

Avionics HMI simulation Tool: AEROPVS037

VirtualSim provides commercial simulation infrastructure software, specifically its flagship product **vsTASKER**, which serves as a complete software development kit (SDK) and code-generation tool for the defense, aviation, and aerospace industries. Instead of writing code from scratch, developers use its graphical interface to design complex simulation scenarios, which vsTASKER then automatically converts into fully functional, scalable **C++ source code**.

Core Architecture & Capabilities

- **Automatic Code Generation:** It acts as a graphical paradigm tool where complex simulation logic is designed visually, producing optimized, maintainable C++ infrastructure code.
- **System Integration:** It is engineered primarily for system integrators and software engineers who need to deploy scalable simulation setups under strict real-time or technical constraints.
- **Open Architecture:** It allows users to study, animate, and visualize multi-domain scenarios of any size.
- **Aviation & Avionics Industry Use Cases**

The VirtualSim vsTASKER platform is utilized across several critical aerospace domains:

Man-in-the-Loop Trainers: Building the foundational logic for flight simulators, air traffic control (ATC) training bays, and radar operator stations.

Air Traffic Control (ATC): Implementing collision forecast modeling and system tracking tools to safely train future controllers.

Unmanned Aerial Vehicles (UAVs): Designing complex, large-scale Monte Carlo simulations to optimize UAV flight paths across millions of iterative software runs.

Common Alternatives in the Avionics SDK Space

If you are evaluating frameworks for modern avionics display or flight dynamic development, you might also cross-reference these alternatives depending on your precise project scope:

- **MSFS Avionics Framework:** A modern, open TypeScript and React-like JSX framework used specifically to develop fluid high-fidelity panel gauges and instrument logic within consumer flight simulators.
 - **JSBSim Flight Dynamics Model:** A sophisticated, open-source software-in-the-loop (SITL) physics framework heavily utilized by flight simulators like FlightGear, Unreal Engine project plugins, and autonomous drone autopilot testing.
-

To point you toward the right documentation or setup, could you specify **what you are looking to build**? If you tell me your **preferred programming language**, whether this is for **commercial defense or hobbyist use**, and if you need a **graphics rendering framework**, I can tailor the development steps directly to your project.

BattleShip Trainer

The "**BattleShip Trainer**" built with vsTASKER is a low-cost, high-efficiency **maritime simulation demo and training application** designed by VirtualSim to showcase rapid simulator development. It serves as a proof of concept for constructing real-time military, tactical, and computer-generated force (CGF) training environments without the high cost of proprietary, rigid engines.

Key Technical Architecture

The trainer architecture combines visual scenario logic with specialized third-party rendering engines:

- **Logic Engine:** vsTASKER drives all scenario parameters, vehicle AI, and tactical interactions. It automatically generates the required C++ code using user-designed state machines and graphical diagrams.
- **Visual Framework:** Built natively for **OpenSceneGraph (OSG)**.

- **Ocean & Wave Rendering:** Integrated with the **Triton ocean library** from Sundog Software to simulate realistic maritime water dynamics, wake patterns, and spray effects.
- **Deployment:** Embedded and run on **vsVIEWER**, the free Qt-based player for the vsTASKER simulation engine.

Features & Capabilities

- **Embedded Ship Dynamics:** Integrated physics components simulate authentic vessel handling, buoyancy, and response to oceanic swells.
- **Direct Weapon Controls:** Allows human-in-the-loop interaction, where a user can operate and fire the battleship's main gun stations using a hardware joystick.
- **Rapid Development:** VirtualSim built the core baseline trainer game from scratch in just a few hours using the visual architecture editor.
- **Scalability:** The tutorial framework is designed to easily scale from a simple game into a high-fidelity naval operator trainer by mapping in High-Level Architecture (HLA) or Distributed Interactive Simulation (DIS) network protocols.

Role in vsTASKER Training & Tutorials

The Battleship project is included as an official sample in the vsTASKER 7 Tutorial Suite. It contains two specific reference modules for developers to study:

1. **Battleship:** Covers the primary maritime movements, target tracking, and scenario flow setup.
2. **MainGun:** Focuses on ballistics, joystick inputs, elevation controls, and firing logic components.

If you are currently developing a naval simulation or trying to run this specific sample, I can help you with deeper technical steps. Tell me:

- Are you trying to resolve a **compiler setup** or integration error with the Triton library?

- Do you need help modifying the C++ **state machine logic** for the ship's behaviors?
- Are you looking to expand the sample into an **HLA/DIS distributed federation**?

Building Simulation Tool

Building a simulator with **vsTASKER** revolves around its core design philosophy: you **visually model behaviours, plans, and environments in the GUI, and vsTASKER automatically generates the underlying C++ source code**. This generated code is then compiled into a high-performance, standalone simulation engine.

