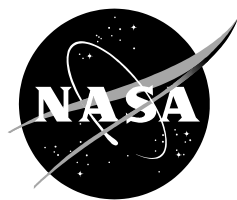


NASA/TM—2020—220511



# **Air Traffic Management TestBed Traffic Viewer: Developer's Guide**

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

---

**March 2020**

## 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:

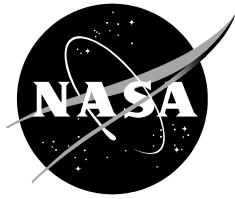
- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing 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](mailto: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—2020—220511



# **Air Traffic Management TestBed Traffic Viewer: Developer's Guide**

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

National Aeronautics and  
Space Administration

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

---

**March 2020**

## **Acknowledgments**

The author would like to thank the other Air Traffic Management TestBed team members, Alan Lee, Dr. Gano Chatterji, Phu Huynh, Huu Huynh, Jimmy Nguyen, and Yun Zheng, for their contributions to the design and development of the Traffic Viewer. The author would also like to thank Irene Smith, Alan Lee, Confesor Santiago, and Katharine Lee for reviewing this technical memorandum.

This report is available in electronic form at  
<http://ntrs.nasa.gov/>

## **Abstract**

The Air Traffic Management (ATM) TestBed is a Platform as a Service that is being developed by the National Aeronautics and Space Administration (NASA) to help design, configure, integrate, run, and monitor air traffic simulations. The platform is designed to provide cloud services including back-end, big-data analytics tools, on-demand computing resource management, data storage, and communication middleware. The ATM TestBed reduces the time to test concepts and technologies, supports interactions among various methods such as human-in-the-loop and automation-in-the-loop simulations, and enables collaborative simulations by sharing technologies and tools in the ATM community. The Traffic Viewer application provides a graphical user interface tool for visualizing real and simulated air traffic as well as airspace definition in two-dimensional space. This guide describes a high-level design and implementation of Traffic Viewer and provides information for a new developer or a user to add new capabilities by following the software design and leveraging existing capabilities.

This page intentionally left blank.

## Table of Contents

1. Introduction.....	9
2. Getting Started .....	10
2.1. Project Directory Layout .....	12
2.2. Configuration Files .....	13
2.2.1. Component Definition File .....	13
2.2.2. User-Defined Properties File .....	15
2.2.3. Simulation Run File .....	15
2.2.4. Application Configuration Files .....	15
3. User Interface .....	15
3.1. Toolbar .....	15
3.2. Viewer Canvas .....	16
3.3. Status Bar .....	16
4. Layers.....	17
4.1. Map Tile Layer .....	20
4.1.1. Web Mercator Projection .....	22
4.1.2. Map Tile Layer Properties .....	25
4.1.3. Map Tile Layer Menu .....	27
4.1.4. Map Tile Setting Menu.....	28
4.2. Boundary Layer .....	28
4.2.1. Boundary Layer Properties .....	30
4.2.2. Boundary Layer Menu.....	31
4.2.3. Boundary Setting Menu .....	31
4.3. Range Ring Layer .....	31
4.3.1. Range Ring Setting Menu.....	32
4.4. Flight Plan Layer .....	32
4.4.1. Flight Plan Layer Properties .....	33
4.4.2. Flight Plan Layer Menu .....	34
4.4.3. Flight Plan Setting Menu.....	34
4.5. Flight Path Layer .....	34
4.5.1. Flight Path Layer Menu .....	35
4.6. Trajectory Layer.....	35
4.6.1. Trajectory Layer Properties.....	36
4.6.2. Trajectory Layer Menu.....	36
4.7. Track Point Layer .....	37
4.7.1. Vehicle Icon.....	37
4.7.2. Vehicle Color .....	38
4.7.3. Vehicle Type.....	38
4.7.4. Vehicle Retention Period .....	39
4.7.5. Track Point Layer Properties .....	40
4.7.6. Track Point Layer Menu .....	40
4.7.7. Track Setting Menu .....	40
4.8. Conflict Layer.....	40
4.8.1. Conflict Layer Properties.....	41
4.8.2. Conflict Layer Menu .....	42
4.8.3. Conflict Setting Menu .....	42
4.9. Flight Tag Layer .....	42
4.9.1. Flight Tag Layer Menu.....	43
4.10. Clock Layer .....	44
4.11. Remote Control Layer .....	44
4.11.1. Remote Control Layer Menu.....	44

