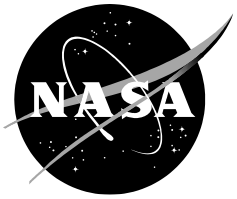
NASA/TM—20260000292



# Development of Traffic Visualization Tool for Visualizing and Analyzing Traffic in National Airspace System

*Chok Fung Lai*
*Ames Research Center, Moffett Field, California*

**January 2026**

# NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

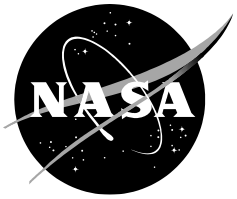
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM—20260000292



# Development of Traffic Visualization Tool for Visualizing and Analyzing Traffic in National Airspace System

*Chok Fung Lai*
*Ames Research Center, Moffett Field, California*

National Aeronautics and
Space Administration

*Ames Research Center*
*Moffett Field, CA 94035-1000*

**January 2026**

## Acknowledgments

The author would like to thank both the National Airspace System Digital Twin and Air Traffic Management TestBed teams for providing the guidance and review of the traffic visualization tool. The author would also like to thank Dr. Christian Plaunt, Dr. Todd Lauderdale, Dr. Aditya Das, and Steven Beard for reviewing this technical document and providing valuable feedback.

# Abstract

An interactive, expandable, two-dimensional, Java-based traffic visualization tool that has been developed for the National Airspace System Digital Twin to support air traffic management research. The visualization tool is designed to acquire data from sources including databases, files, and servers and render airspace elements, geographical information, flown and predicated flight trajectories, historical and real-time air traffic and weather, as well as data recorded during simulation runs. The visualization tool has been used for discrete- and real-time simulations and post-simulation analyses. The development of the visualization tool has four design goals: ease of code base access, ease of data source connectivity, component reusability, and software extensibility. This document describes the user interface design, software capabilities and features of the visualization tool. The author hopes that the code base can be used by aviation projects to reduce software development and maintenance time, thereby increasing productivity.

This page intentionally left blank.

# Table of Contents

# List of Figures



# List of Listings

# 1. Introduction

The National Aeronautics and Space Administration (NASA) has developed a simulation environment for the National Airspace System (NAS) called NAS Digital Twin [1] to support Air Traffic Management (ATM) research. Running a meaningful and successful simulation requires multiple stages: preparing adaptation data, creating scenario files, configuring simulation environments, running simulations and performing post-simulation analyses. In each stage, there is a need to visualize information including airspace elements, geographical information, flown and predicted flight trajectories, historical and real-time air traffic and weather, and data recorded during simulation runs.

An interactive, two-dimensional visualization tool called Traffic Visualization Tool has been created to meet these needs. The tool supports discrete- and real-time simulations. The tool can connect to NASA's Sherlock ATM Data Warehouse, or Sherlock [2], to access airspace adaptation, historical traffic, and weather data. In addition, the tool is expandable by implementing custom drawable elements, layers, dock panels, and element repositories. The development of the visualization tool has four design goals: ease of code base access, ease of data source connectivity, component reusability, and software extensibility. With reduced cost and effort, aviation projects and individual researchers can further design their own, related tools by leveraging any part of libraries including data models, drawable elements, file parsers, and Sherlock connectivity.

The visualization tool is written in the Java [3] programming language and has been developed based on the use cases and needs of supporting NAS Digital Twin simulations and post-simulation analyses. The tool is platform agnostic and can be run on major operating systems such as *Windows*, *Linux*, and *MacOS*. This document describes the user interface design, software capabilities and features of the visualization tool.

# 2. Getting Started

Building the visualization tool project requires three open-source repositories and two private repositories. The open-source repositories are hosted by Central Repository [4], *Unidata* [5], and Open Source Geospatial Foundation [6]. The two private repositories, namely, *AF-Lib* and *AF-Tools*, are hosted by the Aviation Systems Division at NASA Ames Research Center. The former private repository consists of general purpose, common data models and utilities for projects, while the latter private repository consists of specific tools including *Autoresolver* [7], trajectory generators [1, 8], and the traffic visualization tool.

```
$ git clone ssh://git@atmjira.arc.nasa.gov:7999/nasdt/ndt
$ cd ndt/
$ bin/install-ndt
$ cd dev/af-tools/traffic-visualization/
$ gradle run
```

*Listing 2.1 Commands to build and run Traffic Visualization Tool*

Listing 2.1 lists the commands to check out the NAS Digital Twin project repository, install required repositories, and build and run the visualization tool. The initial application frame is shown on Figure 2.1. The main area contains a canvas for showing drawable elements such as airspace boundaries, graticules, map scale, and current simulation time. The canvas supports three mouse interactions: a) Click the left mouse button and drag the mouse to pan the canvas view; b) Use the mouse wheel to zoom in or out the canvas view; and c) Click the right mouse button to show a context-sensitive popup menu. Widget buttons are placed on the four docks located at the top, left, bottom, and right sides of the application frame. Clicking a widget button

will toggle the visibility of the corresponding widget panel. Navigating to a specific widget panel can also be performed from the Tab menu.

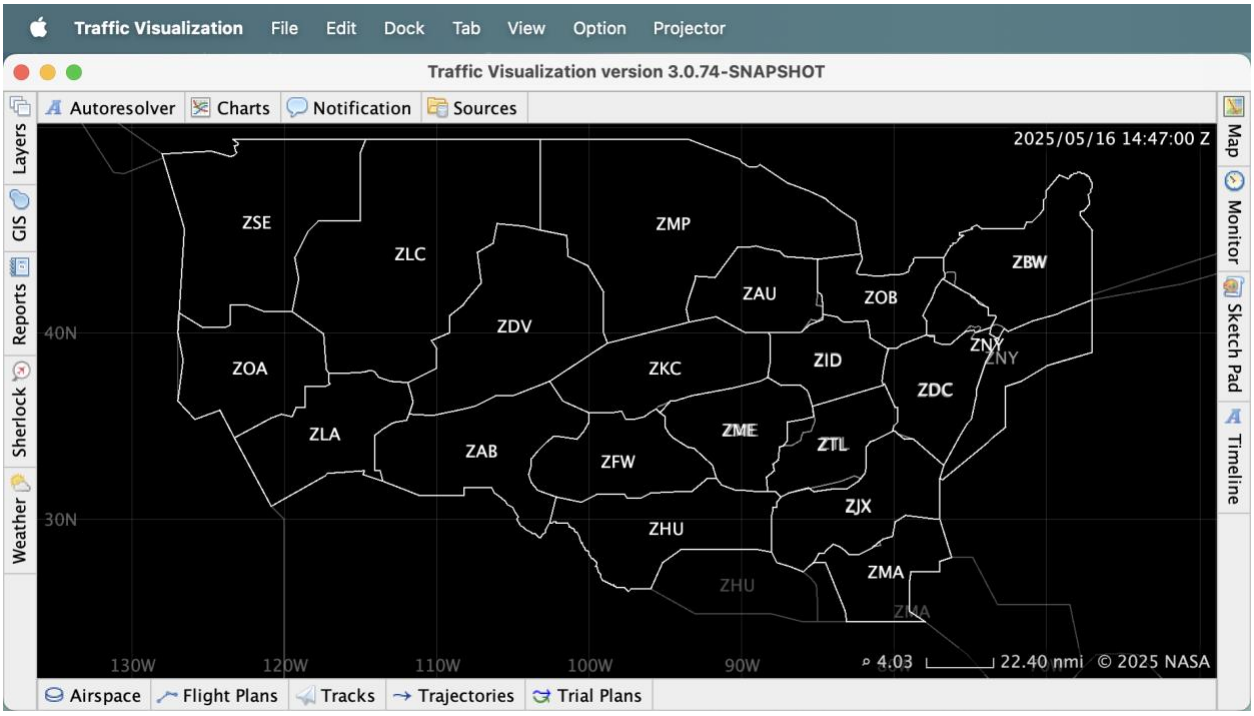



*Figure 2.1 Traffic Visualization Tool application frame*

Section 3 discusses the application specific design. Section 4 describes sources of static airspace elements and their rendering mechanisms. Section 5 presents the charting capability of flight data. Section 6 describes the details of flight information. Section 7 shows supported geographic information system data file formats. Section 8 provides a brief description how to visualize vector map tile elements. Section 9 presents the reporting capability for post-simulation data analyses. Section 10 describes data sources supported in the NASA's data warehouse. Finally, Section 11 provides a brief description and instructions how to visualize weather information.

# 3. Application Basics

Elements storing information including images, locations, paths, texts, etc. for visualization implement the *Drawable* interface (see Section 3.3). Drawable elements are rendered on a canvas using context-based parameters. For example, a path is rendered using a fill color, a line color, and a line stroke, while text is rendered using fill and text colors. The following subsections present application specific components created for the visualization tool.

## 3.1. Canvas

The visualization tool has a canvas, a graphical component, for rendering drawable instances like airspace and traffic elements, layers, texts, and user-defined objects. Figure 3.1 presents the canvas in black, its associated properties in yellow, and the cursor's associated properties in cyan. This section describes these associated properties.

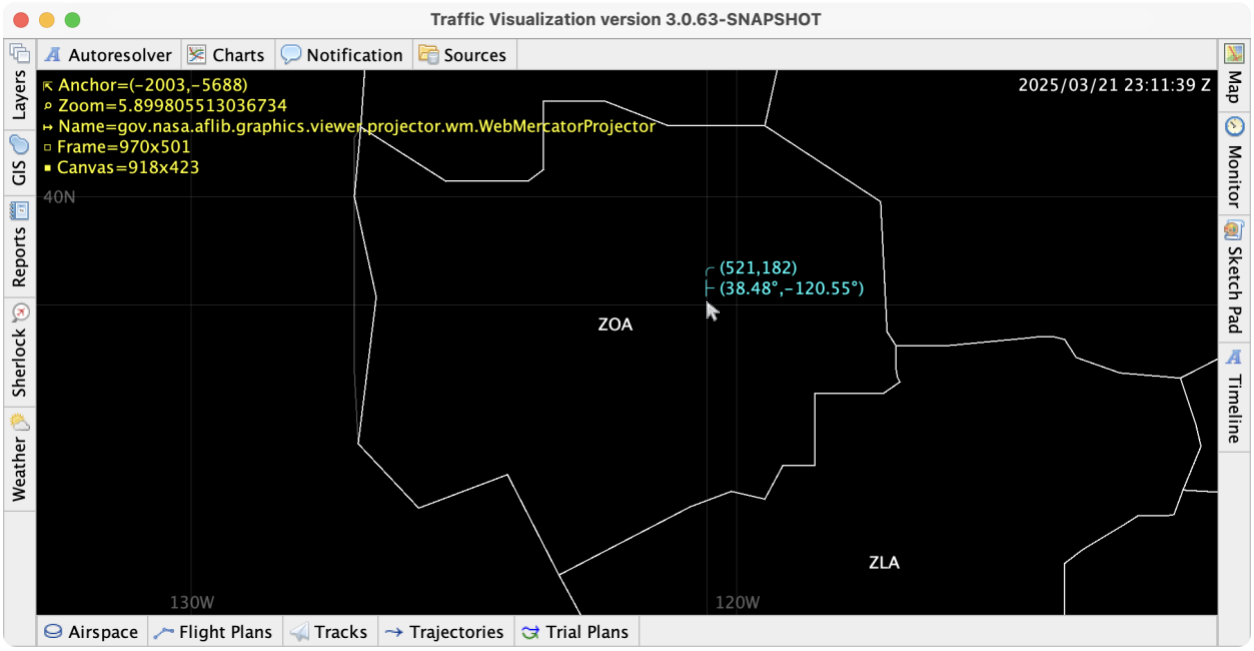



*Figure 3.1 Canvas properties*

An anchor point *A* represents the top-left location of view of a canvas. In Figure 3.1, the view of the canvas has been panned to 2003 pixels towards West and 5688 pixels towards North, the coordinates of the anchor point are A(-2003, -5688). Drawable elements are rendered with respect to the anchor point.