Here is the step-by-step workflow to build a simulator using vsTASKER:

1. Setup the System Database (The Object Model)

Before creating scenarios, you must define the components, attributes, and capabilities of the entities in your simulation.

Define Classes: Create the base types for your simulation (e.g., aircraft, vehicles, sensors, and infrastructure).

- **Assign Attributes:** Add variables such as speed, fuel capacity, coordinates, or health states.
- **Integrate External Code:** If you have existing C/C++ mathematical models (like a JSBSim flight model or specific vehicle physics), bind them to your database objects.

2. Design Behaviours Visually (Logic & Doctrines)

Instead of writing complex state machines by hand, use the visual programming tools to define how entities think and react.

State Machines: Map out operational phases (e.g., Idle → Patrolling → Engaging → Returning).

- **Behaviour Trees:** Chain complex, hierarchical conditions and task execution logic.
- **Embed Custom Logic:** Insert brief C++ snippets directly into the visual nodes to handle custom mathematical calculations or specific logic checks.

- **3. Build the Environment and Terrain**

Load the spatial data that your entities will interact with.

2D Maps & Terrain: Import maps, GIS data, or multi-layered terrain files directly into the environment editor.

- **3D Visuals:** Set up OpenSceneGraph (OsgEarth) or 3D globe models to handle satellite tracking or orbit visualization.
- **Navigation Meshes:** Generate nav-meshes so ground entities can navigate complex terrains or urban pathways automatically.

4. Create the Scenario

Populate your environment and give your entities a baseline mission.

Drop Entities: Physically place your database-defined entities onto the 2D or 3D map.

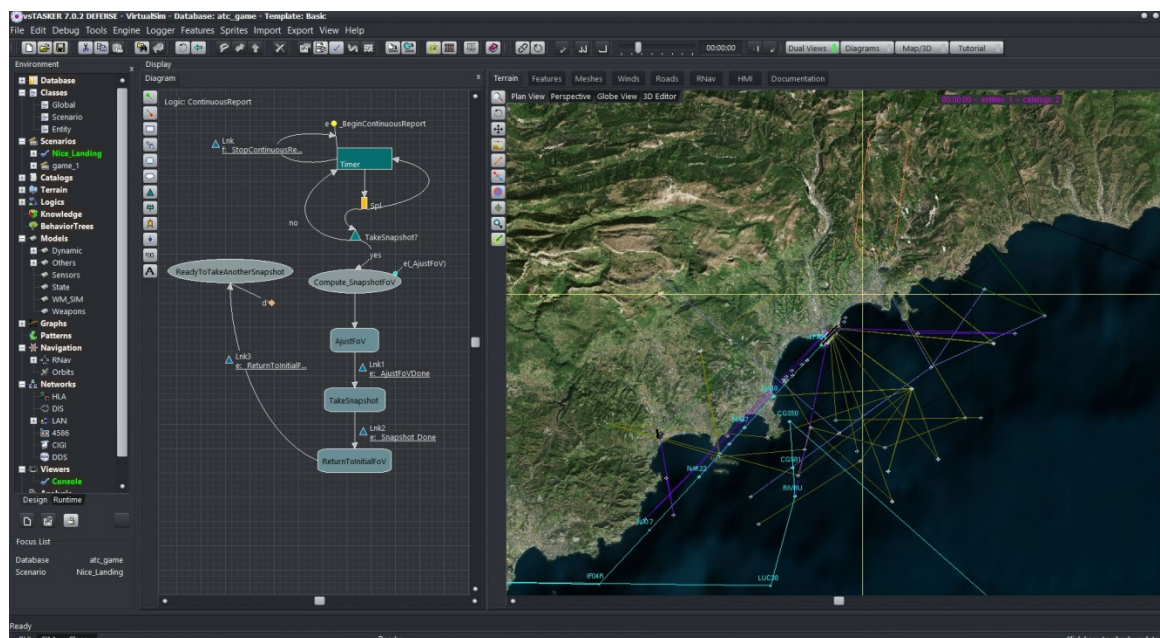
- **Assign Plans:** Give entities waypoints, flight paths, or specific search patterns (e.g., standard search-and-rescue grids).
- **Set Event Triggers:** Define how the scenario reacts to time-based events or runtime communication triggers.

5. Design the Interface (HMI Builder)

Build the control panels your operator will use to interact with the simulation.

Built-in HMI: Use vsTASKER's drag-and-drop HMI builder to create sliders, custom buttons, and tracking windows without needing external tools.

- **External HMI Support:** Alternatively, link your backend to advanced UI libraries like Qt, wxWidgets, or GL-Studio via the C++ API.



6. Generate Code, Compile, and Run

Turn your visual project into an executable simulator program.

