Senior Design Final Report

SUAS Flight Path Visualization

Version 1.0

Team Members

Alex Alcazar
Francisco Brito
Helen Dam
Alex Gaeta
Alberto Gonzalez
Sergio Maradiaga Olivera
Thad Owens
Mychal Salgado
Kevin Tang

Sergio Valadez Polanco

Faculty Advisor
Dr . David Krum

Liaisons
Dr. Paul Fedele - Physical Scientist
Dr. Eric Holder - Research Psychologist

1. Introduction

1.1 - Background

The Army Research Lab (ARL), the US Army's foundational research facility, focuses on scientific discovery and technological innovation to enhance the Army's capabilities in future conflicts. Spanning across regional sites in the West, South, Central, and Northeast of the United States, the ARL harnesses small, uncrewed aircraft, systems, and drones to furnish valuable intelligence, surveillance, and reconnaissance information to small military units. To reduce the risk of detection, these drones must carefully navigate their flight paths while considering variables like environmental conditions, sensor capabilities, noise and visibility signatures, and human perception models.

The project aims to develop, test, and deploy a 3D visualization architecture enabling operators to plan, visualize, and adjust drone flight paths over varied terrains, optimizing sensor use while mitigating detection risks. With Over-The-Air updates facilitated by the web-based platform, future developers can seamlessly integrate and update sensor technologies, human perception models, and drone functionalities.

Our project consists of a web application using HTML5, CSS, JavaScript, and Node.js, with a user interface (UI) for ease of use. Our group set various requirements for the visualization architecture, including drone flight path, ray casting, and the display of specific metrics such as altitude, drone height, total flight path distance, and longitude and latitude coordinates.

1.2 - Challenges

One of our biggest challenges for the project was when attempting to use the past work of the past team, there were a lot more issues with implementing the system than originally anticipated. Even after reviewing past documentation, instructions, and

the contents of the past project, it proved too difficult to build off of it. This resulted in us having to reset and build the project back up.

Raycasting was a significant challenge, as we had to start from scratch when designing the visualization framework. Utilizing a computer graphics rendering method known as "raycasting," we transformed 3D scenes into 2D images through geometric ray tracing. We encountered issues with the raycast being blocked by buildings made available through Mapbox, and the buildings were not interacting with the ThreeJS ray cast. Additionally, we were having trouble getting the ray cast to interact with the 3D drone model. It was also difficult to understand where the ray cast was shooting its rays.

Developing a 3D model for the building meshes/structures presented another difficulty. To enable raycasting obstruction by buildings, we must dynamically generate each building on the map with precision and position them accurately in their designated locations. Additionally, to avoid the program slowing down, limit the number of structures that are generated.

The drone distance team had several difficulties. Among these issues, one identified the most effective approach to recording the drone's position mid-flight to ensure proper resuming after interruption. The team also had to explore different methods to store drone specs for use in instances such as this, calculating the remaining available distance. They considered connecting the project to a database however, it resulted in storing some hard-coded and arbitrary drone specs in an array as a proof of concept for simplicity due to time constraints. Additionally, they struggled with providing interactivity for the drone, transforming it from a moving model on the map to an object capable of interaction with components such as no-fly zones and detectors. Finally, calculating the drone's distance was also a challenge because they had to consider any changes in elevation since the drone would be traveling more distance going up than just going on a straight path with no elevation. The pausing function was also tricky to implement since they had to figure out a way to get the drone's current distance traveled along the path and get it to accurately compare the

distances of each point to determine which two points along the flight path the drone was currently in between.

Another challenge the team faced was implementing the UI with Mapbox because there were times when the UI would not function correctly with Mapbox. Specifically, visual bugs, such as the UI closing incorrectly, moved Mapbox to the left on the screen instead of expanding and taking up the blank space left on the screen. A different visual bug that was known in the UI development was the CSS files incorrectly adjusting the elements on the sidebar and the compass displayed on the Mapbox.

1.3 - Group Members' Contributions

There were several side teams among us, and members would swap sides to lend a hand as needed. Members not only recorded and documented our sessions, but they also actively offered vital information to help with our goals. Online tools like Zoom and Discord were used to help members with conflicting schedules communicate and work together more efficiently. The numerous follow-ups and assistance with missing meetings demonstrated a dedication to inclusivity and ensured every member felt valued and included. To address any issues or concerns that arose during the project, the group promoted open communication and open criticism as well. This collaborative approach resulted in a productive outcome by making all team members feel connected and cohesive throughout the entire development process of the project.