A cursor point *C* indicates the location of the cursor with respect to the top-left corner of the canvas. In Figure 3.1, the cursor is located at 521 pixels East and 182 pixels South of the top-left corner, the coordinates of this cursor point are C(521, 182). The cursor point is also used as the control point when a user interacts with the view such as panning and zooming in or out.

A layer list contains layers and sublayers for rendering drawable elements (see Section 3.4). The canvas creates and initializes a default layer list when launching the visualization tool. Layers are ordered in the sense that elements defined in the first (bottom-most) layer are rendered first, and elements defined in the last (top-most) layer are rendered last.

Application configurations, for example, lists of enabled and disabled layers, are stored in a properties file named `viewer.config`. The visualization tool finds the configuration resources and the file in the following locations. Properties defined in the configuration file will overwrite any existing properties defined in the configuration resources.

1. `gov/nasa/aflib/graphics/viewer/viewer.config` (resources)
2. `config/viewer.config` (file)

A repositories instance stores shareable objects that can be accessed among layers and panels. Objects of similar type are stored in a repository that are keyed by the repository's class name. To illustrate this concept, a Flight Plan Repository stores flight plans while a Track Repository stores tracks. A layer may render drawable elements by utilizing multiple repositories. For example, a trajectory prediction layer may access the flight plan and current location of a specific flight from the Flight Plan Repository and the Track Repository, respectively, and then draw the predicted locations calculated by a trajectory generator.

A control allows users to interact with the canvas, such as showing a context-sensitive popup menu by right click, panning the view of the canvas by left drag, and zooming the canvas by mouse scrolling. Control actions can also be performed by pressing shortcut keys.

User settings of the visualization tool can be saved to and loaded from profiles (see Section 3.7). This feature allows users to restore the application states, for example, the current anchor point and enabled layers, so that the states can be reused upon subsequent runs.

A repaint timer repaints the canvas content at a fixed period. Initially, the canvas is repainted once per half second (or 500 milliseconds). The canvas can also be repainted programmatically by calling the `ViewerCanvas.repaint()` method. For example, geographic data defined in a vector map tile can be rendered on the canvas instantly by calling the repaint method when decoding tile data completes.

## 3.2. Docks

The visualization tool provides four docks to group similar widgets (see Figure 3.2). Docks can be placed on one of the four sides: top, left, bottom, and right. A component added to a dock is associated with a tab. The tab contains a text title and an optional icon. Clicking a tab toggles the visibility of the component, hence the visibility of the dock. Dragging and dropping a tab in the same dock moves the tab's position, while dragging and dropping a tab to a different dock moves the tab from one dock to another. Docked components can also be navigated directly from the Tab menu.


*Figure 3.2 Expanded left and right docks*

A layer can add components to docks by overriding the method `Layer.addDockPanels(DockPanelMap map)` and calling the map's adder or setter methods. In Figure 3.2, the Vector Map Layer added the Map Layers panel to the right-side dock.

## 3.3. Drawable Elements

A drawable element is a primitive item to be rendered on a canvas. To give some examples, a line drawable element draws a line between a pair of geographic coordinates, an image drawable element draws an image centered at canvas display coordinates, and a marker drawable element draws a shaped marker centered at geographic coordinates.

Complex drawable elements can be constructed using multiple primitive or complex drawable elements. For example, a flight plan drawable element consists of drawable elements for drawing callsign text, route path, a truncated line between the first and the second waypoints, waypoint markers, and waypoint names.

In the visualization tool, drawable elements should implement the *Drawable* interface method called "`draw`" which takes two parameters: a Java's *Graphics2D* instance (`g2`) and an AF-Lib's *Draw Parameters* instance (`params`), as shown in Listing 3.1.

```java
/**
 * Draws element content.
 *
 * @param g2 graphics context to use for drawing.
 * @param params parameters to use for drawing.
 */
void draw(Graphics2D g2, DrawParams params);
```

*Listing 3.1 Drawable interface*

The draw parameters instance stores information and references to the current drawing operation including the viewer canvas, projection function for projecting geographic coordinates onto canvas display coordinates, the anchor point indicating the top-left corner location of the view, cursor location, bounds of the component, repositories of the data, text drawer for text drawing, current simulation time, and a monitor for measurement data. The method signature allows the visualization tool to be enhanced in the future by having additional information and references in the draw parameters instance without breaking any existing implementations.

In addition, each drawable element has an associated Drawable Context defining the properties for drawing lines and texts. By default, drawable context instances are grouped by the drawable element's class. Multiple drawable elements can be grouped in a drawable list. Elements in a drawable list are rendered when the drawable list is enabled, which is `true` by default.

## 3.4. Layers

A layer draws a group of drawable elements. For example, a track layer draws a vehicle-type specific icon at the vehicle's last known location and line segments connecting each pair of vehicle's flown locations.

A layer can be enabled or disabled through a configuration file, user interaction via the Layer Tree panel, or programmatically using the layer instance. Drawable elements referenced in a disabled layer will not be rendered on the canvas.

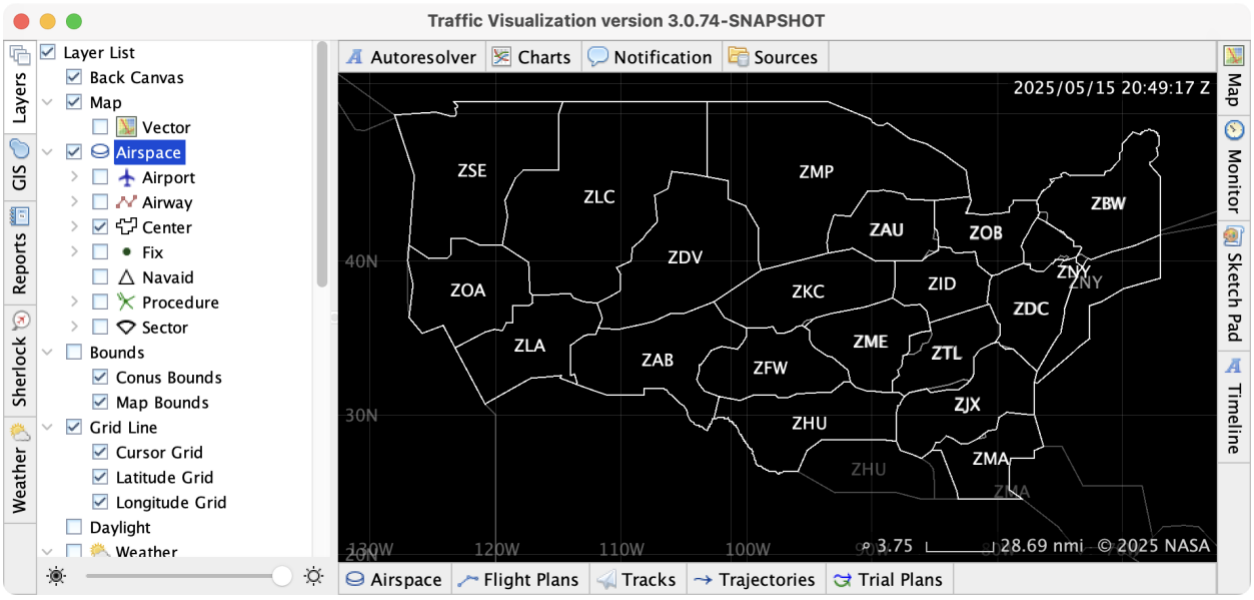



*Figure 3.3 Layer tree panel (left)*

Like the drawable elements, multiple layers can be grouped in a layer list to form a parent-child hierarchy. Figure 3.3 shows a layer tree panel. The airspace element layer list (parent) contains seven airspace element layers (children). Each airspace element layer draws specific type of airspace elements that are described in Section 4. Layers in a layer list are rendered when the layer list is enabled, which is `true` by default. The visualization tool design assumes each layer class has a public constructor without parameters.

## 3.5. Monitor

The Monitor tab shows the rendering performance of individual layers and metrics such as number of visible items on the canvas' viewable area. Custom and layer specific metrics can be added to the monitor defined in the drawable parameter. The Monitor class provides two methods for adding duration and metrics, respectively, as shown in Listing 3.2.

```
/**
 * Adds duration taken to draw layer.
 *
 * @param layer layer to be used.
 * @param startMs start time, in ms, before the draw operation.
 */
public void addDuration(Layer layer, long startMs);

/**
 * Adds key/value pairs to the layer's metrics. This method will
 * overwrite existing pairs.
 *
 * @param layer layer to be used.
 * @param key first key.
 * @param value first value.
 * @param moreKeyValuePairs array of key-value pairs:
 *  <ol>
 *  <li>even index = (String) key.
 *  <li>odd index = (Object) value.
 *  </ol>
 */
public void addMetrics(Layer layer, String key, Object value,
    Object...moreKeyValuePairs);
```

*Listing 3.2 Monitor methods*

## 3.6. Notification

Notification supports information messages such as database and file loading progress, and error messages encountered during operation such as loading a corrupted file. Custom messages can be added via the *Notifier* class. The visualization tool provides two mechanisms to show notification messages: short-term toast messages on the canvas and long-term messages in the Notification panel. Toast messages are shown on the bottom-left corner of the canvas. Toast messages are updated whenever the canvas is repainted. By default, each toast message is shown for three seconds (configurable). The last 100 (configurable) notification messages are listed in the table. Figure 3.4 shows a notification message in the panel and a toast message on the canvas.

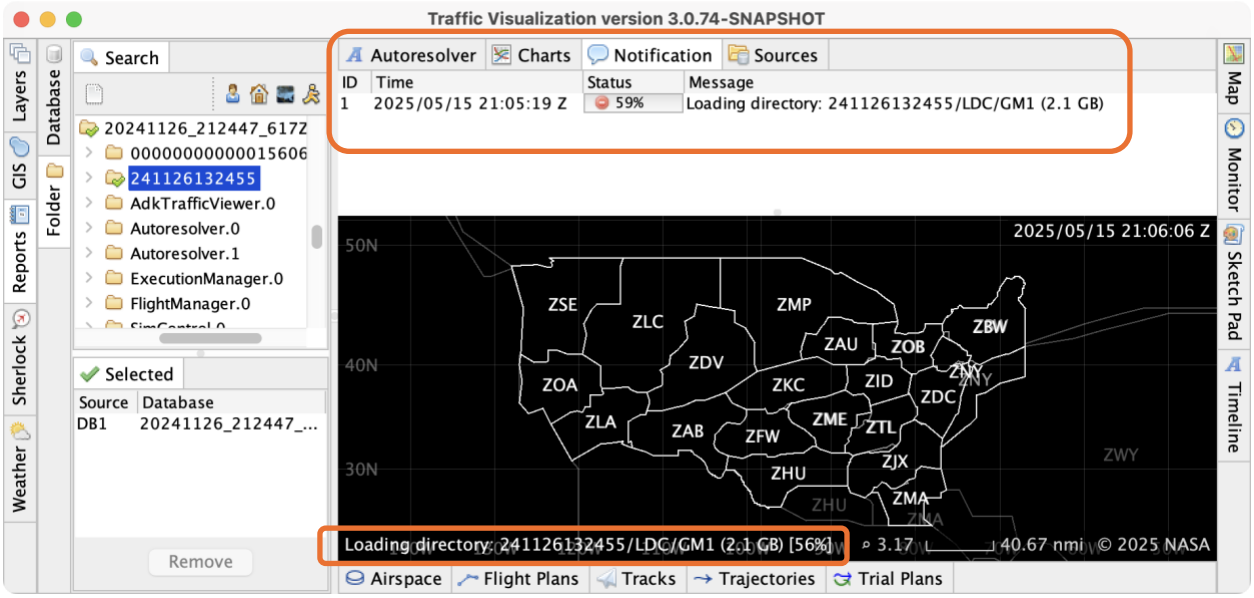



*Figure 3.4 Notification panel (top) and toast message (bottom)*

## 3.7. Profile

A profile stores the current settings in the visualization tool such as current view bounds and enabled map layers. This allows the tool to support user-specific and project-specific views without manual configurations after launching the visualization tool. For example, a project may want to zoom in the canvas to a particular center boundary with three enabled map layers—Land, Airport, and Road.