4.12.	Duration Layer .....	44
5.	Status Bar .....	45
5.1.	Zoom Level .....	46
5.2.	Map Settings .....	46
5.3.	Boundary Settings .....	47
5.4.	Range Ring Settings .....	47
5.5.	Flight Plan Settings .....	47
5.6.	Conflict Settings .....	48
5.7.	Location Information .....	49
5.8.	Track Settings .....	49
5.9.	Cursor Status .....	50
5.10.	User Message .....	50
5.11.	Memory Monitor .....	51
6.	Shortcut Keys .....	52
7.	References .....	53



## List of Figures

Figure 1.1. Traffic Viewer displays four live data feeds.....	9
Figure 2.1. Initial view of Traffic Viewer: (a) toolbar, (b) viewer canvas, and (c) status bar.....	11
Figure 2.2. Initial view of Traffic Viewer with UAM properties .....	12
Figure 3.1. Toolbar with three enabled layers: (a) Map Tile, (b) Boundary, and (c) Clock; (d) is the drop-down menu of the Boundary layer.....	16
Figure 3.2. Status bar.....	17
Figure 4.1. Layers in Traffic Viewer .....	17
Figure 4.2. World map projected onto a single map tile image .....	21
Figure 4.3. World map projected onto four map tile images .....	21
Figure 4.4. Map tile images of weather data: (a) Wind magnitude at 30,000 ft and (b) Total percentage of cloud cover (blue color indicates low value; red color indicates high value) ..	24
Figure 4.5. Two-level caching scheme to load a map tile image .....	24
Figure 4.6. About Map Tiles dialog .....	28
Figure 4.7. Map providers: (a) ArcGIS: World Imagery, and (b) ArcGIS: World Imagery, Boundaries and Places .....	28
Figure 4.8. Oakland ARTCC boundary with high-altitude sectors (a) disabled and (b) enabled.....	29
Figure 4.9. Range ring layer.....	32
Figure 4.10. Flight plan layer: (a) flight plans and waypoints and (b) flight plan setting.....	33
Figure 4.11. Flight path layer .....	35
Figure 4.12. Trajectory layer .....	36
Figure 4.13. Vehicle images: (a) Sample aircraft icon, (b) Supported region, and (c) Heading of 45 degrees. ....	37
Figure 4.14. Conflict layer .....	41
Figure 4.15. Flight tag layer .....	43
Figure 4.16. Clock layer .....	44
Figure 4.17. Duration layer.....	44
Figure 5.1. Status bar.....	45
Figure 5.2. Zoom (a) in and (b) out .....	46
Figure 5.3. Map settings: (a) darker and (b) brighter .....	46
Figure 5.4. (a) Boundary settings and (b) Range ring settings .....	47
Figure 5.5. Flight plan setting panel .....	48
Figure 5.6. Conflict and resolution: (a) Conflict settings panel, (b) Conflict window, and (c) Resolution window .....	48
Figure 5.7. Location information panel: (a) Translate coordinates, (b) World coordinates, (c) Pixel coordinates, (d) Tile coordinates, (e) Cursor coordinates, (f) WGS84 (decimal), and (g) WGS84 ( $d^{\circ}m's''$ ).....	49
Figure 5.8. Track icon size settings: (a) auto-size mode and (b) 32 pixels .....	50
Figure 5.9. Cursor status panel.....	50
Figure 5.10. Memory monitor: (a) status panel, (b) tooltip window, and (c) popup menu .....	52

## List of Listings

Listing 2.1. Commands to check out and build source code on (a) Windows and (b) Mac OS X platforms.....	10
Listing 2.2. Commands to start ActiveMQ on (a) Windows and (b) Mac OS X platforms .....	10
Listing 2.3. Commands to start Traffic Viewer on (a) Windows and (b) Mac OS X platforms.....	11
Listing 2.4. Commands to use properties file on (a) Windows and (b) Mac OS X platforms .....	12
Listing 2.5. Architect block definitions (snippet) .....	13

