Ram Logic | Metal Whisker Modeling

Project Report

SPONSORS
Missile Defense Agency
National Aeronautics and Space Administration
Naval Surface Warfare Center

PREPARED FOR
Dr. Donna Havrisik
Mr. Stephen Wells
Mr. Jay Brusse

PREPARED BY
Cristobal Escobar
Kyle Lim
Alan Sun

Table of Contents

I. PROJECT DESCRIPTION

I.1 Introduction

Metal whiskers are microscopic, hairlike protrusions that spontaneously grow from metal surfaces, specially from tin, zinc, cadmium, or other metals that are widely used in and around electronic circuitry [1]. These whiskers spontaneously grow over time taking from days to months to years. Metal whiskers often attain lengths up to a few millimeters and very occasionally a few centimeters with thicknesses ranging from submicron to tens of micrometers (much thinner than a single human hair) [1]. Their length and thickness distributions have been shown to be nicely-fit by lognormal distributions [2].

Metal whiskers are cause for significant concern in electronic systems due to their potential to cause short circuits, arcing, and electrical malfunctions. Whiskers can cause shorts between the object from which they are growing and nearby conductors. They may also break off from the surface they are growing from and become loose, conductive debris capable of producing shorts at distant locations. These risks are especially critical in high-reliability environments such as aerospace and defense where failure may lead to significant consequences.

Two of the many major field failure cases involving tin whiskers include the GALAXY IV satellite and unintended acceleration of Toyota Automobiles. Galaxy IV, a telecommunications satellite operated by PanAmSat, was lost on May 19, 1998 causing nearly 80% of pagers in the US to stop functioning [3]. The satellite control processors (SCPs) of Galaxy IV relied on tin-plated electromagnetic relays; however, the relays developed tin whiskers that bridged between exposed conductors in the vacuum of space and initiated catastrophic metal vapor arcs that disabled the SCP rendering the satellite inoperable [2]. In 2010 the U.S. Department of Transportation hired NASA to support an investigation of Toyota's electronic throttle control systems to determine if there may be electronic-related causes of unintended acceleration (i.e., cars accelerating without driver pressing on the accelerator pedal). In 2011, NASA presented a report showing how tin whiskers sometimes formed inside of certain types of accelerator pedal position (APP) sensors and produced electrically resistive paths between closely-spaced conductors [4]. Certain combinations of short circuits have been shown to induce unintended acceleration failure modes.

Pagers and automobiles are reliability and safety-critical systems. In hospital and emergency services settings, the loss of critical communications systems may have dire consequences. Similarly, safety-critical automotive systems, such as electronic throttle

controls that are prone to whisker-induced failures can lead to crashes with personal harm and loss of property.
The primary aim of this project is to statistically measure the probability of detached metal whiskers bridging between exposed conductors on a target PCB. To this end the preceding team, the Tin Whisker Investigative Team, simulated forests of metal whiskers landing on simulated PCBs. The nucleation and growth of metal whiskers, however, are not within the scope of this project. This project is positioned in the domain of PCB reliability.

I.2 Background and Related Work

Previous work has been conducted by the Tin Whisker Investigative Team at WSU and will be continued by our team, Ram Logic. The project mentor, Jay Brusse, has mentioned this preceding metal whisker modeling project is state-of-the-art in this context.

The program uses Unity's Physics Game Engine to perform 3-dimensional simulations. First, the program takes user-provided computer-aided design (CAD) files of the target PCB to produce a 3D model of the PCB, components and exposed conductors. Next, the program generates a user-provided number of metal whiskers of length and thickness according to the lognormal distribution having parameters μ and σ provided by the user[3]. The program then simulates dropping the metal whiskers on the PCB in a user-defined area [3]. External forces such as gravity, mechanical shock or vibration can also be simulated to induce the movement of the detached whiskers throughout the simulation. The modeling framework then creates an Excel spreadsheet identifying conductors bridged by individual whiskers [6].

In addition to simulating single runs, the preceding team implemented Monte Carlo simulations to facilitate risk assessment based on the likelihood of whisker bridges between specific conductor pairs.

I.3 Project Overview

The core PCB simulation framework is functional, users can import a PCB, identify conductive components, and run Monte Carlo simulations for metal whisker bridge statistical analysis. Key improvement areas include the following:

1. Automated conductive material detection
2. Enhanced results visualization
3. Optimizing user input parameter ranges
4. Unit and system testing
5. UI fixes for enhanced workflow

6. User guided tutorials and tooltips

By addressing these enhancements, this project aims to increase usability, efficiency, and reliability of the metal whisker modeling system.

I.4 Client and Stakeholder Identification

The sponsors of this project are the Missile Defense Agency (MDA), the National Aeronautics and Space Administration (NASA), and the Naval Surface Warfare Center (NSWC) which all require highly reliable electronics for defense and aerospace.

Beyond the sponsoring organizations, this project has far-reaching application in electrical engineering and PCB reliability analysis. Reliability-critical components such as pacemakers, car accelerators, telecommunication satellites, etc. reveal key stakeholders of this project.

II. REQUIREMENTS AND SPECIFICATIONS

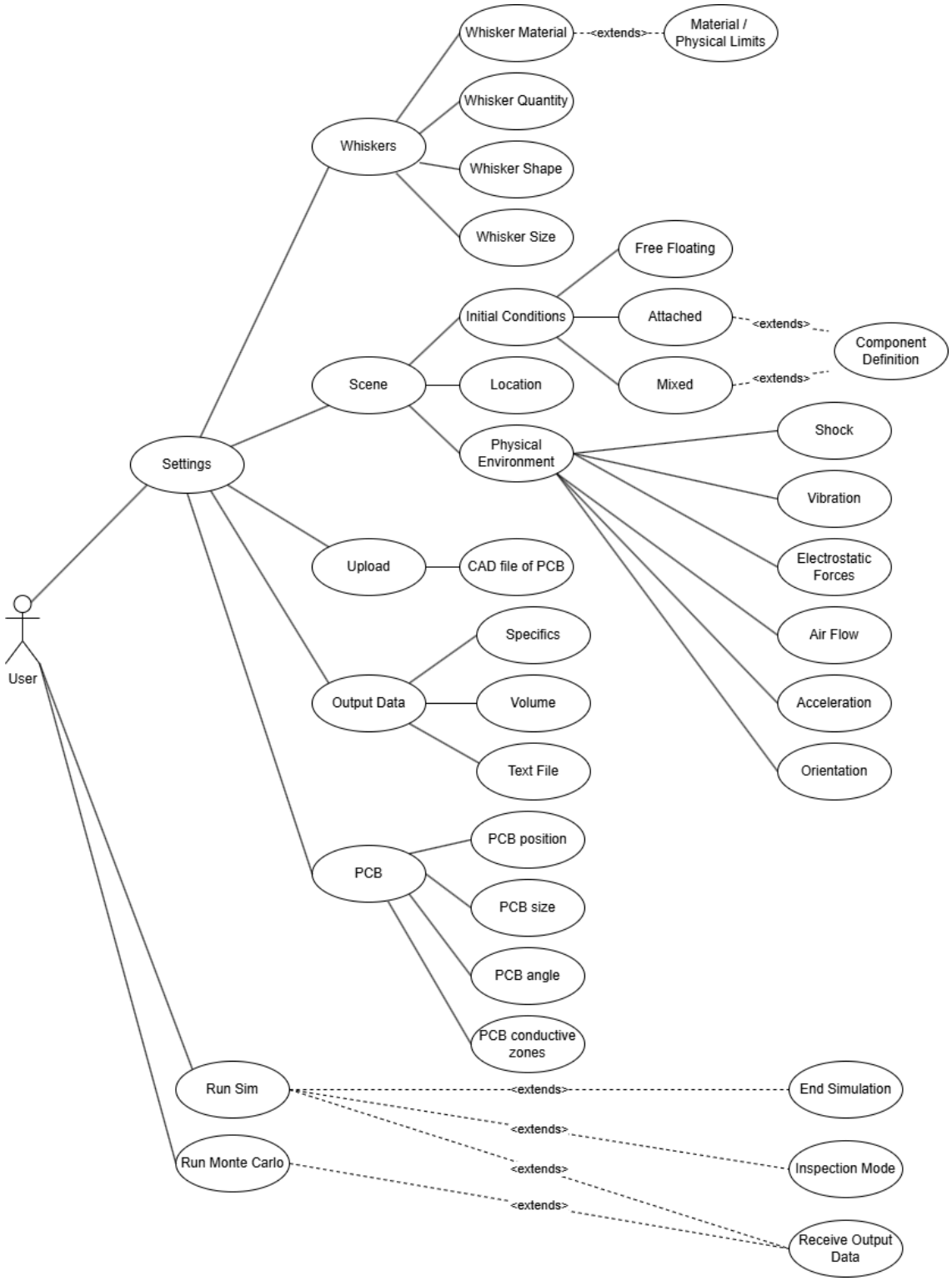
II.1 Introduction

The Tin Whiskers Unity 3D App is a software tool designed to simulate the impact of metal whiskers on printed circuit boards (PCBs). Metal whiskers are electrically conductive hair-like structures that can cause short circuits and system failures in electronic components. The project aims to develop a simulation that captures a 3D model of a PCB, identifies exposed conductors, and simulates detached metal whiskers landing on the board.