To support user-configurable default values, keys with default values are not stored in the profile. For example, assume there are three map layers: `L1`, `L2`, and `L3`. Both layers `L1` and `L2` are enabled but layer `L3` is disabled. If layer `L2` is disabled by a user, then the profile will only store the following (nominal) entries:

```
L2 { "enabled": "false" }
```

instead of

```
L1 { "enabled": "true" },
L2 { "enabled": "false" },
L3 { "enabled": "false" }.
```

This provides a flexibility that if a user wants to change the default configurations of the visualization tool, for example, enable layer L3 by default, then none of the existing profiles need to be updated.

## 3.8. Projectors

A projector contains functions for converting between world and screen coordinates. World coordinates use latitude and longitude degree values, while screen coordinates use x and y pixel values. The project function converts from world to screen coordinates, and the un-project function converts from screen to world coordinates (see Listing 3.3).

```
/**
 * Projects coordinates from map space onto screen space. This method
 * updates the <code>screen</code> parameter.
 *
 * @param map input coordinates on map space.
```

```
     * @param screen output coordinates on screen space.
     * @see #unproject(Point2D, LatLong)
     */
    public abstract void project(LatLong map, Point2D screen);

    /**
     * Unprojects coordinates from screen space onto map space. This method
     * updates the <code>map</code> parameter. This method is considered an
     * inverse projection function.
     *
     * @param screen input coordinates on screen space.
     * @param map output coordinates on map space.
     * @see #project(LatLong, Point2D)
     */
    public abstract void unproject(Point2D screen, LatLong map);
```

*Listing 3.3 Projector methods*

Two projectors are implemented so that two projections are supported in the visualization tool: Web Mercator Projection (EPSG:3857) [13] and Lambert Conformal Conic Projection (EPSG:9802) [14]. Custom projections can be added by creating concrete classes that extends the *Projector* base class.

Fractional zoom levels are supported between 0 and 23, the range commonly used in the Web Mercator Projection. Thus, drawable elements can be rendered using a continuous scaling function that is based on the projector's zoom level.

## 3.9. Sketch Pad

The visualization tool provides a Sketch Pad panel for users to draw elements on the canvas by running text-based commands that can be copied and pasted using text editors. This feature is useful for adding annotations to the canvas at application runtime. The user drawable elements are rendered on a Sketch Pad Layer that is usually near the last (i.e., topmost) layer in the layer list. Commands are methods defined in a class that extends the `DrawCommand` class. The methods must be public, accept a single command parameters instance, and return a `Drawable` instance. For example, the `SketchPadCommand` class has a line method to draw a line and the method signature is listed in Listing 3.4.

```
    /**
     * Returns a line connecting two map points:
     * <pre>
     *  line [latDeg1] [lonDeg1] [latDeg2] [lonDeg2]
     * </pre>
     *
     * @param params command parameters to be used.
     * @return a drawable.
     * @throws NoSuchElementException if not enough parameters.
     * @throws NumberFormatException if numeric value could not be parsed.
     */
    public Drawable line(CommandParams params);
```

*Listing 3.4 Line method in the SketchPadCommand class*

Users can define their own draw command class and specify it in a viewer configuration file. Figure 3.5 shows a screenshot of the visualization tool with two sets of range rings on top of the map layer, where the range rings are user drawable elements generated by the commands in the Sketch Pad panel.

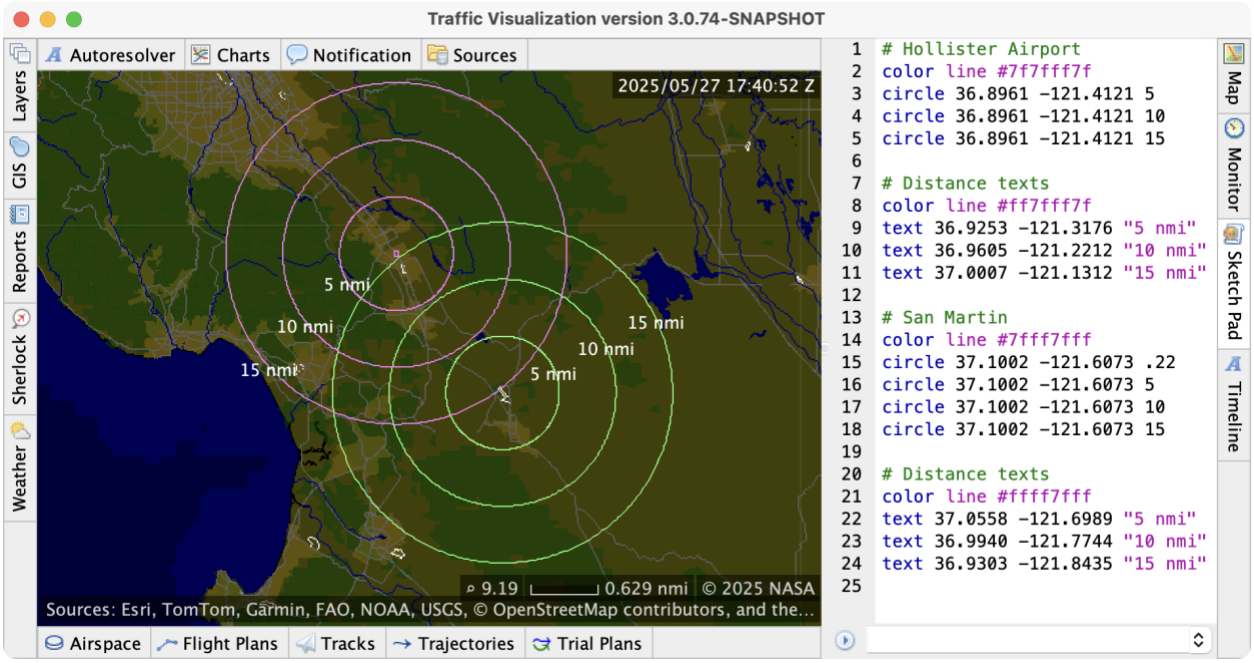



*Figure 3.5 Drawing range rings on map via Sketch Pad*

## 3.10. Sources

Sources are used to define the origin of the drawable elements. For example, when comparing two flights with the same airline number (e.g., `AAL123`) by loading data from two historical days, the two flights will have different sources.

Sources can be used to control the rendering properties. In the above example, the path of the first flight can be rendered in yellow, while the path of the second flight can be rendered in green.

Sources can also be used to control the drawable elements. The flight data associated with the second historical day can be removed from the application storage without affecting the flight data associated with the first historical day. Similarly, adding flight data loaded from a third day will not affect the existing flights.

The visualization tool lists all the sources in the Source tab and provides a mechanism to query the database table names, schema, and data. Figure 3.6 shows the Database tab where table names are listed in a tree widget, both table column schema and table row data are presented using table widgets. Users run `SELECT`-queries in database-specific Structured Query Language (SQL) to select database records.

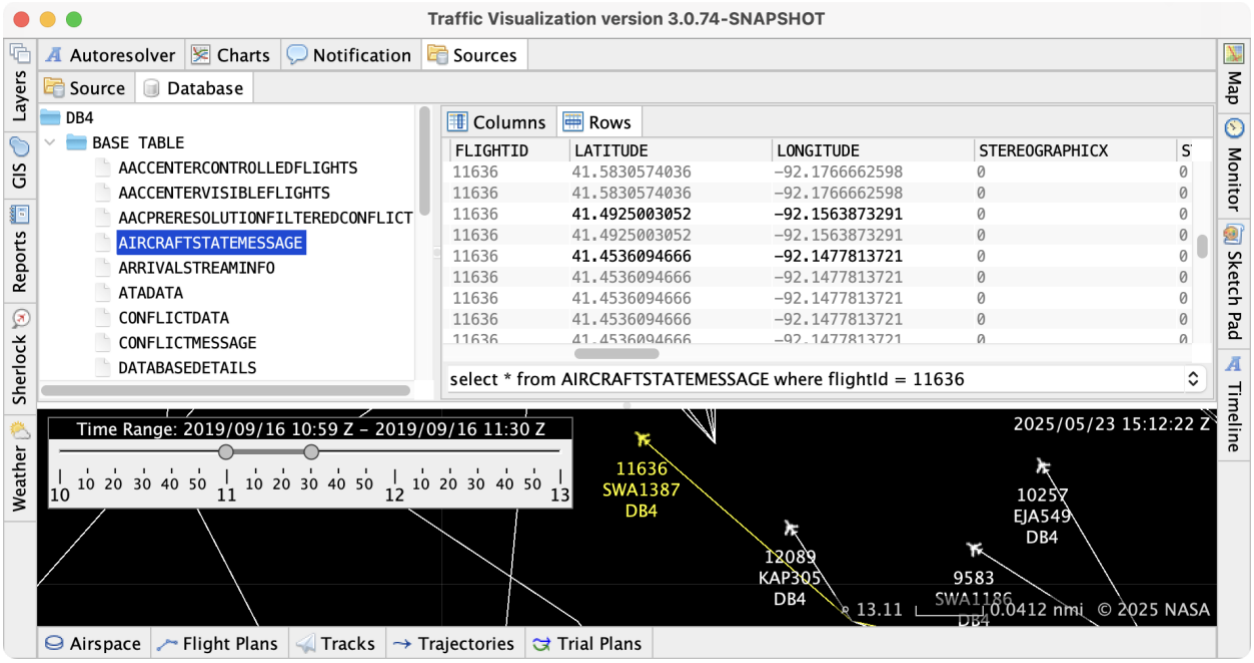



*Figure 3.6 Querying database records using a SQL statement*

## 4. Airspace Elements

The Federal Aviation Administration's (FAA's) Aeronautical Data provides aeronautical information including physical description of the airspace elements such as 28-day National Airspace System Resource [9]. The files can be accessed via the FAA's Aeronautical Information Portal, formerly National Flight Data Center (NFDC) Portal [10], and the NASA's Sherlock. Information presented in this section is based on the 2024/04/18 (Thursday) NFDC data files.

