Software Design Document

for

Small-Unit Drone Optimization (SUDO) (Ver 2.0)

Version 2.0

Prepared by Jonathan Aguirre, Jay Arias, Lloyd Castro, Jean Paul Espinosa, Peter Han, Jorge Hernandez, Hugo Izquierdo, Raymond Martinez, Bruck Negash

Army Research Lab

November 22, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First Draft	11/22/2022	Initial Draft of Document	1.0
Second Draft	3/27/2023	Second Draft of Document	1.3
Third Draft	4/15/2023	Finalizing Document	1.4
Final Draft	4/30/2023	Finalizing Document	2.0

1. Introduction

1.1 Purpose

The Software Design document provides an outline of the design and architecture for the detection models of a Small Unmanned Aerial System (SUAS) and its 3D terrain mapping. Our program is essentially a two-pronged product, using a Python program to calculate the detection distance of a SUAS from a human's perceptual capabilities based on inputs the program is given, and a Unity program to create the 3D simulation of the environment and visually represent our Python outputs by way of a SUAS operating in this environment.

1.2 Document Conventions

- Typographical Conventions - Style Usage
- 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 for developers to overview the implementation of SUDO's features and capabilities in its design and architecture. Future developers can also use this document to expand this project with the provided foundation.

1.4 System Overview

The SUDO project consists of incorporating human perceptual models into a Python program in order to calculate the distance from a human in which a SUAS (drone) is likely to be detected. WebODM assists in creating a virtual representation of the user's environment, and Unity creates a 3D object simulation of that environment. We can then use the Python program's outputs to

simulate a drone operating in the given environment, with visual cues of how it should move through the environment in order to avoid detection.

2. Design Considerations

2.1 Assumptions and Dependencies

The product assumes the following

- The user has access to a drone
- The user has knowledge of controlling and flying the drone
- The user can follow the software instructions of our README file and download the necessary dependencies
- The information and research behind the perception models provided to us are accurate

2.2 General Constraints

- Computer software requirements to run Unity and Python
- Detection model data used for our testing is as follows:
    - Urban Environment
    - ISO Listener Standard
    - Target (drone) noise profile

2.3 Goals and Guidelines

The product will digitize human perception models of sight and sound that have been amalgamated through the decades by the Army Research Lab, and transform those models into a dynamic, easy-to-use, and modern program. We will then be using this newly-developed program to create a simulation in a 3D environment using the Unity Engine to determine the statistical probabilities of a Small Unmanned Aerial System (SUAS) being detected by enemy troops at inputted distances.

2.4 Development Methods

Agile Development Approach

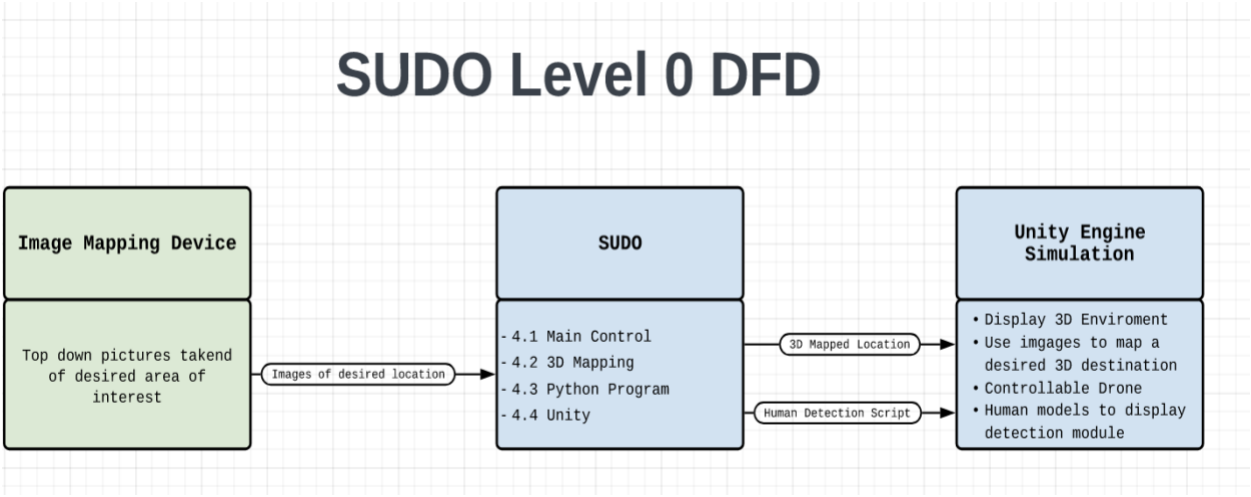
- Our team meets with each other and the liaisons every two weeks to discuss the progress of the project's development.
- Team members dedicated to utilizing Unity to create environment and detection models, and Jupyter Notebook to analyze the acoustic detection model data provided by the liaisons.

3. Architectural Strategies

- **3D Environment**
    - Environmental and game engine known as Unity will be used in order to simulate necessary environments.
    - Unity will then use images taken of desired locations by the user to render an interactive 3D sandbox environment
    - Allows testing of drone flight paths and area traversal in a simulated environment
    - Drone detection can be estimated and represented visually in Unity
    - Using Unity to create our 3D environment will be helpful in that various scripts and algorithms can be used for visual representation and general simulation
- **Mapping Environment**
    - Using a software called WebODM where it will generate 3D models and height maps from images
    - It will use ground control point mapping for increased map creation accuracy when rendering models for Unity
    - Compatible with exporting models to Unity for simulated environment creation
    - Simulated representation of environment will allow for realistic testing without the risk

- **Python Program**
    - Use python libraries and Jupyter notebook to test datasets to make accurate prediction of detection ranges
    - Allows us to easily create scalable data-frames to maintain future flexibility in adding or removing complexities in regards to certain detection parameters.
    - Uses inputs from Unity to create a JSON file which is then converted into a dictionary of key/value pairs. The Python program then creates another JSON file with the calculated outputs, and sends them back to Unity in order to create a visual representation

