Software Design Document
for
Drone Flight Path Simulator

Version 1.2 approved

Prepared by Alex Alcazar, Francisco Brito, Helen Dam, Alex Gaeta, Alberto Gonzalez, Sergio Maradiaga, Thaddeus Owens, Mychal Salgado, Kevin Tang, Sergio Valadez

DEVCOM Army Research Laboratory

1. Introduction.....7
1.1 Purpose.....7
1.2 Document Conventions.....7
1.3 Intended Audience and Reading Suggestions.....7
1.4 System Overview.....8
2. Design Considerations.....9
2.1 Assumptions and Dependencies.....9
2.3 Goals and Guidelines.....9
2.4 Development Methods.....10
3. Architectural Strategies.....11
3.1 Use of Particular Products.....11
4. System Architecture.....12
4.1 DFD Level 0 and 2.....12
4.2 System Development/Implementation.....13
5. Policies and Tactics.....15
5.1 Choice of which specific products used.....15
5.2 Plans for ensuring requirements traceability.....15
5.3 Plans for testing the software.....16
5.4 Engineering Trade-Offs.....16
5.5 Coding guidelines and conventions.....17
5.6 Choice of Algorithms.....17
5.7 Maintaining the software.....18
5.8 Interfaces.....18
5.9 Hierarchical Organization.....18
5.10 System Deliverables.....18
6. Detailed System Design.....20
6.1 Flight Path Simulator.....20
6.1.1 Responsibilities.....20
6.1.2 Constraints.....21
6.1.3 Composition.....21
6.1.4 Uses/Interactions.....21
6.1.5 Resources.....22
6.1.6 Interface/Exports.....22
6.2 3D Mapping with Mapbox & JS Libraries/Packages/Frameworks.....22
6.2.1 Responsibilities.....22
6.2.2 Constraints.....22
6.2.3 Composition.....23
6.2.4 Uses/Interactions.....23

6.2.5 Resources....23
6.2.6 Interface/Exports....23
7. Detailed Lower level Component Design....24
7.1 Frontend: geojson files, JS files, styles, textures, HTML files, Licensing text....24
7.1.1 Classification....24
7.1.2 Processing Narrative (PSPEC)....24
7.1.3 Interface Description....24
7.1.4 Processing Detail....24
7.1.4.1 Design Class Hierarchy....24
7.1.4.2 Restrictions/Limitations....24
7.1.4.3 Performance Issues....25
7.1.4.4 Design Constraints....25
7.1.4.5 Processing Detail For Each Operation....25
7.2 Backend: Weather API/Weather.js....25
7.2.1 Classification....25
7.2.2 Processing Narrative (PSPEC)....25
7.2.3 Interface Description....25
7.2.4 Processing Detail....25
7.2.4.1 Design Class Hierarchy....26
7.2.4.2 Restrictions/Limitations....26
7.2.4.3 Performance Issues....26
7.2.4.4 Design Constraints....26
7.2.4.5 Processing Detail For Each Operation....26
8. Database Design....27
9. User Interface....28
9.1 Overview of User Interface....28
9.2 Screen Frameworks or Images....29
10. Requirements Validation and Verification....32
11. Glossary....33
12. References....35

Revision History

Name	Date	Reason For Changes	Version
First Draft	12/05/2023	Initial Draft of Document	1.0
Final First Draft	5/01/2024	Final First Draft of Document	1.1
Final Draft	5/10/2024	Final Last Draft of Document	1.2

1. Introduction

1.1 Purpose

The purpose of this software is to assist military mission planners in pre-planning and optimizing drone flight paths for stealth operations in preparation for future missions. The software utilizes several sensor properties such as range of sound and detection ranges within an open map to allow mission planners to navigate freely throughout the environment for an accurate location to pre-plan for future flight paths provided with a drone. Information will be displayed when the user interacts with the software such as the flight path distance, the current location based on the mouse inputs on the map, and more. With this software, the mission planners will be able to pre-plan flight paths as well as obtain data on basic information of the path generated by the user, while being on the lookout for enemy units and detectors that may be within range of the assigned path.

1.2 Document Conventions

- Regular Text - Times New Roman, 12 pt font
- Bold Text - Title and Section Headings
- Numbered lists - Ordered Lists
- Bulleted Lists - Unordered Lists

1.3 Intended Audience and Reading Suggestions

This document is intended to guide D.F.P.S.'s software development developers based on the D.F.P.S. software requirements document. The S.D.D. will act as a baseline for future developers to expand the project if desired.

1.4 System Overview

The D.F.P.S. project will use a web framework to build geographic data supporting a spatial database. Using this database, a user can manipulate spatial indexing, geometry fields, and geographical query sets. We will have front-end map rendering to provide interactive map components. With these functionalities, we can implement a software program that will analyze a given drone flight path and provide feedback, and a mission success rate.