Users may interact with the airspace elements using a filter-search-update approach. Figure 4.1 shows the Airports tab consisting of three panels: Filter, Search, and Properties. The Filter panel lists each element type as a check box indicating the visibility of the elements with such type in the search table. Initially, all the check boxes are enabled, thus, all the elements are searchable. The Search panel consists of a table and a search box. The table lists all the filtered elements and their useful information. The search box allows users to refine the items to be listed in the table. Search terms can be a text for subsequence match. For example, searching for the term `SFO` will match both the San Francisco International Airport (`KSFO`) and the Fazenda Novo Horizonte Airport (`SSFO`). Search terms can also be in a `[column]:[text]` format to limit the matching of values in a specific column. For example, searching for `IATA:SFO` will match the San Francisco International Airport but not the Fazenda Novo Horizonte Airport. The Properties panel allows users to preview and update the following seven drawing attributes of selected items in the search table: a) Enabled (selected to enable the attribute filter), b) A Text color, c) A Text fill color, d) Z Line color, d) Z Line fill color, f) Line width and dash style, and g) Stop if applied (selected if subsequent filters will not be applied).

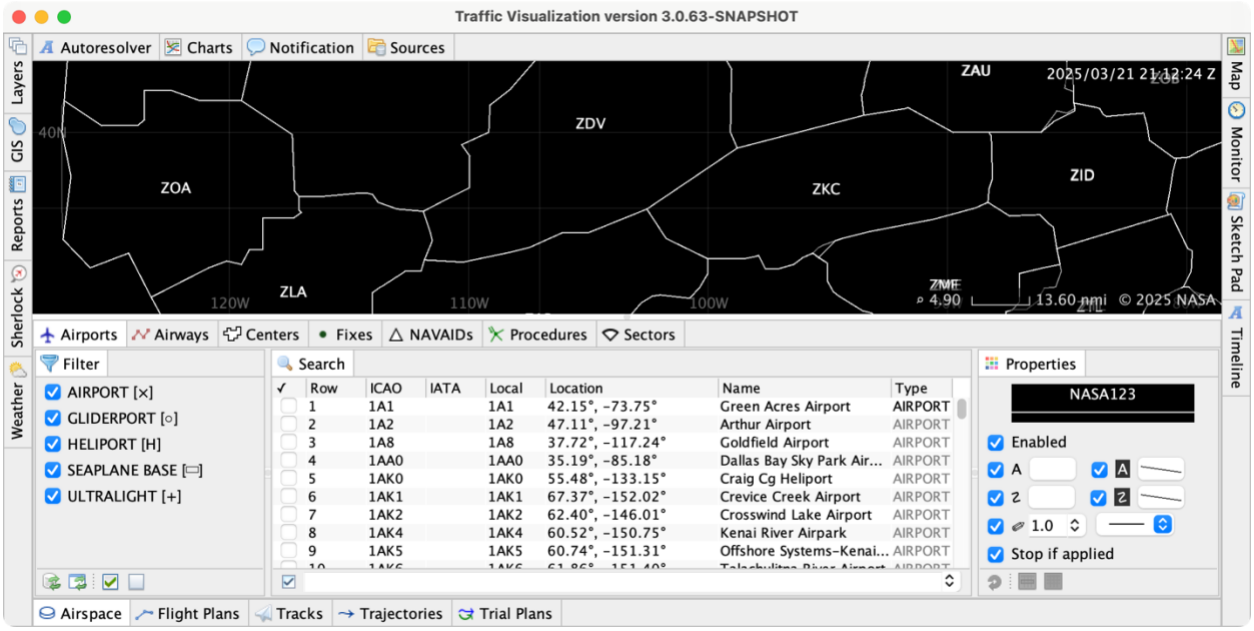



*Figure 4.1 Airports tab from Airspace dock*

Each airspace element has a unique airspace identifier, a name, and a type. When rendering the airspace elements in the visualization tool, their geographic coordinates, latitude and longitude, are projected onto the canvas using the current projector (See Section 3.8). The following subsections describe each airspace element supported in the visualization tool. Detail aeronautical information can be found on the FAA's Aeronautical Information Manual [11].

## 4.1. Airports

Information of airports in the United States are obtained from the FAA's aeronautical data. Airports outside the United States are obtained from the *OurAirports* [12]. The data sources include five airport types. When rendering the airports on canvas (see Section 3.1), these types presented with different markers: airport (x), gliderport (o), heliport (H), seaplane base (▭), and ultralight (+).

In the visualization tool, airports are identified by three codes: a) an International Civil Aviation Organization (ICAO) airport code, b) an International Air Transport Association (IATA) code, and c) a local code. For airports in the United States, the local code is also the FAA identifier in the aeronautical data. For example, the San Francisco International Airport in California has the ICAO airport code `KSFO`, IATA code `SFO`, and the local code `SFO`. Note that not all the codes are used. The small, private, or non-commercial airports do not have the IATA codes. For example, the Carey Airport in Idaho has both the ICAO code and the FAA identifier `U65` but not IATA code.

## 4.2. Airways

Airways are defined route structures a flight usually follows from a source location to a destination location along a series of radio *fixes*. Airways can be considered "highways in the airspace" where the fixes are the route intersections. The FAA's NDFC data files store three types of airways: Alaska Airway, Hawaii Airway, and Federal Airway. These types are defined as enumeration values in the `AirwayType` class, e.g., `AirwayType.FEDERAL`.

In the visualization tool, the airways are identified by a code and a type. The airways are rendered using connected line segments, where connection points are fixes. Figure 4.2 shows

the airway J16 with 18 fixes. The code J16 is shown on top of the first and last fixes, and the name of each fix is shown on the bottom of the fix.

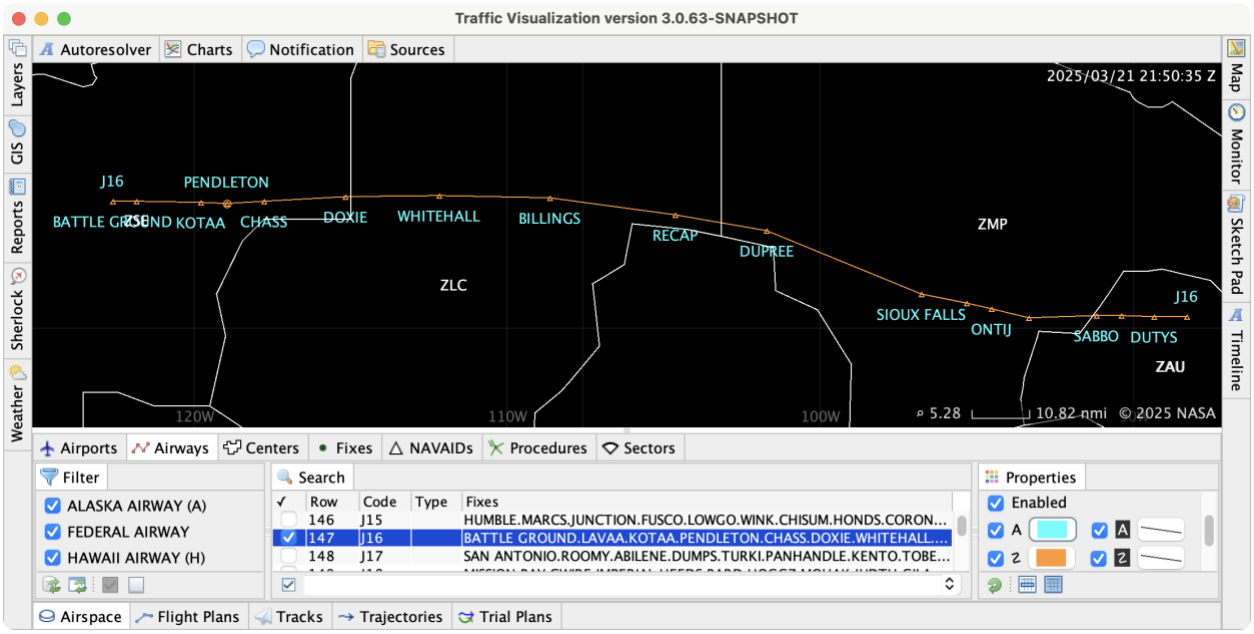



*Figure 4.2 Airway J16*

## 4.3. Centers

A center represents an Air Route Traffic Control Center (ARTCC). A center has three kinds of boundaries: a) a high boundary covering a high-altitude airspace region, b) a low boundary covering a low-altitude airspace region, and c) an optional other boundary covering specific airspace regions like control area, flight information region, and upper control area. A center has a three-letter identifier and a name. For example, ZOA represents the Oakland ARTCC.

The visualization tool draws the high, low, and other center boundaries in white, gray, and light gray, respectively. A user can zoom the canvas view to a center's boundary by right clicking anywhere inside the center boundary and then selecting the "Zoom to *Center*" popup menu. Figure 4.3 shows a case to zoom the canvas view to the Oakland ARTCC.

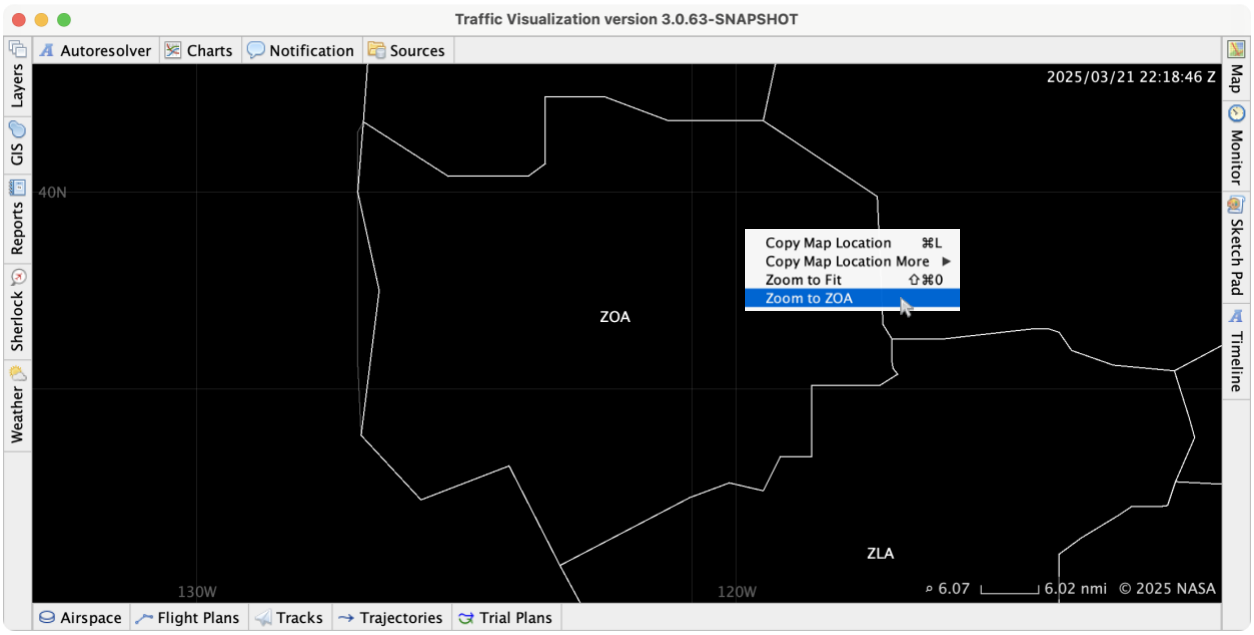



*Figure 4.3 Zoom to ZOA center via popup menu*

## 4.4. • Fixes

Fixes are radio-fixes or reporting points in the National Airspace System. A fix may have one or more properties and their corresponding markers in the visualization tool:

1. An Instrument Landing System (ILS) fix (marker: ○),
2. A Navigational Aid (NAVAID) fix (marker: △),
3. A dynamic fix (marker: +) which is dynamically generated, e.g., using a `latitude/longitude` convention or a `fix/radial/distance` convention,
4. A base fix (marker: +) is none of the above.

