the user to view the movement of drones and robots in real time as the drones complete their deliveries.

New Feature

The feature that our group added to the simulation was CO_2 tracking. We made it possible to track the cumulative CO_2 that all the drones in the simulation would produce if they were flying in the real world.

What the new feature does

This feature accomplishes this by having a singleton called CO2Tracker keep track of the total CO_2 produced by the drones following each movement strategy in addition to the cumulative CO_2 produced by all drones in the simulation. Over the course of the simulation's lifespan, these values are constantly being appended to a .csv file (generated in the root directory of the project) in order to generate data which can be studied to research which movement strategy is the most efficient. We also added a live counter of the accumulation of the CO_2 output so the user can stay informed of their carbon footprint.

Why this new feature is interesting

Electric vehicles are often seen as having a net zero carbon output. Of course, the electricity must come from somewhere, whether that is from a traditional source like natural gas or coal or a renewable source. In the United States on average producing 1 kWh generates 0.81 pounds of CO_2. Our simulation extension has value because it has many real-world applications this could be used for, such as electric vehicle efficiency and their environmental impact. In our project, the efficiency variable is adjusted by changing the number of passengers. By exporting the data to a CSV file, our simulated data can be compared from pathing strategy to pathing strategy and at different masses to show how changing efficiency and having longer paths can affect CO_2 output even from an electric vehicle

What is added to the existing work

Our extension adds the ability for the program to gather useful data that can be used to expose the environmental impact of the drones represented in this simulation. Our extension exports to a CSV file for further data manipulation in addition to displaying the user a live tracker of the total CO_2 produced by all drones over the simulation's runtime

The implemented singleton design pattern

Our singleton file is CO2tracker.h. We chose to use the singleton design pattern because the nature of our extension requires usage of global variables. Such a feature is not available in any other design pattern while also following software engineering best practices.

To use this new feature

In order to use the new feature, follow the above steps of how to run the project until the trip scheduler is reached, where the user enters the number of passengers they want to simulate. Then in the simulation, the total CO_2 output can be viewed.

The sprint retrospective

Things That Went Right

- Adding the drop-down menu in the html code.
- The UML diagram made our initial coding much faster.
- We completed the boiler plate code very quickly.
- Regular meetings kept us on schedule.