2. Design Considerations

2.1 Assumptions and Dependencies

- Accurate environmental data is available for simulation purposes
    - If not available, allow the User to input certain implemented missing data inquiries
- Users possess the requisite knowledge to operate the simulation software
    - Users are provided a guide to install the necessary packages to run the software
- The environment remains static to match the user's entered data
- Flight path may not change dynamically due to it depending on user data
- Users are assumed to have the latest version of browsers
- Users are assumed to have an internet connection to download & operate the software
- Users are recommended to have Visual Studio Code pre-installed to operate, develop, and debug the software
- Users may need to create their own Mapbox account & implement it into the code, in case, the current Mapbox access account is no longer functioning.

2.2 General Constraints

- Can't predict how the opposing forces will move while creating the simulation
- Can't rapidly adjust the flight path whenever a detection sensor changes focus
- Can't route more than one flight path when operating the software
- Limited data from the SUAS sensors
- Limited data for environmental factors
- Limited drone selection provided

2.3 Goals and Guidelines

- User-Friendly Interface: A friendly UI is crucial for seamless interaction with drone flight planning software, enhancing efficiency, reducing learning curves, especially for small military units, and ensuring accessibility to a wide range of users.

- Ability to Adjust to Terrain & Environment: The software should enable operators to adjust drone flight paths in dynamic military operations swiftly, ensuring mission effectiveness despite unexpected challenges.
- Ensure that the architecture for 3D visualization considers legal requirements and ethical issues.
- Drone Distance Ranges: Given selection of certain drone types, enables operators to determine which drone would be more efficient for the upcoming mission as well as to determine the efficient distance that the drone would be able to travel. This allows for extended missions, prolonged surveillance, reconnaissance, and data collection, aligning to utilize sensors effectively.

2.4 Development Methods

Agile development is perfect for dynamic projects with changing demands because it emphasizes prioritizing feedback, working closely with liaisons, and being flexible.

- GitHub repository & branches for testing, development, and delivering necessary functions
- Straightforward & simple installation of packages to run the software
- Different teams focus on separate functions to maximize workflow along with weekly meetings to update the team on progress along with updates from liaisons for feedback to meet requirements. This allowed members to help one another freely and to make sure everyone was up to date on the overall progress of the functions being implemented.

3. Architectural Strategies

The implementation of the services provided below is necessary to develop the overall software starting with a foundation to allow for future developments & of the system expansion.

3.1 Use of Particular Products

3.1.1 Mapping Environment: Mapbox was used to give users a 2.5D visualization of their flight path and an open street map. Mapbox GL JS, a client-side JS library, was also used to help build our web map using modern mapping technology. GeoJSON was also used to help represent and display map data.

3.1.2 Mapbox APIs: Used to access Mapbox tools & services

3.1.3 JavaScript (JS), HTML, & CSS: Programming languages that were used to build the foundation for a web-based application.

3.1.4 Graphical User Interface: The Map will take up most of the application view with a user interface(UI) that can be expanded and minimized.

3.1.5 Node.js & npm: JS runtime environment and package manager that utilizes Express, a web application framework, to help build the application into components and offer better scalability for future endeavors.

3.1.6 Three.js: Cross-browser JS library used to create and display 3D objects and detectors in our software. One of the most important functions implemented in this software is the Raycasting used for detection and being able to manipulate objects along with it.

3.1.6.1 Threebox.js: Three.js Plug-in for Mapbox GL JS to manage objects within the longitude & latitude coordinates on the map, and used to synchronize the map and scene cameras.

3.1.7 Turf.js: Open-source JS library for spatial analysis for browsers, used specifically with Mapbox APIs to help draw the flight path for the drone.

4. System Architecture

4.1 DFD Level 0 and 1

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.