4. System Architecture



4.1 Main Control

- Refer to DFD Level 0 or Level 1. Detailed description available at **Section 6.1**

4.2 3D Mapping

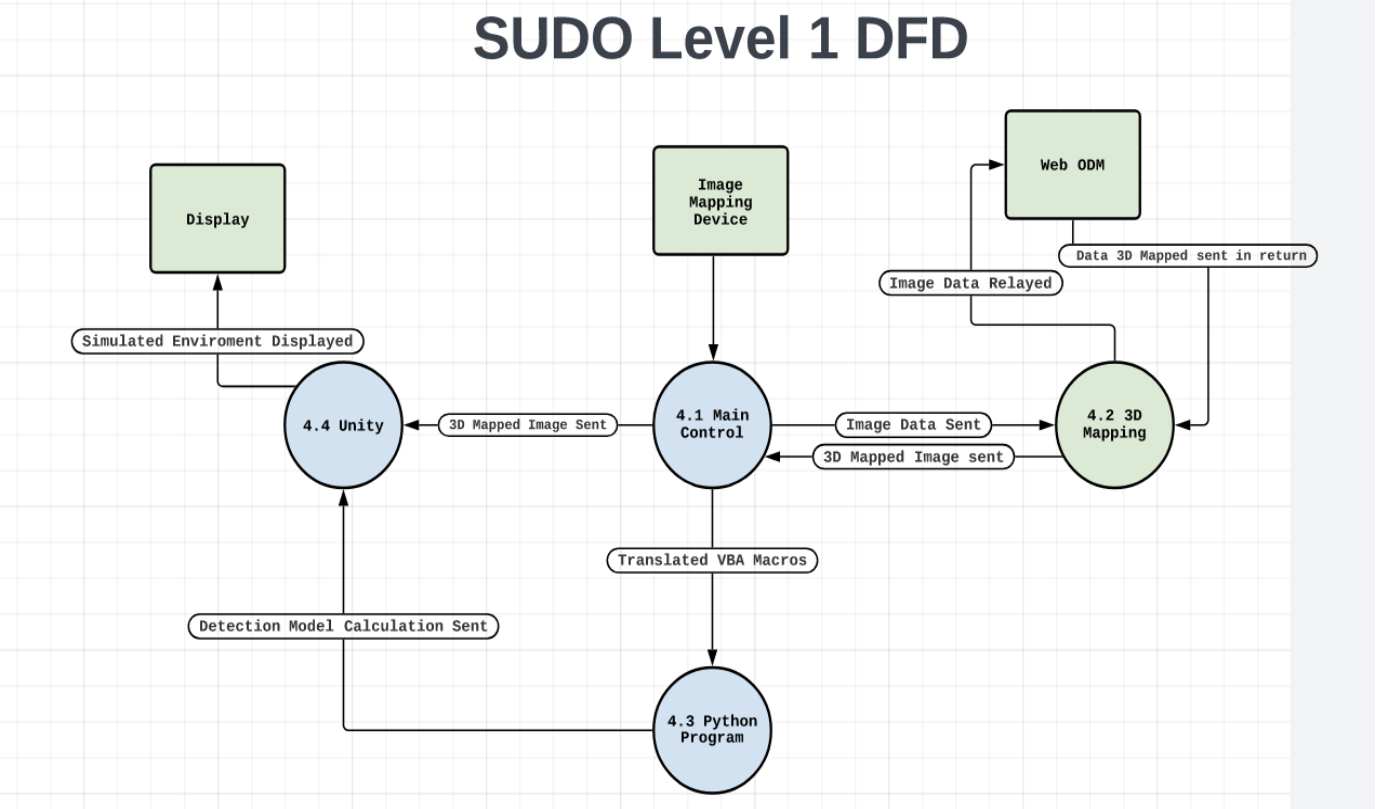
- Refer to DFD Level 0 or Level 1. Detailed description available at **Section 6.2**

4.3 Python Program

- Refer to DFD Level 0 or Level 1. Detailed description available at **Section 6.4**

4.4 Unity

- Refer to DFD Level 0 or Level 1. Detailed description available at **Section 6.5**


SUDO Level 1 DFD
Display
Image Mapping Device
Web ODM
Data 3D Mapped sent in return
Image Data Relayed
Simulated Enviroment Displayed
4.4 Unity
3D Mapped Image Sent
4.1 Main Control
Image Data Sent
3D Mapped Image sent
4.2 3D Mapping
Translated VBA Macros
Detection Model Calculation Sent
4.3 Python Program

5. Development Policies and Tactics

5.1 Specific Products and Development Environments Used

- Unity
- WebODM
- Python
- Software Tools:
    - VSCode
    - PyCharm
    - Github
    - Unity
    - WebODM
- Testing and Debugging tools:
    - TestCase from Unittest
- Libraries and Frameworks:
    - Python Packages:
        -
    - Unity Packages:
        - TextMeshPro
        - Unity UI
        - System.Collections
        - System.Collections.Generic
        - Unity.VisualScripting
        - TMPro
        - UnityEngine
        - UnityEngine.UI
        - UnityEngine.SceneManagement
        - System.IO

5.2 Plans for Ensuring Requirements Traceability

- To ensure that the requirements for this project are documented and reviewed, meetings are recorded on Zoom, meeting notes and action items are written in word documents and the requirements are documented in the Software Requirements document and the Software Design document.

5.3 Plans for Testing the Software

- Test software with variables that have known results from original VBA Macros
- Test the parameters with the Python program and their effect on the detection simulation in the Unity program.
- Test decibels of an object or drone in an Urban environment
    - Try and match data given to us from previous tests in an Urban environment
    - Alter certain algorithms depending on similarity to previously collected data

6. Detailed System Design

6.1 Main Control

6.1.1 Responsibilities

The Main Control module will be responsible for securely combining all the data that is necessary for each file transfer and visual display. It then securely presents the collective information from all the modules to the user.

6.1.2 Constraints

It will display everything as long as one module does not cause a major issue, specifically the Unity module.