- **Code Generation:** Click "Generate" to let vsTASKER automatically write the clean, error-free C++17 code.
- **Visual Studio Sync:** Open the generated code in an IDE like **Microsoft Visual Studio 2019/2022**. The automatic code updater ensures changes made in Visual Studio sync back to the vsTASKER database.
- **Build & Deploy:** Compile the code using the vsTASKER White Box SDK. The resulting standalone executable can run on target machines without requiring the full vsTASKER GUI license.

7. Connect to Distributed Networks (Optional)

If your simulator needs to talk to other hardware or software units, configure the built-in protocols:

- **Military Interoperability:** Enable **DIS**, **HLA**, or **CIGI** to stream data to 3D stealth viewers like Unreal Engine, Unity, or VBS-IG.

- **IoT & Systems Interoperability:** Use **DDS**, **LAN**, or **MQTT** interfaces to stream telemetry data or connect simulated hardware components in real-time.
-

To point you toward the right tutorials or design architectures, could you share a few details?

- What **type of simulator** are you building? (e.g., UAV/flight tracker, autonomous vehicle, maritime, or tactical defense)
- Do you need to connect it to an **external 3D engine**? (e.g., Unreal, Unity, or OpenSceneGraph)
- Do you have **existing C++ code libraries** that you need to plug into the simulator?

Drone Mission Control

vsTASKER is a specialized C++ simulation SDK developed by **VirtualSim** that enables defense industries, software engineers, and system integrators to build high-fidelity drone mission control and UAV simulation infrastructures. Rather than being a basic, ready-to-use consumer app, it serves as a powerful engine used to design custom, complex multi-drone environments, Ground Control Stations (GCS), and synthetic training environments.

Core Drone Capabilities in vsTASKER

- **HMI Builder & Control Panels:** Users can quickly develop a comprehensive **UAV control panel**. This interface supports integrated payload camera views, simulated radar scanning, replicated instruments, map overlays, and target tracking.
- **Standard Interoperability:** The platform natively supports the **STANAG 4586** standard. This ensures cross-platform interoperability for unmanned systems and data links.
- **Flight Plan Customisation:** Operators can easily drop custom **search patterns** and waypoint routes directly onto integrated terrain maps. This allows the platform to simulate automated path optimization routines over millions of test runs.

- **Environmental & Component Replication:** By importing elevation data and terrain maps, vsTASKER can replicate drone flight dynamics (such as a Parrot or quadcopter model). This allows operators to directly compare simulated camera feeds with real-world flight footage.
- **Distributed Network Architectures:** The platform integrates seamlessly with standard military distributed networks using **DIS/HLA** protocols or customized LAN/Socket connections. This means vsTASKER can easily function as a computer-generated force (CGF), a threat generator, or a central data recorder within broader training exercises.

How vsTASKER Compares to Alternatives

If you are evaluating vsTASKER, it helps to understand where it sits relative to other 2026 drone software tools:

Software	Core Use Case	Key Target Audience
vsTASKER	High-fidelity synthetic simulation, military training, C++ code generation, and CGF/threat modeling.	Defense contractors, simulation software engineers, and system integrators.
UgCS / UgCS Commander	Simultaneous multi-drone fleet control, real-world commercial mapping, and advanced LiDAR integration.	Commercial drone pilots, mapping teams, and industrial inspection surveyors.
UAV Mission Control App	Pre-flight weather checks, signal-loss mapping, and safety analysis tailored to consumer/commercial hardware.	Field operators needing instant Go/No-Go safety verdicts based on real-time weather constraints.

Are you looking to **build a custom GCS simulator** using vsTASKER, or are you looking for an **off-the-shelf software solution** to plan a live, real-world drone flight? Turn to these choices to narrow down your requirements.

EOIR SAR Image Generator

vsTASKER is a COTS simulation software designed by Virtual Simulation (VirtualSim) to develop mission planners and manage multi-sensor synthetic environments. It acts as a central **scenario generator and data orchestration engine**, coordinating complex airborne intelligence, surveillance, and reconnaissance (ISR) training operations.

Instead of generating raw image pixels itself, vsTASKER orchestrates the platform dynamics, sensor footprints, and environment tracking. It then hands off these variables to **high-fidelity third-party simulation models** to output Electro-Optical (EO), Infrared (IR), and Synthetic Aperture Radar (SAR) imagery.

The Role of vsTASKER in EO/IR and SAR Generation

Within a multi-sensor simulation loop, vsTASKER Simulation Software provides the fundamental behavioral layer:

Mission Planning & Logic: Models the flight path, altitude, slant range, and roll/pitch/yaw of airborne sensor platforms.

Dynamic Scenarios: Controls entity movements (e.g., ships, aircraft, ground radars) across global geographic layers via custom UDP packets or standard protocols like DIS/HLA.

Sensor Footprint Calculations: Generates the precise Field of View (FOV) cones, coverage swaths, and look angles for optical and radar pods.