A fix has an identifier and a location. For example, `EBAYE` is a base fix located at `35.87°N/120.26°W`; `BAYFA` is both an ILS and a NAVAID fix located at `37.74°N/122.27°W`; `PAYSO189052` is a dynamic fix in the fix/radial/distance convention located at `33.50°N/110.96°W` which is 189 degree and 52 nautical miles from the NAVAID fix `PAYSO`.

## 4.5. △ NAVAIDs

Navigational Aids, or NAVAIDs, are physical ground equipment. In the FAA's NFDC files, each NAVAID has an airspace element identifier with format `id/name/type`. The NFDC files have the following types defined.

1. DME: Distance Measuring Equipment
2. FAN MARKER: Fan Marker
3. MARINE NDB: Marine Nondirectional Radio Beacon
4. NDB/DME: Nondirectional Radio Beacon/Distance Measuring Equipment
5. TACAN: Tactical Air Navigation
6. VOR: Very High Frequency (VHF) Omni-directional Range
7. VOR/DME: VHF Omni-directional Range/Distance Measuring Equipment
8. VORTAC: VHF Omni-directional Range/Tactical Air Navigation
9. VOT: VOR Test Facility

An example NAVAID identifier is `EWR/MARYANN/FAN MARKER`. NAVAIDs are indicated by a white up-pointing triangle (△) in the visualization tool.

## 4.6. Procedures

A procedure defines a series of fixes for a pilot to follow. The visualization tool renders both the departure and arrival procedures defined in the FAA's NFDC data files. Departure procedures are in blue and arrival procedures are in green. Each procedure has six properties: a procedure identifier, a code, a name, a sequence, a list of fixes, and a flag indicating whether it is a departure procedure or not. Departure and arrival procedure codes have the formats `[departureName].[transition]` and `[transition].[arrivalName]`, respectively. For example, Figure 4.4 shows two departure routes from the San Francisco International Airport (KSFO) with the departure procedure "NIITE THREE" (code `NIITE3.NIITE`) and their fixes are:

1. `SFO.GALOO.NIITE` (cyan dashed line)
2. `SFO.MDBAY.HUSSH.NIITE` (blue solid line)

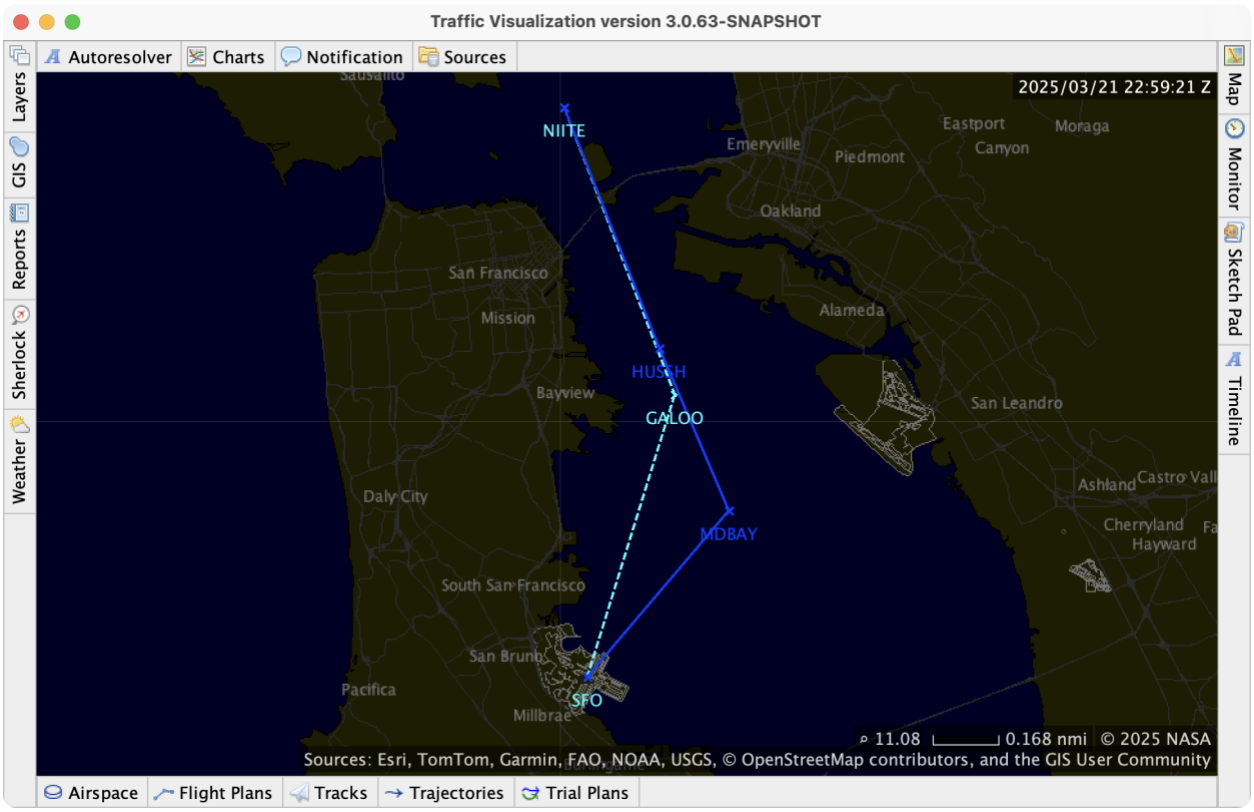



*Figure 4.4 Two routes for departure procedure NIITE THREE*

## 4.7. Sectors

A sector represents an airspace region in a center or an ARTCC. Each sector consists of one or multiple smaller airspace regions called subsectors. A sector's type depends on its altitude coverage between ground and 60,000 feet (or Fight Level 600). Depending on the center and traffic patterns, a center may have low sectors, high sectors, and superhigh sectors.

# 5. Charts

The visualization tool utilizes *JFreeChart*, an open-source two-dimensional chart library for Java applications [15], for plotting trajectory information of highlighted flights in any of the flight information panels (see Section 6). To avoid plotting excessive flight information in the charts, up to 100 selected flight items will be handled. The Charts tab consists of a legend panel, a settings panel, and seven chart panels to be described in this section. The size of each of these

panels can be adjusted by dragging the split bar located on the panel's right-hand side. Figure 5.1 shows a subset of chart panels of three selected flights.

The separation charts show the value difference vs. time of the selected flights. The base flight is the first selected flight in the legend panel. These charts are useful for understanding the separation values based on the trajectories of the selected flights.

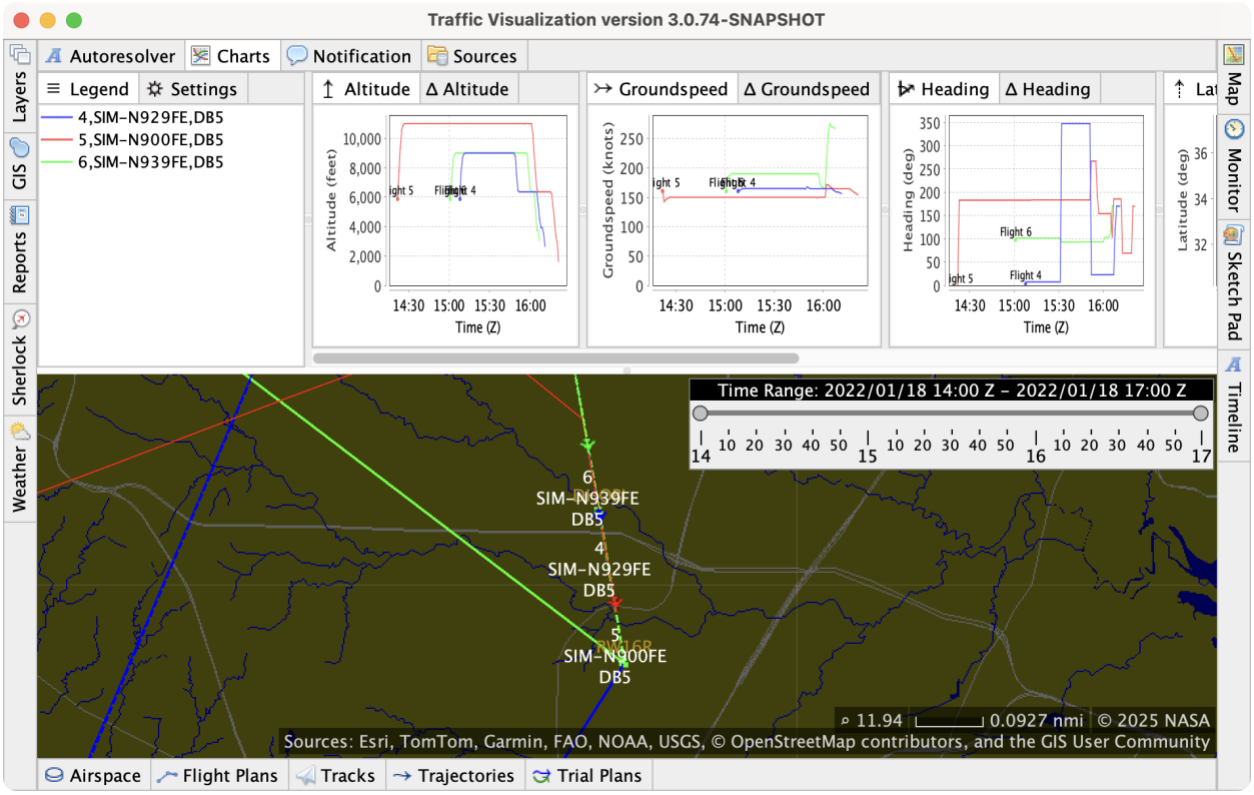



*Figure 5.1 Charts showing flight profiles*

The Charts tab contains the following charts and panels for visualizing flight profiles and configurating charts:

1. The ≡ Legend panel shows the legend of individual track identifiers (See Figure 5.1). Each track identifier consists of three components: a numeric flight identification number, a callsign string, and a data source identifier. Highlighting track identifiers in the legend panel will also highlight the corresponding datasets among the charts. Like the concept of freezing the first column in Microsoft Excel sheets, the legend panel is "frozen" in the Charts such that horizontally scrolling the six chart panels will not affect the view of the individual track identifiers.
2. The ⛭ Settings panel shows chart settings controlling the following features of each chart: a) series labels, b) data points, c) crosshairs, d) date and time axes. In addition, the settings panel allows a user to resize the charts so that their widths and heights are the same.
3. The ↥ Altitude vs. Time chart shows the altitude profiles of the selected flights. The x-axis represents the time, and the y-axis represents the altitude values, in feet. This chart is useful for understanding the climb, cruise, and descent segments of flights.
4. The ↣ Groundspeed vs. Time chart shows the groundspeed profiles of the selected flights. The x-axis represents the time, and the y-axis represents the groundspeed values, in knots. This chart is useful for understanding the speeds along the climb, cruise, and descent segments of flights.

5. The Heading vs. Time chart shows the heading profiles of the selected flights. The x-axis represents the time, and the y-axis represents the heading values, in degrees. This chart is useful for understanding the heading change in the climb, cruise, and descent segments of flights.
6. The Latitude vs. Time chart shows the latitude profiles of the selected flights. The x-axis represents the time, and the y-axis represents the latitude values, in degrees.
7. Like the Latitude chart, the Longitude vs. Time chart shows the longitude profiles of the selected flights. The x-axis represents the time, and the y-axis represents the longitude values, in degrees.
8. The Path chart shows the latitude vs. longitude of the selected flights. The x-axis represents the longitude, in degrees, and the y-axis represents the latitude values, in degrees. This chart is useful for understanding the trajectories of the selected flights on the horizontal plane.

