PAPER • OPEN ACCESS

Integration of JSBSim and Unreal Engine for Flight Simulator Development: A Case Study on the Cessna T-37

To cite this article: Mohamed Awd Saber *et al* 2025 *J. Phys.: Conf. Ser.* **3070** 012009

Integration of JSBSim and Unreal Engine for Flight Simulator Development: A Case Study on the Cessna T-37

Mohamed Awd Saber[1*], **Mohamed Y. Zakaria**[2], **Ashraf M Kamal**[3]

[1] M.SC. Student, Aircraft Mechanics Department, Military Technical College, Cairo, Egypt.
[2] Associate Professor, Aircraft Mechanics Department, Military Technical College, Cairo, Egypt.
[3] Assistant Professor, Aircraft Mechanics Department, Military Technical College, Cairo, Egypt.
[*]E-mail: mohamed_mouter@yahoo.com

Abstract. This study presents systematic procedures for the development of a cost-effective flight simulator by integrating multiple software tools into a unified simulation environment. The simulator framework combines both Microsoft Flight Simulator and Blender for 3D realistic CAD model generation, JSBSim as the aerodynamics and flight dynamics model, and Unreal Engine as the primary simulation platform. The Cessna T-37 aircraft is used as a case study to develop the simulator. The development process includes the preparation of the 3D CAD model, the integration of the aerodynamic data, and real-time control surface animations, ensuring an immersive and accurate representation of the aircraft behavior. The JSBSim is implemented to handle the flight dynamics while Unreal Engine is used for visualization and interactivity. Additionally, the simulation includes an analysis of the aircraft's natural modes, such as phugoid, short-period, rolling, spiral and dutch-roll modes, to assess its dynamic stability. The response characteristics of these modes are examined to ensure consistency with theoretical expectations and to identify nonlinear effects present in the simulation. The results demonstrate the feasibility of integrating open-source tools for flight simulation and highlight the efficiency of the proposed cost-effective framework in replicating realistic flight performance.

1 Introduction

1.1 Background and Importance of Flight Simulation

The origins of flight simulation date back to the early 20th century, beginning with basic mechanical devices designed to provide pilots with limited training in flight mechanics [1]. As aviation technology evolved, flight simulators became increasingly sophisticated. A significant breakthrough came in the 1960s with the introduction of computer technology, which allowed for more precise and immersive simulations [2].

Currently, flight simulation is crucial in aviation industry as it is used in a variety of applications, ranging from fundamental pilot training to advanced aeronautical research and development. Modern simulators incorporate high-fidelity graphics, accurate flight dynamics, and immersive environments capable of replicating diverse weather conditions and flight scenarios. Such capabilities allow enhancing safety, improving training efficiency, and making aviation more accessible. Flight simulation offers several key advantages that make it an essential tool in aviation. It provides a controlled and risk-free environment where pilots can practice essential skills and handle emergency scenarios, allowing them to refine their decision-making abilities without real-world dangers. Compared to actual flight training, simulators significantly reduce costs by eliminating expenses related to fuel, aircraft maintenance, and operational wear, making pilot training more affordable and accessible. Additionally, training in simulators enhances aviation safety by preparing pilots to manage critical situations and emergencies more effectively. Beyond training, flight simulators play a crucial role in aerospace research and innovation, enabling engineers to evaluate new aircraft designs, analyze performance, and study aerodynamics without requiring physical prototypes. They also increase accessibility to aviation, allowing aspiring pilots to experience flying from home, fostering interest in the field, and helping individuals grasp fundamental flight concepts before formal training. Even experienced pilots benefit from simulation-based training, as it allows for periodic skill development and ensures they remain proficient with evolving technologies, regulations, and procedures.

1.2 Literature Review

This section examines several flight simulators designed for different user groups, from aviation enthusiasts to professional pilots and military personnel. These simulators vary in fidelity, customization, and accessibility, serving distinct training and operational needs. By assessing key factors such as flight dynamics modeling, realism, graphics quality, and FAA certification, this analysis provides a comprehensive overview of their capabilities [3][4][5][6][7].

Microsoft Flight Simulator (MSFS) 2020 excels in graphical fidelity, real-world mapping through Bing Maps, and real-time weather simulation via Azure, offering a highly immersive experience for general aviation. X-Plane 12, using Blade Element Theory (BET) for real-time aerodynamic calculations, provides the most accurate physics-based flight dynamics, making it suitable for FAA-certified training. DCS (Digital Combat Simulator) remains the best option for military applications, utilizing wind tunnel data, computational fluid dynamics (CFD), and flight test validation to achieve high-fidelity aerodynamics, particularly for fighter jets.

Prepar3D and Flight Simulator X (FSX), while widely used for training, rely on older lookup table-based architectures, limiting their adaptability to changing aerodynamic conditions. Prepar3D remains a stable choice for structured training, while FSX benefits from an extensive add-on ecosystem. FlightGear, as an open-source alternative, offers extensive customization with multiple flight models (JSBSim, YASim), though its graphical quality and real-time weather features are less advanced.

The choice of Flight Dynamics Model (FDM) significantly impacts realism. X-Plane 12 dynamically calculates aerodynamic forces per surface section, ensuring highly accurate flight physics. DCS and MSFS 2020, with CFD-based modeling, enhance airflow and turbulence realism. Prepar3D and Falcon BMS use precomputed lookup tables, prioritizing stability over adaptability. FlightGear's flexibility allows users to select between data-driven and geometry-based solvers.

Studies such as [6] have investigated flight dynamics through modeling, simulation, and flight testing, particularly for unconventional configurations like tailless UAVs. Similarly, [7] validated small airplane flight dynamics models, reinforcing the importance of accurate aerodynamic modeling in flight simulation research.

DCS excels in combat physics, while X-Plane 12 is preferred for civil flight accuracy. The choice of simulator depends on its intended use—training, research, or entertainment. Advances in flight modeling, weather simulation, and control systems continue to enhance realism in aviation and aerospace research.