Simulation Orchestration: Syncs the core engine clock with secondary imagery renderers, ensuring that EO, IR, and SAR sensor outputs capture identical frames of a physical target simultaneously.

Third-Party Plugins Integrated with vsTASKER

To generate the final high-resolution synthetic image datasets, vsTASKER relies on distinct physics-based modeling tools:

1. EO/IR Image Generation

Optical (EO) and Thermal Infrared (IR) systems rely on radiometry, atmospheric radiance, and surface temperatures:

Thermo-Optical State Plugins: Software like TAItherm/MuSES by ThermoAnalytics calculates real-time internal heat signatures (engines, electronics) on 3D CAD target models.

- **Radiance Rendering Engines:** Modules like OKTAL's SE-Workbench-EO or DIRSIG ray-trace the scene to output photorealistic imagery across multi-spectral bands (NIR, MWIR, LWIR).
- **2. SAR Image Generation**

Synthetic Aperture Radar behaves differently than optical lenses, requiring coherent complex radar return simulations instead of ambient light capture:

Electromagnetic Ray Tracing: Engines such as Xpatch, MOCEM, or modern ray-tracers estimate multiple backscatter reflections off structural facet surfaces.

- **Echo Time-Domain Models:** Convert physical facet shapes and radar pulse widths into raw 2D SAR transfer functions, outputting high-resolution speckled radar backplates.

vsTASKER is a high-performance, real-time simulation toolkit developed by **VirtualSim** that uses a visual architecture and automatic C++ code generation to develop complex flying object and synthetic airspace environments. It allows defense, aerospace, and engineering teams to model aircraft, drones, missiles, and spacecraft behaviors without wrestling with low-level manual programming.

Key Capabilities for Flying Object Simulation

- **Aircraft & Flight Dynamics:** Supports linear models for fast 2D scenarios and integrates high-fidelity dynamic flight parameters for realistic 3D output. It also features native **JSBSim Flight Model integration** and allows engineers to import custom MATLAB/Simulink models directly.
- **Unmanned Aerial Systems (UAS):** Fully supports the **STANAG 4586** standard for UAV and ground control station interoperability. It is widely used to develop Autonomous UAV mission planning and AI-driven target coordinates extraction.
- **Missile Trajectories & Countermeasures:** Built to test complex guidance laws, weapon payloads, missile counter-measures, and terrain-following (Nap-of-the-

Earth) optimal search paths using fast, high-frequency **Monte Carlo batch simulations**.

- **Space & Orbital Objects:** Includes an orbital propagator supporting **SGP4/SDP4 models**. Users can point-and-click to load public satellite databases from NORAD or use Keplerian parameters to simulate custom satellites and intercontinental ballistic missiles.
- **Air Traffic Management (ATM):** Capable of hosting large-scale air traffic simulations with hundreds of concurrent arrivals, testing ATC Tower separation rules, and injecting real-world **ADS-B data**.
- **Core Architecture & Features**

[Visual Editor] -> [State Machines / Behavior Trees] -> [Automatic C++ Code Gen] -> [Standalone Sim Engine]

- **Visual Logic Design:** You design vehicle logic using state machines and behavior trees. The system then generates optimized, license-free executable C++ code.
- **High Performance Engine:** Executes logic loops anywhere from 1Hz up to **frequencies greater than 1000Hz**, managing thousands of moving entities simultaneously.
- **Graphics & Game Engine Integration:** While it has its own 2D/3D map viewers, it functions as a "plug-and-play" plug-in to drive high-end image generators like **Unreal Engine, Unity, Unigine, Titan Vanguard, and VegaPrime**.
- **Military Connectivity Protocols:** Seamlessly pushes real-time spatial data over standard distributed simulation networks including **DIS, HLA, CIGI, and DDS**.
- **Technical Details & Implementation**

To evaluate or test your aerospace code, VirtualSim provides a free discovery download version alongside formal evaluation requests for full production SDK testing. []

If you are developing a specific flight system, please tell me:

- What **type of flying object** are you modeling? (e.g., commercial aircraft, quadcopter drone, guided missile, satellite)

- Do you need to connect to an external **3D graphic engine** or **hardware-in-the-loop** setup?

I can provide specific implementation steps or connection templates for your target setup!

Metro Simulation

vsTASKER is a professional, high-end C++ code generation GUI and simulation engine developed by [VirtualSim](#) that is widely used to power commercial and military **metro and heavy train driver training cabs**.

Instead of being a consumer video game like Metro Simulator on Steam, vsTASKER acts as the underlying **simulation core engine (TrainSim Server)** for industrial hardware setups.

Core Architecture & Implementation

According to [VirtualSim's technical architecture](#), a standard industrial metro simulator using vsTASKER is structured around several interconnected components:

The Simulation Engine: vsTASKER automatically generates the core C++ logic handling train dynamics, track geography, routing, and rail physics.