6.1.3 Composition

The Main Control module falls between the images of the specified area and the final simulated environment.

6.1.4 Uses/Interactions

The Main Control module will be used to interpret the user-provided images of the specified area and start to distribute it towards the 3D Mapping Module. Then it will take the 3D model from the Mapping module and save it for the Unity module. After that process, the 3D model is sent to Unity in order to create the sandbox environment and allow for modification of said environment via user chosen values which will then be sent to the Python Program module. The Python Program module will run these values through several methods and formulas specifically made for simulation and will return outputs related to object detection. The user will then be able to simulate drone detection and other environmental factors that may be present in a close to realistic sense based on the perception models.

6.1.5 Resources

The Main Control module will require every other module as mentioned before, including the raw user images of the location.

6.2 3D Mapping

6.2.1 Responsibilities

The 3D Mapping module will be responsible for creating a 3D environment based on the images the user sends as well as coordinates from the general area covered using ODM/WebODM.

6.2.2 Constraints

In order to have accurate 3D models, the images obtained by the user need to be clear, have a general overlap between each other, and good hardware may be necessary for creation.

6.2.3 Composition

The 3D Mapping module falls in between the Drone Camera and the final display of the 3D simulated environment.

6.2.4 Uses/Interactions

The 3D Mapping module will collect the data sent by the Main Control module and build an environment based on it. After the 3D model is made, it will then be sent back to the Main Control module so it can then be imported into Unity for testing and general sandbox purposes.

6.2.5 Resources

As mentioned before, ODM/WebODM will be used to create the 3D model. Ground Control Points based on the images can be used and will be made with the help of OpenStreetMap which is built into it. This will create accurate models based on the location of said images.

6.3 Python Program

6.3.1 Responsibilities

6.3.1.1 The Python program module is responsible for processing auditory input parameters and generating accurate predictions of a person's ability to detect a target Small Unmanned Aircraft System (SUAS) based on those parameters. The program outputs a range of detectability "zones" based on possible hearing-quality thresholds of the person. These ranges are visually represented within the Unity Engine to show the maximum distance that an SUAS can approach a target before being detected.

6.3.2 Constraints

6.3.2.1 The current data inputs for the program are sourced from the Army Research Lab's "ADM_from_Joel" Excel sheet. While the program works with these inputs to generate detectability zones for visualization, it assumes the accuracy, mathematics, and physics of the provided data are correct.

6.3.2.2 The program's reliance on a dictionary for data inputs could be considered a constraint as it limits the flexibility and scalability of the program in handling other data formats. If the program needs to handle inputs in a different format, changes, or updates to the code may be required to accommodate the new format. Therefore, it is important to carefully consider the data inputs the program will be handling and ensure that the chosen data structure can accommodate future changes or expansions.

6.3.3 Composition

6.3.3.1 Auditory Detection Model (ADM)

6.3.3.2 Drone Mapping Software (WebODM)

6.3.4 Uses/Interactions

The components are utilized for retrieving and interpreting data from WebODM, retrieving data from the ADM Excel sheet, using ADM data to generate detectability ranges, and sending those ranges to the Unity Engine for visualization.

6.4 Unity Engine

6.4.1 Responsibilities

The main responsibility of the Unity module is to test our detection models in a simulated environment.

6.4.2 Constraints

The amount of time it may take to show the rendered environments may depend on the hardware of the devices being used to interact with the software.

6.4.3 Composition

Unity will fall in between what the user processes in terms of models, images, and data as well as the final representation of the environment captured.

6.4.4 Uses/Interactions

The Unity module will essentially receive 3D rendering data from the 3D Mapping module and visually display that in the Unity engine itself. Then this will allow for movement and modification of the created environment for testing purposes which will act as a sandbox for users. Combined with the Python Program module, data will be reflected as closely as possible through the data given via user input and received from the related script outputs in a visual representation.

6.4.5 Resources

Unity will require the 3D mapped data from the 3D Mapping module in order to display everything required.

7. Detailed Lower Level Component Design

7.1 Unity Engine Module

7.1.1 Classification

Unity is an engine and package of libraries of code to help allow us to visualize 3D objects. Also can help to simulate the physics of the environment.

7.1.2 Processing Narrative (PSPEC)

The Unity engine will receive images from the user wherein WebODM will render the images as 3D objects in an environment in Unity if necessary. This 3D environment will then be sent back to the main module.

7.1.3 Interface Description

The Unity engine renders a 3D environment that the user can see and interact with to explore the 3D environment seeing distances from objects provided by other modules. The Options menu will allow the user to change the parameters necessary for the Python script in order to replicate the environment data as needed wherein any output will be visually represented in the simulation.

7.1.4 Processing Description

The Unity engine will process the data taken from WebODM and the Python Program Module to create a 3D environment and display detection ranges to the user.

7.1.4.1 Design Class Hierarchy

Not Applicable

7.1.4.2 Restrictions/Limitations

The limitation to the engine would consist of sending inaccurate data wherein it does not render an accurate simulation, as well as the use of blurry images that will not clearly represent a clear 3D model. The Unity engine is based on C# natively and will usually only allow for C# related files unless some workarounds are implemented. The limitation of the engine rendering power and speed would be the OS and GPU that is in use.

7.1.4.3 Performance Issues

The Unity engine performance will be determined by the amount of data that is sent to the engine for rendering the environment. The more accurate the simulated

model is, the more time it takes for the engine to create the simulated environment of the location.

7.1.4.4 Design Constraints

The Unity engine constraints would be the engine can only render these simulations when a certain filetype is given to the engine.

7.1.4.5 Processing Detail For Each Operation

Not Applicable

7.2 Python Program Module

7.2.1 Classification

The Python program module will process the calculations of the provided auditory perception models, interacting with the Unity module in real time.