Table 1. Comparison of Available Flight Simulators

Feature	**FlightGear**	**Prepar3D**	**DCS**	**Microsoft Flight Simulator (MSFS) 2020**	**X-Plane 11/12**	**Flight Simulator X (FSX)**
Open-Source	Yes	No	No	No	No	No
Customization	High (fully modifiable)	Moderate (Software Development Kit (SDK) available)	Limited (restricted by developers)	High (SDK + add-ons)	High (SDK available)	High (SDK & SimConnect)
Graphics Quality	Moderate	Moderate	High	Very High	High	Moderate
Weather Simulation	Moderate	Moderate	High	Advanced (real-time, Azure)	Advanced (volumetric clouds)	Moderate
Flight Dynamics Model (FDM)	JSBSim, Yet Another Simulator (YASim)	Table-based (FSX legacy)	Wind tunnel, Computational Fluid Dynamics (CFD), and flight test data	Computational + empirical (CFD voxel grid, lookup tables)	Blade Element Theory (BET)	Table-based
Methodology	JSBSim: Data-driven (wind tunnel/CFD), YASim: Geometry-based solver	Empirical lookup tables	Wind tunnel, CFD, and flight test data	CFD-based modeling with lookup tables	Divides surfaces into elements, calculating forces per section	Empirical (lookup tables)
Turbulence & Non-linear Flow	YASim models airflow iteratively; lacks high-fidelity turbulence	Precomputed datasets, lacks real-time turbulence modeling	CFD for specific aircraft, models dynamic airflow	Simulates wake turbulence and airflow	Simplified turbulence modeling, lacks real-time CFD	No real-time CFD, relies on precomputed datasets
Realism & Accuracy	User-defined models, highly flexible	Stable and widely used for professional training	High fidelity, especially for combat aircraft	High realism for general aviation, simplified control surface modeling	Realistic for general aviation, but lacks high-fidelity combat dynamics	Large add-on ecosystem, but outdated flight physics
Failure Modeling	Fully customizable	Moderate	Advanced	Limited	Customizable but limited	Basic
Loading Speed	Fast	Moderate	Slow	Moderate	Slow	Moderate
Ease of Use	Complex (Extensible Markup Language (XML) edits)	Moderate (User Interface (UI))	Moderate	Easy (intuitive UI)	Moderate (detailed UI)	Moderate
Scenery Coverage	Global (basic)	Global (add-ons)	Limited (combat zones)	Global (Bing Maps, real-time)	Global (detailed terrain)	Global (low detail)
Multiplayer	Available	Available	Yes (Player vs. Player (PvP), Co-op)	Available	Available	Available
MATLAB/Simulink Support	Yes	Yes (SimConnect)	No	Yes (SDK support)	Yes (User Datagram Protocol (UDP) communication)	Yes (SimConnect API (Application Programming Interface))
Federal Aviation Administration (FAA) Certification	No	Yes (training)	No	No	Yes (FAA-certified)	No
Main Strength	Open-source, research flexibility	Professional use, stable	Best for combat sim	Best visuals, real-world mapping	Best flight physics, FAA-cert	Large add-on ecosystem

1.3 Research Objective
The primary objective of this research is to develop a procedure for integrating JSBSim into Unreal Engine to create a real-time flight simulation environment. The study focuses on analyzing the resulting flight dynamics and evaluating the feasibility of using this platform for simulation in terms of realism, precision, and applicability for training or research purposes. By examining the behavior of the simulated aircraft, this research provides insights into the potential of game engines for professional flight simulators and identifies areas for further development and improvement.

2 Flight Simulator Development
The following flowchart illustrates the proposed methodology for developing a flight simulator in Unreal Engine 5 (UE5) using JSBSim as a flight dynamics model.

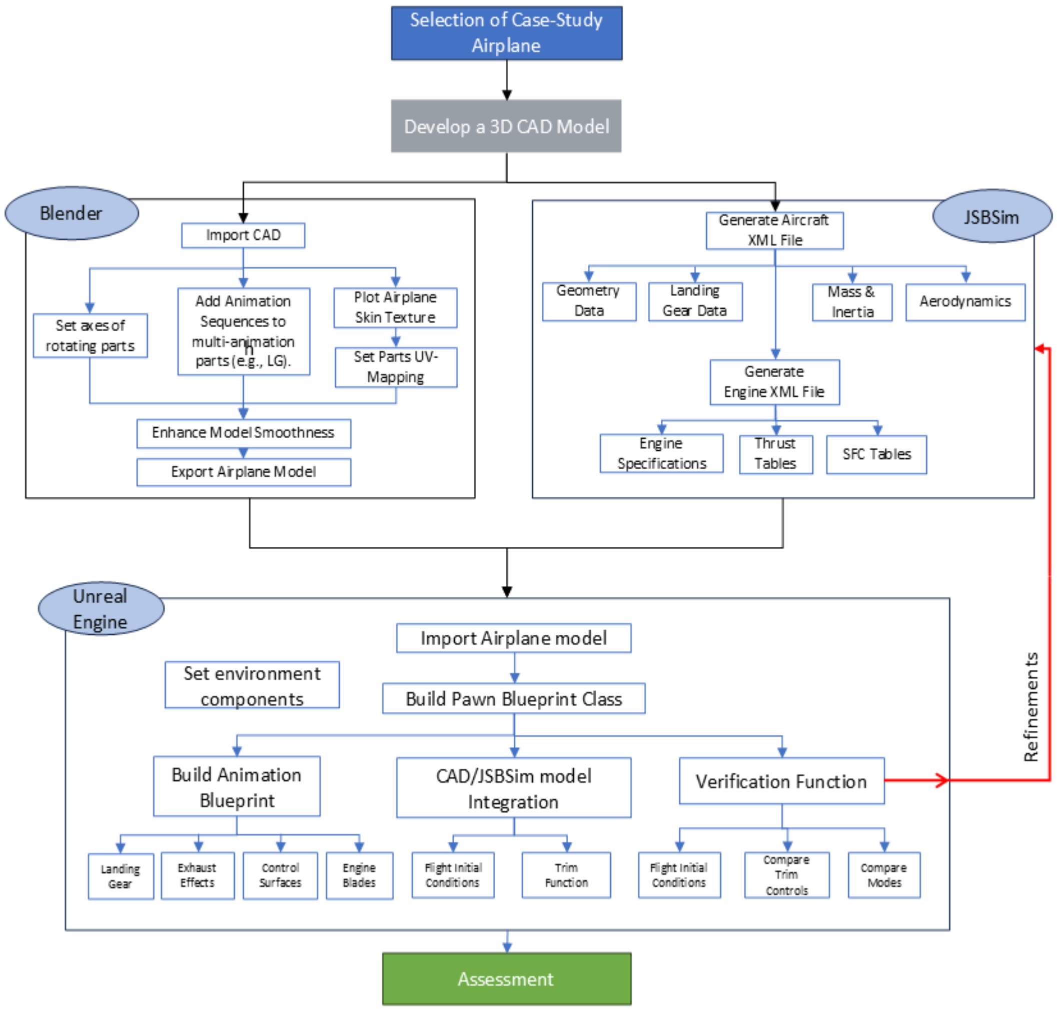


Figure 1. Flowchart for the Proposed Methodology to Develop a Flight Simulator

2.1 Airplane Selection
The selection of an aircraft with well-documented data is essential for accurate simulation. Among various options, the **Cessna T-37** is chosen, commonly known as the Tweet, a twin-engine jet trainer developed by the Cessna Aircraft Company. Introduced in 1955, the T-37 has been a cornerstone of military pilot training, providing hands-on experience in jet operations. Table 2 summarizes the main characteristics of the Cessna T-37 airplane.