This project is beneficial to any company that relies on electronic circuitry and wants to estimate the probability of detached metal whiskers leading to bridging events. The results from this analysis may be used with other information available to the customer (e.g., consequence of specific bridging events) to device mitigation strategies to reduce the threat of metal whisker induced harm.

II.2 System Requirements Specification

II.2.1 Use Cases



Upload File:

Pre-condition	Application running and on the main menu.
Post-condition	PCB loaded and configured
Basic Path	1. Click on the Load button. 2. Select file.
Related Requirements	File must be .obj or .mtl

View Monte Carlo Simulation Results:

Pre-condition	On the main menu.
Post-condition	Display results of the simulation.
Basic Path	1. User has configured preferred settings. 2. Navigated back to the main menu. 3. Run Monte Carlo Simulation and input the preferred number of simulations. 4. Click on Sim Results. 5. Select View Monte Carlo Simulation Results
Related Requirements	A Monte Carlo simulation was run.

Edit box parameters:

Pre-condition	Application running and on the main menu.
Post-condition	Box settings updated.
Basic Path	1. Locate Sim Settings 2. User enters in parameters of the box (xyz position and xyz size)

Edit Whiskers Parameters:

Pre-condition	Application running and on the main menu.

Post-condition	User will have selected preferred type of whisker (size, shape, material, amount)
Basic Path	1. Locate Sim Settings. 2. User selects the preferred material type. 3. User enters in parameters of the whisker itself (i.e., # of detached whiskers, whisker length, and whisker thickness distribution parameters).

Edit Board Parameters:

Pre-condition	Application running and on the main menu.
Post-condition	User will have selected preferred board settings (size, position, and angle)
Basic Path	1. Locate Board Settings. 2. User enters in parameters of the board itself (i.e., size in xyz, position in xyz, select conductive materials, tilt on x and z axis).
Related Requirements	Board has been uploaded.

Run Simulation:

Pre-condition	Application running and on the main menu.
Post-condition	Receive Data Output.
Basic Path	1. User has configured preferred settings. 2. Navigated back to the main menu. 3. Run Simulation. 4. Collect Output .
Related Requirements	User has loaded and configured preferred settings.

Run Monte Carlo Simulation:

Pre-condition	Application running and on the main menu.

Post-condition	Receive monte carlo data output.
Basic Path	1. User has configured preferred settings. 2. Navigated back to the main menu. 3. Run Monte Carlo Simulation and input the preferred number of simulation.. 4. Collect Output.
Related Requirements	User has loaded and configured preferred settings.

Inspect Simulation:

Pre-condition	A simulation is running.
Post-condition	The simulation is paused to be inspected.
Basic Path	1. User has configured preferred settings. 2. Navigated back to the main menu. 3. Run Simulation. 4. Click on the Inspection Mode button.
Related Requirements	Simulation must be running and the user must configure preferred settings.

End Simulation:

Pre-condition	Application running and in the middle of simulation.
Post-condition	Simulation stops and results are saved.
Basic Path	1. User has configured preferred settings. 2. Navigated back to the main menu. 3. Run simulation and end the simulation. 4. Collect Output.
Related Requirements	Simulation must be running and the user must configure preferred settings.

II.2.2 Functional Requirements

II.2.2.1 Software Assurance and Testing

Software Assurance Check:

Description	Identify and fix anomalies, defects and issues in existing software as well as the new software being developed at each stage.
Source	Mandatory request from Dr. Havrisik.
Priority	0

Unit and System Testing:

Description	Ensure the functionality of individual functions by including unit testing and to verify overall reliability of the system by including system testing.
Source	Suggested by the previous team and required from Dr. Havrisik.
Priority	0

II.2.2.2 Performance and Simulation

GPU Acceleration for Monte Carlo Simulation:

Description	The program currently has no GPU acceleration as Unity's physics engine implementation is CPU-bound. The metal whisker simulation must include GPU acceleration for reasonable simulation time.
Source	Future work left by the preceding metal whisker modeling team confirmed by Dr. Havrisik in an email on 2/4/2025.
Priority	0

Optimization of User Parameters Ranges:

Description	The program can break as a result of user input. Parameters for different inputs in the program should be restricted to an acceptable range.

Source	Mentioned in future work left by the preceding team and confirmed by Dr. Havrisik.
Priority	1

II.2.2.3 User Interface and Visualization

Enhanced Results Visualization:

Description	The application should include simulation results visualization. In the current implementation, users must navigate to a results directory and open CSV files using a spreadsheet application. The application should be unified to fulfill user needs.
Source	Mentioned by preceding confirmed by Dr. Havrisik in an email on 2/4/2025.
Priority	0

Automated Conductive Material Detection:

Description	The application should detect conductive surfaces on CAD modeled PCB files without user interaction.
Source	Mentioned by preceding team confirmed by Dr. Havrisik in an email on 2/4/2025
Priority	0

Guided Tutorials and Tooltips:

Description	The application should include a guided tutorial to briefly run through a set of simulations using different μ, σ, mechanical vibration, shock, and vibrate parameters.
Source	Left as future work from the preceding team confirmed by Dr. Havrisik in an email on 2/4/2025
Priority	1

User Interfaces Fixes:

Description	Resolve issues related to the design, layout, and functionality of the user

	interface. Fixes may include resolving visual inconsistencies, improving navigation, and enhancing responsiveness.
Source	Left as future work from the preceding team confirmed by Dr. Havrisik in an email on 2/4/2025
Priority	2

II.2.2 Non-Functional Requirements

Performance:

- The system should provide visualization results with minimal time delays.
- The system should process and simulate large scale PCBs without performance degradation.

Compatibility:

- The system should work in Windows, Linux and MacOS.
- PCB models files should be supported in formats OBJ and MTL.

Scalability:

- The system should support future refinements for physics models.
- There should be room to enhance GPU acceleration.

Maintainability:

- Documentation should be comprehensive and clear.
- Future updates and modifications should not affect main functionality.

Security:

- The system should follow software assurance best practices to ensure robustness and reliability.

II.3 System Evolution

This project is dependent on several key assumptions that influence design and implementation of the Unity Metal Whisker Modeling project. As the project evolves, we anticipate some assumptions may be revisited and revised due to changes in software and hardware constraints as well as user needs.

This project assumes the following:

- GPU acceleration is feasible and necessary.
    - Simulating a high-density forest of metal whiskers falling on and interacting with a PCB board is computationally complex, our team assumes GPU acceleration will provide necessary performance gains.
    - We assume we can utilize unity shaders and/or third party plugins to access the GPU.
- Automated conductive material detection for CAD files is viable.
    - Our team assumes .OBJ CAD files can be reliably parsed and processed for conductive material extraction.
    - Our team assumes inaccuracies in conductive area detection can be remedied with preprocessing techniques.
- User interface can be implemented without major overhaul.
    - Our team assumes a user will benefit from a structured onboarding process and the tooltips will be readable without information overload.
    - The project includes updates to the UI, including general tooltips, and adding a tutorial to the existing project (could add video tutorial with instruction audio). Our team assumes these additions can be implemented without significant structural changes.

Anticipated changes and risks:

- Hardware limitations and compatibility issues:
    - If our GPU acceleration methods do not work as expected for any of the following reasons: driver incompatibility, limited GPU support, performance bottlenecks, etc., then we may need to explore alternative methods.
- User experience challenges:
    - If users struggle with tutorial design, then additional design iterations of UI may be required.
    - If guided tutorials prove insufficient in the user structured onboarding process, then alternative methods must be explored such as an interactive walkthrough.

III. SOLUTION APPROACH

III.1 Introduction

This section provides a comprehensive overview of the design of the metal whisker simulation project. The document will focus on three key design elements: Architecture, Data, and User Interface.

Architecture: This section will explore the subsystems that form the foundation of the project, detailing their concepts, algorithms, and interface properties.

Data: The document will outline the primary data types and database structures involved in the simulation.

User Interface: The UI design will be covered with mock-ups and descriptions of components and functionalities.

The primary goal of this design document is to establish a guideline for developers to follow throughout the implementation phase. Developers will be able to verify the system's design by referencing the following components:

- Subsystem descriptions
- Subsystem relationships
- Data type objects
- Data type and database relationships
- Database schema
- User Interface mock-ups

Additionally, this document serves as a reference for stakeholders, ensuring that the prototype aligns with the intended objectives and design specifications.

III.2 System Overview

Metal whiskers are microscopic conductive fibers that can cause electrical failures by bridging exposed conductors on a PCB. This project aims to develop a software tool that statistically measures the probability of detached metal whiskers bridging conductors on a target PCB. The tool leverages Unity's Physics Engine to simulate metal whiskers landing on PCBs, providing critical data for risk assessment.

Key enhancements over previous work include:

- GPU acceleration for faster Monte Carlo simulations
- Enhanced results visualization
- Automated conductive material detection
- UI improvements
- Unit and system testing
- User-guided tutorials and tooltips

These improvements will enhance efficiency and usability, making the tool more effective for reliability-critical applications.

