AIR TRAFFIC CONTROL: AEROPVS02

The vsTASKER ATC Tower Simulator is a professional, high-fidelity real-time simulation system designed by VirtualSim for air traffic control (ATC) training, ground management, and terminal airspace studies. Unlike commercial video games, vsTASKER functions as a C++ code-generating engine and development toolkit, allowing organizations to program complex logic, custom human-machine interfaces (HMIs), and advanced automation rules for aviation infrastructure.

Core Architecture & Workstations

The simulator operates over a decentralized, multi-station network with specialized modules:

- Flight Server (FS): Simulates all aircraft in real-time, executing specific operations and answering commands from other network stations.
- ATC Radar View: Feeds from the Flight Server to display real-time aircraft positions to trainee controllers.
- Pseudo Pilot Station: Allows an instructor or operator to role play as a pilot, talking to and executing instructions given by the ATC trainee.
- 3D Visual Engine: Renders a 360-degree, highly immersive visual simulation of the airport tower cabin.
- Instructor Station: Empowers trainers to launch specific scenarios, manipulate real-time weather conditions, or inject critical incidents like bird strikes and sudden wind gusts.

Key Technical Capabilities

- Data Integration: Built to import and handle ARINC 424 RNAV navigation databases to build real-world flight paths.
- Live Feeds: Can optionally overlay live ADS-B data streams, allowing simulated aircraft to seamlessly interact within real-world localized traffic structures.

- HMI Builder: Features embedded GUI layout toolkits (supporting OpenGL, Qt, or wxWidgets) to build functional, high-fidelity replica ATC consoles.

- Separation Logic: Uses sophisticated math and user-defined script rules to safely guide hundreds of simultaneous aircraft across runways, approach corridors, and holding patterns.

- Interoperability: Connects into existing infrastructure using HLA protocols or DIS visual interfaces, and can embed its generated code directly into external rendering platforms.

If you would like to explore this system further, tell me:

- Are you evaluating this software for academic research, military applications, or commercial ATCO training?
- Do you need details on how it integrates with specific game engines or protocols like VBS-IG / CIGI?
- Are you looking for information on its latest version 7/8 roadmap capabilities?
-

vsTASKER by VirtualSim is a specialized C++ code-generation toolkit used to build high-fidelity Air Traffic Control (ATC) and tower training simulators. Rather than being a standalone video game, it is a professional development environment designed for aviation schools, defense contractors, and airspace researchers.

Core Capabilities for ATC Simulation

- Traffic & Flight Plan Generation: Developers can build complex terminal maneuvering area (TMA) scenarios by importing standard ARINC 424 RNAV navigation databases. This allows aircraft to automatically execute standard instrument departures (SIDs) and standard terminal arrival routes (STARs).

- HMI and Console Customization: It features a built-in Human-Machine Interface (HMI) builder. This allows designers to build functional radar screens, electronic flight strips, and control panels. It seamlessly links with external C++ libraries like Qt, wxWidgets, VAPS, or GL-Studio .

- Multi-Station Distributed Architecture: It facilitates training environments by splitting the software architecture across multiple dedicated systems:

  - **Flight Server**: Simulates all aircraft physics and logic in real-time.
  - **ATC Radar View**: Visualizes the simulated aircraft for the trainee air traffic controller.
  - **Pseudo Pilot Station**: Allows an operator to act as the pilot, responding to the trainee's commands and adjusting flight parameters.

- **Instructor Station**: Enables trainers to dynamically inject weather shifts or emergencies like bird strikes.

- **Live Data & Engine Integration**: It supports the injection of real-world **ADS-B traffic data** so controllers can interact with actual local aircraft in real-time. Because it generates pure C++ code, it embeds directly into professional 3D graphic engines like **Presagis VegaPrime** or visual setups via **CIGI / DIS** protocols to create 360° airport tower environments