1.4 - Key Design Principles

The HTML5, CSS, JavaScript, and Node.js architecture for the 3D visualization produces a flexible, scalable, and user-focused solution. We prioritize providing operators with an easy-to-use interface and making future updates and technological integrations simple. An extensible and modular design allows for the independent development and deployment of components such as sensor technologies, drone

capabilities, real-time rendering processes, and data processing. This helps with performance and optimization. Compatibility with many browsers and devices and responsive design principles guarantee broad accessibility and utility. In addition to feedback systems, thorough documentation—which includes user manuals and tutorials—also improves the user experience and encourages efficient usage of the visualization system.

1.5 - Design Benefits

There are several benefits to using HTML5, CSS, JavaScript, and Node.js architecture for 3D visualization. JavaScript and HTML5 offer strong capabilities for developing engaging and easy-to-use user interfaces. Using well-known web interface paradigms, operators can effectively plan, visualize, and alter drone flight trajectories over terrain, improving mission planning's usability and effectiveness. Overall, using HTML5, CSS, JavaScript, and Node.JS architecture for 3D visualization in military drone operations provides a strong and adaptable solution that supports mission planning and execution in demanding and dynamic environments by combining accessibility, usability, scalability, and security.

Utilizing Mapbox allowed us to open a web browser and view Mapbox maps in our application, integrate user interaction, and personalize the map experience.

2. Related Works and Technologies

2.1- Review of Existing Works

The team found out that when attempting to use the past work of the past team, there were a lot more issues with implementing the system than originally anticipated. Even after reviewing past documentation, instructions, and the contents of the past project, it proved too difficult to build off of it. This resulted in us having to reset and build the project back up.

2.2- Justification of Process

We collectively decided to scrap the program and start over. That way, we could maximize our time on the new program. We were able to work in smaller groups and be more hands-on by creating new software, which allowed us to concentrate on obtaining all the criteria for the web application.