Listing 2.6. Default snfb.run properties .....	15
Listing 4.1. Abstract methods a layer subclass must implement.....	18
Listing 4.2. Example of drawing to geo-coordinates space .....	19
Listing 4.3. Example map provider properties.....	27
Listing 4.4. Example snippet of boundary definition.....	29
Listing 4.5. Example route of flight in flight plan.....	32
Listing 5.1. Abstract methods a status bar button subclass must implement.....	45
Listing 5.2. Default Track.sample file content .....	51

## List of Tables

Table 2.1. Required software .....	10
Table 2.2. Project directory layout.....	13
Table 4.1. Affine matrix: (a) Geo-coordinates and (b) Screen-coordinates.....	18
Table 4.2. Layer properties .....	19
Table 4.3. Registry retention properties .....	20
Table 4.4. Four types of coordinates in Map Mercator Projection.....	22
Table 4.5. Projection functions implemented in Traffic Viewer .....	23
Table 4.6. Inverse projection functions implemented in Traffic Viewer .....	23
Table 4.7. Map tile properties.....	25
Table 4.8. Map tile layer properties.....	25
Table 4.9. Map provider properties .....	26
Table 4.10. Map tile layer menu.....	27
Table 4.11. Supported boundary definition file formats.....	29
Table 4.12. Map tile layer properties.....	30
Table 4.13. Boundary layer properties .....	30
Table 4.14. Boundary layer menu .....	31
Table 4.15. Range ring layer properties.....	32
Table 4.16. Flight plan layer properties.....	33
Table 4.17. Flight plan layer menu.....	34
Table 4.18. Flight path layer menu.....	35
Table 4.19. Trajectory layer properties .....	36
Table 4.20. Trajectory layer menu .....	36
Table 4.21. Supported vehicle types and icons .....	37
Table 4.22. Zoom levels and vehicle dimensions in px.....	38
Table 4.23. Vehicle color properties.....	38
Table 4.24. Vehicle type properties.....	39
Table 4.25. Vehicle type properties.....	39
Table 4.26. Track point layer properties.....	40
Table 4.27. Track point layer menu.....	40
Table 4.28. Conflict layer properties .....	41
Table 4.29. Conflict layer menu .....	42
Table 4.30. Flight tag layer menu.....	43
Table 4.31. Remote control layer menu .....	44
Table 6.1. Shortcut keys in Traffic Viewer.....	52

# 1. Introduction

Running air traffic simulations usually requires a way to visualize and interact with traffic data and airspace. Though existing simulation tools [1, 2, 3] may be used, they are heavily integrated with software modules that have been built to meet specific research and/or simulation needs. In addition, in order to easily add new capabilities to visualize geospatial, air traffic, weather and environment data, there is a need to have a general, simple yet powerful, responsive and extensible visualization tool.

To address these challenges, the Air Traffic Management (ATM) TestBed under the ATM-eXploration (ATM-X) Project [4] provides a two-dimensional, layered traffic and airspace visualization tool called Traffic Viewer that allows users to customize features and extend capabilities to meet their project needs. Based on the ATM TestBed architecture, Traffic Viewer can be used to visualize live, historical and simulated traffic and airspace data. For example, Figure 1.1 shows a screenshot of a Traffic Viewer (version 2.0a) subscribing to four live data feeds from the Federal Aviation Administration’s (FAA’s) System Wide Information Management (SWIM) via the NASA’s Sherlock ATM Data Warehouse [5]. Aircraft icons are color-coded to indicate their originating data feeds.