7.2.2 Processing Narrative (PSPEC)

The processing begins when the Python program is initiated. The program retrieves data from our dictionary database, which includes auditory perception model values, environmental sound measurements, and SUAS metrics provided by the Army Research Lab. Using the given parameters, the program performs the following computations:

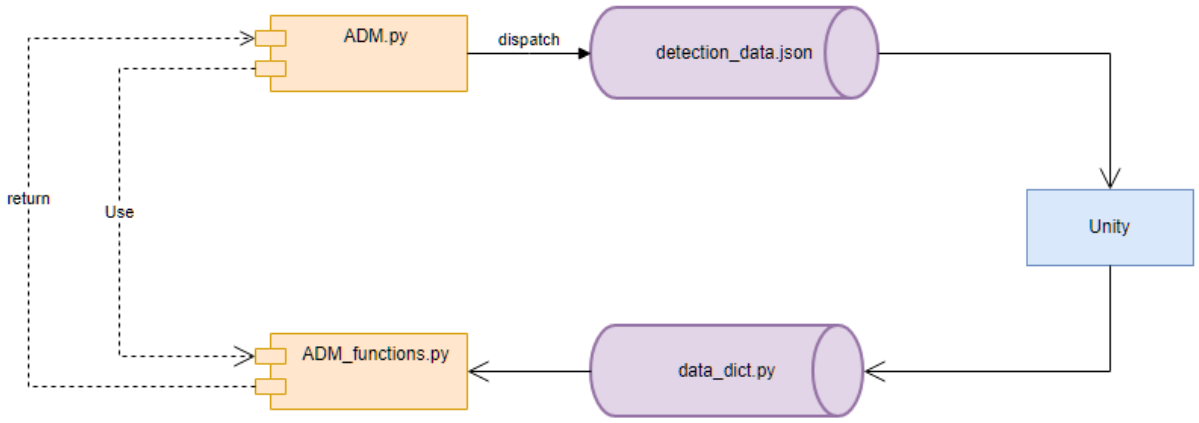
7.2.2.1 Calculates the distance at which a target can be detected based on the data from the auditory and visual perception models and environmental sound measurements.

7.2.2.2 Computes the propagation losses and noise spectrum at any distance from a measured noise source using the SUAS metrics.

Once the computations are complete, the results are transferred or passed to the Unity user interface for display to the end-user. The Unity engine receives the computed data and renders it in the appropriate format to display the relevant information to the user.

7.2.3 Interface Description

This module will have no direct interface for the end user.



7.2.4 Processing Description

Python program is processed to be used as a JSON file to be used in Unity. Given the inputs of specific variables, either from hardcoded values or from the dictionary, it is processed for use in Unity.

7.2.4.1 Design Class Hierarchy

Not applicable.

7.2.4.2 Restrictions/Limitations

Currently at this level there is no well known restrictions and limitations

7.2.4.3 Performance Issues

Currently at this level there is no well known performance issues

7.2.4.4 Design Constraints

This module works with a strict hardcoded value in place, doesn't take into account dynamic ever changing variables.