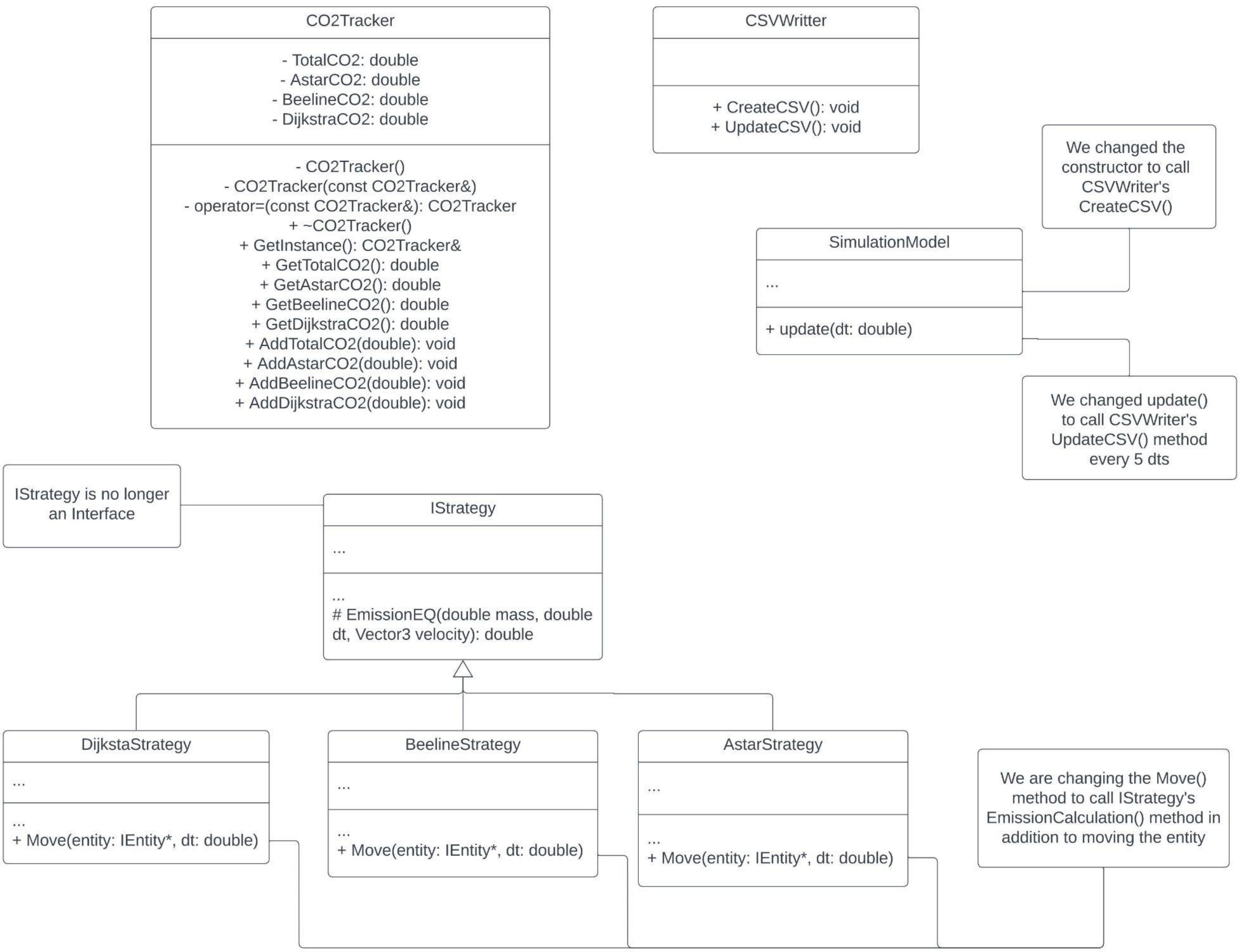
Things That Went Wrong

- Not enough time allocated for testing.
- Our requirements were originally not in line with what we could do in one sprint.

- We did not complete any of the project progress outside of meetings.

- GitHub pushes were not as frequent as would have been ideal.

UML Diagram



Trello Board (Kanban)

https://trello.com/b/ZtwfJpcQ/csci-3081w-project-4-sprint

Github

https://github.umn.edu/umn-csci-3081-F22/Team-010-30-homework04

Presentation

https://drive.google.com/file/d/1NV1Tqw-XZVYU0L3wCzXqk8z-v_oyUet8/view?usp=sharing

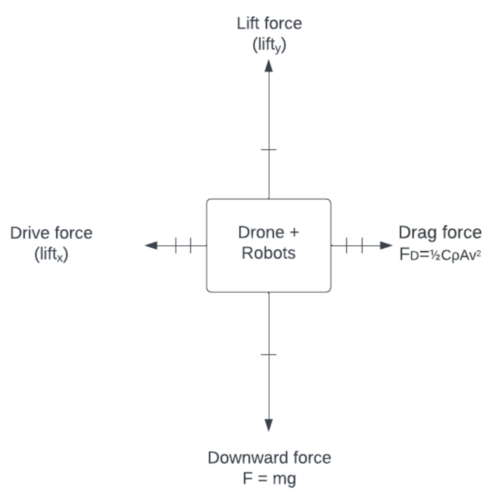
Demo

https://drive.google.com/file/d/11xWSP8FqtVrRva3Ig_1dhU1C2gY6BIY5/view?usp=sharing

Docker

https://hub.docker.com/repository/docker/devri212/homework4

CO_2 output calculations



$lift_y$ = gravity
$lift_x$ = drag

Energy Expended = ($lift_y$ + $lift_x$)*dt*speed
Energy Expended = ($9.8m_d + \frac{1}{2}C\rho Av^2$)*dt*speed
Energy Expended = ($9.8m_d + 0.58434375v^2$)*dt*speed

Fluid density of air
p = 1.2466

Cross-sectional area of drone
a = $1.875m^2$

Drag Coefficeint
c = 0.5

Gravity
g = $9.8m/s^2$

Mass of drone
m_d = 30kg

CO2 output (in pounds) = (Energy Expended * .818 pounds of CO2 per KiloWatt Hour) / 3600000 Kilowatt Hours per pound CO2

(pounds of CO2 per kWh provided by carbonfund.org)