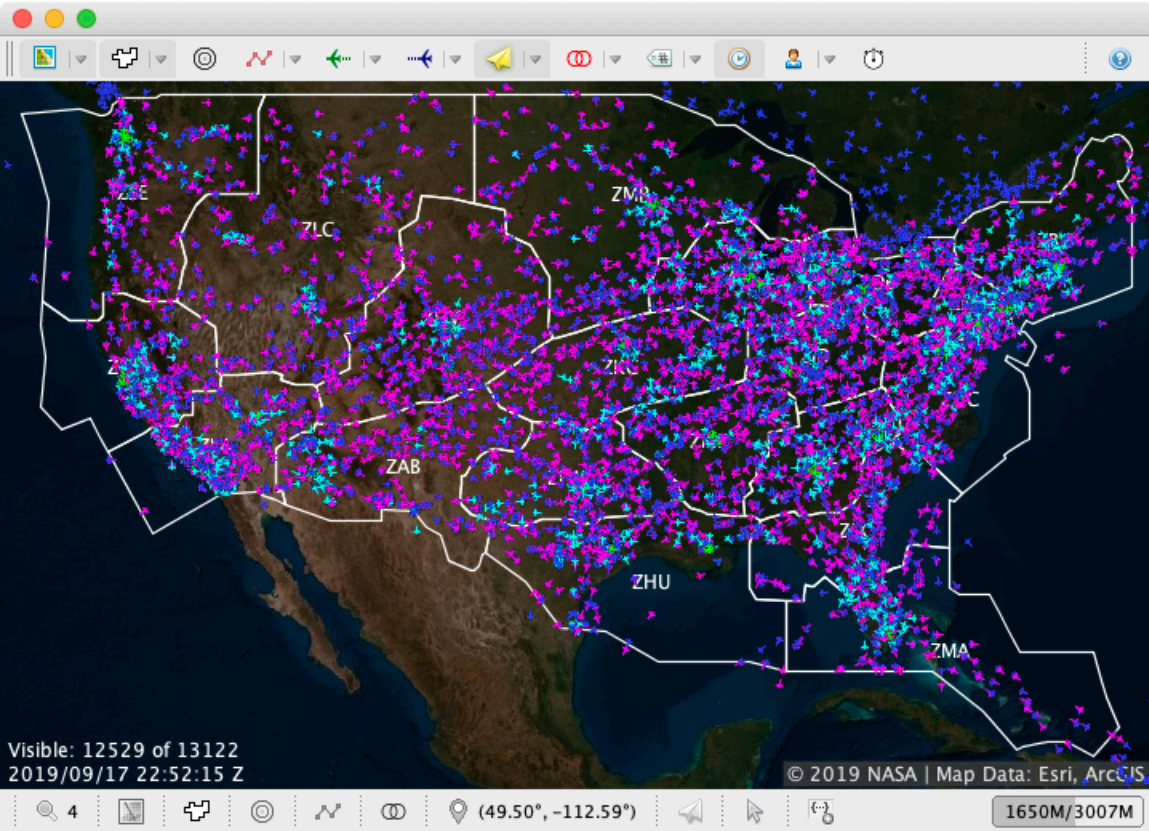


Figure 1.1. Traffic Viewer displays four live data feeds

This Developer’s Guide provides instructions for a user to access the Traffic Viewer source code, build the project, configure and run the application. The following sections describe the user interfaces, developer manual and widget details.

## 2. Getting Started

The Traffic Viewer project is stored in a Git repository<sup>1</sup> located at NASA Ames Research Center. The project can be built using Gradle build tool and Oracle Java Standard Edition (SE) Development Kit 8. The tool requires Apache ActiveMQ and Oracle Java 8 to run; the required software is listed in Table 2.1.

Table 2.1. Required software

Software	Description	Version Used	Reference
Apache ActiveMQ	Messaging middleware.	5.15.8	[6]
Git	Version control system.	2.20.1	[7]
Gradle	Project build tool.	4.8	[8]
Oracle JDK 8	Java development kit.	1.8.0_201	[9]

Since the visualization tool is developed in the Java programming language, it can be built on major operating systems including Linux, Mac OS X, and Windows. Listing 2.1 lists the commands to check out a copy of source code of version 2.0a to a fictitious directory marked in blue<sup>2</sup>.

Listing 2.1. Commands to check out and build source code on (a) Windows and (b) Mac OS X platforms

(a)	<pre>&gt; cd c:\path\to\project\ &gt; git clone ssh://git@atmjira.arc.nasa.gov:7999/tv/testbed-traffic-viewer.git &gt; cd testbed-traffic-viewer\ &gt; git checkout version-2.0a</pre>
(b)	<pre>\$ cd /path/to/project/ \$ git clone ssh://git@atmjira.arc.nasa.gov:7999/tv/testbed-traffic-viewer.git \$ cd testbed-traffic-viewer/ \$ git checkout version-2.0a</pre>

ATM TestBed architecture provides a messaging abstraction that supports publish and subscribe communication middleware for message exchanges. This Developer's Guide uses Apache ActiveMQ, an open source messaging server. Listing 2.2 lists the commands to start an Apache ActiveMQ server instance. To stop a running server instance, press <Ctrl>-<C>. Note that it is not necessary to stop and restart the ActiveMQ server during the development.