7.2.4.5 Processing Detail For Each Operation

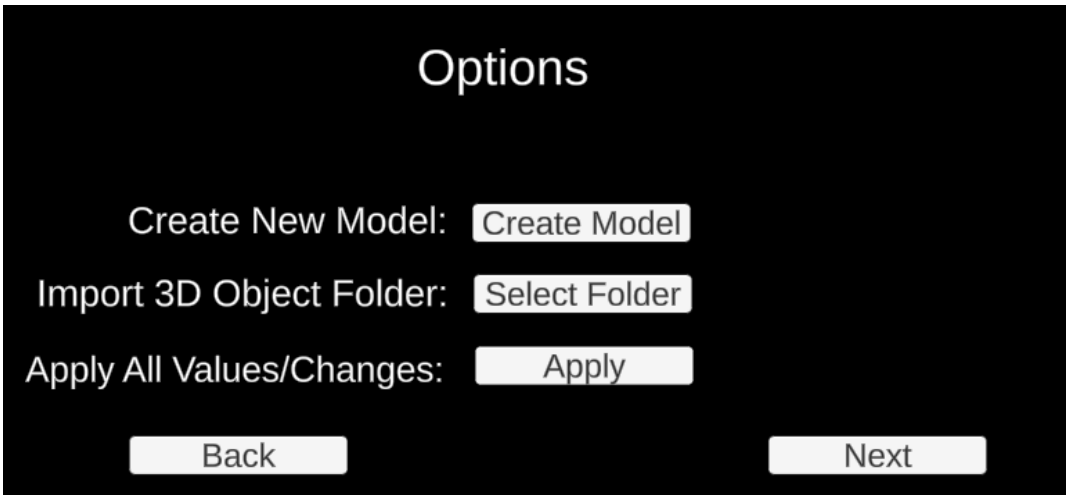
Not Applicable

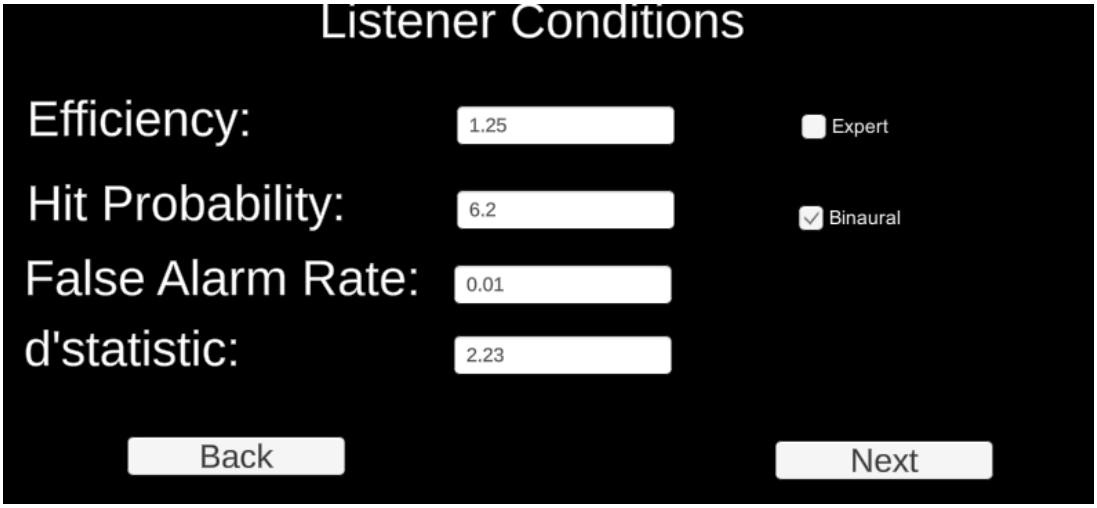
8. User Interface

8.1 Overview of User Interface

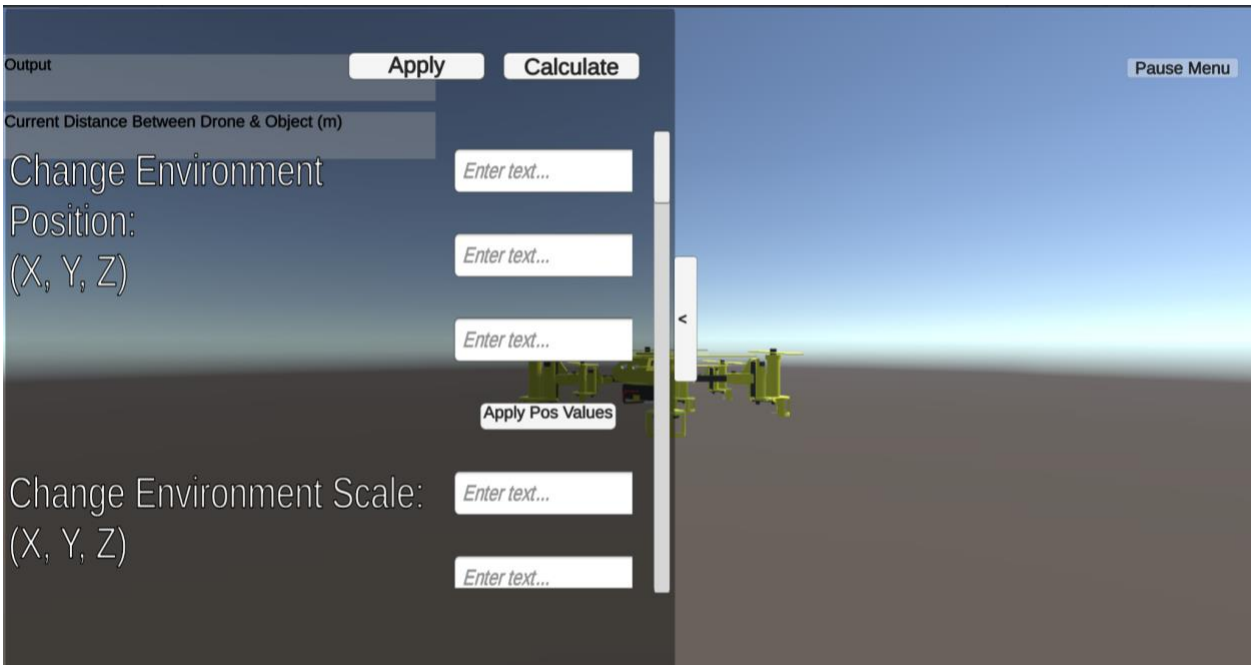
The user interface will display all information related to reconnaissance environments, allow for input data designed for visual manipulation, and will reveal detection distances between the drone and possible human locations.

8.2 Screen Frameworks or Images




Listener Conditions
Efficiency:
1.25
Expert
Hit Probability:
6.2
Binaural
False Alarm Rate:
0.01
d'statistic:
2.23
Back
Next


163.8128m



8.3 User Interface Flow Model

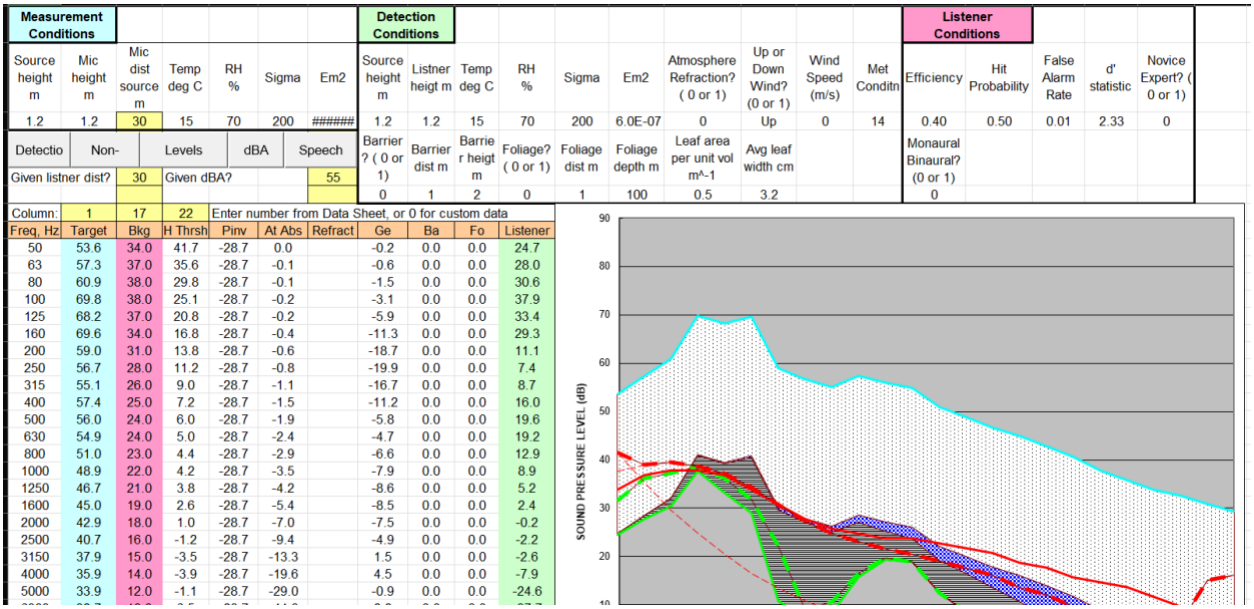
The user interface will have several sections that will help display what the user requires from the program and the Python scripts provided. The Options menu will let the user input values via textbox, checkboxes, dropboxes, and buttons. A few values that may be

interchangeable will include hearing thresholds, background noise settings, and environmental changes which will ultimately impact the detection distance output received from the Python script. Condition categories will also be changeable which include listener, detection, measurement, and environmental. These changes will be set once the apply button is clicked in which the values will be used in a Python script via virtual environment and output the related detection fields.