Table 2. Main Characteristics of the Cessna T-37

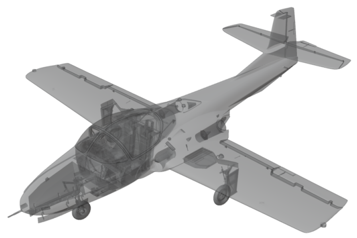
Characteristic	**Value**
Wingspan	33.83 ft (10.32 m)
Wing Area	182 ft^2 (16.91 m^2)
Mean Aerodynamic Chord	5.47 ft (1.67 m)
Horizontal Tail Area	31.77 ft^2 (2.95 m^2)
Horizontal Tail Arm	11.73 ft (3.57 m)
Vertical Tail Area	28.59 ft^2 (2.66 m^2)
Empty Weight	4,056 lbs (1,840 kg)
Maximum Weight	6,580 lbs (2,984 kg)
Moment of Inertia (I_{xx})	7,985 slug-ft^2
Moment of Inertia (I_{yy})	3,326 slug-ft^2
Moment of Inertia (I_{zz})	11,183 slug-ft^2
Engine Type	2 × Continental J69-T-25
Thrust per Engine	1,025 lbs (4.56 kN)
Fuel Capacity	1,400 lbs (635 kg)
Maximum Speed	425 mph (370 knots, 684 km/h)
Cruise Speed	350 mph (304 knots, 563 km/h)
Range	650 miles (1,046 km)
Service Ceiling	35,000 ft (10,668 m)

2.2 Aircraft 3D Model Development
Developing a high-fidelity 3D aircraft model is essential for accurate flight simulation. The model must capture both the aircraft's exterior and its functional components, including control deflections and system interactions. Typically, users create such models from scratch using 3D modeling software like Blender or Autodesk Maya, or they modify existing models obtained from online repositories or other simulators. In my case, I imported the T-37 aircraft model using the MSFS Importer plugin from Microsoft Flight Simulator (MSFS). This method was chosen because MSFS provides a highly detailed and visually accurate 3D model, reducing the need for extensive manual modeling while ensuring a high level of realism suitable for integration into Unreal Engine and JSBSim.

2.2.1 Importing the Cessna T-37 into Blender
The process began with importing a freeware Cessna T-37 model from Microsoft Flight Simulator (MSFS) using the MSFS glTF Importer plugin, a specialized tool that allows the extraction and conversion of MSFS aircraft models from the glTF (Graphics Library Transmission Format) used by the simulator. This plugin enables users to import and modify MSFS assets within 3D modeling software such as Blender, ensuring compatibility with other simulation platforms. Once imported, the model underwent geometric refinements, surface topology improvements, and optimizations to prepare it for seamless integration with Unreal Engine and JSBSim.

2.2.2 Texturing and Material Design
To enhance realism, existing textures are refined, and material properties are adjusted for accurate visual representation. Physically-based rendering (PBR) is used to achieve realistic reflections and lighting, while normal maps are applied to simulate fine surface details such as rivets and panel lines.

2.2.3 Animation
Dynamic animations ensure interactive realism in the simulator, including the proper rotation of control surfaces such as elevators, ailerons, rudder, and flaps. The landing gear follows accurate retraction and extension sequences, while the engine blades rotate correctly along their intended axis. These refinements allow the aircraft model to respond realistically to user inputs and environmental conditions.


(a) Model Detailed Components.

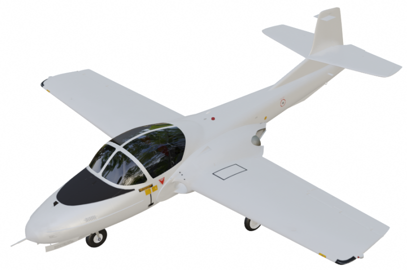

(b) Model with Applied Textures and Material Details.

Figure 2. Visuialization of Cessna T-37 Model Details

2.3 JSBSim Airplane Model
The process of inputting the flight model data into JSBSim, an open-source flight dynamics simulator, is presented [8]. JSBSim enables accurate aircraft simulation by incorporating parameters such as geometry, landing gear, mass, inertia, aerodynamics, and engine performance. This flight model can later be integrated with Unreal Engine for visualization and interaction.

In JSBSim, an aircraft XML(Extensible Markup Language) file defines the geometric, aerodynamic, mass, propulsion, and control characteristics necessary for simulation. Table 3 summarizes the key variables used to model the T-37 aircraft.

Table 3. Summary of Aircraft Data

Category	Description
Aircraft Geometry	Wing area, wingspan, mean aerodynamic chord, horizontal and vertical tail areas, moment arms, CG(Center of Gravity), Neutral Point.
Mass and Inertia	Total empty weight, moments of inertia (I_{xx}, I_{yy}, I_{zz}), CG location.
Landing Gear Data	Positions of nose and main landing gear, friction coefficients, spring and damping coefficients, retraction capability, and steering limits.
Propulsion System	Engine model file defining thrust characteristics, number of engines and their locations, fuel tanks with capacity and initial fuel levels.
Flight Control System	Control surface inputs and limits, deflections and limits of elevator, ailerons, rudder, actuation of flaps, landing gear, speed brakes, and trim settings.
Aerodynamic Properties	Lift and drag dependencies on angle of attack, Mach number, and control surface positions, pitching, rolling, and yawing moment coefficients, ground effect corrections.

2.4 Unreal Engine Simulator Development

2.4.1 Environment Components

The environmental setup in this Unreal Engine-based flight simulator is designed to enhance realism and ensure accurate geographic representation. The key elements include geo-referencing, high-resolution scenery, and dynamic weather simulation.

The GeoReferencing plugin is utilized to integrate real-world coordinates into the simulation. This plugin allows Unreal Engine to interpret latitude, longitude, and altitude data, ensuring precise positioning of 3D models within a global coordinate system. It converts geographic coordinates into Unreal Engine's world space, enabling accurate placement of scenery and terrain. In this project, the Cairo Airport 3D model was imported and aligned with real-world Geographic Information System(GIS) data using this method. The process involved defining a georeferenced origin, ensuring that the airport layout matched real-world terrain and navigation points.