Listing 2.2. Commands to start ActiveMQ on (a) Windows and (b) Mac OS X platforms

(a)	<pre>&gt; cd c:\path\to\activemq\bin\ &gt; activemq.bat start</pre>
(b)	<pre>\$ cd /path/to/activemq/bin/ \$ ./activemq console</pre>

Listing 2.3 lists the commands to create an uber jar file containing all dependencies required to start a Traffic Viewer instance. To stop a running instance, press <Ctrl>-<C>. Note that

<sup>1</sup> Please contact the ATM TestBed Development Team, email: [chok.f.lai@nasa.gov](mailto:chok.f.lai@nasa.gov)

<sup>2</sup> All the fictitious directories are marked in blue in the instruction listings throughout this document.

restarting Traffic Viewer is needed if settings in a configuration file have been changed, or the source code has been modified and rebuilt.

Listing 2.3. Commands to start Traffic Viewer on (a) Windows and (b) Mac OS X platforms

(a)	<pre>&gt; set SNTB_HOME=c:\path\to\project\testbed-traffic-viewer\ &gt; cd %SNTB_HOME%\TestBedPlugins\TrafficViewer\ &gt; gradle shadowJar &gt; .\bin\run_traffic_viewer.bat</pre>
(b)	<pre>\$ set SNTB_HOME=/path/to/project/testbed-traffic-viewer/ \$ cd \$SNTB_HOME/TestBedPlugins/TrafficViewer/ \$ gradle shadowJar \$ ./bin/run_traffic_viewer.sh</pre>

After building the source code and launching the application in the TrafficViewer directory, a user will see an application frame. Figure 2.1 shows an initial view of Traffic Viewer on the Mac OS X platform. The application consists of three important widgets:

- (a) A toolbar provides menu buttons for accessing layer features.
- (b) A viewer canvas displays traffic and airspace information.
- (c) A status bar allows a user to control layer settings.