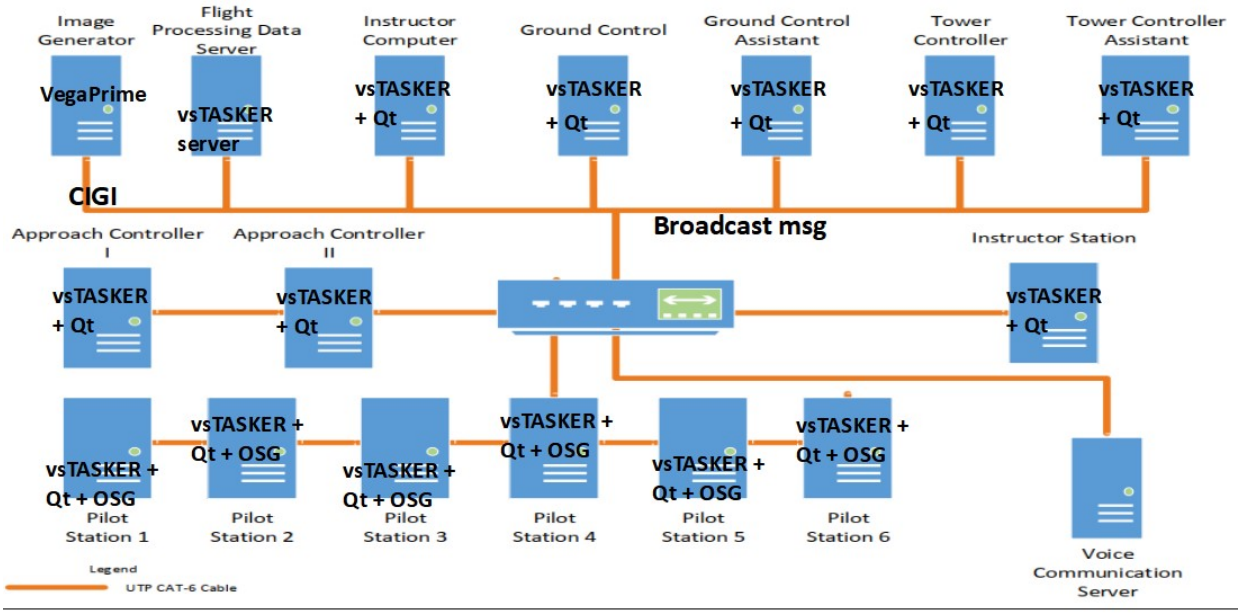
The purpose of this ATC & Tower simulator is to provide real-time air traffic control training for multiple airports to several trainees simultaneously.

- **3D tower view** of the airport area in 360°
- High fidelity **3D models** for aircraft, helicopter and vehicles
- **Day** and **night** visual **operations** combined with **different weather** conditions
- **Multiple airports** with runways, taxiways, parking gates, utility vehicles, etc.
- One **Instructor station** to prepare offline scenarios stored into SQL database
- **Record** & **Replay** of the full system with pause, rewind, **forward** and **fly-out**.
- **Faster** than real-time simulation
- **ARINC 424** support for flight plans definition and map display
- Up to 6 ATC **radar view stations** including flight **strip boards**
- Up to 6 **pilot stations** to translate **radio communication** to aircraft controls
- **2D display** of the airport, flight-plan and surrounding
- Automatic **push-back**, **taxi**, **follow-me** and **parking**
- **ILS** capture and landing
- **Collision detection** and ground avoidance
- **Formation flight** for up to 4 aircraft with special maneuvers for landing
- **VFR** approaches, **Touch & Go** and **Go Around** capabilities
- **Failures** (gears, engine…) and landing/take-off **accident** support

- Ground and **Service vehicle** controls (fire & fuel truck, baggage cart, ambulance…)
- Up to **100 simultaneous aircraft**

Main code simulation engine produced by vsTASKER, Qt + vsTASKER backbone and 2D maps for all stations, VegaPrime to handle the 3D view on multiple channels and screens

Architecture



The **Flight Server** (FS) acts as the central hub which sends/receives data and events to/from all others work stations. All the network traffic is recorded for later replay. As the simulation engine generated by vsTASKER is license free, the cost for a large number of running instances remains the same, mostly when embedded into a Qt visual interface.

A second network is used for the voice communication between ATC operator and Pseudo Pilots. This network has its own recording capability. vsTASKER GUI is used to define all the logic and components shared among the various stations. Four databases are created, one for the **FS**, one for **ATC Radar Views**, one for the **Pseudo**

Pilots and one for the Scenario Preparation. Each database holds as many scenarios as supported airports. Aircraft models are defined in the Catalog and are instantiated at runtime. During a training session, any station sends messages to the Flight Server (FS) to control various aircraft or initiate specific operations. The FS processes every command and return acknowledge to the sender.