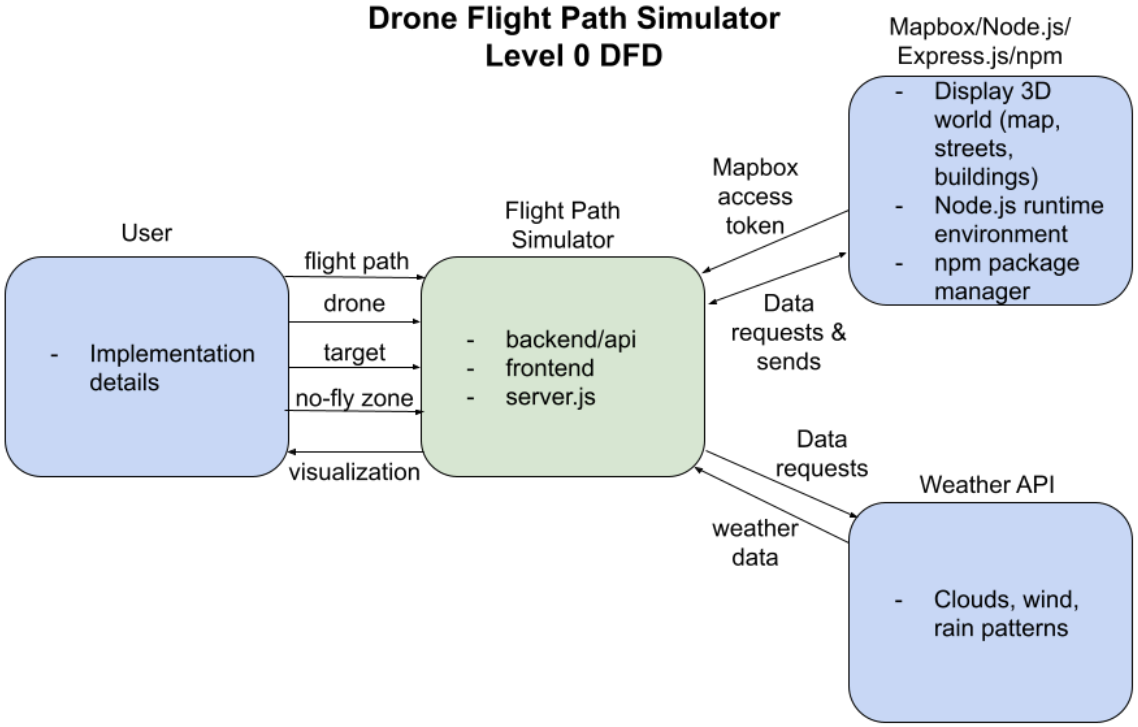


Figure 1

Four Main Functionalities:

- User: A person who operates and interacts with the system as needed for inputs.
    - Inputs can include settings points for the path, mouse input for the current location on the map, drone selection of choice, no-fly zone layer, and button functions.
- Flight Path Simulator: Central part of the system. Includes the necessities of the application needed for the entire architecture to function correctly.
- Weather API: API that helps with receiving data requests & sending weather data

- Mapbox & Other JS Libraries: External resources that assist with running the system's architecture. Mapbox provides the main world map. The other JS libraries such as the runtime environments are to help start and properly run the server. The package manager assists with booting the overall system.

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

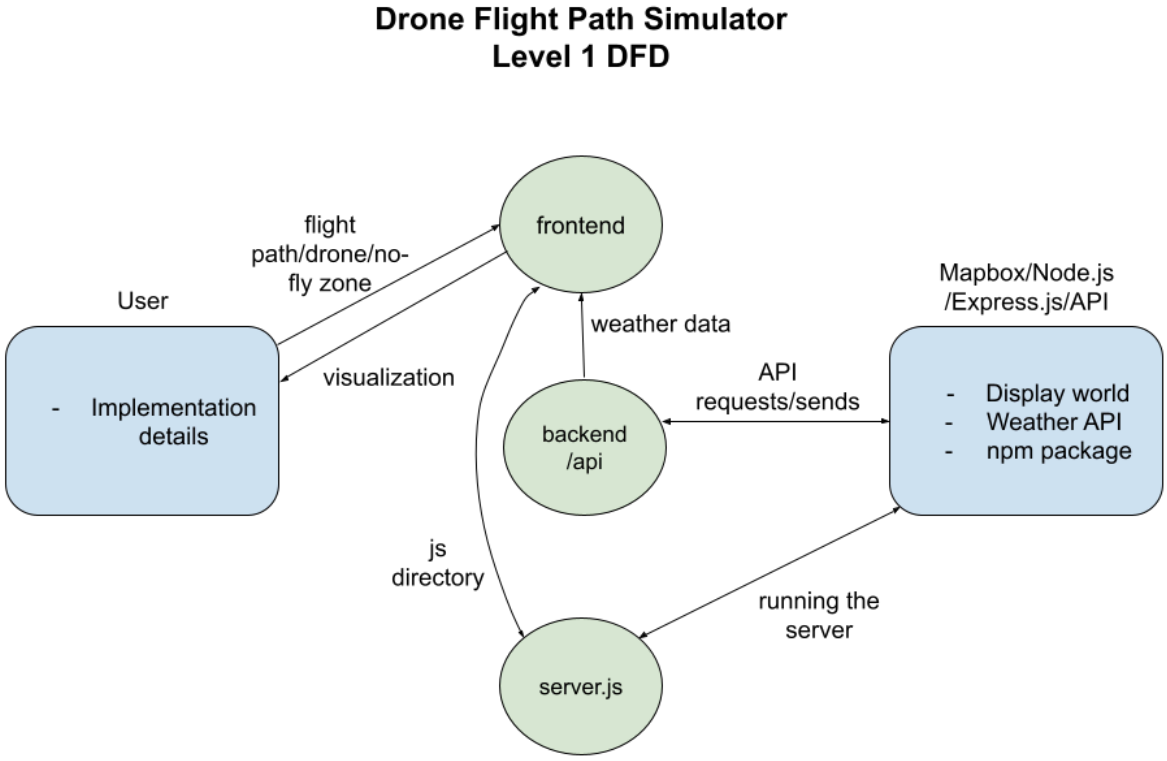


Figure 2

4.2 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

- 4.2.1: 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.
- 4.2.2: 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.