Cesium for Unreal was employed to stream high-resolution satellite imagery and digital elevation models, enhancing terrain accuracy. The integration process involved importing the Cairo Airport model, enabling the GeoReferencing plugin, and setting its global reference point. This ensured that the airport's runways, taxiways, and surrounding structures were correctly positioned relative to real-world coordinates.

For atmospheric realism, a dynamic weather system was implemented to simulate real-time environmental changes. The Dynamic Weather plugin was configured to generate cloud formations, fog, and wind variations, affecting visibility and flight dynamics. The lighting system was adjusted to reflect time-of-day changes, ensuring accurate sun positioning and shadow casting.

By combining geo-referencing, high-resolution terrain data, and dynamic weather effects, the simulator achieves a high degree of realism, accurately representing the Cairo Airport environment under various atmospheric conditions.

2.4.2 Model Preparation

In this subsection, the process of preparing a detailed aircraft model for integration into Unreal Engine is outlined, along with the creation of animations and interactive elements using Unreal Engine's Blueprint system. The preparation process includes defining the model's geometry, animations, and textures, while Blueprint scripting ensures the aircraft's dynamic functionality and user control.

The aircraft model was imported as a glTF-2 file, a widely used format for 3D models that ensures compatibility with Unreal Engine without losing textures or materials. Upon import, the model was treated as a Skeletal Mesh, which means it includes a rigged skeleton structure that allows for animation. Along with the Skeletal Mesh, Unreal Engine also generated a Skeleton asset, which defines the bone structure, a Physics Asset, which handles collision and physics interactions, and additional components such as textures and animations.

The Skeletal Mesh was structured by segmenting the aircraft into multiple static meshes, including the fuselage, wings, tail, and landing gear. Each of these components was individually rigged with bones, enabling the animation of movable control surfaces such as ailerons, elevators, and rudders, as well as mechanical parts like landing gear retraction and engine fan rotation.

To achieve realistic visuals, the model's textures were carefully applied and verified. Texturing involves assigning realistic material properties to the model, allowing surfaces to reflect light naturally and create an immersive experience. Once the model and textures were refined, the next step was to integrate the aircraft into Unreal Engine using Blueprint scripting.

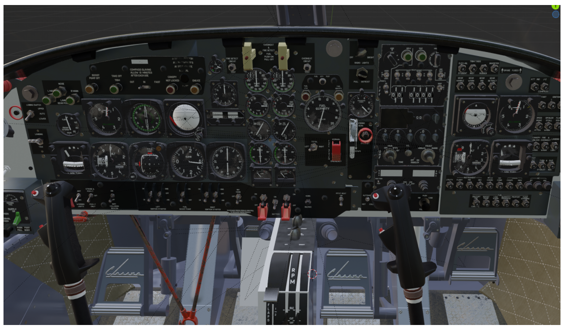
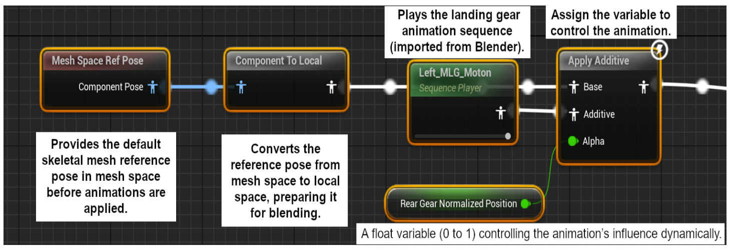


Figure 3. Cessna T-37 Cockpit Texture

2.4.3 Animation Blueprint Setup

An Animation Blueprint was created to control the animation of the aircraft's moving parts. An Animation Blueprint is a specialized scripting system in Unreal Engine that enables complex animation logic. It uses a State Machine, which is a graphical system that defines different animation states and transitions between them. For example, the state machine controls animations for the elevator, rudder, ailerons, flaps, and landing gear, ensuring that they respond dynamically to user inputs while respecting the aircraft's physical constraints.

Each animation is stored as an individual Animation Sequence as shown in Figure 4. For instance, the landing gear animation is represented as a normalized value ranging from 0 (fully retracted) to 1 (fully extended), allowing smooth transitions between states . In the other hand control surface deflections are defined by their respective input angles as shown in Figure 5.



Plays the landing gear animation sequence (imported from Blender).

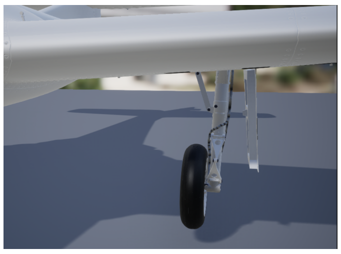
Assign the variable to control the animation.

Provides the default skeletal mesh reference pose in mesh space before animations are applied.

Converts the reference pose from mesh space to local space, preparing it for blending.

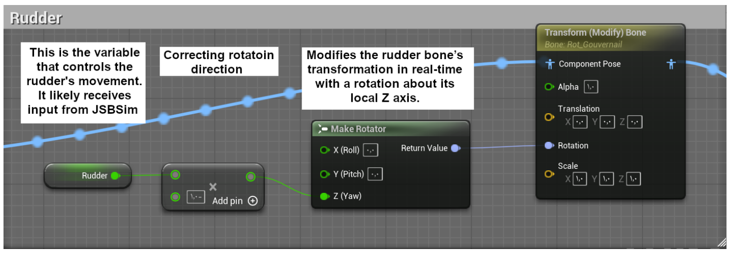
A float variable (0 to 1) controlling the animation's influence dynamically.

(a) LMG = 1



(b) LMG = 0.4

Figure 4. Left Main Landing Gear (LMG) animation blueprint flow.

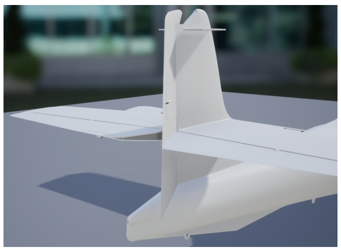


This is the variable that controls the rudder's movement. It likely receives input from JSBSim

Correcting rotatoin direction

Modifies the rudder bone's transformation in real-time with a rotation about its local Z axis.

(a) Rudder Angle = 30



(b) Rudder Angle = -30

Figure 5. Rudder animation blueprint flow.

2.4.4 Blueprint Class Implementation

The aircraft's interactive behavior was implemented using a Blueprint Class, a system in Unreal Engine that defines reusable objects with both visual and logical properties. The aircraft was assigned to a custom Blueprint Class derived from the Pawn class, which represents any controllable object. Within this class, the aircraft's Skeletal Mesh was added as a component, along with a Camera, Camera Spring Arm (for smooth camera movement), lights, exhaust effects, and the JSBSim flight model. These elements contribute to an accurate and visually realistic flight experience.