# 6. Flight Information

The visualization tool can also access flight information such as historical flight plans and tracks stored in the Sherlock. Section 10 describes in detail the user interface for accessing Sherlock data.

## 6.1. Flight Plans

In the visualization tool, a flight plan consists of a flight plan identifier, a vehicle type, a flag that indicates whether the waypoints between the first and second waypoints are truncated or not, a departure airport, an arrival airport, and a list of waypoints along the flight route. The truncated flag, both the departure and arrive airports, as well as the list of waypoints can be derived from a flight plan route. For example, the route `KPHX./.PAYSO189052..MAXXO..CNX..MMB.J26.JOT.J146.GIJ.J554.CRL..DKK..BUF..KBUF` has the following properties:

1. Truncated = true
2. Departure airport = `KPHX` (Phoenix Sky Harbor International Airport)
3. Arrival airport = `KBUF` (Buffalo Niagara International Airport)
4. List of waypoints = `PAYSO189052`, `MAXXO`, `CNX`, …, `DKK`, and `BUF`.

In the Finder panel, like the airspace elements, users may filter, search, view, and update the flight plan elements. The Filter panel lists each flight plan type as a check box indicating the visibility of the elements with such type. The Search panel allows users to refine, mark, or highlight the items listed in the table.

In the User Plans panel, a user may enter custom callsigns and flight plan routes. The visualization tool will render the custom flight plans on the canvas, as shown in Figure 6.1. The input text area supports syntax highlighting: Valid waypoint names and route delimiters are in black, invalid or unknown waypoints are highlighted in red, resolvable waypoints such as latitude/longitude coordinates in a lenient format (not officially supported in the FAA's flight plan format) are highlighted in yellow.

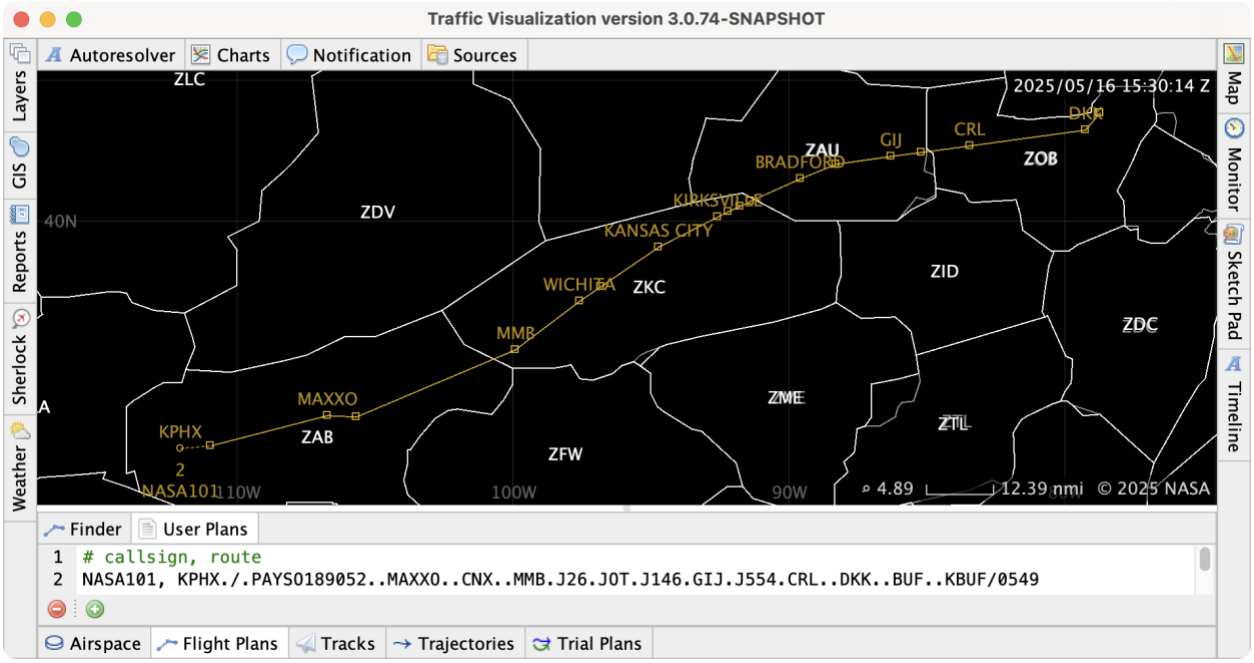



*Figure 6.1 Custom flight plan route*

The visualization tool includes a flight plan parsing library to extract information from flight plan routes such as departure and arrival airports, departure and arrival transition routes, and waypoint's geographic locations (latitude and longitude) by names. In the canvas, the flight plan identifier is shown at the bottom of the departure airport location indicated by a circle marker (○), each waypoint has a square marker (□), and the name of waypoint is drawn on top of each marker. When a flight plan is truncated, the segment between the departure airport and the first waypoint is rendered in a dotted line.

## 6.2. Tracks

A track consists of an identifier, a vehicle type, a resizable array of track points, and a retention period. A track point contains time-specific vehicle telemetry data. Whenever a track point is added to a track, the elements in the array are ordered based on the track point's timestamp. In the visualization tool, a track is rendered by three drawable elements: a vehicle image at the last known location, a polyline with each line segment connecting two consecutive track points, and a tag showing the identifier at the last known location.

The retention period specifies the time period during which the track data is stored in application memory. If the difference between the current simulation time and the last track point time exceeds the period, the track is considered outdated and will be removed from the track repository. During a simulation run, the retention period is set to 5 minutes. For post-analyses, the retention period is set to unlimited so that historical tracks are kept in the memory.

## 6.3. → Trajectories

Users can add predicated trajectories based on flight plan information and current aircraft states. The following three types of trajectory generators are supported:

a. Interpolation Trajectory Generator [1] based on kinematic trajectory generation,
b. Multi-Purpose Aircraft Simulation [8] based on four degree-of-freedom equation of motions, and
c. Point Mass Trajectory Generator [1] based on point mass equations along a prescribed trajectory in energy coordinates.

Custom trajectory generators can be added by creating concrete classes that extend the *Trajectory Generator* base class.

# 7. Geographic Information System Data

Geographic Information System (GIS) data describes information associated with one or more locations on Earth. Locations are recorded using latitude and longitude coordinates. In the GIS tab, GIS data files can be loaded, and GIS information can be visualized. Currently, both the Shapefile [16] and Keyhole Markup Language (KML) file formats are supported.

The Shapefile file format, developed by the Environmental Systems Research Institute, Inc. (ESRI) [17], stores GIS data. Certain noise analysis tools [18, 19] involving wing designs, flight performance, and trajectory generations [1, 20] utilize Shapefile formats to store geographic information. The visualization tool supports loading Shapefile and renders the contours on the canvas. Moreover, the Census Bureau's Census Block geography and attributes [21] are supported.

The Keyhole Markup Language (KML) file format, maintained by Open Geospatial Consortium, focused on geographic visualization on Earth applications such as ArcGIS Earth and Google Earth [22, 23, 24]. The file format is an Extensible Markup Language (XML) [25]. KML files have the `*.kml` file name extension. A main KML file and zero or more supporting files can be stored in a zip-compressed file with the `*.kmz` file name extension [26]. The visualization tool can open both types of files.

# 8. Map

The visualization tool renders tiled vector data from *ArcGIS* World Basemap [27] encoded using the *Mapbox* Vector Tile Specification version 2.1 [28]. Vector tiles also use layers to group geometric features. The visualization tool provides a map layer panel for the users to search, enable or disable map layers, and change the line and text rendering properties. Figure 8.1 shows a screenshot of the tool covering the San Francisco Bay Area in California with the following seven map layers enabled: Marine area, Bathymetry, Land, Vegetation small scale, City, Road, and Urban area.

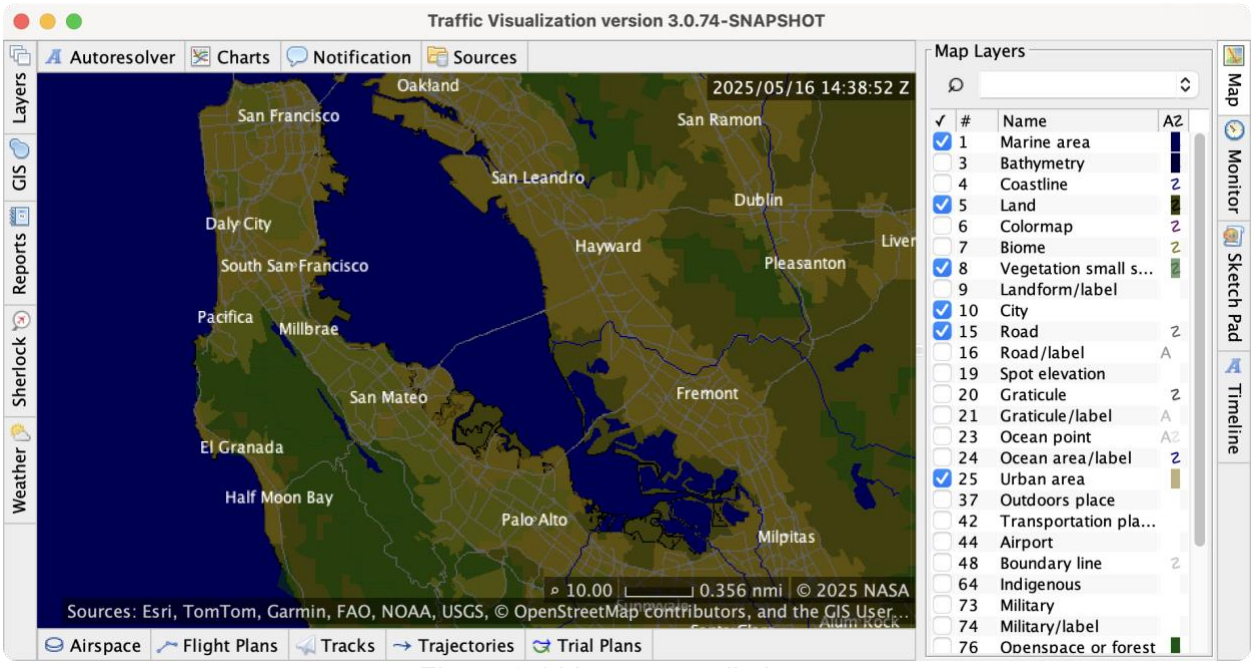



*Figure 8.1 Vector map tile layers*

The map layer is useful for geographically visualizing airspace elements, flight plans, and track paths. Since the Web Mercator projection is used in the tiled vector data, the vector tiles are displayed when the Web Mercator projector is selected in the visualization tool. Additional map sources and projections can be included in the future.

# 9. Reports

During a simulation run, data generated by individual components can be analyzed and visualized using the reporting capability in the visualization tool. This section describes the commonly supported file and record formats.

## 9.1. Local Data Collection Files

Data generated in the Local Data Collection (LDC) format of the Airspace Concept Evaluation System [29] can be imported into a *MySQL* database [30]. The visualization tool can also process the LDC files and import data into a *Hypersonic 2* (H2) database file [31]. This feature allows users to visualize LDC data files without accessing a MySQL server. Custom databases can be supported by creating concrete classes that extends the *Database* base class.