5. Policies and Tactics

5.1 Choice of which specific products used

- Javascript (JS)
    - npm: Default package manager for JS Runtime Environment
    - Node.js: Open-source JS Runtime Environment
    - Turf.js: JS Library for Advanced Geospatial Analysis
    - Three.js: JS Library for 3D objects for web browsers
        - Threebox.js: Three.js plugin for Mapbox GL JS
- GitHub: Developer Platform
- Visual Studio Code (VS Code): Source-code Editor by Microsoft
- CSS: Style Sheet Language for a document written in a markup language
- HTML: Standard Markup language for documents designed in a web browser
- MapBox
    - Mapbox APIs: To allow access to Mapbox tools & services
    - Mapbox GL JS

5.2 Plans for ensuring requirements traceability

Our meetings with liaisons and contacts have been recorded to ensure all requirements can be traced to their desired specifications. In addition, all user stories and similar planning steps will be maintained to show how each condition is tied to a user's need. All requirements are up to date and contain everything needed to follow along with the project's progress

5.3 Plans for testing the software

- Implement Continuous Implementation using a platform like GitHub repositories & branches to test and ensure code quality before merging into the main branch.
- Because the architecture is relatively new, being built from scratch by this year's (Spring 2024) team, testing will be done within the following groups (Note that it is not open to the public as of May 2024):
    - Internal Portions of the Team
    - Liaisons
    - Contacts of Liaisons
    - Advisors
- As described in Section 2.4 Development Methods, the Agile Software Development method is used continuously to test, develop, and debug each part of the development stage.

5.4 Engineering Trade-Offs

- Performance vs Complexity:
    - Goal: High performance and accuracy in drone flight path optimization
    - Approach: Implementing efficient algorithms that optimize flight paths without excessive computational overhead. Some algorithms that have been considered but have yet to be implemented include A* and Dijkstra's for pathfinding.
- Hardware Limitations:
    - Challenge: Lack of access to drone instrumentation readings.
    - Solution: Create a robust web app with plugin-based architecture to interface with drones through standardized APIs, assuming primary telemetry data. The system should be designed to adapt, implement, and integrate more data as it becomes available.
- Accuracy vs. Computation Time:
    - Priority: High accuracy in flight path optimization

    - Balancing Act: We will consider leveraging cloud computing resources to handle complex computations without compromising response time. Current considerations include AWS Lambda or Azure Functions.

- Safety and Compliance:
    - Consideration: Ensure the flight paths comply with local aviation regulations and safety standards
    - Implementation: Integrate an existing simple map projection known as an equirectangular project that allows the showcasing of no-fly zones along with a toggleable feature to also remove the no-fly zones as well.

- Scalability:
    - Objective: Create a scalable solution.
    - Strategy: Implementing a plug-in-based architecture, allowing easy scaling and integration of new features or data sources in the future.

5.5 Coding guidelines and conventions

An essential aspect of our design is ensuring readability since future teams will likely build upon this project. Extensive use of descriptive but concise comments will be implemented throughout the code. All relevant files will also be kept in an organized manner so that each can be found easily. In addition, all general code formatting conventions will be followed in a way that will be consistent across all code. This includes but is not limited to using CamelCase for naming proper indentation, and descriptive error handling.

5.6 Choice of Algorithms

The algorithms involved with the software are the spatial analysis and the Mapbox tilesets to display the dynamic visualizations of the map such as elevation data, the overall satellite map view, the generation of the 3D terrain mesh of the map, the terrain slopes of mountains, hills, and other terrains, and the map style.

5.7 Maintaining the software

All code will be claimed via GitHub, and team members' contact information will be provided for future developers and users. Extensive documentation will also be held. Utilizing the latest and most stable versions of software for long-standing support.

5.8 Interfaces

The use of a browser is necessary for the software to display all information that will take place based on the user's interactions within the web application. Throughout the development phase of the software, testing the UI to be clear and straightforward to follow at all times along with being synchronized with the map interactions is important to maintain one of the requirements of the project. The user interface will be applied with CSS and HTML styles to be displayed properly within the web application, this will allow for easier future expansions to the UI, as well as simple transitions to another platform with a relaxed type of style.

5.9 Hierarchical Organization

The team was able to get feedback from a senior software developer during the development phase of the project’s file organization. The team considered the feedback and implemented changes to balance the readability along with the encapsulation of the files.