Flight-Server

All stations are connected to the broadcast network, listening from messages sent by the Flight Server at 2 Hz frequency for aircraft positions and every 5 seconds for some low band entity status. The Flight Server runs at 30 Hz and controls the 3D visual using CIGI protocol at a dynamic frequency (high for visible entities, low for out of sight ones).
The Flight Server (FS) is the core simulation engine which handles all aircraft, airports, logic and ATC components. It is responsible for the ground and air movement of all vehicles. It uses embedded dynamic models which are fast enough to support 30hz base frequency with hundreds of entities.

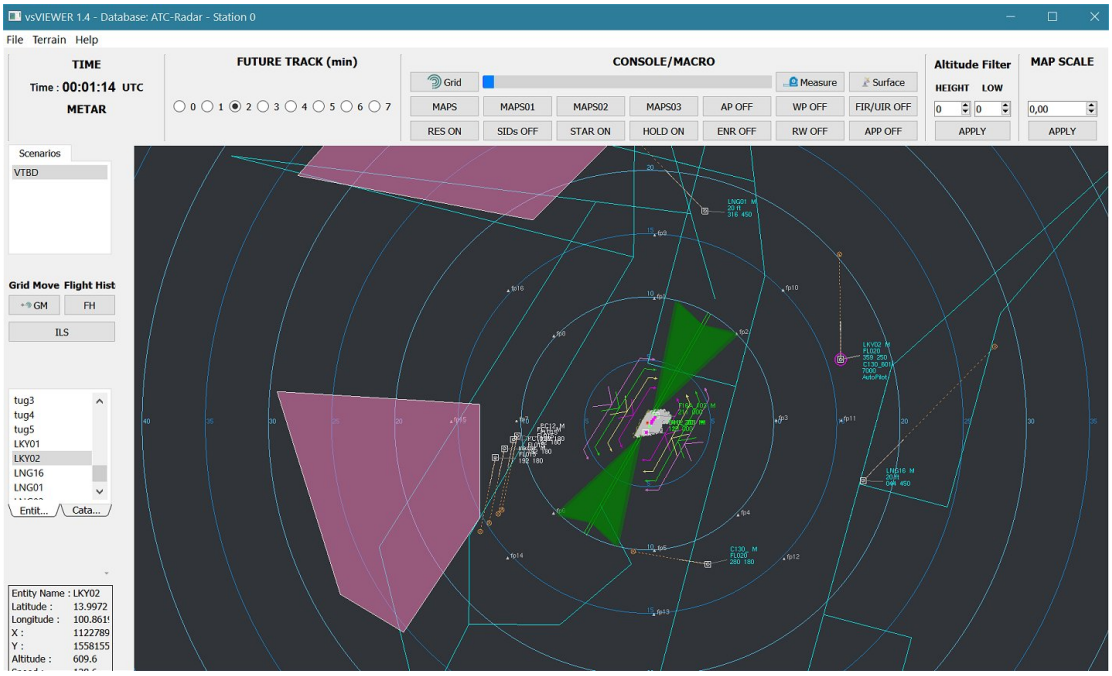
The FS uses a unique Catalog of all aircraft shared by all scenarios. Each aircraft has been designed to represent a specific model. It holds the 3D OpenFlight model, the associated flight dynamics, some bespoke components required by the simulator and a flight management system for automatic landing and flight plans following.

The above screen shot shows a portion of the list of available aircraft and helicopters. The FS can instantiate hundreds of these Catalog's entities and position them anywhere on the airport ground or controlled airspace.

Each entity refers to some logic and components which are shared by all. Only parameters differ and are defined by the user to specialize the model. Aircraft share same behavior. Each vehicle has its own logic according to its type. See below

ATC Radar View

These dedicated stations are used for the training of controller students. There can be several of them to manage dedicated area, airspace or ground traffic. Trainees are monitoring the real-time traffic simulated by the flight server, can question each aircraft, reorder their progress strips and use the radio to give orders to pilots. On another rooms, pseudo pilots listen to controller commands and controls aircraft accordingly.



The Radar display is generated using a free Qt SimpleUI example provided with vsTASKER software. As the Qt code and UI are supplied, user can modify and customize it according to its need. The map display is embedded. The sample uses the vsTASKER libraries and specific code generated data. Both Qt and vsTASKER libraries are linked together to produce the license free deployable application.