Buttons for model creation and importation are also present in the Options menu so the user can easily select the option best suited for them. The Model Creation button will send the user to a localhost website that will let the user customize the 3D model from texture details to inserting exact camera location. Depending on the amount of options changed as well as the amount of pictures inserted the model will take a certain amount of time to generate. A recommended amount of 15 photos with mostly default settings is recommended as changes may be unstable.

The Start button will visually display the 3D environment obtained from either WebODM or any previously made object file and its related textures as well as any additional modifiers made by the user. The user will be able to get a clearer view of the rendered environment by using a simulated drone to move around the sandbox area and get a clearer idea of the surrounding area without having to risk another reconnaissance mission. A Pause menu has also been added in order to help users navigate back and forth between menus.

9. Database Design



Measurement Conditions							Detection Conditions									Listener Conditions				
Source height m	Mic height m	Mic dist source m	Temp deg C	RH %	Sigma	Em2	Source height m	Listner heigt m	Temp deg C	RH %	Sigma	Em2	Atmosphere Refraction? (0 or 1)	Up or Down Wind? (0 or 1)	Wind Speed (m/s)	Met Conditn	Efficiency	Hit Probability	False Alarm Rate	d' statistic	Novice Expert? (0 or 1)
1.2	1.2	30	15	70	200	######	1.2	1.2	15	70	200	6.0E-07	0	Up	0	14	0.40	0.50	0.01	2.33	0
Detectio	Non-	Levels	dBA	Speech			Barrier ? (0 or 1)	Barrier dist m	Barrie r heigt m	Foliage? (0 or 1)	Foliage dist m	Foliage depth m	Leaf area per unit vol m^-1	Avg leaf width cm			Monaural Binaural? (0 or 1)				
Given listner dist?	30	Given dBA?		55			0	1	2	0	1	100	0.5	3.2			0				

Column:	1	17	22	Enter number from Data Sheet, or 0 for custom data						
Freq, Hz	Target	Bkg	H Thrsh	Pinv	At Abs	Refract	Ge	Ba	Fo	Listener
50	53.6	34.0	41.7	-28.7	0.0		-0.2	0.0	0.0	24.7
63	57.3	37.0	35.6	-28.7	-0.1		-0.6	0.0	0.0	28.0
80	60.9	38.0	29.8	-28.7	-0.1		-1.5	0.0	0.0	30.6
100	69.8	38.0	25.1	-28.7	-0.2		-3.1	0.0	0.0	37.9
125	68.2	37.0	20.8	-28.7	-0.2		-5.9	0.0	0.0	33.4
160	69.6	34.0	16.8	-28.7	-0.4		-11.3	0.0	0.0	29.3
200	59.0	31.0	13.8	-28.7	-0.6		-18.7	0.0	0.0	11.1
250	56.7	28.0	11.2	-28.7	-0.8		-19.9	0.0	0.0	7.4
315	55.1	26.0	9.0	-28.7	-1.1		-16.7	0.0	0.0	8.7
400	57.4	25.0	7.2	-28.7	-1.5		-11.2	0.0	0.0	16.0
500	56.0	24.0	6.0	-28.7	-1.9		-5.8	0.0	0.0	19.6
630	54.9	24.0	5.0	-28.7	-2.4		-4.7	0.0	0.0	19.2
800	51.0	23.0	4.4	-28.7	-2.9		-6.6	0.0	0.0	12.9
1000	48.9	22.0	4.2	-28.7	-3.5		-7.9	0.0	0.0	8.9
1250	46.7	21.0	3.8	-28.7	-4.2		-8.6	0.0	0.0	5.2
1600	45.0	19.0	2.6	-28.7	-5.4		-8.5	0.0	0.0	2.4
2000	42.9	18.0	1.0	-28.7	-7.0		-7.5	0.0	0.0	-0.2
2500	40.7	16.0	-1.2	-28.7	-9.4		-4.9	0.0	0.0	-2.2
3150	37.9	15.0	-3.5	-28.7	-13.3		1.5	0.0	0.0	-2.6
4000	35.9	14.0	-3.9	-28.7	-19.6		4.5	0.0	0.0	-7.9
5000	33.9	12.0	-1.1	-28.7	-29.0		-0.9	0.0	0.0	-24.6