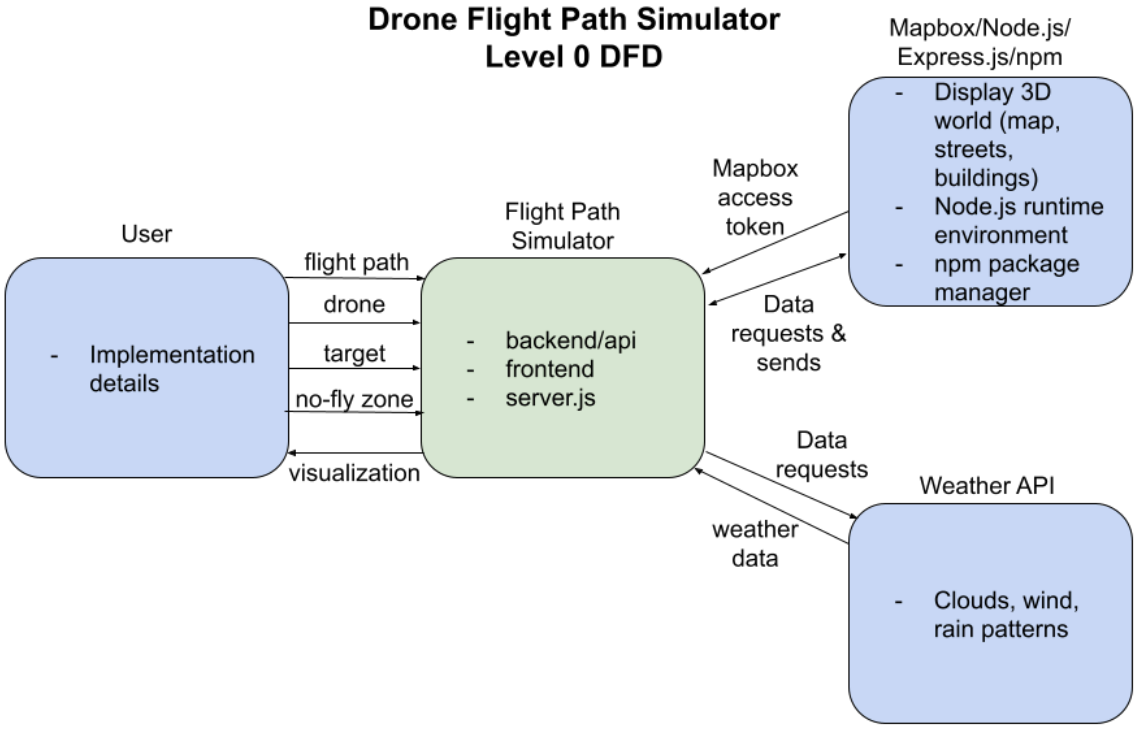
5.10 System Deliverables

The team provided a README file along with the code of the program, that allows any user that has access to the GitHub repository to follow along and be able to run the software properly step-by-step.

If the project is not accessed by cloning the GitHub repository, then downloading the zip file of the project works as well. After downloading the zip file, follow these steps:

1. Extract the “Drone_Flight_Path_Simulator-development” file.
2. Right-click the extracted file and open it in Visual Studio Code.
3. Open a terminal in VS Code and ensure that the file is in the extracted file directory.
4. Then run the following command: ‘npm install’
5. After all, packages are installed from #4 and there are no issues, to start the server run the command: ‘npm start’.
6. After the server starts showing ‘Server is listening on port 3000’, open the web browser of choice, and navigate to the local host link: http://localhost:3000 for the main software or http://localhost:3000/areaRecon.html for the visualization of the no-fly zone.
7. To stop the server, go back to VS Code and use keyboard commands in the terminal of CTRL and c. Once asked to ‘Terminate batch job (Y/N)?’, enter y for yes otherwise n for no.

6. Detailed System Design



6.1 Flight Path Simulator

6.1.1 Responsibilities

The Flight Path Simulator's frontend module will be responsible for all the necessary Javascript functionalities that are reused within the local file to balance the readability. The frontend module also relates to the browser and anything related to manipulating the document object model (DOM) and user events within the browser.

6.1.2 Constraints

One of the constraints of the software is that each connection to the server will only allow the user to create one flight path at a time. To reset and map out another pathway, the user would have to either restart the server or refresh the web application. The web application and local host page should be displayed properly as long as there are no issue with the modules or the services used within this project.

6.1.3 Composition

The front-end module will include all the functions to load the visualizations of the DFPS as well as display objects such as drones, 3D buildings, the detection range of the drone, the drone flight pathway, enemy units of detectors, and other functions included within the software. The module also includes other functions such as the user interface display, the CSS document, the standard file format for the three-dimensional scenes and models (TfL), the textures of the drone/portable network graphic (PNG) files, and the licensing of the drone model used. The main module will also provide the overall final the visual architecture of the world map.

6.1.4 Uses/Interactions

The front-end module will handle most, if not all, of the mouse events provided by the user within the web application, it will also handle the user inputs taken for flight altitude during the process of setting the pathway. The uses will also include the detector ranges around the drone and the additional units that can be placed in the 3D environment. Additional basic controls such as the zooms and the compass are provided to be handled along with the mouse interactions. The server will try to be set up within the backend of the module, however, if not possible, it will be set up within the project scope of the overall code. The backend module will handle the weather API to obtain certain weather effects.

6.1.5 Resources

The Flight Path Simulator requires all modules to be able to display and function based on the requirements made for the project. Additional installations and factors that are included within the system are required to display the full functionality of what DFPS is.