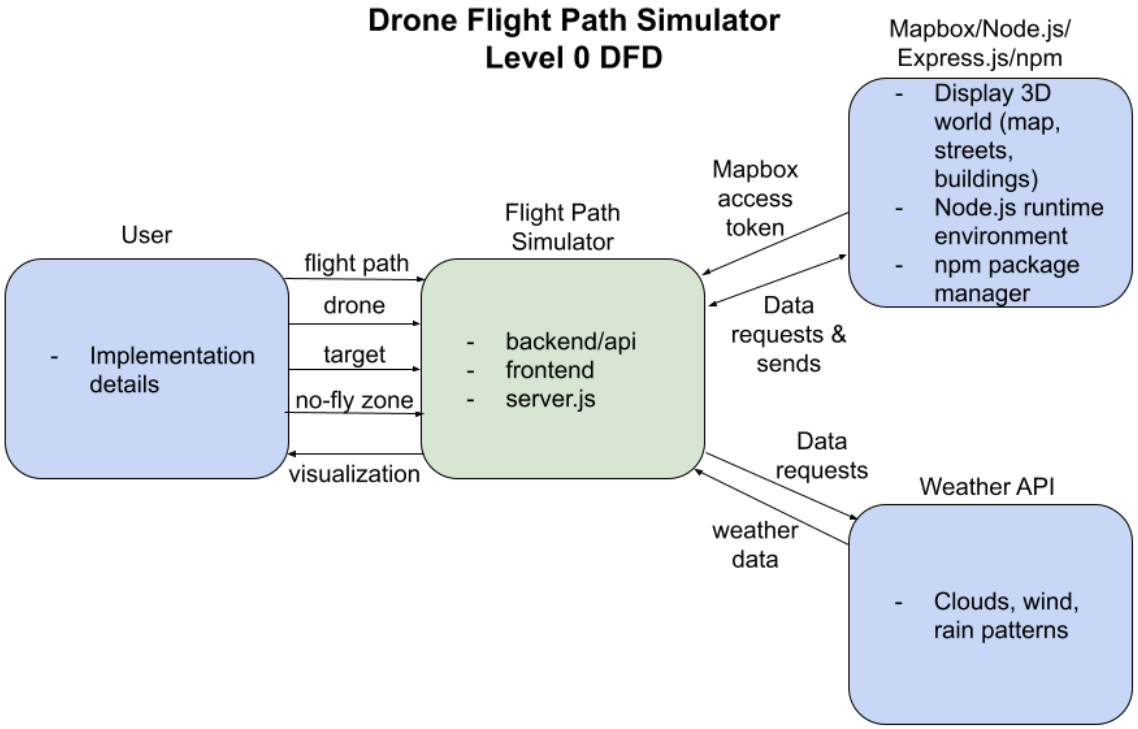
3. System Architecture

3.1- Description of System Architecture

The DFPS architecture is summarized in the Context Diagram, Data Flow Diagram (DFD) Level 0, given below. The diagram provides the overall structure of the functionalities, the responsibilities, and inputs and the outputs of the software modules.

3.2- DFD Graph Level 0

The Level 0 DFD graph and a Level 1 DFD graph were developed to visualize the program’s functionality for the system's architecture. The below shows the Level 0 DFD.



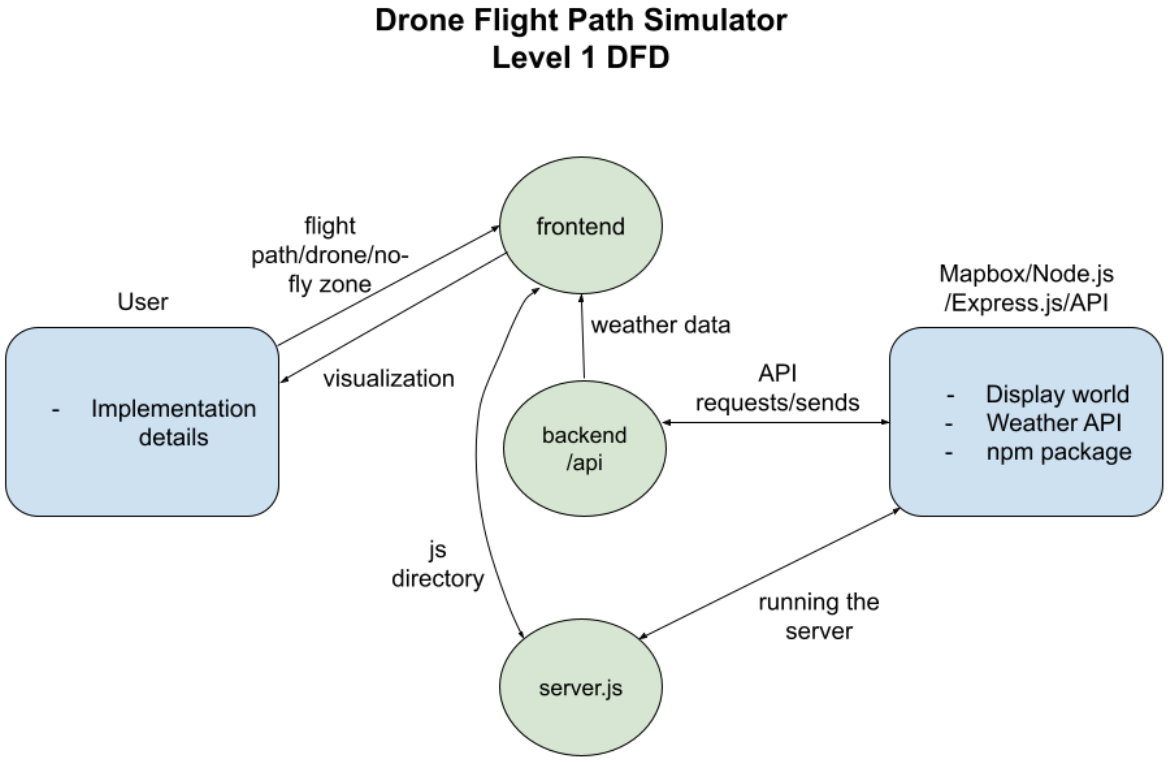
The Level 0 DFD graph provides a high-level overview of the functionalities and responsibilities of the system architecture. Its purpose is to understand the system's design and how the individual parts work together to provide its functionalities. This DFD has four main parts: the user, the flight path simulator, and the two external sources: Mapbox/Node.js/Express.js/npm and Weather API.