The above figure is an example of the raw data shared with us by the Army Research Lab. We examined the VBA Macros from these excel sheets and translated the background macros into a Python program which gets exported into Unity. The python program extracts the data from the excel sheet currently for certain standard values like MIL STD for threshold of hearing but allows the code to be modified to use any source of data. i.e. A database, a cleaned up csv, etc. To be able to separate the excel sheet from the python program we'll have a data dictionary that is able to pull the necessary data required for the functionality of the functions to process the same way they did in the excel sheet. The data dictionary will be set with parameters that are referenced in the model sheet. Thave values in the data dictionary can be modified through Unity via JSON or set by the user for the purposes of field testing and validating the accuracy and precision of the theoretical model used. This simulates the idea of a No-SQL database like MongoDB which can be an alternative method of storing these values.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	Freq Hz	Typical Vehicle	BBN Default Target	Jungle Day	Jungle Night	Desert Low Wind	Desert Moderate Wind	Urban	TACOM Default	MIL_STD 1474	TACOM '77 #12292	Upper Limit Rural EPA	Lower Limit Rural EPA	Grand Canyon Wilder-ness Critical	NATO	STANAG (Future Main Battle Tank)	BBN Noisy Jungle	ISO Standard	1.5-2.4 years service time	7.5-12.4 years service time	17.5-22.4 years service time	Poor hearing: Auditory Handicap	Poor + Temp Threshold Shift	Avg REAT EAR form plug
2	50	53.6	66.0	34.0	38.0	34.0	44.0	51.0	26.0	15.0	34.0	43.0	34.0	18.0	40.0	29.0	34.0	41.7	44.0	44.0	46.0	58.0	66.0	35.0
3	63	57.3	73.0	34.0	38.0	33.0	43.0	51.0	25.0	15.0	35.0	45.0	37.0	17.0	37.0	29.0	35.0	35.6	41.4	41.5	43.4	51.0	60.0	35.0
4	80	60.9	81.0	33.0	35.0	32.0	42.0	51.0	24.0	15.0	35.0	48.0	38.0	16.0	38.0	28.0	35.0	29.8	38.8	38.9	40.9	45.0	55.0	35.0
5	100	69.8	83.0	30.0	33.0	30.5	41.0	50.0	22.0	15.0	35.0	47.0	38.0	15.0	37.0	26.0	35.0	25.1	36.2	36.3	38.3	40.0	51.0	35.0
6	125	68.2	82.0	28.0	30.5	29.0	40.0	50.0	21.0	15.0	35.0	45.0	37.0	14.0	32.0	24.0	35.0	20.8	33.6	33.8	35.7	36.0	48.0	35.0
7	160	69.6	81.5	27.5	28.0	26.0	38.0	49.0	20.0	15.0	34.0	42.0	34.0	13.0	27.0	23.0	34.0	16.8	31.0	31.3	33.2	32.0	46.0	35.0
8	200	59.0	82.0	27.5	26.0	23.5	35.0	48.0	19.0	15.0	33.0	41.0	31.0	12.0	26.0	21.0	33.0	13.8	28.4	28.7	30.6	30.0	44.0	35.0
9	250	56.7	79.5	27.0	24.5	21.0	31.0	47.0	18.0	15.0	32.0	40.0	28.0	11.0	25.0	19.0	32.0	11.2	25.9	26.1	28.0	28.0	43.0	36.0
10	315	55.1	78.0	26.0	24.0	18.5	28.0	46.0	17.0	15.0	31.0	39.0	26.0	10.0	24.0	18.0	31.0	9.0	23.2	24.0	26.0	25.0	42.0	38.0
11	400	57.4	77.0	25.0	24.0	16.5	25.5	45.0	16.0	15.0	30.0	37.0	25.0	9.0	24.0	16.0	30.0	7.2	21.2	22.2	24.7	24.0	41.0	40.0
12	500	56.0	75.0	24.5	24.0	14.0	23.0	44.0	15.0	15.0	29.0	36.0	24.0	8.0	23.0	15.0	29.0	6.0	19.8	21.2	23.6	23.0	40.0	41.0
13	630	54.9	71.0	24.0	25.0	12.5	20.5	42.0	14.0	15.0	27.0	34.0	24.0	6.0	21.0	14.0	27.0	5.0	18.6	20.5	23.0	23.0	40.0	41.5
14	800	51.0	70.0	24.0	26.0	11.0	18.0	40.0	14.0	15.0	25.0	34.0	23.0	5.0	20.0	13.0	25.0	4.4	17.7	20.2	23.2	23.0	42.0	41.5
15	1000	48.9	66.0	24.0	26.5	10.0	16.0	38.0	13.0	15.0	23.0	33.0	22.0	5.0	18.0	12.0	23.0	4.2	17.4	20.0	24.0	24.0	44.0	41.0
16	1250	46.7	62.0	24.0	27.0	9.0	14.0	36.0	12.0	15.0	21.0	32.0	21.0	5.0	19.0	11.0	21.0	3.8	16.9	20.3	25.2	25.0	46.0	39.5
17	1600	45.0	62.0	24.0	27.0	8.0	12.0	34.0	12.0	15.0	19.0	30.0	19.0	4.0	18.0	10.0	19.0	2.6	16.1	21.6	27.1	27.0	49.0	38.0
18	2000	42.9	61.5	24.0	27.0	7.0	10.0	32.0	11.0	15.0	17.0	28.0	18.0	4.0	18.0	9.0	17.0	1.0	16.0	24.5	30.3	30.0	53.0	38.0
19	2500	40.7	63.0	23.0	28.5	6.0	8.0	29.0	10.0	15.0	14.0	26.0	16.0	3.0	18.0	8.0	14.0	-1.2	16.8	29.8	38.3	38.0	65.0	40.5
20	3150	37.9	63.0	22.5	30.0	5.0	6.0	26.0	10.0	15.0	11.0	23.0	15.0	2.0	19.0	7.0	11.0	-3.5	19.0	37.0	46.5	46.0	76.0	43.0
21	4000	35.9	61.0	22.0	31.5	4.0	4.0	23.0	9.0	15.0	8.0	22.0	14.0	1.0	19.0	7.0	8.0	-3.9	24.9	46.4	57.5	57.0	82.0	44.0
22	5000	33.9	58.0	22.0	35.0	4.0	4.0	20.0	8.0	15.0	5.0	20.0	12.0	0.0	19.0	6.0	5.0	-1.1	30.4	52.5	63.4	63.0	85.0	44.5
23	6300	32.7	54.0	22.0	40.0	4.0	4.0	16.0	8.0	15.0	0.0	19.0	10.0	0.0	19.0	6.0	0.0	6.5	30.4	52.5	63.4	72.0	91.0	44.5
24	8000	31.0	50.0	20.0	43.0	4.0	4.0	11.0	8.0	15.0	0.0	17.0	9.0	0.0	21.0	6.0	0.0	15.3	30.4	52.5	63.4	76.0	95.0	44.5
25	10000	29.4	46.0	20.0	38.0	4.0	4.0	11.0	8.0	15.0	0.0	16.0	8.0	0.0	21.0	6.0	0.0	16.4	30.4	52.5	63.4	72.0	91.0	44.5
26																								
27	Column#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
28		Typical Vehicle		Noise														Threshold of Hearing						