The Radar application uses its own vsTASKER database with logics to process HMI actions and network access (aircraft messages from Flight Server and other stations).

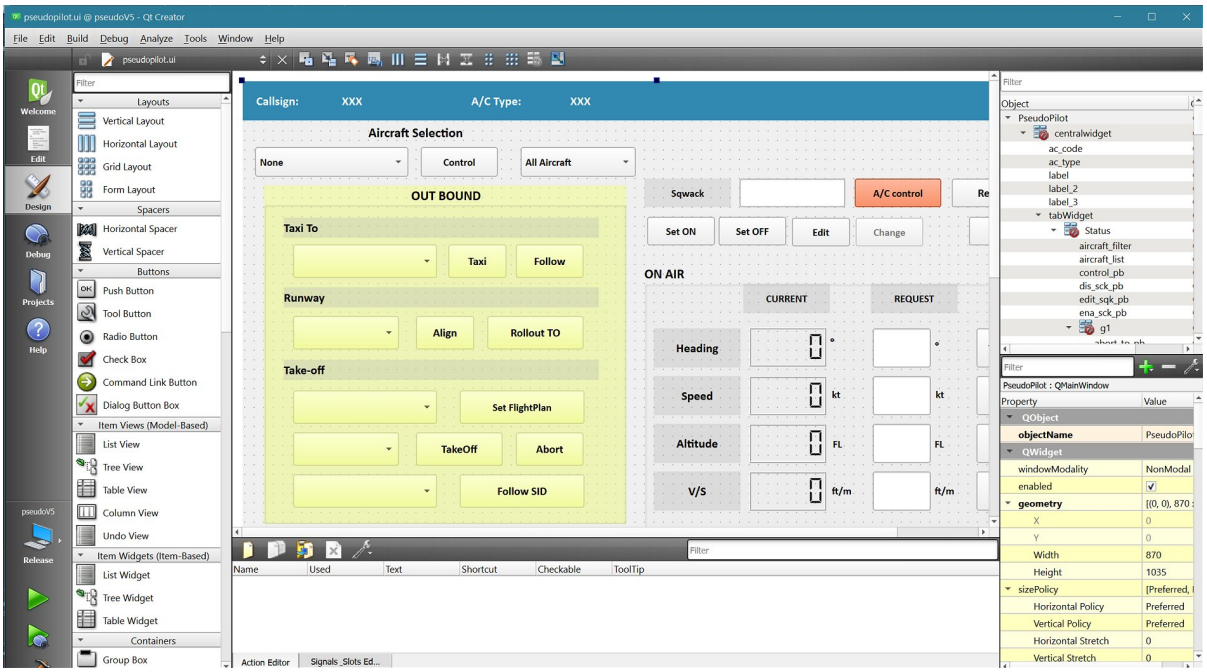
Pseudo Pilot

Several stations are used by operators to simulate pilot. Operators respond to ATC requests by using the user interface to control specific aircraft. Operator use radio device and listen to the ATC air and ground channels. He can take control of any available aircraft and acquire it. Other operators see owned aircraft as unavailable. Operator control owned aircraft for the duration of the exercise.