6.1.6 Interface/Exports

Please refer to the source code for all available interfaces/ exports.

6.2 3D Mapping with Mapbox & JS Libraries/Packages/Frameworks

6.2.1 Responsibilities

The 3D environment will be provided by Mapbox using Mapbox APIs to utilize their services and tools of an open street map. The architecture will also include plug-ins for Mapbox, Javascript libraries, packages, and frameworks to include event functions such as longitude and latitude coordinates, formulas to calculate distances of the path, elevation provided by certain services, 3D toggleable buildings, and runtime environments to handle packages necessary for DFPS. There is also another 2D map that displays the no-fly zones along with an optional toggleable layer to be rid of the no-fly zones as well as styles to display the 2D map.

6.2.2 Constraints

The constraint of the software to handle objects is that there will only be one type of 3D drone object visualization will be the only default drone object available. Decent hardware may be necessary to run the software as loading the 3D objects can take a while if not loaded properly. Another constraint is that possibly the no-fly zones may be only displayed within the 2D map and not the 3D map within the main software.

6.2.3 Composition

The 3D mapping with Mapbox falls along with the frontend module and the final display of the 3D world map.

6.2.4 Uses/Interactions

The 3D map interactions and uses will be passed within the main control module, which in this project would be the front-end modules, this includes the user input of flight altitude during the mapping out of the pathway, and the mouse interactions along the 3D map such as the coordinates and elevation of the selected location.

6.2.5 Resources

As mentioned along with 3D mapping, Mapbox is the main web application for 3D map the model used in this software as well as Mapbox APIs to utilize the services and tools including threebox.js, a Mapbox plug-in, to assist the other necessary functionalities within the main requirements provided.

6.2.6 Interface/Exports

Please refer to the source code for all available interfaces/ exports.

7. Detailed Lower level Component Design

7.1 Frontend: geojson files, JS files, styles, textures, HTML files, Licensing text

7.1.1 Classification

The kind of components range differently depending on the file. The classifications include text documents, packages, classes, functions, files, subsystems, etc.

7.1.2 Processing Narrative (PSPEC)

Encapsulation of the functionalities that are meant to be reused within its own local modules.

7.1.3 Interface Description

The frontend module has no interface as it contains files and documents that are used for the development of the foundation of DFPS.

7.1.4 Processing Detail

Processes all the development files that handle user interactions, mouse events, texturing of objects, licenses of the models used, as well as the functionality of the DFPS.

7.1.4.1 Design Class Hierarchy

Parent/ Main Module to handle requests.

7.1.4.2 Restrictions/Limitations

There were no restrictions or limitations.

7.1.4.3 Performance Issues

No performance issues within the final development phase.

7.1.4.4 Design Constraints

There were no design constraints.

7.1.4.5 Processing Detail For Each Operation

Processes all the operations that are functional in the project, including user inputs, mouse events, JSON text into objects, and textures of the 3D objects.

7.2 Backend: Weather API/Weather.js

7.2.1 Classification

Javascript file/ Class

7.2.2 Processing Narrative (PSPEC)

Encapsulates the functions needed to access the weather API.

7.2.3 Interface Description

The backend module has no interface as it only contains the weather API.

7.2.4 Processing Detail

Processes the weather API such as the wind speed depending on the altitude and elevation of the selected map coordinate.

7.2.4.1 Design Class Hierarchy

Parent/ Main Module to handle weather API requests.

7.2.4.2 Restrictions/Limitations

Certain weather effects cannot be displayed within the UI.

7.2.4.3 Performance Issues

No performance issues within the final development phase.

7.2.4.4 Design Constraints

The only constraint within this weather API is needing an account to access the weather API.

7.2.4.5 Processing Detail For Each Operation

Processes the operations to obtain weather information and be displayed within the UI.

8. Database Design

No databases were used in this development phase of the Drone Flight Path Simulator.

9. User Interface

9.1 Overview of User Interface

The user interface will provide users with information about the flight & map coordinate location:

- The Longitude & Latitude of the drone on any given point of the flight path
- Elevation of the drone
- Wind Speed during the planned flight
- Total distance of the planned flight path
- Remaining Drone range of the flight path
- Basic controls for map: Zoom in or out, Compass location facing North
- Toggleable button for plotting points of the pathway for drone
- Move & Pause button during flight animation
- Reset the button to remove the drone from the flight path
- Additional Detectors to pose as enemy units or detector objects
- Toggleable building generator to remove or show 3D buildings on a map
- Toggleable button to minimize UI
- Drone selector & information provided based on the user's selected drone