## 9.2. Comma-Separated Values Files

The report capability also allows users to visualize Comma-Separated Values (CSV) data files with custom column names and value formats. The column name patterns are case insensitive, and two wildcards are supported:

1. The symbol * matches zero or more characters. For example, `*id*` will match any string containing the token `id` such as `aircraftID` and `identification`.
2. The symbol ^ indicates negation of the matching result. For example, `^_` will match any string that does not have an underscore prefix such as `_callsign` and `_id`.

The default column name patterns and value formats are defined in the resource file `RecordFormatConfig.properties` located at the project directory `src/main/resources/gov/nasa/aftools/tv/layer/report/folder/`. The default file assumes CSV files storing track point records with the following column name patterns and values. Note that the order of the column names is not important.

1. `*alt*`: altitude, in feet.
2. `*callsign*` and `*id*`: callsign string.
3. `*head*`: heading, in degrees.
4. `*lat*`: latitude, in degrees.
5. `*lon*`: longitude, in degrees.
6. `gsKt` and `*speed*`: groundspeed, in knots
7. `*time*`: simulation time, in milliseconds elapsed since the epoch (midnight, January 1, 1970 UTC), `yyyy-MM-dd HH:mm:ss.SSS`, or `yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'`.

A custom resource file can be placed in the directory of the CSV files for custom column names and value formats. For example, when the groundspeed column name pattern is cleared in the custom resource file, the groundspeed column values, if present in the CSV file, will be excluded from the loading operation. To illustrate this capability, a sample file named `NASA123.csv` with five track points in a custom CSV format is added to a user-specific directory `/path/to/sample/`, as shown in Listing 9.1. A custom resource file named `RecordFormatConfig.properties` is also placed in this user-specific directory, and its content is shown in Listing 9.2. Note that the pattern value of the `SPEED_KT` property is cleared.

```
# acid, time, lat, lon, alt, head, speed
NASA123, 2025-05-29 10:00Z, 37.57, -122.54, 25000, 65, 100
NASA123, 2025-05-29 10:01Z, 37.57, -122.52, 24500, 66, 103
NASA123, 2025-05-29 10:02Z, 37.58, -122.50, 24000, 67, 100
NASA123, 2025-05-29 10:03Z, 37.59, -122.48, 25000, 68, 115
NASA123, 2025-05-29 10:04Z, 37.60, -122.45, 26000, 69, 110
```

*Listing 9.1 File content of NASA123.csv*

```
TrackPointRecord.SPEED_KT =
TrackPointRecord.TIME_MS.timeFormats = yyyy-MM-dd HH:mm'Z'
```

*Listing 9.2 File content of RecordFormatConfig.properties*

After opening the `NASA123.csv` file from the Reports→Folder→Search panel, the groundspeed information of the flight is not available in the Groundspeed chart, as indicated in Figure 9.1.

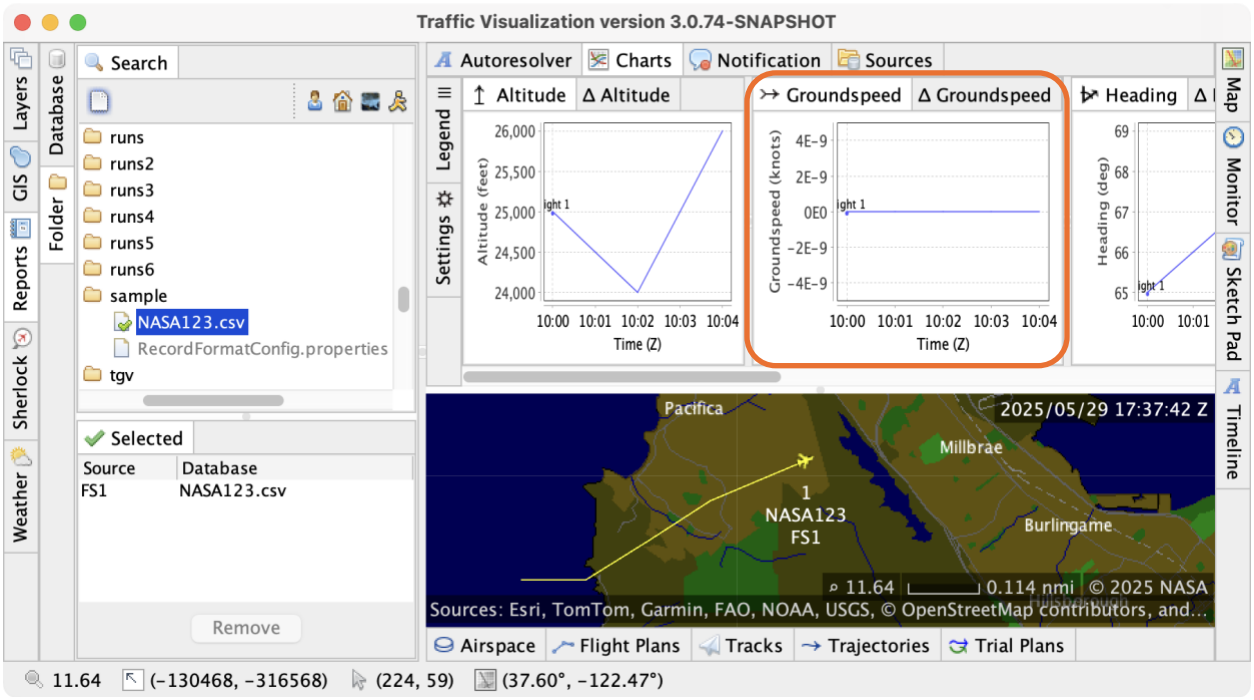



*Figure 9.1 Groundspeed data are excluded*

## 9.3. ATM TestBed Data Exchange Model Files

When running a NAS Digital Twin simulation, components can be configured to record incoming and/or outgoing messages in the ATM TestBed Data Exchange Model [32] data format. The recorded files storing messages in a JavaScript Object Notation [33] lines format with the `*.sndem` file extension. The visualization tool can import the recorded files.

## 9.4. Autoresolver Records

*Autoresolver* [4] is an automated tool for solving separation assurance and related problems including arrival sequencing and weather-cell avoidance. When running simulations with Autoresolver, depending on the enabled features, the following information may be generated

and collected in the LDC format. Since the visualization tool supports reading LDC data files and databases, this group of information can also be presented in the application.

In the visualization tool, there are five tabs in the Autoresolver for showing tables of records:

1. The Resolutions tab lists *resolution* records storing information associated with resolving a specific conflict, where a unique conflict is defined as involving the same two aircraft with the same first loss of separation time.
2. The Attempts tab lists *resolution attempt* records describing one attempt to solve a conflict and is associated with one *resolution* record. Not all resolution attempt records are associated with a call to the trial planner; some may represent a resolution that was generated but then determined to be inappropriate before it is sent. Others are notifiers, pointing out that are not valid solutions.
3. The Arrivals tab lists *arrival stream information* records storing information associated with a flight's arrival such as scheduled and estimated time of arrival, arrival airport and arrival fix. Arrival flights and their timelines are presented in the Timeline panel.
4. The Conflicts tab lists *conflict* records containing data pertaining to the conflict along with data pertaining to each flight involved in the conflict. Two flights are in conflict if their positions are predicted to be within a minimum horizontal and vertical separation distance at the same time.
5. The Recaptures tab lists *recapture* records tracking all the information associated with a recapture maneuver that rejoins an aircraft who is not following its route or assigned altitude back to flight plan conformance.

### 9.5. Trial Planning Records

Trail planning records generated by the Autoresolver are grouped in the *Trial Plans* tab, which contains the following tables:

1. The *Flights* table lists flights that have associated resolution attempt, resolution record, and/or conflict records.
2. The *Trial Plan Times* table lists trial plan times associated with selected rows in the Flights table that have a resolution attempt or resolution record.
3. The *Trial Plans* table further lists trial plans associated with selected rows in both the Flights table and Trial Plan Times table.
4. The *Timeline* panel shows timelines when one or more arrival flights were entering a region of a specific arrival airport and fix.

## 10. Sherlock

The visualization tool can load historical traffic data from the NASA's Sherlock via the Sherlock Application Programming Interface (API). Currently, the tool supports two file formats: Integrated Flight Format (IFF) and Reduced Data (RD). The IFF files contain both flight plan and track point records, while the RD files contain flight plan records.

Figure 10.1 shows the traffic pattern from the IFF track point records obtained from the Airport Surface Detection Equipment, Model X (ASDE-X) at the San Francisco International Airport on 2025/02/05 (Wednesday). Note that the Sherlock API credentials are masked on the figure. Please send email to Sherlock Support at `sherlock-support@lists.nasa.gov` to request Client Credentials.

In addition to loading traffic data using the Sherlock API, historical traffic data files downloaded from the Sherlock can also be loaded from the file panel. Supported CSV and gzip-compressed files with a checkmark indicator ( ) can be selected and opened from the Search tree panel.

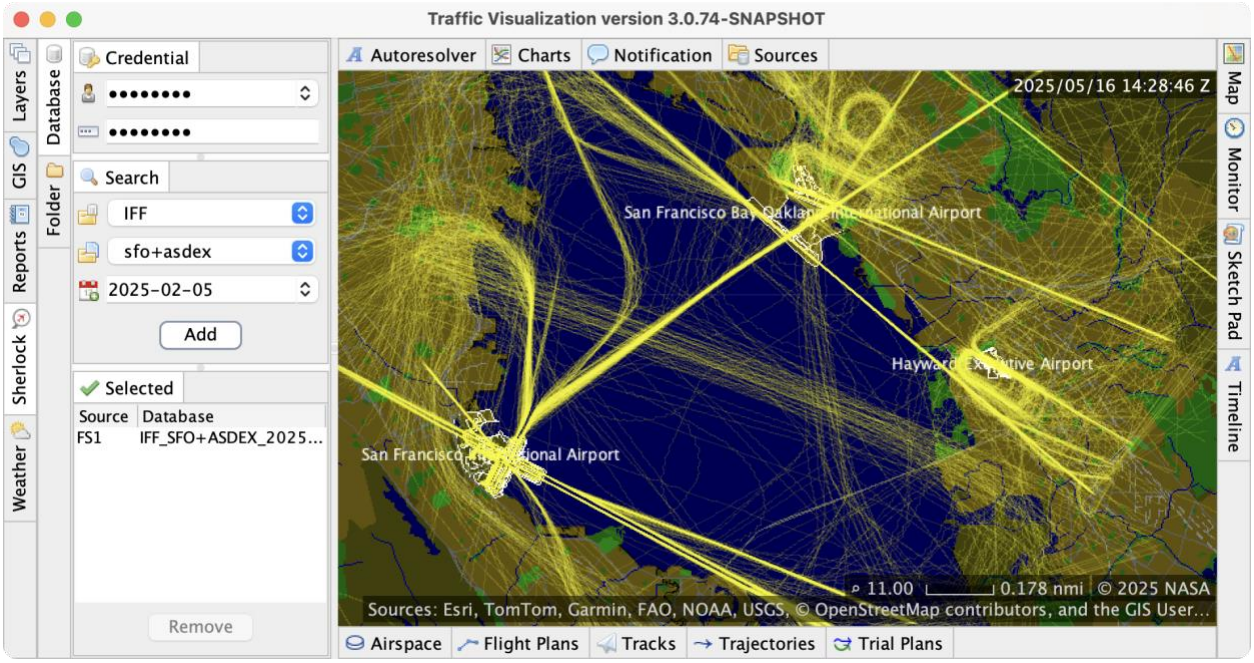



*Figure 10.1 Historical traffic pattern from Sherlock*

## 11. Weather

The visualization tool can load Convective Weather Avoidance Model (CWAM) [34] and render the polygons on canvas. Each polygon represents a forecast of convective weather and probability that pilots will avoid the enclosing airspace. Figure 11.1 shows forecasted CWAM polygons on 2022/10/29 (Saturday) at 17:15 Z at Flight Level 250. The green, yellow, and red polygons indicated that pilots would have 60% to 70%, 70 to 80%, and 80% to 100% chances, respectively, to avoid the weather regions.