To simulate realistic physics, the JSBSim flight dynamics engine was integrated. JSBSim is an open-source model that calculates aerodynamic behavior using predefined derivatives stored in an XML file. The key flight parameters, including thrust, pitch, roll, and yaw, are governed by JSBSim and configured to respond dynamically to control inputs. Unreal Engine's input system maps user commands, allowing the aircraft to react to throttle, pitch, and yaw as expected.

The Animation Blueprint is linked to the Blueprint Class, ensuring that animations are triggered based on real-time flight data from the JSBSim Movement Component. This integration creates a seamless connection between the animation and physics systems, enabling realistic aircraft motion.

Additionally, an automated control system is implemented using a Data Table, which stores time-sequenced control inputs such as (Time, Throttle, Elevator, Rudder, Aileron). This system enables automated flight tests to evaluate the aerodynamic model's accuracy, analyze flight performance, and study dynamic stability.

2.4.5 Integrating JSBSim
To integrate JSBSim into Unreal Engine, several steps are performed to ensure proper flight dynamics simulation. First, the JSBSimMovement component is added to the Component Graph Editor within the blueprint pawn class, allowing the aircraft to respond to aerodynamic forces in real time.

Next, the necessary parameters are configured, including specifying the aircraft XML file containing aerodynamic data and performance characteristics. Finally, the JSBSim model is aligned with the aircraft's 3D model by correctly positioning the center of gravity (CG) and landing gear attachment points. Proper alignment ensures accurate physics calculations, enabling realistic aircraft movement.

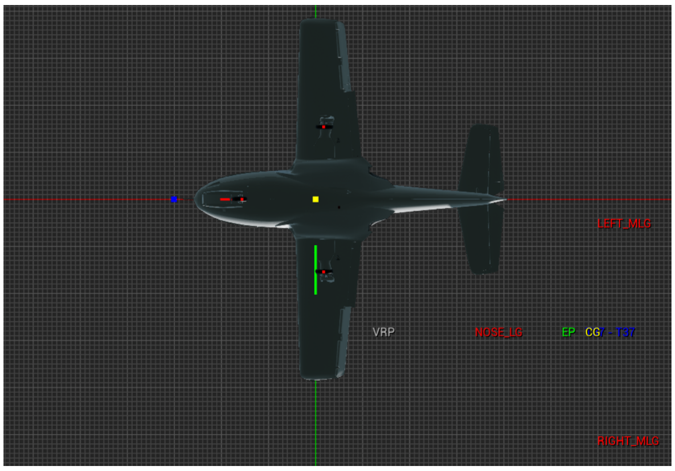
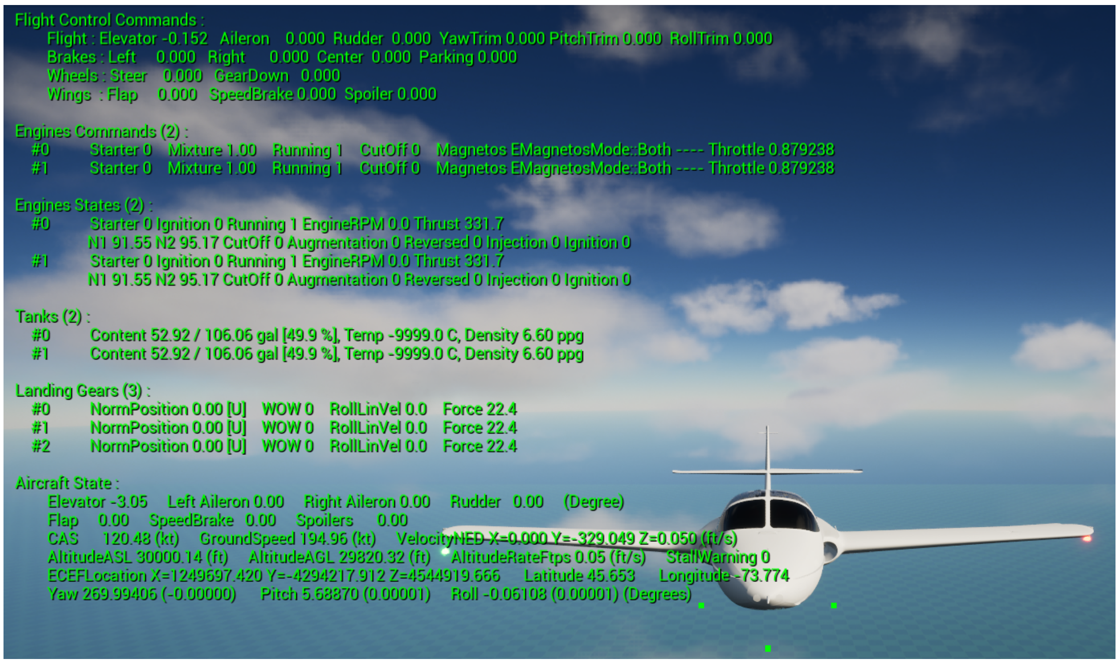


Figure 6. Alignment of Cessna T-37 with JSBSim Reference Points

3 Analysis of the Simulator Results
The analysis of the JSBSim-based Unreal Engine simulation focuses on evaluating flight dynamic characteristics with respect to common mode shapes, as described in [9]. This evaluation is conducted entirely within JSBSim, configured as the primary flight dynamics model in Unreal Engine. By analyzing the system's response, key flight dynamic characteristics are examined, categorized into longitudinal and lateral-directional stability.



```
Flight Control Commands :
    Flight : Elevator -0.152   Aileron    0.000  Rudder  0.000  YawTrim 0.000 PitchTrim 0.000  RollTrim 0.000
    Brakes : Left     0.000   Right      0.000  Center  0.000  Parking 0.000
    Wheels : Steer    0.000   GearDown   0.000
    Wings  : Flap     0.000   SpeedBrake 0.000  Spoiler 0.000

Engines Commands (2) :
  #0      Starter 0   Mixture 1.00   Running 1   CutOff 0   Magnetos EMagnetosMode::Both ---- Throttle 0.879238
  #1      Starter 0   Mixture 1.00   Running 1   CutOff 0   Magnetos EMagnetosMode::Both ---- Throttle 0.879238

Engines States (2) :
  #0      Starter 0 Ignition 0 Running 1 EngineRPM 0.0 Thrust 331.7
          N1 91.55 N2 95.17 CutOff 0 Augmentation 0 Reversed 0 Injection 0 Ignition 0
  #1      Starter 0 Ignition 0 Running 1 EngineRPM 0.0 Thrust 331.7
          N1 91.55 N2 95.17 CutOff 0 Augmentation 0 Reversed 0 Injection 0 Ignition 0

Tanks (2) :
  #0      Content 52.92 / 106.06 gal [49.9 %], Temp -9999.0 C, Density 6.60 ppg
  #1      Content 52.92 / 106.06 gal [49.9 %], Temp -9999.0 C, Density 6.60 ppg

Landing Gears (3) :
  #0      NormPosition 0.00 [U]   WOW 0   RollLinVel 0.0   Force 22.4
  #1      NormPosition 0.00 [U]   WOW 0   RollLinVel 0.0   Force 22.4
  #2      NormPosition 0.00 [U]   WOW 0   RollLinVel 0.0   Force 22.4

Aircraft State :
    Elevator -3.05    Left Aileron 0.00    Right Aileron 0.00    Rudder   0.00    (Degree)
    Flap    0.00    SpeedBrake  0.00    Spoilers     0.00
    CAS     120.48 (kt)    GroundSpeed 194.96 (kt)    VelocityNED X=0.000 Y=-329.049 Z=0.050 (ft/s)
    AltitudeASL 30000.14 (ft)    AltitudeAGL 29820.32 (ft)    AltitudeRateFtps 0.05 (ft/s)    StallWarning 0
    ECEFLocation X=1249697.420 Y=-4294217.912 Z=4544919.666    Latitude 45.653    Longitude -73.774
    Yaw 269.99406 (-0.00000)    Pitch 5.68870 (0.00001)    Roll -0.06108 (0.00001) (Degrees)
```