9.2 Screen Frameworks or Images

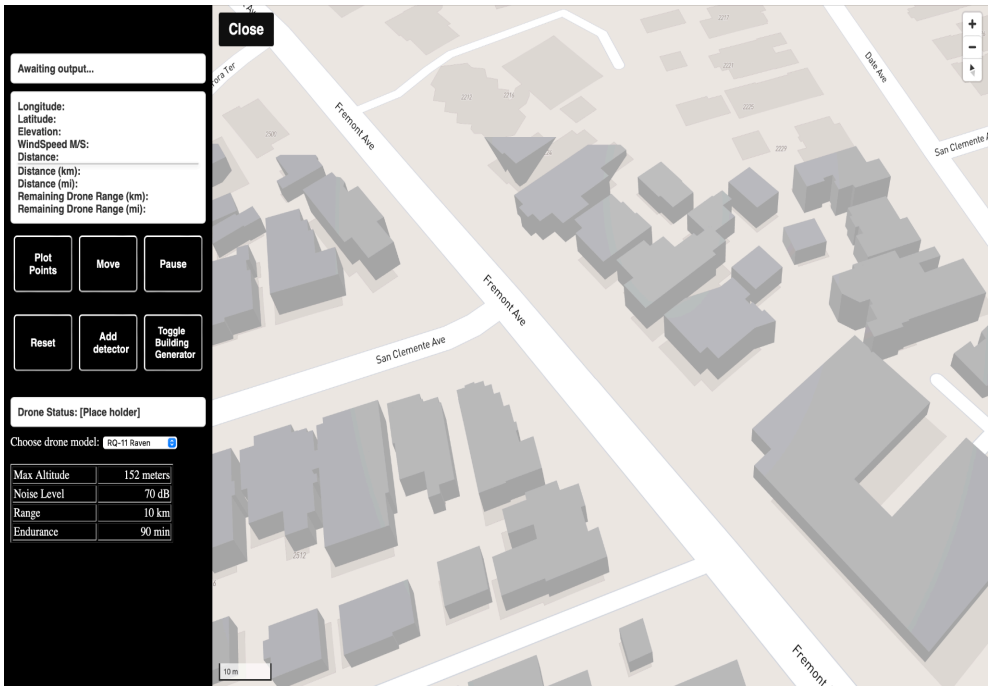


Figure 1 (pg 25): link → http://localhost:3000
The 3D visualization architecture of the Drone Flight Path Simulator.

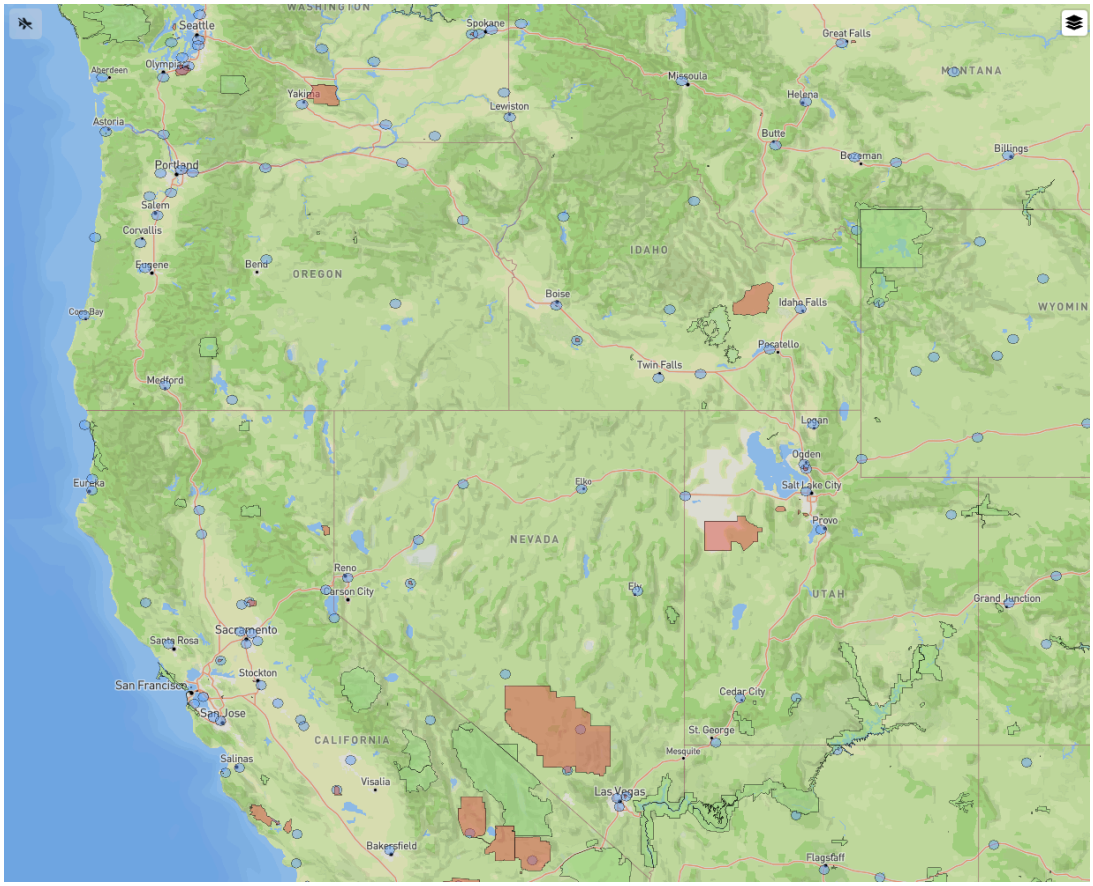


Figure 2: link → http://localhost:3000/areaRecon.html

The 2D map visualization of the no-fly zones. Red represents default no-fly zones such as airports, U.S. national parks, and U.S. military bases.