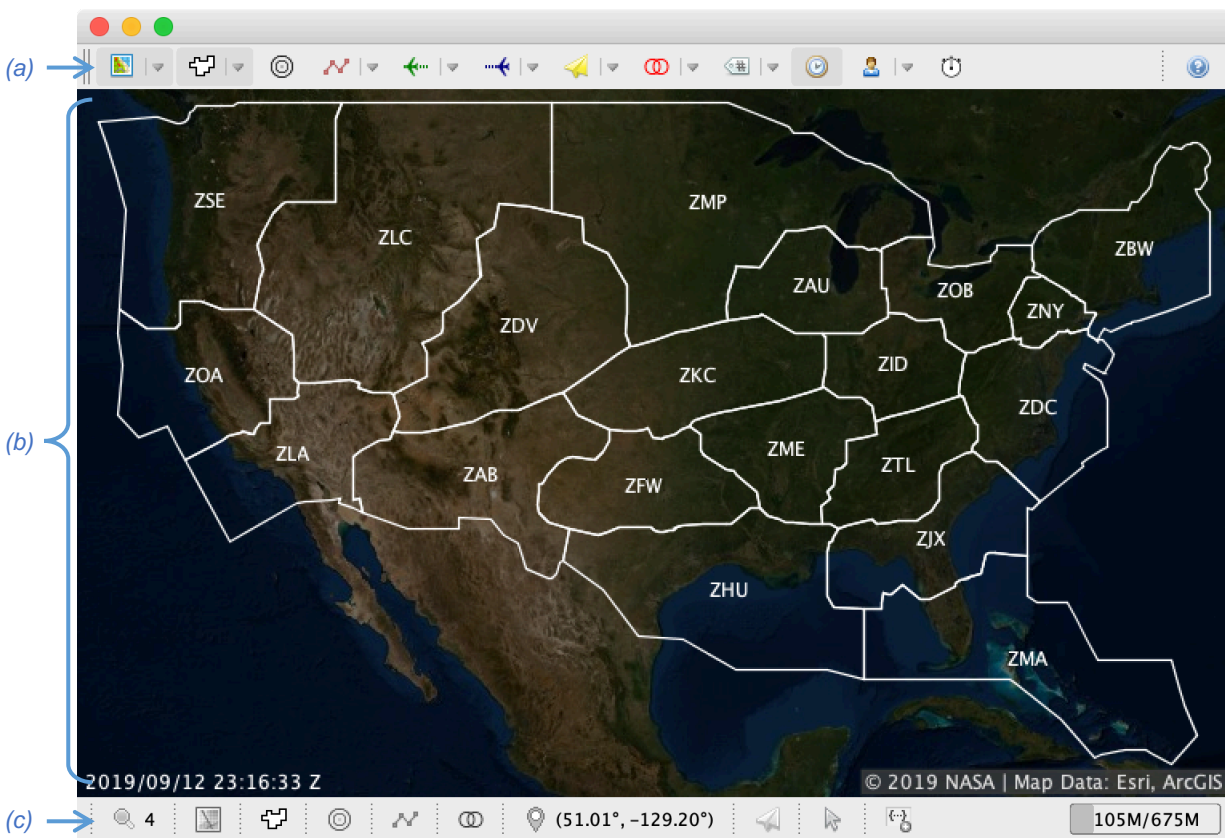


Figure 2.1. Initial view of Traffic Viewer: (a) toolbar, (b) viewer canvas, and (c) status bar

The Traffic Viewer project stores default application settings in a configuration file located at `config/TrafficViewer.properties`. Default settings are always loaded during start up and their values may be overridden in a separate, project-specific configuration file. The run script accepts an optional parameter `-config` and the parameter expects a path of a configuration file to be used. Configuration files typically use the file extension `.properties` and are stored in the

config directory. Listing 2.4 lists an example to run the visualization tool using an Urban Air Mobility (UAM) project configuration file:

Listing 2.4. Commands to use properties file on (a) Windows and (b) Mac OS X platforms

(a)	> .\bin\run_traffic_viewer.bat -config config\UAM.properties
(b)	\$ ./bin/run_traffic_viewer.sh -config config/UAM.properties

After specifying the UAM configuration file, upon startup, the initial view is the terminal area airspace surrounding Dallas/Fort Worth International Airport in Texas, and proposed routes are shown in white (see Figure 2.2).