Figure 7. Steady-Level Flight Initial Conditions

3.1 Steady-Level Flight

In steady-level flight, all forces and moments are balanced. The trim condition is found by determining the trim angle of attack (α_{trim}) and elevator deflection ($\delta_{e,\text{trim}}$).In the other hand trim is accomplished in Unreal Engine using JSBSim built-in functions [10].

Equilibrium Conditions: The lift force must balance weight, and the pitching moment coefficient must be zero; hence, thrust must balance the drag force:

$$L = W = \frac{1}{2}\rho V^2 S C_L, \quad C_{L,\text{trim}} = \frac{2W}{\rho V^2 S}, \tag{1}$$

$$C_L = C_{L_\alpha}\alpha_{\text{trim}} + C_{L_{\delta e}}\delta_{e,\text{trim}}, \tag{2}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha_{\text{trim}} + C_{m_{\delta e}}\delta_{e,\text{trim}} = 0, \tag{3}$$

$$\alpha_{\text{trim}} = \frac{C_{L,\text{trim}}C_{m_{\delta e}} + C_{m_0}C_{L_{\delta e}}}{C_{L_\alpha}C_{m_{\delta e}} - C_{m_\alpha}C_{L_{\delta e}}}, \quad \delta_{e,\text{trim}} = -\frac{C_{L_\alpha}C_{m_0} + C_{m_\alpha}C_{L,\text{trim}}}{C_{L_\alpha}C_{m_{\delta e}} - C_{m_\alpha}C_{L_{\delta e}}}. \tag{4}$$

$$T = D = \frac{1}{2}\rho V^2 S C_D \cos(\alpha_{\text{trim}}), \quad C_D = C_{D_0} + C_{D_\alpha}\alpha_{\text{trim}}, \tag{5}$$

Figure 8 shows a steady-level flight condition, where altitude, velocity components, and attitude angles remain nearly constant over time. Small variations in pitch rate, pitch angle, and angle of attack indicate minor oscillations due to numerical trimming. This confirms that the aircraft is in a trimmed equilibrium state, maintaining stable flight dynamics without additional control inputs.

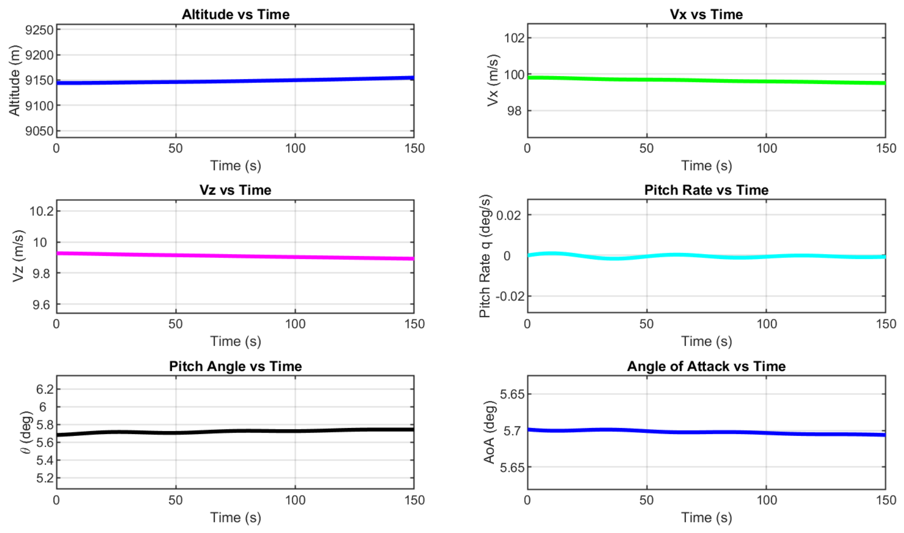


Figure 8. Steady-Level Flight Trace in Unreal Engine

3.2 Longitudinal Motion

The observed responses align with the expected behavior of the aircraft's natural modes. he initial elevator pulse induced a transient response that excited the phugoid mode, while the short-period mode was initiated exclusively by an elevator doublet. The phugoid mode, a low-frequency oscillation, is evident in the periodic variations of altitude and velocity, reflecting the interplay between kinetic and potential energy. Similarly, the short-period mode manifests in the rapid initial changes in pitch rate (q) and angle of attack (α), which quickly damp out. The system's response follows the logical progression dictated by aerodynamic stability, validating the expected control-input-to-motion dynamics.