III.3 Architecture Design

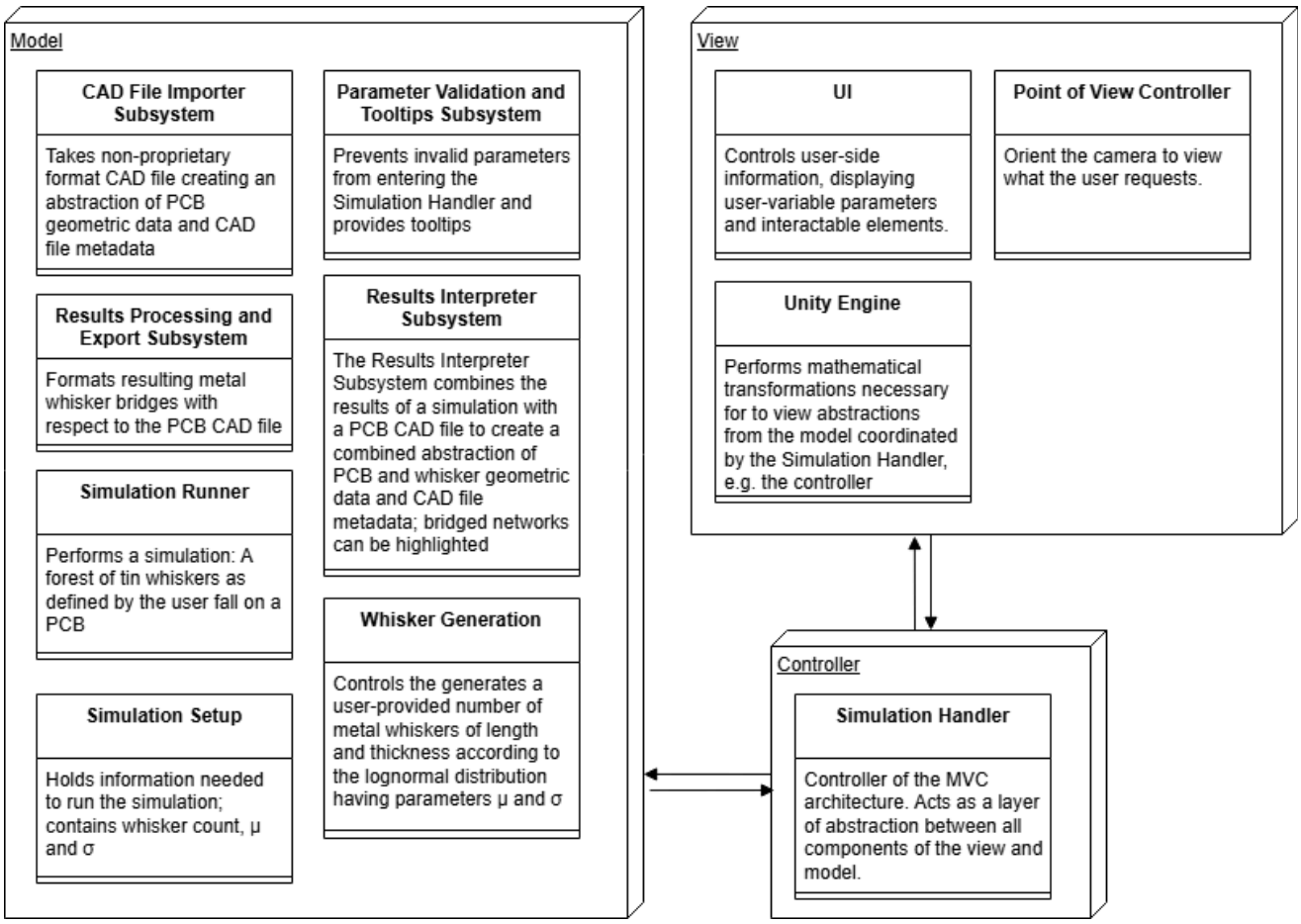
III.3.1 Overview

The software architecture chosen for the metal whiskers modeling and simulation uses the Model-View-Controller (MVC) pattern to ensure separation of concerns, maintainability and scalability. We believe this architecture follows a modular and layered approach and it allows for independent development and modification of important components such as the user interface, file handling subsystems and simulation engine. Since Monte Carlo simulation needs GPU acceleration, this structure could help optimize the application performance as well as keeping the user intuitive.

We believe the MVC architecture is the best option for this application since it separates the simulation logic, the user interface and the application behavior which should make it easier for future modifications.

- **Model**: Responsabile of the simulation core and data handling of the application, it includes generating metal whiskers, running single simulations or Monte Carlo simulations, and processing simulation results.
- **View**: Provides a graphical user interface(GUI) and handles user interaction with the system such as simulation configurations, visualization of single simulations and results, as well as results of Monte Carlo simulations.
- **Controller**: Contains the application logic and manages interactions between the model and view components, including validation of configurations, inputs processing and workflow management.

System Diagram:



III.3.2 Subsystem Decomposition

III.3.2.1 ****New Subsystem**** CAD File Importer Subsystem

Description:

In a typical well-documented PCB computer-aided design (CAD) file, nets are created and defined by a file's creator. This implementation enables the following feature:

Automated Netlist Generation: The tool automatically extracts and generates netlists from PCB designs. All conductive networks can be mapped without manual intervention.

Subsystem Responsibilities:

- **File Parsing**: Reading a CAD file format (e.g., STEP, DXF, Gerber, etc.) and extracting geometric information and metadata.
- **Data Conversion**: Transforming the imported CAD data into the program's internal representation of a PCB.
- **Abstraction for Downstream Processing**: Providing a standardized interface other parts of the system can rely on without reliance on CAD file raw data.

Concepts and Algorithms Generated:

- **File I/O Parsing**: Parsing to interpret structured CAD file formats.
- **Data Mapping and Transformation**: Abstraction techniques to extract three-dimensional geometries and map them to networks defined by the PCB designs creator.
- **Factory Pattern**: Constructing CAD file abstractions in steps to enable complex PCB CAD file imports.

Interface Description:

The CAD file importer subsystem exposes an interface that abstracts details of file parsing and data conversion. The CAD File Import Subsystem provides a standardized interface for interacting with a PCB.

Services Provided:

Service Name	Service Provided To	Service Description
PCB CAD File Loader	Simulation Handler	Enables users to search directories and select a PCB CAD file to load and validate those files.
CAD Parser	Simulation Handler	Parses the validated PCB CAD into an abstraction; creates internal representation of CAD geometries mapped to nets.
Get PCB	Simulation Handler	Returns PCB abstraction for use in the simulation.

Services Required:

Service Name	Service Provided From
File Search	Unity Engine
User Input	Simulation Handler
PCB CAD File	Non-proprietary CAD File

III.3.2.2 ****New Subsystem**** Results Processing and Export Subsystem

Description:

The results Processing and Export Subsystem provides additional context and positional information on electrical networks and components compromised.

Subsystem Responsibilities:

- **Metal Whisker Bridge Recording**: Captures metal whisker bridge positions as points on a set of conductive components over time.
- **File Output**: Generates a partial CAD file of *only* conductive bridges on a component with a coordinate system corresponding to the PCB file.

Concepts and Algorithms Generated:

- **Coordinate Recording**: Generates an abstraction of relevant simulation results with respect to three-dimensional coordinates.

Interface Description:

The Results Processing and Export Subsystem exposes an interface to abstract methods to record simulation bridging results.

Services Provided:

Service Name	Service Provided To	Service Description
Metal Whisker Bridge Recorder	Simulation Handler	Records metal whisker bridges, mapping bridge coordinates as a set of points relative to conductive areas bridged.
Results File Generator	Output File	Parses the validated PCB CAD into an abstraction; creates internal representation of CAD geometries mapped to nets.

Services Required:

Service Name	Service Provided From
Results from Simulation	Simulation Handler

III.3.2.3 ****New Subsystem**** Parameter Validation and Tooltip Subsystem

Subsystem Responsibilities:

Prevents invalid parameters from entering the Simulation Handler and provides valid parameter ranges to the simulation handler.

Subsystem Responsibilities:

- **Return Valid Parameter Ranges**: Numerical limits on parameters prevent users from breaking the program.
- **Return Tooltips**: Return a tooltip specific to a parameter element in the UI.

Concepts and Algorithms Generated:

- **JSON Parsing**: Parse central JSON objects for information retrieval.
- **Data Mapping**: After data is parsed, it must be hashed to match UI components to tooltips and valid parameter ranges.

Interface Description:

Validation Subsystem exposes an interface to the valid range of parameters and tooltip information. When called a method will return a valid parameter range and tooltip for a specific tool.

Services Provided:

Service Name	Service Provided To	Service Description
Parameter Validation	UI	Returns valid parameter ranges.
Tooltip Getter	UI	Returns the combination of the tooltip with its respective valid range.