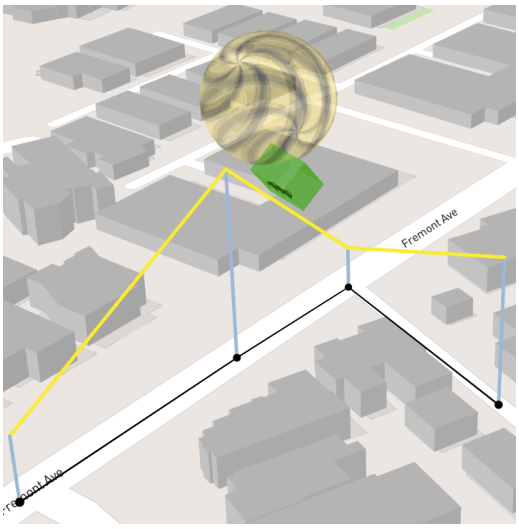


Figure 3: Flight path plotted along with altitude, drone (green box) along detector bubble around it.

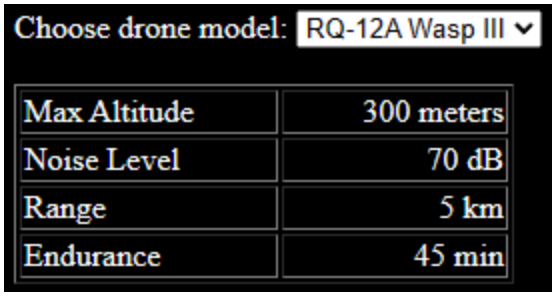


Figure 4: Drone model information based on user selection, information will be changed as the drone model changes.

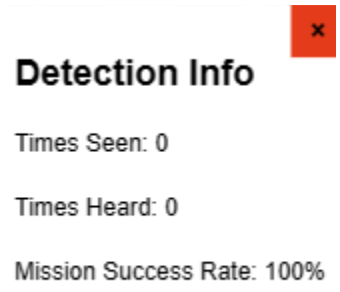


Figure 5: Detection information that is displayed when the drone is paused or finishes its flight path.

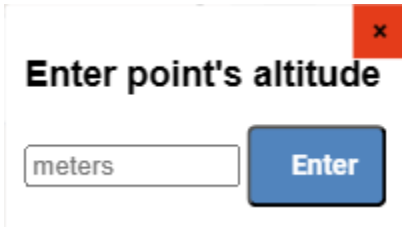


Figure 6: User input for drone altitude when plotting each point on the flight path.

10. Requirements Validation and Verification

Functional Requirements	Component Modules
D.F.P.S. shall provide a basic 3D geographic environment.	Frontend → js → main.js
D.F.P.S. shall provide map information such as map coordinates, elevation, and wind speed.	Frontend → js→ threejsHandlers & Backend/api → weather.js
D.F.P.S. shall allow users to input parameters for the altitude during the flight path.	Frontend → js → mapClickHandlers.js
D.F.P.S. shall allow users to plot endpoints as the flight pathway.	Frontend → js → mapClickHandlers.js

D.F.P.S. shall display the inputted altitude & distance information on the flight path.	Frontend → js → mapClickHandlers.js
D.F.P.S. shall provide a selection of drones & display information based on the selection.	Frontend → js → mapClickHandlers.js
D.F.P.S. shall provide a move & pause button for the drone flight path animation.	Frontend → js → threejsHandler.js
D.F.P.S. shall provide basic sensor characteristics such as audio and visual for drone detection ranges.	Frontend → js → threejsHandler.js
D.F.P.S. shall a visualization of no-fly zones within a 2D map.	Frontend → js → areaRecon.js & Frontend → areaRecon.html & Frontend → geojson
D.F.P.S. shall provide basic controls such as compass & zoom functions.	Frontend → js → main.js

11. Glossary

- **2D:** Two Dimension
- **3D:** Three Dimension
- **A.R.L.:** DEVCOM Army Research Laboratory
- **CTRL**: Control key command on the keyboard
- **D.F.P.S.:** Drone Flight Path Simulator

- **DOM:** Document Object Model
- **O.T.A.:** Over-The-Air
- **PDF:** Portable Document Format
- **SDD:** Software Design Document
- **S.U.A.S.:** Small Unmanned Aircraft System (i.e. Drone)
- **UI:** User Interface
- **Front-end**: the development of the graphical user interface of a website through the use of HTML, CSS, and JavaScript so users can view and interact with that website.
- **Back end:** build and maintain the mechanisms that process data and perform actions on websites.

12. References

https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc

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