Hardware Integration (MQTT): The simulator runs continuously and uses the MQTT message protocol to interface directly with real physical train cabin controls, gauges, and driver indicators.

- **3D Visualization:** It integrates with high-end visualization tools like *Vega Prime* to render true-to-life, multi-channel 3D visual environments across cabin windows, windshields, and mirrors.
- **Passenger Simulation:** It incorporates third-party human simulation software (such as *DI-Guy*) to handle dynamic passenger flow, platform boarding, and realistic crowd behaviors at stations.

Real-World Use Cases

vsTASKER has been utilized globally to develop complex rail transit training infrastructures:

- **Bangkok Metro:** Powers the core driver trainer cabs for the Blue and Orange lines, managing full train rail system logic and instructor-led training exercises.
- **Algeria & Turkey:** Employed to develop embedded heavy train simulators complete with physical motion platforms and immersive visuals.

Key Features For Metro Training

When built with vsTASKER, these heavy-duty metro training setups typically include:

Instructor Module: A central control panel allowing an instructor to view the live metro network map, track train positions, inject equipment malfunctions, and trigger emergency situations.

- **Audio Environments:** Precise audio mapping that recreates the sounds of electric traction motors, pneumatic door systems, track vibrations, and station environments.
- **Scenario Management:** The capacity to train operators through over 100 typical line malfunctions, block signaling challenges, and weather disruptions.

If you are trying to implement or study a specific aspect of a vsTASKER deployment, please let me know:

- Are you looking to write custom **C++ code or logic** within vsTASKER?
- Do you need information on setting up the **MQTT communication layer**?
- Are you researching **commercial licensing** or technical support?

I can provide more targeted technical steps or architecture details depending on your goals.

Moving Objects on Map Server

vsTASKER is a professional C++ code-generation framework and simulation software used primarily for defense, tactical, and air traffic control (ATC) applications. It serves as a

comprehensive tool to build, coordinate, and execute real-time simulations of moving objects (such as aircraft, ground vehicles, drones, and naval ships) over 2D and 3D maps.

The system functions as a **"white box" developer environment**, meaning instead of relying on a locked-down interface, it generates source-ready C++ code that connects directly with third-party rendering engines.

Key Capabilities for Moving Object Simulation

- **2D Map and 3D Globe Visualization:** It provides a built-in 2D GIS map display alongside an OpenSceneGraph (OSG) 3D globe. This dual-view setup allows you to drop assets onto a location, plot flight paths, and see real-time positional changes represented simultaneously in both layouts.
- **Visual Logic and Behavior Trees:** You do not have to script every individual entity choice manually. You can model behaviors using visual State-Charts, Grafcets, and **Behavior Trees** to dictate how a moving object reacts when it encounters obstacles, hits waypoints, or receives radar/sonar triggers.
- **High-Fidelity Dynamics:** For complex object handling, you can use linear dynamic equations or integrate external high-fidelity models (like MATLAB Simulink) to calculate realistic pitch, yaw, wind effects, turning radii, and acceleration.

Standalone Deployments: Once your scenario—composed of terrains, moving convoys, or aircraft paths—is configured, vsTASKER converts your visual architecture into C++ code. You can compile and deploy a license-free standalone runtime panel without any ongoing seat costs.

vsTASKER utilizes its core real-time simulation engine to model, assess, and optimize **radio communications (Radio Com)** under diverse and complex operational scenarios.

Developed by VirtualSim, vsTASKER acts as a "white-box" simulation toolkit that automatically generates C++ code from graphical state machines and diagrams, allowing engineers to design high-fidelity synthetic communication environments.

The framework simulates everything from basic signal propagation to complex, multi-station distributed networks.

Core Radio Simulation Capabilities

- **High-Fidelity Wave Propagation:** Models how radio waves travel across environments, accounting for frequencies, power levels, and signal degradation over distances.
- **Terrain Occultation:** Simulates how physical obstacles, hills, and mountainous terrain block or weaken line-of-sight radio signals.
- **Flexible Duplex Modes:** Supports both full-duplex (simultaneous two-way transmission) and half-duplex (one-way at a time) communication modes.
- **Custom Protocol Frameworks:** Allows users to define custom, user-defined protocols and frequencies to test electronic warfare or specific military/civilian hardware.
- **Data Transfer Simulation:** Measures the real-time efficiency, latency, and effectiveness of data transferring across moving nodes.
- **Typical Use Cases & Architectures**

vsTASKER is widely used across defense and aerospace sectors to build distributed training networks and test autonomous behaviors:

[Instructor Station] ----- (Scenario & Weather Control)

|

v

[Flight Server] <===== > [Radio Com Simulation Engine]