Services Required:

Service Name	Service Provided From
Parameter Validation	Central Parameter Parameter JSON file
Results Generator	Central Parameter Parameter JSON file Central Tooltip JSON file

III.3.2.4 ****New Subsystem**** Results Interpreter Subsystem

Description:

The Results Interpreter Subsystem will revamp the current results visualization implementation. Currently, only bridged conductive areas are recorded; however, this subsystem will display more precise positional data and electrical networks compromised as they exist on the PCB after simulation.

Subsystem Responsibilities:

- **Display Results Directly on a PCB**: Enable direct display of bridges as they exist on a PCB.

- **Results Displayed**: Highlight bridged electrical networks as well as exactly displaying bridges on conductive components.

Concepts and Algorithms Generated:

- **CAD File Integration**: Parse and integrate result CAD file with respective PCB CAD file by mapping positional data of whisker bridging to PCB.
- **Electrical Network Detection and Highlighting**: Determine and highlight the path of each electrical network, and color code bridged electrical networks.

Interface Description:

The Results Interpreter Subsystem integrates the result non-proprietary CAD file with a respective PCB non-proprietary CAD file.

Services Provided:

Service Name	Service Provided To	Service Description
PCB Whisker Interpreter	CAD File Importer Subsystem	Parse the result and respective PCB non-proprietary CAD files and integrate both into an XML file.

Services Required:

Service Name	Service Provided From
Collection of Whiskers	PCB and whisker bridge CAD files

III.3.2.5 ****Modified**** UI

Description:

The UI controls all of the user-side information and displays. This includes getting user inputs and updating the display as specified. Concepts and Algorithms Generated: The built-in UI in Unity controls most of the complex algorithms concerning displays and user input. However, this component will still be split into subclasses that perform the tasks of the UI component.

Services Provided:

Service Name	Service Provided To	Service Description
Handle Input	Simulation Handler	Once the user inputs are received, the UI component is meant to send necessary commands and directions to the Simulation Handler.
Display Output	Simulation Handler	On parameter update, validate parameters. If a parameter is invalid highlight invalid parameter and deactivate the simulate button. Send the updates to display to Unity UI.
Tooltip Display	Simulation Handler	Display a tooltip on parameter range hover event

Services Required:

Service Name	Service Provided From
Receive Input	Unity UI

III.3.2.6 ****Modified**** Simulation Handler

Description:

The simulation handler component is meant to be the main part of the View Model layer. It receives the directions from UI (View) and sends them to be processed in the Model layer. Then it sends the results from the model layer back to the BAY (View) layer to be handled and displayed.

Concepts and Algorithms Generated:

This component is the interface between the View layer and the model layer. It controls directing all of the Model components and sending the necessary information back to the View Layer to be processed and displayed (through databinding).

Services Provided:

Service Name	Service Provided To	Service Description
Handle Tasks	UI	Call and control all of the components in Model layer as specified
User Inputs	Point of View Handler Simulation Setup Results Handler	Gives the necessary user inputs to components as needed
Collection of Whiskers	Result Processing and Export Subsystem	Gives the gathered collection of whiskers from the Whisker Generator.
Simulation Settings	Simulation Runner	Provides the simulation settings for the simulation runner to use in its simulation.
Results from Simulation	Results Processing and Export Subsystem	Gives the results gathered from a simulation to the results handler.

Services Required:

Service Name	Service Provided From
User Inputs	UI
Camera orientation and position updates	Point of View Handler
Collection of Whiskers	Whisker Generation
Simulation Settings	Simulation Set up
Collision Detection	Simulation Runner
Metal Whisker Bridge Recorder	Results File Generator

III.3.2.7 ****Modified**** Simulation Runner

Description:

Perform a simulation of tin whiskers falling as specified.

Concepts and Algorithms Generated:

This component contains logic for running a simulation (whiskers on a PCB) and collecting and recording data throughout the simulation. This will require collision detection algorithms to find when and where nodes are connected by a metal whisker

Interface Description:

Services Provided:

Service Name	Service Provided To	Service Description
Run Simulation	Simulation Handler	Drops a forest of tin whiskers on PCB
Collision Event	Simulation Handler	Determines whether a whisker has collided with more than one conductive area providing positional information.

Services Required:

Service Name	Service Provided From
Collection of Generated Whiskers	Simulation Handler

III.3.2.8 **Original** Whisker Generation

Description:

The Whisker Generation component controls the creation of all of the metal whiskers.

Concepts and Algorithms Generated:

This component uses the content in the Simulation Set up component to create a collection of metal whiskers as specified by the user.

Services Provided:

Service Name	Service Provided To	Description
Whisker Creation	Simulation Handler	Creates a collection of whiskers as specified by the user

Services Required:

Service Name	Service Provided From
Whisker Settings	Simulation Settings

III.3.2.9 **Original** Point of View Controller

Description:

Orient the camera to view what the user requests.

Concepts and Algorithms Generated:

The matrix manipulation and other mathematical concepts used in this component will be handled by the Unity Engine. The Unity Engine has extensive tools dealing with view handling.

Services Provided:

Service Name	Service Provided To	Description
Camera Orientation and Position Updates	Simulation Handler	Sends camera orientation and position updates to the simulation handler

Services Required:

Service Name	Service Provided From
User Inputs	Simulation Handler

III.3.2.10 ****Original**** Simulation Setup

Description:

The simulation set up component holds all the information needed to run a simulation. This includes fields such as whisker count, whisker size, object positions, and so on.

Concepts and Algorithms Generated:

This component receives directions from the simulation handler and sets up a simulation accordingly.

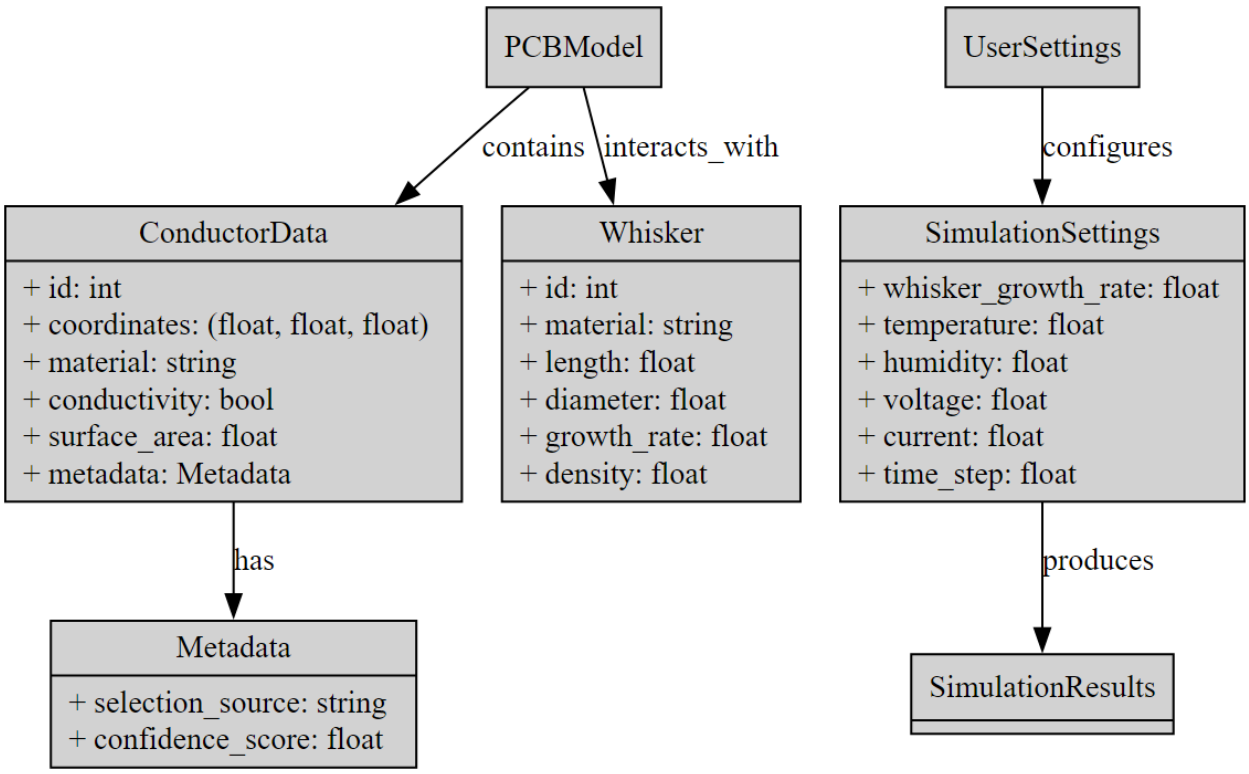
Services Provided:

Service Name	Service Provided To	Description
Set Simulation Content	Simulation Handler	This service sets up a simulation with specified inputs by setting the necessary values of a simulation's content.

Services Required:

Service Name	Service Provided From
User Inputs	Simulation Handler

III.4 Data design



Data Structures:

1. Printed Circuit Board (PCB) Model:
    - Represents the physical layout of the printed circuit board, including components, traces, and conductive paths.
    - Data Structure:
        - Graph Representation: Nodes represent PCB components and junctions, while edges represent conductive traces.
        - 3D Mesh: Used for visualization and finite element analysis (FEA).
        - Storage Format: Adjacency matrix or adjacency list (for graphs), and mesh files (e.g., .obj, .stl) for 3D visualization.