X	Y	Z	AA	AB	AC	AD	AE	AF
Avg REAT EAR form plug	1/3 OB weights for masking bands					B3 Array for TSD calc	Awt weights	Cwt weights
35.0	0	0	1	1	0.5	10.158	-30.2	
35.0	0	0.5333	1	0.6683	0.45	9.48	-26.2	-0.8
35.0	0.3048	0.4355	1	0.5176	0.3846	9.054	-22.5	-0.5
35.0	0.1521	0.3565	1	0.3999	0.1321	8.848	-19.1	-0.3
35.0	0.07568	0.2917	1	0.309	0.04539	8.747	-16.1	-0.2
35.0	0.03776	0.2388	1	0.2388	0.0156	8.747	-13.4	-0.1
35.0	0	0.195	1	0.1845	0	8.747	-10.9	0
36.0	0	0.1596	1	0.1429	0	9.002	-8.8	0
38.0	0	0.1306	1	0.1104	0	9.535	-6.6	0
40.0	0	0.1069	1	0.08531	0	10.395	-4.8	0
41.0	0	0.0875	1	0.06592	0	11.398	-3.2	0
41.5	0	0	1	0	0	12.498	-1.9	0
41.5	0	0	1	0	0	13.703	-0.8	0
41.0	0	0	1	0	0	15.199	0	0
39.5	0	0	1	0	0	17.054	0.6	0
38.0	0	0	1	0	0	19.134	1	-0.1
38.0	0	0	1	0	0	21.469	1.2	-0.2
40.5	0	0	1	0	0	24.089	1.3	-0.3
43.0	0	0	1	0	0	27.028	1.2	-0.5
44.0	0	0	1	0	0	30.326	1	-0.8
44.5	0	0	1	0	0	34.026	0.5	-1.3
44.5	0	0	1	0	0	38.178	-0.1	-2
44.5	0	0	1	0	0	42.837	-1.1	-3
44.5	0	0	1	0	0	48.064	-2.5	-4.4
23								
	Internal Data For Model							

```
#Dictionaries to store parameters referenced in 'Model' sheet
#For now it is set at constants, these will be adjusted based on user/program input in the future

meas_cons = {
    'mic_height_meas':  1.2,
    'celsius_degrees_meas': 15,
    'relative_humid_percent_meas': 70,
    'sigma_meas': 200,
    'em2_meas': 0.0000006,
}

det_cons = {
    'source_height_det': 1.2,
    'listener_height_det': 1.2,
    'celsius_degrees_det': 15,
    'relative_humid_percent_det': 70,
    'sigma_det': 200,
    'em2_det' : 0.0000006,
    'wind_speed': 0,
    'wind_flag': False,
    'wind_direction': 'UP',
    'barrier_on': False,
    'barrier_dist': 1,
    'barrier_height': 2,
    'foliage_on': False,
    'foliage_dist': 1,
    'foliage_depth': 100,
    'leaf_areapervol': 0.5,
    'leaf_width': 3.2,
}


lis_cons = {
    'observer_efficiency': 0.40,
    'hit_prob': 0.50,
    'false_alarm_rate': 0.01,
    'd_stat': 2.33, #this is calculated using Dprime
    'is_expert': True,
    'is_binaural': False,
}

given_cons = {
    'm_measure_distance': 250,
    'lis_dist_given': 30,
    'dba_given': 55,
    'trg_name': 'M60 Tank idling at 30 meters',
    'bkg_name': 'Urban',
    'hth_name': 'ISO Std',
}
```

10. Requirements Validation and Verification

10.1 Functional Requirements

Functional Requirements	Component Modules
SUDO shall create a simulation based on images received	- Unity engine will import images from WebODM
SUDO shall have a customizable environment for simulation	- Toolbar UI element and Unity Engine
SUDO shall let users input parameters needed for a detection distance output	- Toolbar UI element and Unity Engine
SUDO shall estimate detectability distances of an SUAS based on target location parameters	- Python Program Component
SUDO shall create statistical detection zones in the form of spheres for target objects	- Unity Engine will simulate sphere-like detection zones around the drone and targets
SUDO shall detect how far an target object is away from drone in our 3D environment	- 3D mapping of the SUAS's images, providing the distances with the correct scaling - WebODM
SUDO shall detect what range is needed for drone detection based on a fixed-target distance	- Python Program Component

11. Glossary

- **DFD** - data flow diagram
- **SUAS** - Small Unarmed Aerial System (aka drone)
- **SUDO** - Small-Unit Drone Optimization
- **Unity** - Cross-platform game engine developed by Unity Technologies, used to create our 3D environments and visualization of our detection model
- **ODM -** OpenDroneMap
- **Jupyter Notebook-** Open source web application used to create code, equations, and data visualizations.

12. References

- Auditory Detection Model by Joel (2013) *Excel Sheet*
- Experimental Study of Quadcopter Acoustics and Performance at Static Thrust Conditions - https://www.bu.edu/ufmal/files/2016/07/aiaa-2016-2873.pdf
- Proposed Aural Detectability - Garinther (1985)
- Modeling human visual perception for target detection in military simulations - Jungkunz (2009)

Add/Remove from Design Document

- Update the unity engine to introduce the new interface / interaction with python program**(Later date)**
- Remove the **Machine learning** aspect from **Architectural strategies**, **System Architecture, Detailed lower level design**
- Replace machine learning with python program details etc
- Replace DFD with new and updated details
- Stress the details when it comes to translating the macros, and the process we took, including the research for functions.
- 9. Database design involve aspects of the python program and how we translated the the macros and put them into a data dictionary the section will emphasize the translation of the data
- 5 Dev policy , replace yolov5 with python program
- 3. Replace machine learning python program