(Aircraft Dynamics) (Terrain, Signal, Duplex Modes)

|

v

[3D Engine / VBS / HMI] ---- (Visualizes Effects & Layouts)

- **ATC Tower Simulators:** Manages the simulated radio network connecting pseudo-pilots, air traffic controller trainees, and automated radar stations.
 - **UAV Radio Relays:** Simulates the use of unmanned aerial vehicles acting as flying radio relays to bridge communications over mountainous or blocked terrains.
 - **Distributed Network Scenarios:** Connects simulated radio stations with standard simulation protocols like DIS/HLA or custom LAN/Socket integrations to interact with third-party software.
 - **Mass Batch Testing:** Leverages its high-performance runtime engine to run millions of automated scenario variations, storing communication failure data into SQL databases for statistical analysis.
-

If you are looking to design a specific radio setup, tell me:

- What **specific frequency band** or **protocol** are you trying to model?
- Are you integrating this with an **external 3D visual engine** (like Virtual Battlespace / VBS)?
- Do you need assistance writing the **C++ logic structures** for your custom radio nodes?

vsTASKER is a professional simulation software platform developed by VirtualSim that is widely used to develop **intelligent Search and Rescue (SAR) mission planning and support systems**. It serves as a core scenario editor and simulation engine for both civilian emergency responders and defense agencies.

How vsTASKER is Used in Search and Rescue

- **Autonomous Drone & UAV Missions:** The platform is used to program autonomous behavior and search patterns for Unmanned Aerial Vehicles (UAVs) tasked with scouring vast areas. For example, it was deployed to model SAR logic over the English Channel.
- **Sensor & Image Generation (EO/IR):** It integrates with high-fidelity Electro-Optical and Infrared (EO/IR) airborne sensor models. This simulates thermal imaging to

help operators identify missing persons or vehicles through dense terrain, at night, or in adverse weather.

- **Intelligent Emergency Support Systems:** Agencies like local fire departments use the system to build real-time tactical overlays, manage emergency communication networks, and project drift estimation for maritime operations.
- **Multi-Agency Coordination Training:** It builds complex, repeatable training environments where instructors can simulate emergencies to train responders on rapid resource allocation and path optimization.

Core Technical Capabilities

- **Scenario Editing:** Allows users to quickly build complex 2D/3D environments on raster or vectorized maps.
- **Open Hardware Connectivity:** Connects easily to real training cabs, cockpits, and control interfaces using IoT protocols like MQTT.
- **License-Free Deployment:** Organizations can design custom console panels and system logic in-house, then export them into standalone applications without paying downstream licensing fees.
- If you are looking to deploy a SAR simulation project, you can request a evaluation version on the official vsTASKER Demo Page.

Are you exploring this for **military defense modeling**, **civilian first-responder training**, or a specific **UAV/drone application**? Let me know so I can share more tailored details on its logic modules.

Simulation Builder

When comparing "simulator builder" as a functional concept to vsTASKER, the primary distinction is that vsTASKER is a professional, commercial-grade C++ code-generation toolkit, whereas a general "simulator builder" refers to the broad category of software tools or game engines used to construct synthetic environments.

While a generic simulator builder might focus purely on 3D graphics or simple gaming physics (like Builder Simulator or game engines), vsTASKER by VirtualSim is specifically

engineered for military, aerospace, and defense industries to model high-fidelity, real-time logic and distributed training systems.

Core Structural Differences

Feature / Aspect	Generic Simulator Builder (e.g., Unity, Unreal, Game Engines)	vsTASKER (by VirtualSim)
Primary Focus	Visual aesthetics, rendering pipeline, and generalized physical interactions.	Complex entity behavior, real-time logic, and tactical scenario generation.
Output Type	Compiled proprietary application binaries or standalone game files.	License-free, standalone C++ source code that compiles into an independent engine.
Logic Design	Scripting (C#, C++) or proprietary visual scripting (Blueprints).	Hierarchical State Machines, Behavior Trees, and graphical logic diagrams.
Graphics Engine	Built-in monolithic renderer (DirectX/Vulkan-based).	Graphics-agnostic. Deploys onto VBS-IG, Unreal Engine, Unity, or VegaPrime.
Military Protocols	Requires third-party plugins or manual networking setup.	Native support for DIS, HLA, DDS, CIGI, and LAN out of the box.

Key Strengths of vsTASKER

- **Automated Code Generation:** Users design scenarios, flight paths, and tactical logic using an intuitive 2D map and visual editor. vsTASKER then automatically generates optimized, error-free C++ code.
- **Built-in HMI & UI Builder:** It includes an integrated drag-and-drop Human-Machine Interface (HMI) builder. This allows you to construct custom instrument panels,

control knobs, and interactive telemetry screens without relying on external UI suites.