2. Conductor Identification Data:
    - Stores information about the location and properties of exposed conductors on the PCB surface.
    - Data Structure:
        - Dictionary or list containing:
            - Coordinates (X, Y, Z)
            - Material Properties (resistivity, thickness)
            - Conductivity Status (boolean flag: *Conductive/Non-Conductive*)
            - Surface Area (computed for whisker interaction likelihood)
3. Whisker Characteristics:
    - Defines the characteristics of detached metal whiskers, such as material composition, length, thickness, and spatial distribution.
    - Data Structure:
        - Object or dictionary storing properties as key-value pairs:
            - Material Composition (e.g., Tin, Zinc, Cadmium)
            - Length Distribution (Normal or Weibull)
            - Diameter (Micron range)
            - Growth Rate (Function of environmental factors)
            - Whisker Density (Region-based probability)
4. Simulation Parameters:
    - Stores user-defined simulation parameters, including the number of whiskers, environmental conditions, and simulation duration.
    - Data Structure:
        - Object or dictionary storing:
            - Whisker Growth Rate
            - Temperature & Humidity Settings
            - PCB Surface Conditions
            - Voltage & Current Levels
            - Time Step Resolution
5. Simulation Results:
    - Stores the results of whisker simulation, including information about bridged conductor pairs and probability of failures.
    - Data Structure:
        - List or database table storing:
            - Whisker-Bridge Events (Start & End Node)
            - Probability of Conduction Failure
            - Time to Failure Predictions
            - Graph Representation of Failures (Nodes & Edges modified dynamically)

Databases:

1. Simulation Data Repository:
    - Stores simulation input and output data for future reference and analysis.
    - Tables:
        - PCB Models: Stores information about PCB geometries and components.
        - Conductor Data: Stores details of exposed conductors identified in PCB models.
        - Whisker Characteristics: Stores parameters defining whisker properties.
        - Simulation Settings: Stores user-defined simulation parameters.
        - Simulation Results: Stores output data from whisker simulations, including bridged conductor pairs and failure probabilities.
2. User Settings Database:
    - Stores user preferences and configurations for customizing the software tool's behavior.
    - Tables:
        - User Profiles: Stores user information and preferences.
        - Default Settings: Stores default values for simulation parameters and tool configurations.

Graph-Based Conductive Path Modeling:

- The conductive paths on a PCB are represented as a graph, where:
    - Nodes represent PCB components, junctions, and exposed conductors.
    - Edges represent electrical connections and potential short-circuit paths.
- This approach enables efficient:
    - Shortest Path Calculations (to identify high-risk areas)
    - Real-Time Updates (as whiskers grow or bridge connections)
    - Graph-Based Failure Analysis (tracking changes dynamically)

Metadata for Conductive Material Selection:

- To track which materials are identified as conductive, the software will store metadata within the conductor identification data structure.
- Metadata Structure:
    - Selected Conductive Materials:
        - User-defined conductive material selection
        - Auto-detected conductors (via image processing or simulation data)
    - Selection Source:
        - User Input (manual override)

        - Algorithmic Detection (automated classification)
    - Confidence Score: Probability of correct classification (based on historical data and training sets)

III.5 User Interface Design

The Ram Logic team has developed a functional UI mockup for their metal whisker simulation tool, prioritizing a user-friendly experience for researchers and engineers. The design emphasizes clear navigation and intuitive controls to streamline the workflow of PCB analysis. As depicted in the provided images, the user interacts with several key interfaces. The main menu (Image 1) presents a central 3D viewport for visualizing the PCB model, surrounded by essential controls. Buttons labeled "Open Sim Settings," "Open Board Settings," "Heat Map," "Sim Results," "Load," "Run Sim," and "Monte Carlo" provide access to the core functionalities of the application. The top left of the main menu displays viewport manipulation instructions ("Hold RMB to pan," "Press keys WASD to move"), a "Go To Part" feature with a text input for part selection, and a close button. The loading dialog (Image 2), a standard file explorer interface, allows users to navigate their file system and select PCB model files in ".obj" and ".mtl" formats.

After loading in the PCB from the main dashboard, the following options are available:

- **Simulation Setup:** Users can define parameters for whisker simulations, such as whisker material, density, length, and environmental conditions. (3)

- **PCB Design:** Users can create and edit 3D models of printed circuit boards, adding components, traces, and conductive paths. (Image 4)

- **Conductor Identification:** This feature allows users to mark exposed conductors on the PCB, either manually or automatically, for use in whisker simulations. (Image 4)

- **Results Analysis:** The tool provides visualization and analysis of simulation results, showing bridged conductors and failure probabilities. (Image 8)

IV. Testing and Acceptance Plans

IV.1 Introduction

IV.1.1 Project Overview

The *Metal Whiskers Unity App* aims to perform the actions below:

- Simulate metal whiskers falling onto a printed circuit board (PCB).
- Identify the bridges between a PCB's conductive components.
- Save simulation results to a file.
- Run Monte Carlo simulations.
- Adjust simulation parameters within valid ranges.
- Access simulation statistical analysis.
- Autonomously identify conductive PCB materials after import
- Provide user tutorials and tooltips.

The aim of our test process is to ensure logically sound components and maximize computational resource efficiency to improve accuracy, reduce resource usage, and increase simulation speed.

IV.1.2 Test Objectives and Schedule

In support of the aim of our test process, we will generate test coverage for the following:

- Subcomponent Testing: Low-level testing for non-trivial algorithms providing a single function.
- Component Testing: Middle-level testing for components composed of two or more subcomponents.
- System Testing: High-level testing for the final expected outputs of the application.

The above items create an end-to-end testing foundation for the *Metal Whiskers Unity App* in efforts to ensure a robust, high-quality codebase. Our team will generate tests before making any changes to the program for a *test-driven development approach* (TDD).

The TDD approach sets the foundation for a continuous integration and continuous deployment (CI/CD) pipeline. CI/CD prevents incorrect code from merging with production code by alerting a user if a test fails. Developers are more likely to catch logical errors which may impact overall performance and/or simulation accuracy.

Finally, we will develop documentation in support of tests. This creates a knowledge base and pathway to modify and further build the application. It is for these reasons our team, Ram Logic, will adopt a test-driven approach.

IV.1.2.1 Required Testing Software Resources

To facilitate end-to-end testing, our team will generate unique tests with the *Unity test framework*. For CI/CD we propose the free third-party software, *GameCI*. Since Unity's CI/CD proprietary framework requires payment our team cannot provide, *GameCI* is greatly preferred.

IV.1.2.2 Milestones and Deliverables

With software requirements and an outline of our team's high-level plan, the deliverables are as follows: (1) unit tests, (2) integration tests, (3) system tests, and (4) documentation. Two milestones encapsulate these deliverables as below:

1. Test Generation:
    a. Produce tests at every level.
    b. Produce rough documentation for internal use.
2. Test Documentation: Produce quality documentation of testing for an industry professional without formal education in a field related to computer science, software engineering.

Our team will develop tests and documentation simultaneously. This approach prevents loss of information as tests develop and provides the client with a clear outline of our team's motivations as developers.

After the Test Generation Milestone is complete, our team will dedicate efforts to refine preliminary documentation by rephrasing overly technical and/or abstract language. This may ease comprehension difficulty for professionals without formal education in some field related to software engineering or computer science.

IV.1.3 Scope

This section provides our team's test procedure which is to ensure software quality, detect bugs, and provide shape to project requirements from client outlines.

IV.2 Testing Strategy

The testing strategy proposes *Ram Logic* implements continuous integration and continuous deployment (CI/CD) for a robust, test-driven framework. Currently, the tin whisker simulator lacks unit, integration, and system tests. Test-coverage across the codebase will likely improve the logical quality and reliability of code. CI/CD may also provide another layer to protect the main codebase branch against logically incorrect and unreliable code.

To give context, the *Metal Whisker Unity 3D App* currently implements the following:

- Metal Whisker Generator: Widths, lengths, and spawn positions, are log-normal, log-normal, and uniformly random, respectively, to generate a multivariate stochastic distribution model of metal whiskers. Whisker positions can only exist within the bounded 3D spawn box a user defines.
- Unity Physics Engine: As the metal whisker set generator creates whiskers, the Unity Physics Engine applies gravity and the forces of collision to these objects.
- PCB Parser and Loader: Parses and abstracts a blender file to graphically and logically represent a PCB.
- Flags for Conductive Materials: Allows users to manually define conductive components on a PCB.
- Short Detector: Continuously detects shorts across conductive components for the duration of a simulation.
- Monte-Carlo Simulator: Implements a simulation method to define PCB risk as a set of probabilities.
- UI and Graphics Display: Uses the Unity framework to create UI and provide simulation graphics.

The team will first generate tests of the items above before tests to support supplementary functionality.

IV.3 Test Plans

IV.3.1 Unit Testing

Unit tests aim to take indivisible units of testable code, isolate them, and test for bugs and/or unexpected behavior.