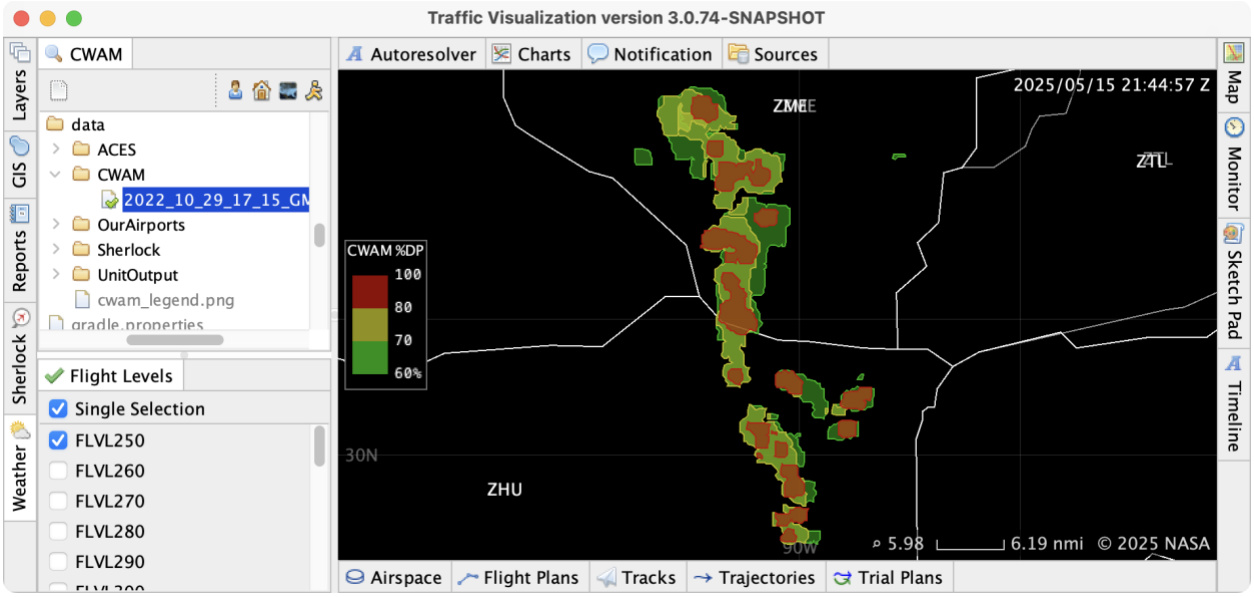



*Figure 11.1 CWAM polygons shown on the Weather layer*

CWAM files with a checkmark indicator ( ) can be selected and opened from the Search tree panel located on the top-left area. Flight levels can be selected or deselected on the Flight Levels panel located on the bottom-left area.

In the future, additional weather files such as Corridor Integrated Weather System [35] and Rapid Refresh [36] will be supported.

## 12. Concluding Remarks

This document presents an interactive, two-dimensional visualization tool called Traffic Visualization Tool for the National Airspace Digital Twin project. The visualization tool is designed to accommodate project's use cases and needs to view airspace elements, flight information, flown and predicated trajectories, simulation and weather data. The visualization tool is capable of loading records from a wide range of data sources including databases, files, and the NASA's Sherlock. Customization is supported such that existing drawable elements, layers, and capabilities can be enabled, disabled, enhanced, and extended; new drawable elements, layers, and capabilities can also be added. The visualization tool is developed for aviation research projects with an objective that developers and researchers can leverage existing code base to create their own tools in a straightforward manner, thereby reducing software development and maintenance cost as well as increasing productivity. Future works include supporting additional traffic and weather data types provided by the Sherlock as well as showing animated and live traffic and weather patterns.

## 13. References


1. Lauderdale, T.A., Windhorst, R.D., Coppenbarger, R.A., Thipphavong, D., and Erzberger, H., "Overview of the National Airspace System (NAS) Digital Twin Simulation Environment," AIAA Aviation Forum and Exposition, Las Vegas, NV, US, 29 July-2 August 2024.
2. M. M. Eshow, M. Lui and S. Ranjan, "Architecture and capabilities of a data warehouse for ATM research," 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC), Colorado Springs, CO, USA, 2014, pp. 1E3-1-1E3-14, doi: 10.1109/DASC.2014.6979418.
3. "Java | Oracle" (Online), https://www.java.com/en/. Oracle. Retrieved 2025/02/19.
4. "Maven Central Repository" (Online), https://repo1.maven.org/maven2/. Sonatype. Retrieved 2025/02/24.
5. "Unidata | NetCDF" (Online), https://www.unidata.ucar.edu/software/netcdf/. University Corporation for Atmospheric Research Community Programs. Retrieved 2025/02/24.
6. "GeoTools - OSGeo" (Online), https://www.osgeo.org/projects/geotools/. Open Source Geospatial Foundation. Retrieved 2025/02/25.
7. Erzberger H, Lauderdale TA, Chu Y-C. "Automated Conflict Resolution, Arrival Management, and Weather Avoidance for Air Traffic Management." Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering. 2012;226(8):930-949. doi:10.1177/0954410011417347.
8. Peters, M., Kimmet, T., Henthorn, R., Krotoski, S., "The Detailed Design and Software Implementation of the Aircraft Dynamics Model for the FAA Target Generation Facility," Seagull Technology, Inc., FAA Prime Contract No. DTFA03-94-C-00042, June 1998.
9. "Aeronautical Data" (Online), https://www.faa.gov/air_traffic/flight_info/aeronav/aero_data/. Federal Aviation Administration. Retrieved 2025/02/19.
10. "Aeronautical Information Portal Login" (Online), https://nfdc.faa.gov/nfdcApps/ . Federal Aviation Administration. Retrieved

11. “Aeronautical Information Manual – AIM” (Online), https://www.faa.gov/air_traffic/publications/atpubs/aim_html/. Federal Aviation Administration. Retrieved 2025/03/06.
12. “Open data @ OurAirports” (Online), https://ourairports.com/data/. OurAirports. Retrieved 2025/02/21.
13. “WGS 84 / Pseudo-Mercator - Spherical Mercator, Google Maps, OpenStreetMap, Bing, ArcGIS, ESRI - EPSG:3857” (Online), https://epsg.io/3857. Klokan Technologies GmbH, Switzerland. Retrieved 2025/02/19.
14. “Lambert Conic Conformal (2SP) - EPSG:9802” (Online), https://epsg.io/9802-method. Klokan Technologies GmbH, Switzerland. Retrieved 2025/02/19.
15. “GitHub - jfree/jfreechart: A 2D chart library for Java applications (JavaFX, Swing or server-side)” (Online), https://github.com/jfree/jfreechart. David Gilbert. Retrieved 2025/02/19.
16. “Shapefile Plugin — GeoTools 32-SNAPSHOT User Guide” (Online), https://docs.geotools.org/stable/userguide/library/data/shape.html. Open Source Geospatial Foundation. Retrieved 2025/02/19.
17. “GIS Software for Mapping and Spatial Analytics | Esri” (Online), https://www.esri.com/en-us/home. Environmental Systems Research Institute, Inc. Retrieved 2025/02/19.
18. Li, J., Sridhar, B., Xue, M., and Ng, H., “AIRNOISE: A Tool for Preliminary Noise-Abatement Terminal Approach Route Design,” 16th AIAA Aviation Technology, Integration, and Operations Conference, 13-17 June 2016, Washington, DC. doi:10.2514/6.2016-3905.
19. “UNIT SBIR - ATAC” (Online), https://www.atac.com/innovative-research/unit-sbir/. ATAC Corporation. Retrieved 2025/02/19.
20. Wells, D.P., Gatlin, G.M., June, J.C., and Marien, T.V., “NASA Transonic Truss-Braced Wing Studies,” 34th Congress of the International Council of the Aeronautical Sciences (ICAS), 9-13 September 2024, Florence, Italy.
21. Technical Documentation: 2024 TIGER/Line Shapefiles Technical Documentation / prepared by the U.S. Census Bureau, 2024, https://www2.census.gov/geo/pdfs/maps-data/data/tiger/tgrshp2024/TGRSHP2024_TechDoc.pdf. The U.S. Census Bureau. Retrieved 2025/05/16.
22. “Keyhole Markup Language (KML) Standard | OGC Publications” (Online), https://www.ogc.org/publications/standard/kml/. Open Geospatial Consortium. Retrieved 2025/02/28.
23. “3D Earth Map | Earth App for Desktop & Mobile | ArcGIS Earth App” (Online), https://www.esri.com/en-us/arcgis/products/arcgis-earth/overview. Environmental Systems Research Institute, Inc. Retrieved 2025/02/28.
24. “Google Earth” (Online), https://www.google.com/earth/about/. Google. Retrieved 2025/02/28.
25. “Extensible Markup Language (XML) 1.0 (Fifth Edition)” (Online), https://www.w3.org/TR/xml/. World Wide Web Consortium. Retrieved 2025/02/28.
26. “KMZ Files | Keyhole Markup Language | Google for Developers” (Online), https://developers.google.com/kml/documentation/kmzarchives. Google. Retrieved 2025/02/28.
27. “ArcGIS World Basemap v2” (Online), https://basemaps.arcgis.com/arcgis/rest/services/World_Basemap_v2/VectorTileServer. Environmental Systems Research Institute, Inc. Retrieved 2025/02/19.
28. “Vector tiles standards | Tilesets | Mapbox Docs | Mapbox” (Online), https://docs.mapbox.com/data/tilesets/guides/vector-tiles-standards/. Mapbox. Retrieved 2025/02/19.

29. S. George and F. Wieland, "Build 8 of the Airspace Concept Evaluation System (ACES)," *2011 Integrated Communications, Navigation, and Surveillance Conference Proceedings*, Herndon, VA, USA, 2011, pp. 1-21, doi: 10.1109/ICNSURV.2011.5935378.
30. “MySQL” (Online), https://www.mysql.com/. Oracle. Retrieved 2025/02/18.
31. “H2 Database Engine” (Online), https://www.h2database.com/. H2 Group. Retrieved 2025/02/18.
32. Lai, C.F., “Air Traffic Management TestBed Data Exchange Model,” NASA/TM-20205001252, NASA Technical Memorandum, May 1, 2020.
33. Bray, T., “RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format” (Online). https://tools.ietf.org/html/rfc8259. Internet Engineering Task Force. Retrieved 2025/05/23.
34. Shu-Chieh Wu, R. Luna and W. W. Johnson, "Flight Deck Weather Avoidance Decision Support: Implementation and Evaluation," 2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC), East Syracuse, NY, USA, 2013, pp. 5A2-1-5A2-13, doi: 10.1109/DASC.2013.6712594.
35. Klingle-Wilson, D. and Evans, J., “Description of the Corridor Integrated Weather System (CIWS) Weather Products,” Project Report ATC-317, Lincoln Laboratory, Massachusetts Institute of Technology, August 1, 2005.
36. “Rapid Refresh (RAP)” (Online), https://rapidrefresh.noaa.gov/. Global Systems Laboratory, National Oceanic & Atmospheric Administration. Retrieved 2025/02/19.