- **Extreme Performance:** The core simulation engine is optimized to handle thousands of complex, interactive AI entities concurrently. It operates seamlessly across high real-time frequencies ranging from 1Hz up to **1000Hz+**.
- **No Deployment Royalties:** Unlike many commercial engine licenses that require runtime seats or ongoing fees, the simulation engines and control panels built via **vsTASKER** are completely royalty-free to distribute.
- **When to Use a Standard Simulator Builder**

You should lean toward a traditional game engine or general-purpose simulation builder if your objective relies heavily on state-of-the-art visual rendering, consumer-facing entertainment physics, or out-of-the-box asset stores for commercial gaming.

When to Choose vsTASKER

Opt for **vsTASKER** if you are an engineer or defense contractor tasked with building a complex, high-consequence synthetic environment—such as an Air Traffic Control (ATC) tower trainer, drone flocking (Boids) mechanics, or military distributed simulation networks—where predictable backend logic and precise standard protocol compliance are mandatory.

To help narrow down your selection, could you clarify:

- What **specific industry** or domain is your simulation project targeting (e.g., military defense, industrial automation, or commercial gaming)?
- Do you require native support for **distributed simulation protocols** like DIS or HLA?
- What is your team's preferred **programming language and framework** layout?

Train simulator

vsTASKER is a professional Software Development Kit (SDK) created by **Virtual Simulation** to design and power complex, real-time simulation frameworks. In the rail industry, it is utilized as the **core simulation engine (trainsim server)** for high-end, mission-critical train cabin and railway network training systems.

Unlike commercial video games like Train Sim World, vsTASKER is a commercial-grade platform tailored for software engineers and systems integrators.

Core Architecture for Train Simulation

When building a train simulator (such as a Metro Cab Trainer), vsTASKER serves as the master coordinator:

The Simulation Engine (trainsim server): Generated directly by vsTASKER, this core engine handles the train's physics, rail network dynamics, and logic.

MQTT Message Interfaces: Connects the simulation engine seamlessly to physical cab hardware (throttles, brakes, dashboards) and external train scheduling software.

Visual & Entity Integration: Works out of the box with third-party components like **VegaPrime** (for multi-channel, multi-screen 3D cab visuals) and **DI-Guy** (for realistic passenger behaviors at stations).

Key Technical Capabilities

- **Graphical Logic Design:** Developers can build complex railway operations, signal behaviors, and Automatic Train Control (ATC) systems using intuitive **Sequential Function Charts**.
- **Automatic C++ Generation:** The system translates visual diagrams into optimized, **high-performance C++ code** with a single click, removing the manual management of messy codefiles.

- **System Interoperability:** Fully supports distributed architectures and industry-standard protocols, allowing training systems to scale from a single desk to multi-station networks.
- **Real-World Rail Use Cases**

Organizations use vsTASKER to deliver heavy industrial solutions:

Train Control-Command & Cab Trainers: Building full-scale physical replicas of metro and train cabins for real-time driver training on newly deployed lines.

- **ATC Collision Forecasting:** Creating safe, virtual spaces to train railway dispatchers and Air Traffic Controllers on emergency procedures and predictive conflict management.
 - **Human Factor & Ergonomics Studies:** Simulating emergencies to train and analyze the communication efficiency between drivers, controllers, and trackside signalers.
-

To help you explore further, could you share the **scope of your project**?

- Are you looking to develop a **custom railway logic system** using C++?
- Do you need to interface vsTASKER with **physical hardware buttons and levers**?
- Are you evaluating it for **military, aviation, or transit training**?
- Let me know your primary goal, and I can detail the specific integration steps.

Warfare Gaming Simulator

vsTASKER is a professional, high-performance C++ software framework and code generator developed by VirtualSim. It is widely used across the defense and aerospace industries to design, develop, and run complex **avionics, tactical, and Electronic Warfare (EW)** simulation scenarios.

Unlike closed "black-box" systems, vsTASKER serves as an open "white-box" development environment. It lets systems engineers graphically design complex logic, combat scenarios, and vehicle behaviors, and then automatically compiles that data into high-performance, real-time native C++ code.

Core Capabilities in Avionics & Electronic Warfare

The toolkit provides specialized features tailored for modeling air defense, avionics, and radar countermeasures:

- **Behavior and Logic Modeling:** Engineers can graphically build complex artificial intelligence (AI), mission tactics, state machines, and pilot behaviors without manually typing thousands of lines of code.
- **Radar & Electronic Warfare (EW):** The software can simulate multi-emitter tactical battlefields. It supports complex threat behaviors and has been used to model detailed anti-ship missile countermeasure scenarios, such as electronic decoys and jamming algorithms.
- **Avionics & Flight Planning:** vsTASKER features native support for standard aviation navigation data, including the **ARINC 424** format. This allows realistic simulation of Flight Management Systems (FMS), flight plans, and air traffic operations.
- **Environment & Synthetic Battlefield:** Users can set up high-fidelity, geo-specific terrain maps (both online and offline) and simulate hundreds to thousands of entities concurrently—running faster than real-time for comprehensive "what-if" analytics.