IV.3.2 Integration Testing

Integration testing in Unity has limitations. The Unity Engine requires the team to isolate and test code in clusters rather than all components simultaneously. A developer may

need to determine whether certain scripts can integrate with others for testing, otherwise the developer may decide to test a script independently.

IV.3.3 System Testing

IV.3.3.1 Functional testing:

The team's functional testing plan uses tests developers generate. We aim to create test-coverage for items in the *II. Requirements and Specifications* section outlines. Each requirement will correspond to at least one functional test to validate behavior and logic.

Note: If tests on certain Unity Engine functions fail, then the responsible developer will document the test conditions and request corrections from the original developer.

IV.3.3.2 Performance testing:

Performance testing will utilize the *Unity Profiler* tool to measure key metrics including CPU and memory usage. The *Tin Whiskers Unity 3D App* generates a multivariate stochastic distribution set of metal whiskers and applies physics to these whisker objects, thus the items in the outline below hold importance by descending order:

1. Gauge the *Physx Engine's* floating point precision to determine information loss in results.
2. Optimize logical operations in the *Unity* application for improved speed and memory usage.

The team has tested the application under different configurations to evaluate the speed of computation against PCB complexity and Monte-Carlo total simulations in an unfinished draft within the following report:

https://github.com/WSUCptSCapstone-S25-F25/-mda-unity3dapp-/blob/main/GPU_physics_proposal.pdf

Note: *The document above is unfinished and is subject to change.*

IV.3.3.3 User Acceptance Testing:

User acceptance testing (UAT) will involve collaboration with industry professionals to evaluate the application's usability and accuracy. UAT is an iterative process in which our team collects and integrates industry professional feedback into subsequent development cycles.

Key aspects to be assessed include (1) simulation accuracy, (2) robustness and usability of UI, and (3) clarity in output visualization. The UAT process will ensure the application meets customer expectations and can deploy.

IV.4 Environment Requirements

The figure, *Fig. 1. Unity 6 Environment Requirements.*, below defines software and hardware requirements as defined by *Unity Technologies*:

Operating system	Operating system version	CPU	Graphics API	Additional requirements
Windows	Windows 10 version 21H1 (build 19043) or newer	X64 architecture with SSE2 instruction set support, ARM64	DX10, DX11, DX12 or Vulkan capable GPUs	Hardware vendor officially supported drivers
macOS	Big Sur 11 or newer	X64 architecture with SSE2 instruction set support (Intel processors) Apple M1 or above (Apple silicon-based processors)	Metal-capable Intel and AMD GPUs	Apple officially supported drivers (Intel processor) Rosetta 2 is required for Apple silicon devices running on either Apple silicon or Intel versions of the Unity Editor
Linux	Ubuntu 22.04, Ubuntu 24.04	X64 architecture with SSE2 instruction set support	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs	Gnome desktop environment running on top of X11 or Wayland windowing system, Nvidia official proprietary graphics driver, or AMD Mesa graphics driver. Other configuration and user environment as provided stock with the supported distribution (Kernel, Compositor, etc.) **Notes:** • **Ubuntu 22.04:** Wayland is supported with AMD graphics cards. • **Ubuntu 24.04:** Wayland is supported with AMD graphics cards and Nvidia graphics cards utilizing Nvidia proprietary graphics drivers 550 and above.

Fig. 1. Unity 6 Environment Requirements. [7]

GLOSSARY

Computer-Aided Design (CAD): Software allowing users to create, modify, and analyze digital models of physical objects.

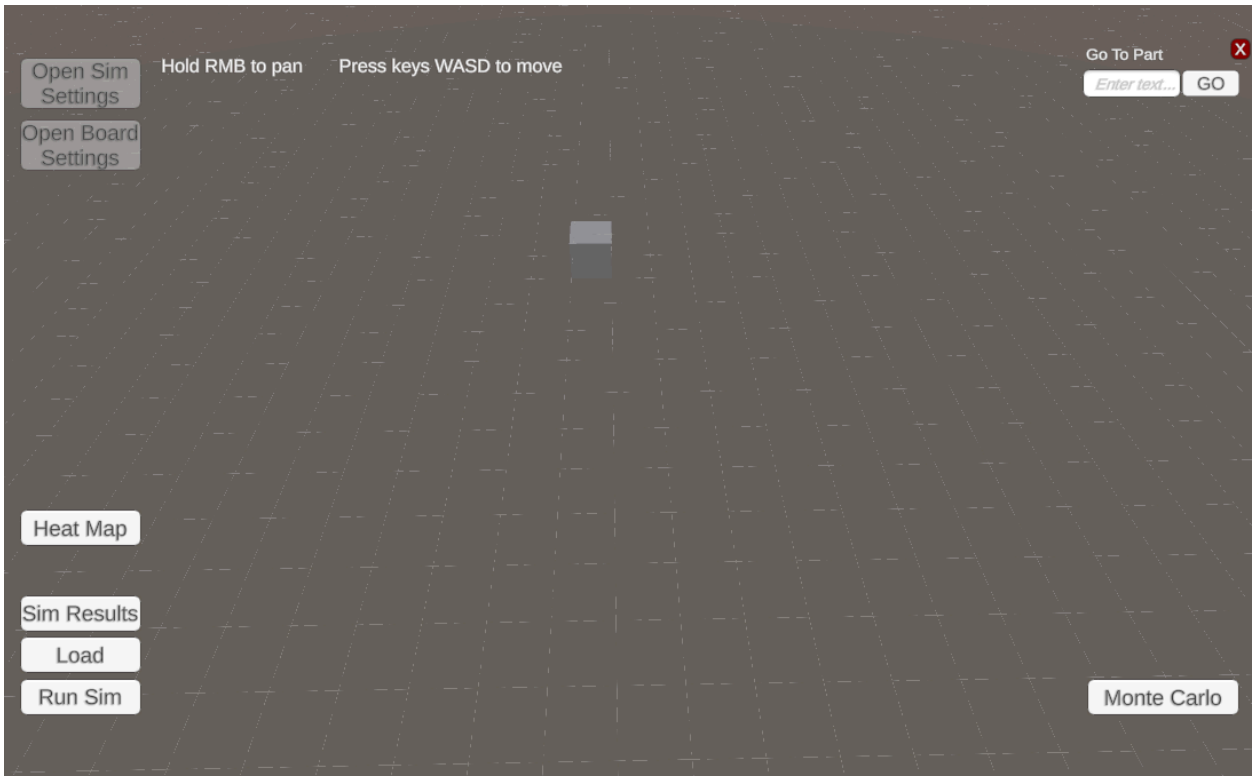
Continuous Integration and Continuous Deployment (CI/CD): Software development practice that automates the integration, testing, and delivery of code, accelerating deployment.

Metal Whiskers: Microscopic, hairlike protrusions that spontaneously grow from metal surfaces, specially tin, zinc, cadmium, or other metals.

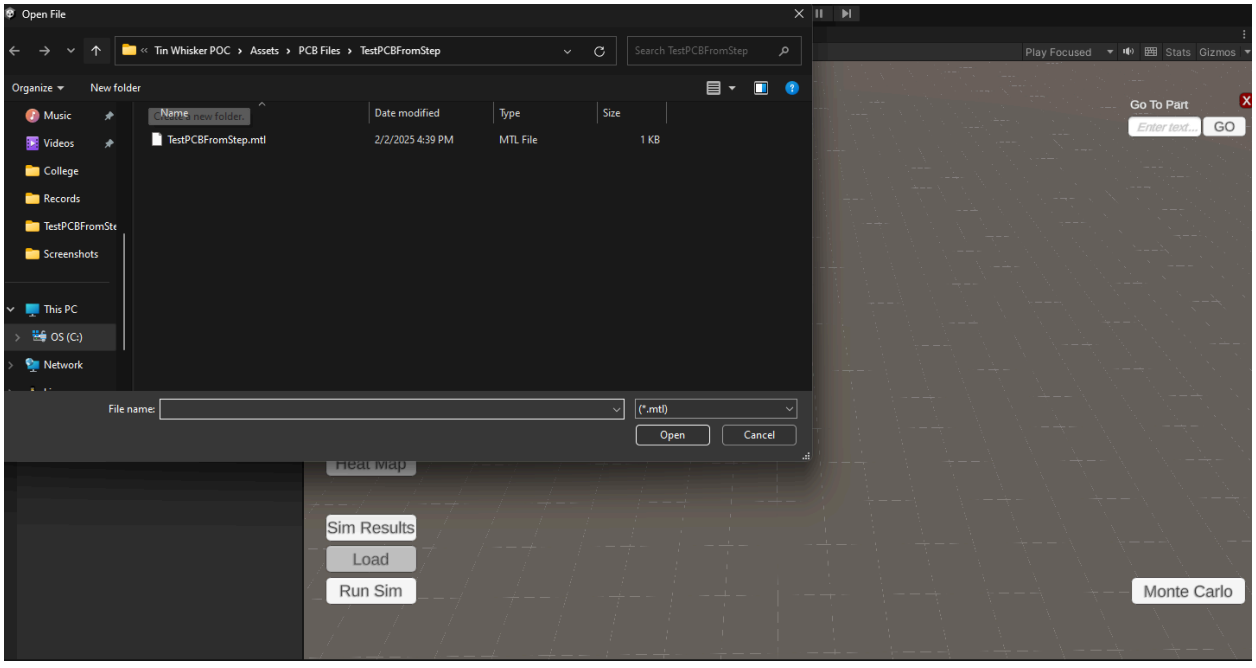
Multivariate Stochastic Distribution Modeling: Interdependent set of variables dependent on multiple distributions.