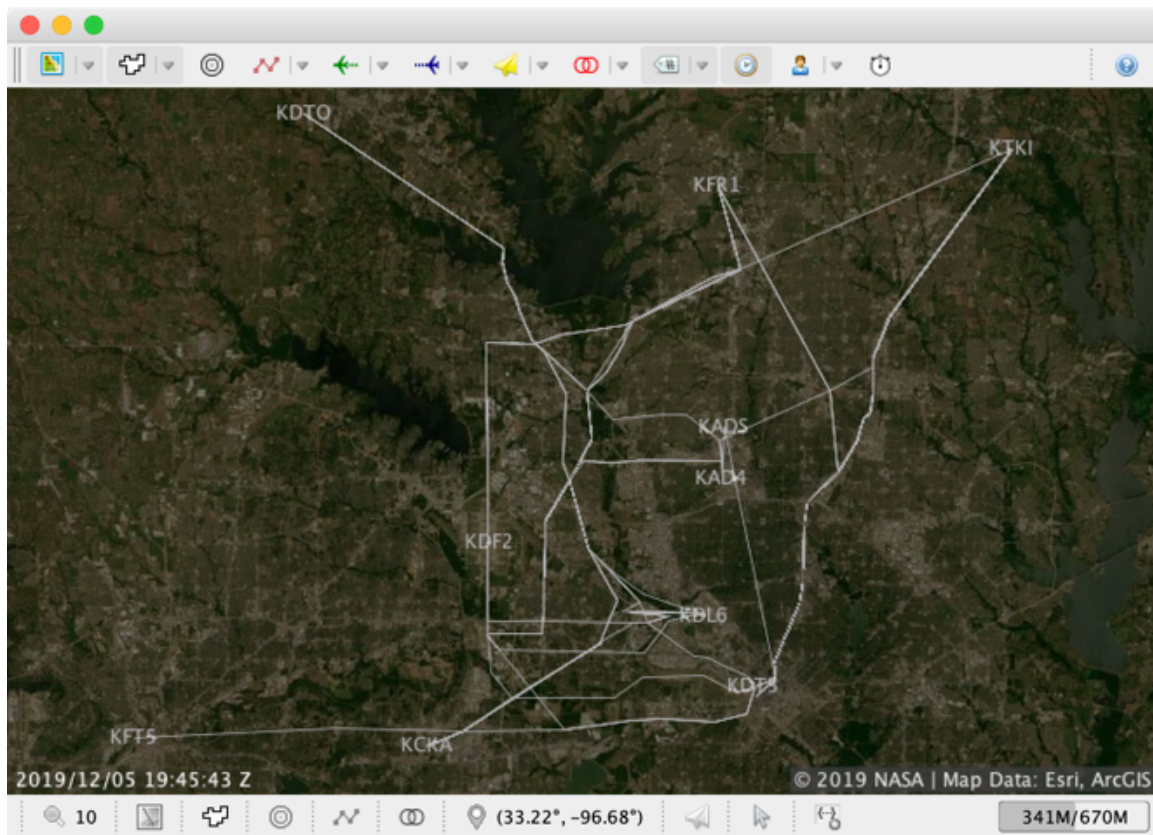


Figure 2.2. Initial view of Traffic Viewer with UAM properties

## 2.1. Project Directory Layout

The ATM TestBed and Traffic Viewer projects use the standard Maven project directory layout [10]. The description of each major directory in the project is listed in Table 2.2. Whenever building or running the visualization tool, additional directories will be created in the project directory. These directories can be kept or deleted without affecting the functionality of the application:

1. The `build` directory stores the build-related folders and files generated by the build tool.
2. The `logs` directory stores the log files generated by application loggers.
3. The `sntb.record` directory stores the metadata generated by TestBed framework.

Table 2.2. Project directory layout

Name of Directory	Description
\$SNTB_HOME/	Home directory of the TestBed project.
config/	Directory of TestBed configurations.
data/	Directory of system-wide airspace adaptation files related to the TestBed core components.
lib/	Directory of prebuilt TestBed libraries.
project/	Directory of Gradle build tool files.
TestBedPlugins/	Home directory of the TestBed plugins.
TrafficPub/	Home directory of the Traffic Publisher project.
TrafficViewer/	Home directory of the Traffic Viewer project.
bin/	Directory of executable files.
config/	Directory of configuration files.
src/	Directory of source code.

## 2.2. Configuration Files

Configuration files store application settings that are bound at launch time and the settings can be modified without rebuilding the project. User-configurable settings can also be declared in the Component Definition File. During development, a user may override the user-configurable settings in the User-Defined Properties File and modify simulation run settings in the Simulation Run File. When adding new features to the visualization tool, a user may introduce project-specific configuration files. By coding convention, all configuration files are stored in the `$SNTB_HOME/TestBedPlugins/TrafficViewer/config` directory. The following subsections describe each configuration file in detail.

### 2.2.1. Component Definition File

The visualization tool is developed based on the ATM TestBed adapter design. The locations of component definition file and adapter icon are `config/sntb.component` and `config/TrafficViewer.png`, respectively. These files are used by Simulation Architect [11] in order to render a block in a palette pane or an editor content. A snippet of the component definition file is listed in Listing 2.5.

Listing 2.5. Architect block definitions (snippet)

```

1 exports.components = [{
2   "architectBlockDefinitions": {
3     "ucid": "SNTB Simulation/Traffic Viewer/1.0",
4     "icons": {
5       "icon": "TrafficViewer.png",
6     },
7     "texts": {
8       "title": "Traffic Viewer",
9       "author": "SNTB Simulation Team",
10    },
11    "interfaces": {
12      "category": "simulation",
13      "group": "Visualization",
14      "component": "gov.nasa.sntb.trafficviewer.ui.TrafficViewer",
15      "fast-time-support": true,

```

```

16     "inputs": [
17         {
18             "type": "datasets.TrackDataset",
19             "required": false
20         },
21         {
22             "type": "datasets.FlightPlanDataset",
23             "required": false
24         }
25     ],
26     "outputs": [
27         {
28             "type": "None",
29             "required": false
30         }
31     ]
32 }
33 }
34 }]]

```

A user may modify any of the following three sections in the component definition file for specific project needs.