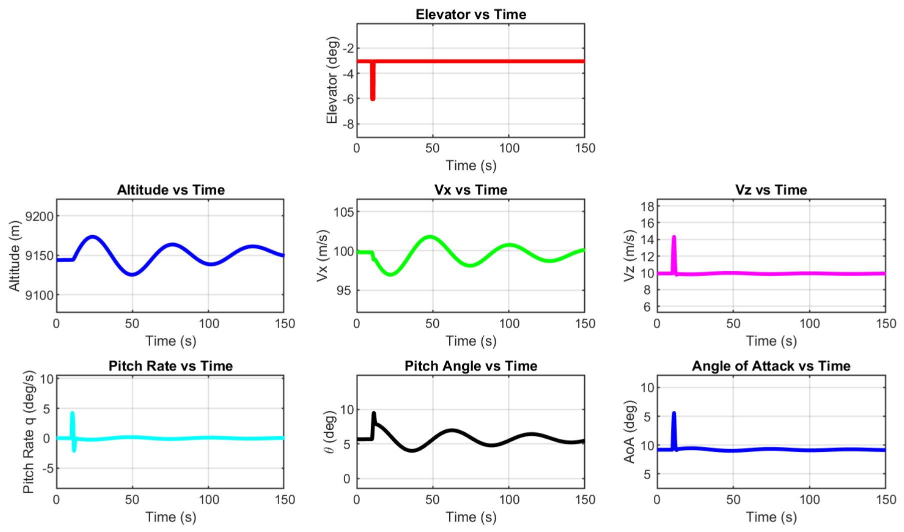


Figure 9. Phugoid Mode

As shown in Figure 9 The elevator pulse caused an initial sharp decrease in elevator deflection, leading to a transient response in the aircraft's motion. The altitude exhibited an oscillatory behavior, characteristic of a phugoid response, with periodic variations over time. The longitudinal velocity (V_x) also showed oscillations, while the vertical velocity (V_z) experienced a brief spike before stabilizing. The pitch rate (q) and pitch angle (θ) responded immediately to the elevator input, showing rapid initial changes followed by gradual damping. The angle of attack (AoA) increased momentarily before stabilizing, indicating the phugoid oscillations.

As shown in Figure 10 The elevator pulse induced a sharp initial change in the elevator deflection, leading to an immediate transient response in the aircraft's motion. The pitch rate (q) and pitch angle (θ) exhibited rapid variations before quickly stabilizing, characteristic of the short-period response. The angle of attack (α) momentarily deviated before settling, indicating a quick damping effect. The velocity components (V_x and V_z) showed minor transient oscillations, with vertical velocity experiencing a brief fluctuation before stabilization. The altitude demonstrated a slight initial drop due to the momentary downward motion before recovering as the oscillations damped out.

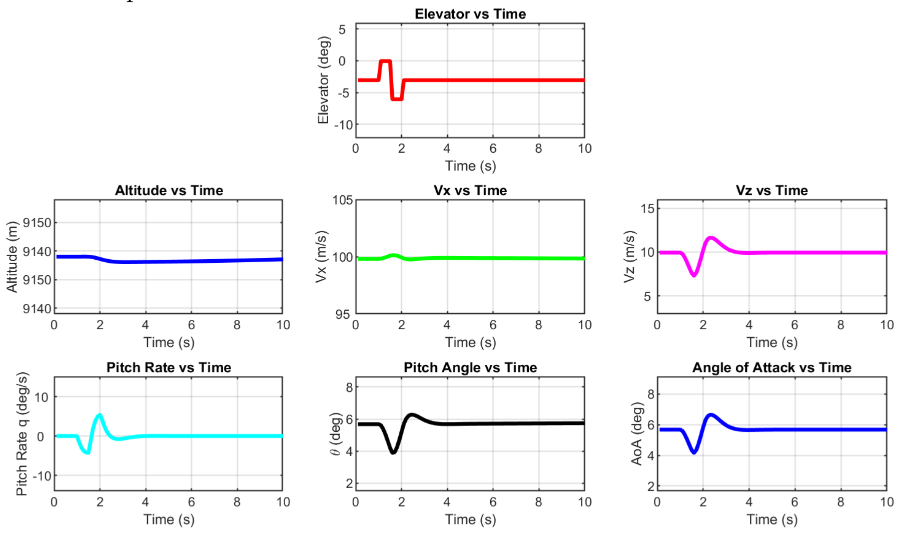


Figure 10. Short Period Mode

3.3 Lateral-Directional Motion

The simulation results exhibit responses that align closely with the expected dynamic characteristics of the aircraft's natural modes.

The Rolling mode, a high-frequency and heavily damped motion, demonstrated an immediate response to the aileron pulse input. This response was characterized by a sharp and sudden increase in roll rate (p), followed by a rapid decay due to strong aerodynamic damping. The significant damping ensures that roll oscillations subside quickly, preventing excessive lateral instability. As a result of coupling effects, transient oscillations were observed in the lateral velocity (V_y) and yaw rate (r). These secondary responses highlight the interconnected nature of lateral-directional motion, where roll perturbations influence yaw dynamics. Additionally, the bank angle (ϕ) initially experienced a sharp decrease before gradually stabilizing at a steady-state value, indicative of the aircraft's inherent roll stability. The yaw angle (ψ) exhibited minimal deviation throughout the response, reinforcing the notion that rolling motion primarily affects bank angle rather than heading direction. Furthermore, the sideslip angle (β) oscillated briefly, reflecting the aircraft's lateral stability characteristics. This short-lived sideslip motion, governed by sideslip damping and dihedral effect, was quickly attenuated, confirming the aircraft's ability to resist prolonged lateral disturbances. Overall, the rapid damping of these oscillations, particularly in roll rate and sideslip angle, validates the heavily damped nature of the rolling mode. This behavior is crucial for maintaining stable lateral control and ensuring that the aircraft remains responsive without exhibiting excessive roll-induced instability.

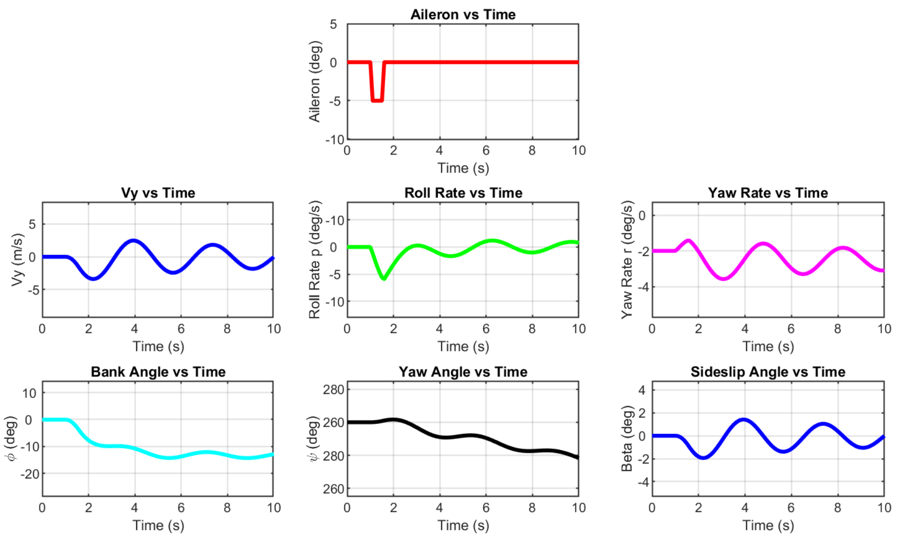


Figure 11. Rolling Mode

The Spiral mode, a low-frequency, weakly damped, or even unstable motion, was observed as a slow divergence in bank angle (ϕ), indicating a gradual rolling tendency over time. The roll rate (p) and yaw rate (r) exhibited initial oscillations before stabilizing, while the lateral velocity (V_y) and sideslip angle (β) demonstrated weak damping. The continuous change in yaw angle (ψ) further confirms the characteristic deviation of spiral mode.