Printed Circuit Board (PCB): Flat, rigid board providing mechanical support and electrical connections for electronic components.

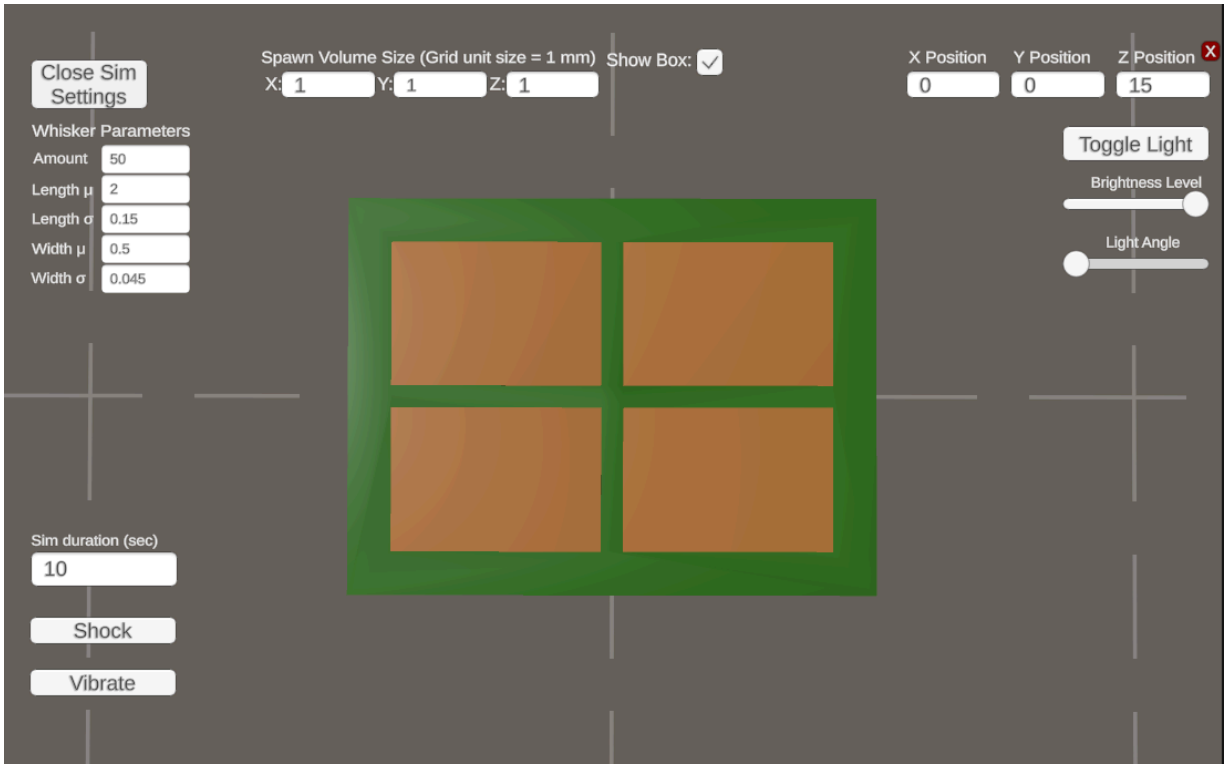
APPENDIX



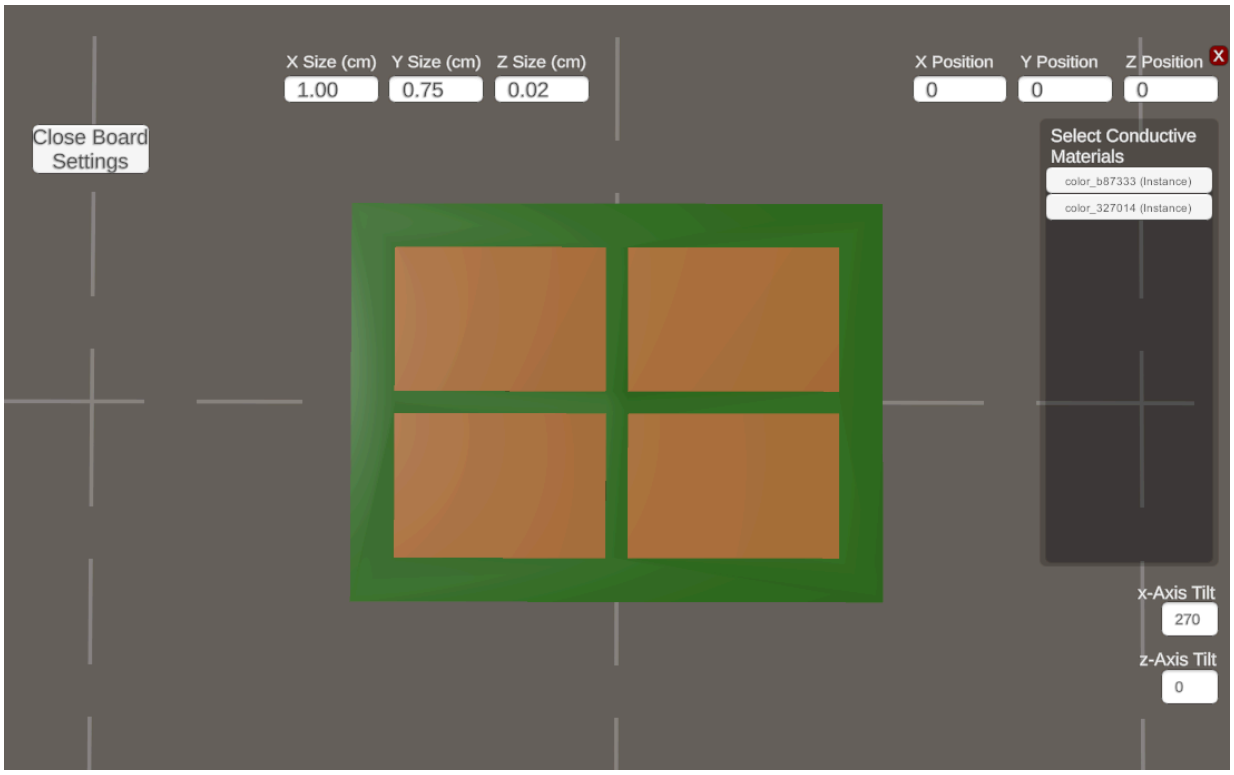
(Image 1)



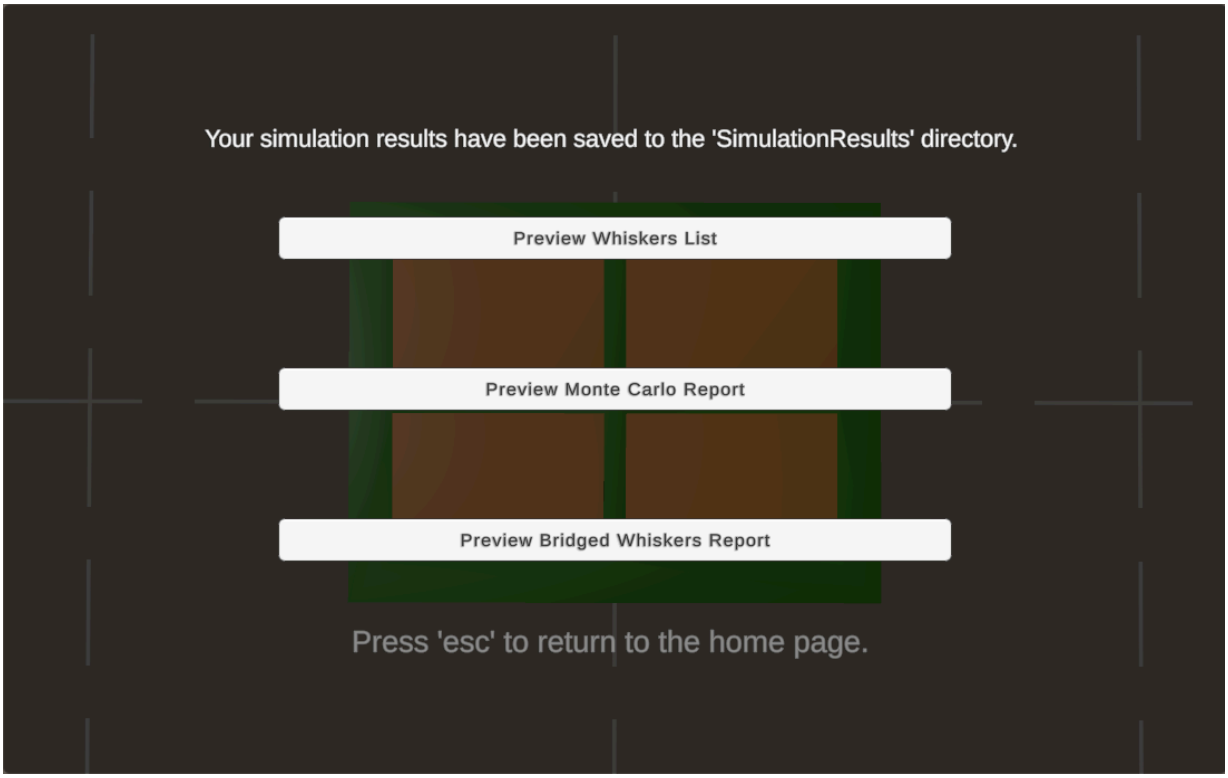
(Image 2)