- User: The user is the individual who engages with the system. They perform implementation details and interact with the system as needed with the necessary input.
    - Such implementations could be setting the flight paths, choosing which drone to use in the program, and creating a no-fly zone if necessary.
- Flight Path Simulator: The central part of the system. It includes the backend/APIs, the frontend, and server.js. These subsystems contain the code needed for the entire system architecture to work.

- Mapbox/Node.js/Express.js/npm: These are the external resources that assist with running the system's architecture. Mapbox provides the world map, the "node.js" and "express.js" help run the server and other necessary processes for the user. The npm package manager assists with booting up the system.
- Weather API: The weather API helps with receiving data requests from the Flight Path Simulator and sending weather data to the Flight Path Simulator.

3.3 - Workflow

The workflow consists of the Level 1 DFD for the Drone Flight Path Simulator, which is shown below.



3.4 - System Development/Implementation

- Software/Algorithm/System Development and Implementation.
    - Describe implementation steps and details.
    - Describe the algorithms implemented.

The program is split into two parts to allow both the user and the program to be as efficient as possible: The Satellite Startup and the Planning Phase

3.4.1: Satellite Startup

During this phase, you view the map in 2D from above, then transition to a 3D view when you move from the 2D to the 3D view. Here, the user could choose a starting point anywhere in the world to determine where the user would like to plan a reconnaissance mission with the drones. Following their location, selection, and confirmation of an area, the user will proceed to the program's planning step.

3.4.2: Planning Phase

Using Mapbox, Three.js, and other backend files/applications, the user is greeted with the project's Planning Phase. Here, the user can use the sidebar to select a drone of their choice, create a drone flight path, create objects like no-fly zones and detectors, and control the momentum of the drone on the flight path by pausing. Additionally, the sidebar displays essential info for the user, ranging from how much mileage the selected drone has left, the total distance the drone has traveled when the drone gets detected through raycasting and collisions, and the total distance the drone has traveled thus far.

4. Results and Conclusions

4.1 - Description of Results

Results

- Web Application: A complete working web application created with Node.js, HTML5, CSS, and JavaScript.

- User Interface (UI): Because of the user-friendly design of the user interface, operators may readily engage with the visualization architecture. Features like interactive controls for Zoom, Compass, and map rotation data would be included.
- Drone Flight Path Visualization: The project is planning drone flight paths, and creating an intuitive way to allow the user to plan and visualize movement is important. Animating the drone along the line, not only does it create a way for the drone to interact with objects such as detectors as the flight is ongoing, but it also allows the user to more intuitively see what is happening during the flight. Features like the distance traveled and distance remaining inform the user of relevant mission parameters. In addition, doing all of this in a clear and easy-to-read way is especially important given the potential and often impeded mental states that operators in the field might be dealing with.
- Ray Casting: Allows drone operators to plan, visualize, & adjust flight paths while accounting for obstacles, terrain, and sensor capabilities. Allows objects to interact with each other and “see” each other or block vision. It’s implemented to act as a sightline for entities like drones & spotters.

4.2 - Successes and Failures

Successes:

- Our group was able to successfully implement the foundations of a simulation of a drone doing observations on a map. We also feel like we left a good foundation for future teams to build upon.

Failures:

- Our group was unable to build on the SAUS development team's efforts from the prior year. Despite our best efforts to go through the documentation to follow their installation process, the team needed help to launch the project correctly, forcing us to salvage, recycle, and create a new project from the ground up for the Army Research Lab.

4.3 - Future Research and Development

- Debugging of Current Functions
- Refinement of the 3D objects
- Integrating a drone database
- Upgrading the UI to include new features
- The operator can adjust parts of the planned path if they feel like the drone has been detected too much
- Option to upload pre-made paths
- Drone battery life

5. References

https://github.com/jscastro76/threebox/blob/master/examples/18-extrusions.html

https://docs.mapbox.com/mapbox-gl-js/example/measure/

https://docs.mapbox.com/mapbox-tiling-service/examples/building-footprints/

https://github.com/cambecc/earth

https://github.com/maptalks/maptalks.routeplayer/

https://discourse.threejs.org/t/limiting-raycast-on-multiple-objects/27593

https://threejs.org/docs/index.html?q=sphere#api/en/geometries/SphereGeometry

https://github.com/gkjohnson/three-mesh-bvh