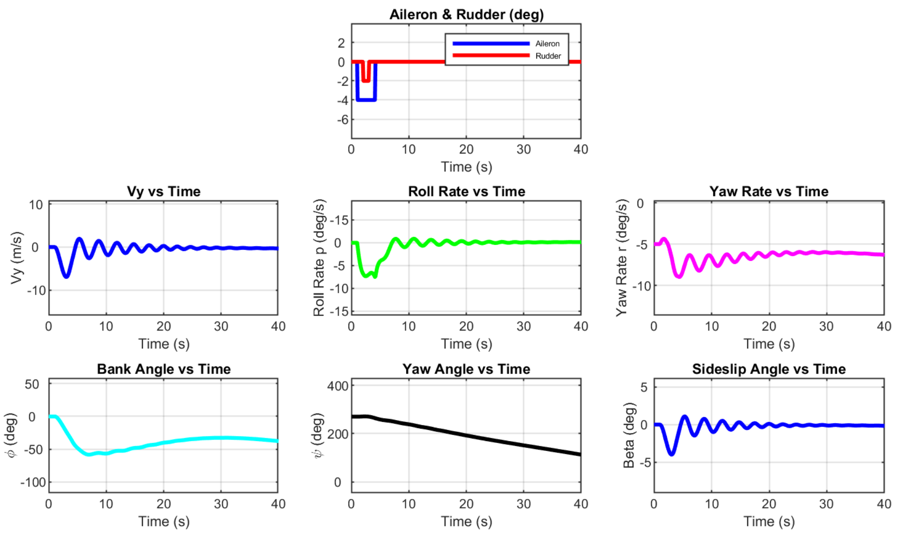


Figure 12. Spiral Mode

The Dutch-Roll mode, a moderate-frequency and lightly damped oscillation, emerged following an initial rudder input, leading to coupled yawing and rolling motion. The roll rate (p) and yaw rate (r) displayed sustained oscillations before gradually damping out, while lateral velocity (V_y), roll angle (ϕ), and sideslip angle (β) followed similar oscillatory patterns. The yaw angle (ψ) exhibited small oscillations, highlighting the expected yaw-roll coupling. The observed damping confirms the aircraft's natural aerodynamic stability in mitigating Dutch-Roll oscillations over time.

Overall, these results validate the logical aerodynamic response of the aircraft to corresponding control inputs, reinforcing the established characteristics of these lateral-directional dynamic modes.

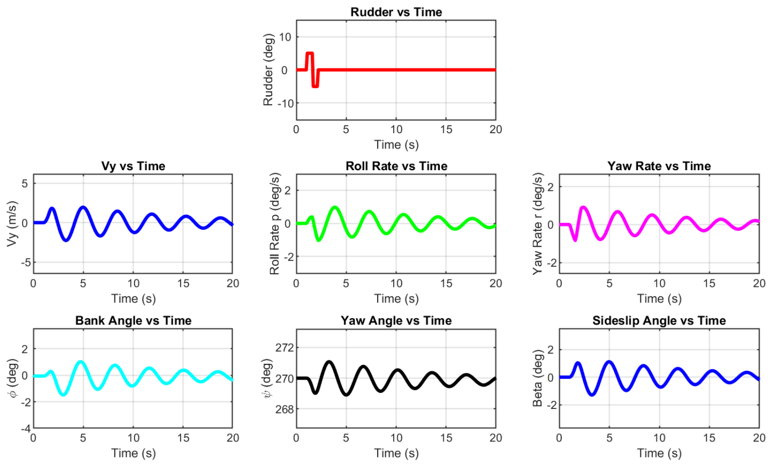


Figure 13. Dutch-Roll Mode

4 Conclusion

This study successfully demonstrates the integration of JSBSim with Unreal Engine to develop a cost-effective and immersive flight simulator for the Cessna T-37 aircraft. By leveraging open-source tools, the simulator provides a realistic flight experience, combining accurate flight dynamics modeling together with high-fidelity visualization. The verification of the developed simulator through the successful simulation of trim steady-level flight and the aircraft's natural flight modes (i.e., Phugoid, short period, spiral, roll, and Dutch roll) confirms its logical operation and consistency with theoretical expectations. The results validate the feasibility of using JSBSim within Unreal Engine for real-time flight simulation and highlight the potential of this framework for future applications in pilot training and flight dynamics research. Future work will focus on further verifying the simulator's accuracy by performing analytical calculations based on the nonlinear equations of motion and comparing the results with simulation outputs. Additionally, when flight test data becomes available, validation will be conducted by comparing the simulator's response with actual flight test results.

References

[1] Goldsman D, Nance R and Wilson J A brief history of simulation revisited 2010 pp 567–574

[2] Allerton D The impact of flight simulation in aerospace 2010 *Aeronautical Journal* **114** 747–756

[3] del Barrio L, Korek W, Millidere M and Whidborne J Analysis of visualization systems in flight simulators 2023

[4] Christensen C and Salmon J An agent-based modeling approach for simulating the impact of small unmanned aircraft systems on future battlefields 2022 *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*

[5] Do M H, Lin C E and Lai Y C Validation of the flight dynamics engine of the x-plane simulator in comparison with the real flight data of the quadrotor uav using cifer 2023 *Drones* **7** 548

[6] Ahmed N, Zakaria M Y and Kamal A M Investigating tailless uav flight dynamics through modeling, simulation, and flight testing 2024 *Unmanned Systems* 1–21

[7] Kamal A, Aly A M and Elshabka A Modeling, analysis and validation of a small airplane flight dynamics 2015 *AIAA Modeling and Simulation Technologies Conference* p 1138

[8] Kunz D L and Kim J P Evaluation of unmanned aircraft flying/handling qualities using a stitched learjet model 2021 *Journal of Guidance, Control and Dynamics*

[9] Napolitano M R 2011 *Aircraft Dynamics: From Modeling to Simulation* (John Wiley & Sons) ISBN 978-1-118-34491-7

[10] Millidere M, Karaman U, Uslu S, Kasnakoglu C and Cimen T Newton-raphson methods in aircraft trim: A comparative study 2020 *AIAA AVIATION Forum*