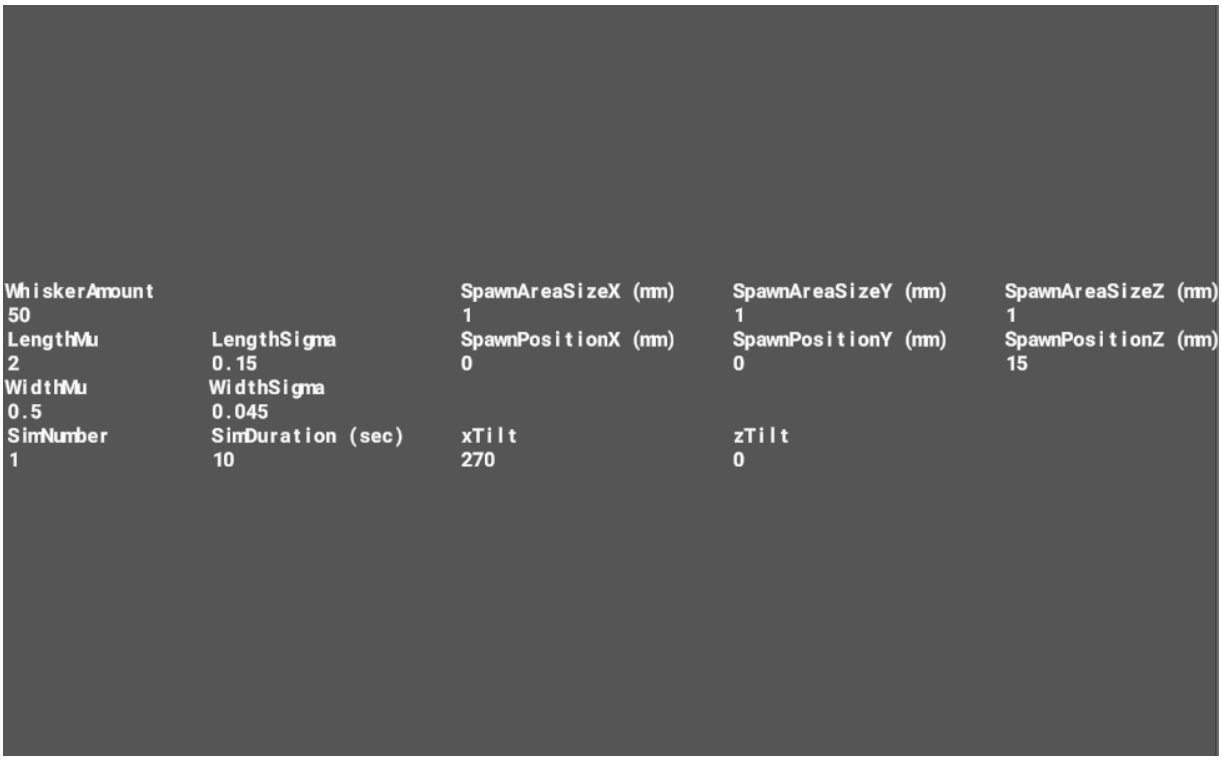
(Image 3)



(Image 4)

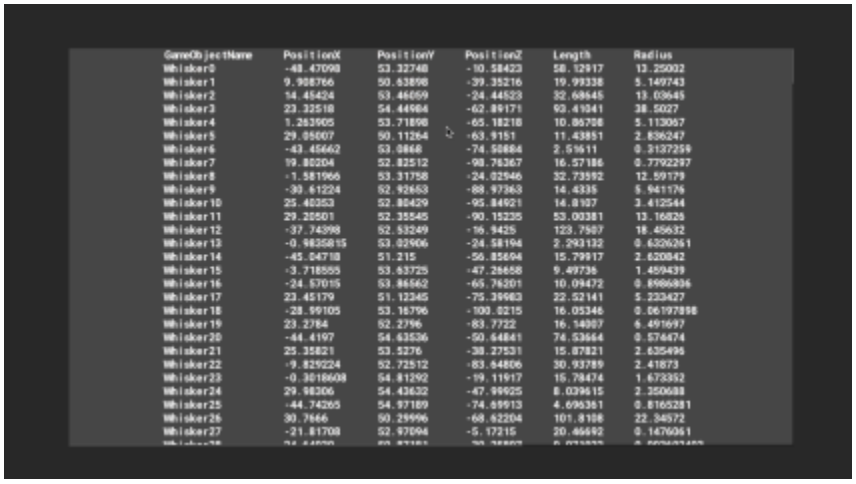


(Image 5)



(Image 6)

(Image 7)



GameObjectName	PositionX	PositionY	PositionZ	Length	Radius
Whisker0	-48.47098	53.32748	-90.58423	58.12917	13.25032
Whisker1	9.908766	50.63898	-39.35216	19.99338	5.149743
Whisker2	14.45424	53.46059	-24.44523	32.68645	13.03645
Whisker3	23.32518	54.44984	-62.89171	93.41041	38.5027
Whisker4	1.263905	53.71898	-65.18218	10.86708	5.113057
Whisker5	29.05007	50.11264	-63.9151	11.43851	2.836247
Whisker6	-43.45662	53.0868	-74.50884	2.51611	0.3137259
Whisker7	19.80204	52.82512	-90.76367	16.57186	0.7792297
Whisker8	-1.581966	53.31758	-24.02946	32.73592	12.59179
Whisker9	-30.61224	52.92653	-88.97363	14.4335	5.941176
Whisker10	25.40353	52.80429	-95.84921	14.8107	3.412544
Whisker11	29.20501	52.35545	-90.15235	53.00381	13.16826
Whisker12	-37.74398	52.53249	-16.9425	123.7587	18.45632
Whisker13	-0.9835815	53.02906	-24.58194	2.293132	0.6326261
Whisker14	-45.04718	51.215	-56.85694	15.79917	2.620842
Whisker15	-3.718555	53.63725	-47.28658	9.49736	1.459439
Whisker16	-24.57015	53.86562	-65.76201	10.09472	0.8986836
Whisker17	23.45179	51.12345	-75.39983	22.52141	5.233427
Whisker18	-28.99105	53.16796	-100.0215	16.05346	0.06197898
Whisker19	23.2784	52.2796	-83.7722	16.14007	6.491697
Whisker20	-44.4197	54.63536	-50.64841	74.53664	0.576474
Whisker21	25.35821	53.5276	-38.27531	15.87821	2.635496
Whisker22	-9.829224	52.72512	-83.64806	30.93789	2.41873
Whisker23	-0.3018609	54.81292	-19.11917	15.78474	1.673352
Whisker24	29.98306	54.43432	-47.99925	8.039615	2.350688
Whisker25	-44.74265	54.97189	-74.59913	4.696361	0.8165281
Whisker26	30.7666	50.29996	-68.82204	101.9108	22.34672
Whisker27	-21.81708	52.97094	-5.17215	20.46692	0.1476061

(Image 8)

REFERENCES

[1]“NASA Goddard Tin Whisker Homepage,” nepp.nasa.gov. https://nepp.nasa.gov/whisker

[2]NASA “Examples of Lognormal Distributions of Tin Whisker Lengths and Thicknesses” https://nepp.nasa.gov/whisker/reference/2009Panashchenko_Log_Normal_L_and_T.pdf

[3]On-Orbit COMMERCIAL (non-NASA) Satellite Failures https://nepp.nasa.gov/whisker/failures/index.htm#satellite

[4]H. Leidecker, L. Panashchenko, J. Brusse, "Electrical Failure of an Accelerator Pedal Position Sensor Caused by a Tin Whisker and Investigative Techniques Used for Whisker Detection", 5th International Tin Whisker Symposium, Sept. 2011 https://nepp.nasa.gov/whisker/reference/tech_papers/2011-NASA-GSFC-whisker-failure-app-sensor.pdf

[5]“Galaxy IV,” wikipedia.org. https://en.wikipedia.org/wiki/Galaxy_IV.

[6]“Tin Whiskers: Unity 3D App,” github.com. https://github.com/WSUCptSCapstone-S24-F24/-mda-unity3dapp-/blob/main/Project%20Report%20Final.pdf

[7]Unity Technologies, “Unity - Manual: System requirements for Unity 6,” Unity3d.com, 2019. https://docs.unity3d.com/6000.0/Documentation/Manual/system-requirements.html