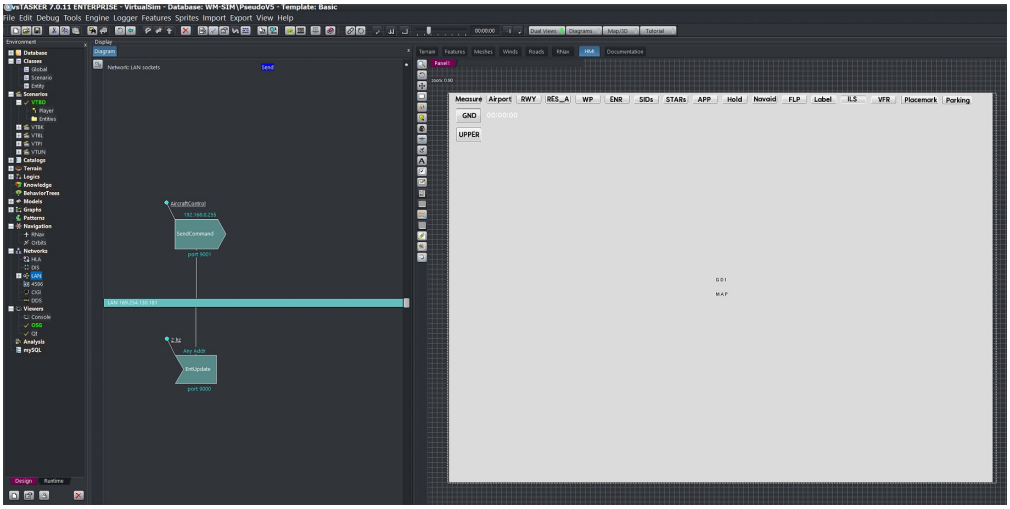
Pseudo Pilot Interface example of use

The Pseudo Pilot is also responsible for controlling ground vehicles. He may send bag carts to some designated aircraft, send ambulance and fire-trucks to damaged aircraft. Tugs and follow-me cars are automatically handled by the Flight Server if required by specific procedures. Below are some of its main features:

- Aircraft selection, acquire and release ownership
- Command to taxi to a specified location, alone or with follow-me car
- Command to align on a runway before stop or rollout take-off
- Authorization to initiate or abort take-off (before V1)
- Set or change of a flight plan
- Command to follow a specified SID or STAR procedure
- Activate ILS landing, VFR approach and landing
- Set or change the Squawk code
- Control flight parameters (heading, speed, altitude and climb/descent)
- Control holding procedure (immediate or planned)
- Manually retract or extend landing gears
- Management of formation (members join or leave, leader control, landing)
- Activate or deactivate failures
- The Pseudo Pilot has been developed with Qt and vsTASKER

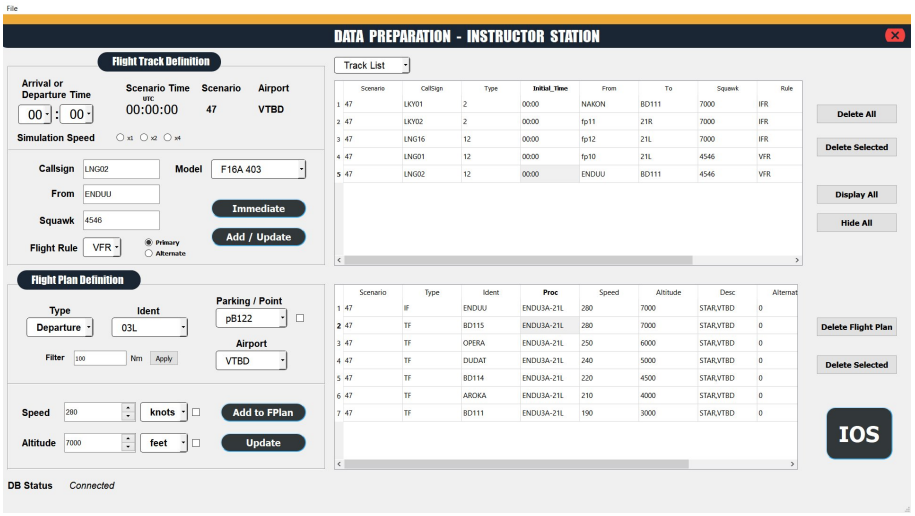


The Pseudo Pilot is a combination of Qt and vsTASKER. The Map display (on the right) is generated using the vsTASKER HMI toolkit. Buttons (top), reactions, logics, network connection (…) are defined in vsTASKER and code generated. The left inside panel is designed under Qt. It uses the vsTASKER libraries and specific code generated data. Both Qt and vsTASKER libraries are linked together to produce the license free control station.



Flight Plan Editor

The flight plan editor is a Qt and vsTASKER application designed to prepare scenario and exercise. The operators work offline and populate the airport and the airspace with entities. Aircraft and helicopter can be put on ground, on any parking gates or in air, at a designated Waypoint or Navaid and with or without a flight plan. Each aircraft has a beginning and ending time, to sequence traffic during the course of the exercise, without manual interaction.



- Outgoing aircraft may have a departure time and a flight plan including a SID.
- Incoming aircraft may have a flight plan including a STAR
- All defined scenarios are stored into an SQL database

OS & HARDWARE SUPPORT	INDUSTRY APPLICATION

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