□ Interoperability and Architecture

To function within multi-domain warfare simulators, vsTASKER integrates seamlessly with industry-standard military simulation layers:

Feature	Supported Protocols & Engines	Purpose
Distributed Simulation	DIS (Distributed Interactive Simulation), OpenDIS, MQTT	Networks multiple simulators, ground units, and threats into one joint airspace.
Visual Systems (IG)	CIGI (Common Image Generator Interface), Unigine,	Connects the underlying logic code to photorealistic 3D virtual views and pilot

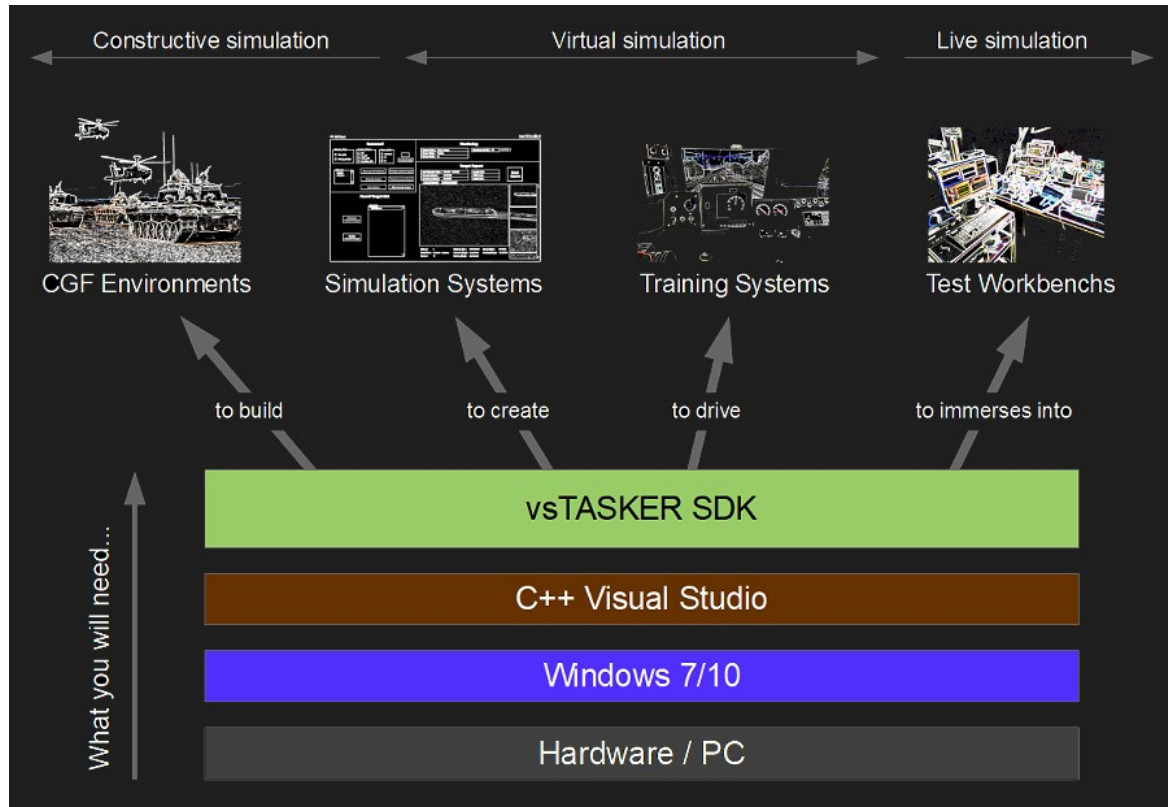
	VegaPrime	cockpits.
Serious Games Integration	VBS (Virtual Battlespace / VBS-IG)	Drives the tactical tactical logic behind low-cost commercial-off-the-shelf (COTS) training platforms.

□ Accessing the Software

Because it is a highly specialized defense product, access to the software is strictly regulated:

- **Corporate Filter:** You can request a license-free evaluation version directly via the vsTASKER Demo Portal. However, VirtualSim will reject public domains (like Gmail, Yahoo, or Outlook) and requires a legitimate **business/institutional email address**.
- **Documentation:** Detailed structural architecture and implementation guides are available in the official vsTASKER User Manual.

If you are currently evaluating this software for an upcoming project, I can provide more targeted information. Would you like to explore **how it handles radar cross-section (RCS) data**, look into **sample C++ code generation patterns**, or see how it compares to alternative frameworks like **AFSIM**?



OS & HARDWARE SUPPORT

INDUSTRY APPLICATION

--	--

Note: ** Specifications are copied from manufacturer data. Please contact before buying.