1. Unique Component Identifier (line 3)—The `ucid` field value is a JavaScript Object Notation (JSON) [12] string consisting of three segments separating by a forward-slash (/) delimiter:
  - a. Project name, e.g., SNTB Simulation
  - b. Component name, e.g., Traffic Viewer
  - c. Component version, e.g., 1.0

Any or all of these segment values may be modified to support project branding and versioning control, e.g., My Project/My Traffic Viewer/2.0a. Note that the field value must also be updated in the simulation run file located at `config/sntb.run`.

2. Input interfaces (lines 16 to 25)—The `inputs` field value is a JSON array of input interfaces supported by Traffic Viewer. For example, the input interface `datasets.TrackDataset` is supported and whenever a Track message is received from the messaging middleware, the method `TrafficViewer.processTrackDataset(String, Track)`<sup>3</sup> will be called. A user may add new input interfaces and their corresponding process dataset methods must be added.
3. Output interfaces (lines 26 to 31)—The `outputs` field value is also a JSON array of output interfaces supported by Traffic Viewer. By default, no output interface is specified. A user may replace the value “None” with a specific dataset type, e.g., `datasets.TrackDataset` and the method `PluginAdapter.publish(Track)` defined in the superclass is expected to be called inside the Traffic Viewer class. Additional output interfaces may be added.

In addition, the component definition file stores user-configurable settings and their default values in the `properties` field (omitted in Listing 2.5).

---

<sup>3</sup> The fully qualified name of the class Traffic Viewer is `gov.nasa.sntb.trafficviewer.ui.TrafficViewer`.



### 2.2.2. User-Defined Properties File

During development, a user may override any default values in the User-Defined Properties File located at `config/sntb.properties` in JSON format.

### 2.2.3. Simulation Run File

ATM TestBed supports concurrent simulations whereby their simulation settings are defined in a JSON file located at `config/sntb.run` (see Listing 2.6). Each simulation is uniquely identified by a 24-hexadecimal simulation run identifier. The identifier value is specified in the `runId` field and its default value is a string with 24 zeros (“000000000000000000000000”). Whenever the file is loaded, the default value will be replaced based on the simulation launch mode:

1. For simulations launched from a web browser accessing the TestBed Web Portal, the value will be replaced with a unique string generated by a back-end database, e.g., “5de9596a4a57cbd97d445d01.”
2. For simulations launched from a local development environment, the value will be automatically resolved into a string using the local Internet Protocol (IP) address prefixed with 12 zeros, e.g., “000000000000192168001002” for the IP address 192.168.001.002.

Keep in mind that if a simulation requires running multiple Traffic Viewer instances, the instances need to have unique, positive component sequence values. The sequence value is specified in the `componentSeq` field and its default value is one (1).

Listing 2.6. Default `sntb.run` properties

```
{
  "componentId": 0,
  "componentSeq": 1,
  "componentUcid": "SNTB Simulation/Traffic Viewer/1.0",
  "runId": "000000000000000000000000"
}
```

### 2.2.4. Application Configuration Files

The default application settings are stored in the configuration file located at `config/TrafficViewer.properties`. A project may override any default settings in a project-specific configuration file. The settings are fully discussed in Section 4.

## 3. User Interface

Traffic Viewer is a graphical user interface application. A user interacts with the application via toolbar, viewer canvas and status bar widgets using mouse and keyboard. This section briefly describes each widget.

### 3.1. Toolbar

The toolbar located at the top of the application frame provides menu buttons for accessing features supported by layers (see Section 4). Initially, each layer has a menu toggle button to allow a user to enable (menu button is selected) or disable (menu button is deselected) the layer. Whenever a layer menu has more than one menu item, the menu toggle button becomes a split drop-down button. A split drop-down button has a down-pointing arrow on the right-hand side. Clicking on the split drop-down button will pop up a menu listing all available menu items. To illustrate differences between a menu toggle button and a split drop-down button, Figure 3.1 shows a toolbar with three selected layer menu buttons: (a) Map Tile, (b) Boundary, and (c) Clock. Notice that both (a) and (b) are split drop-down buttons while (c) is a toggle button. In addition, the drop-down menu (d) of the Boundary layer is shown.

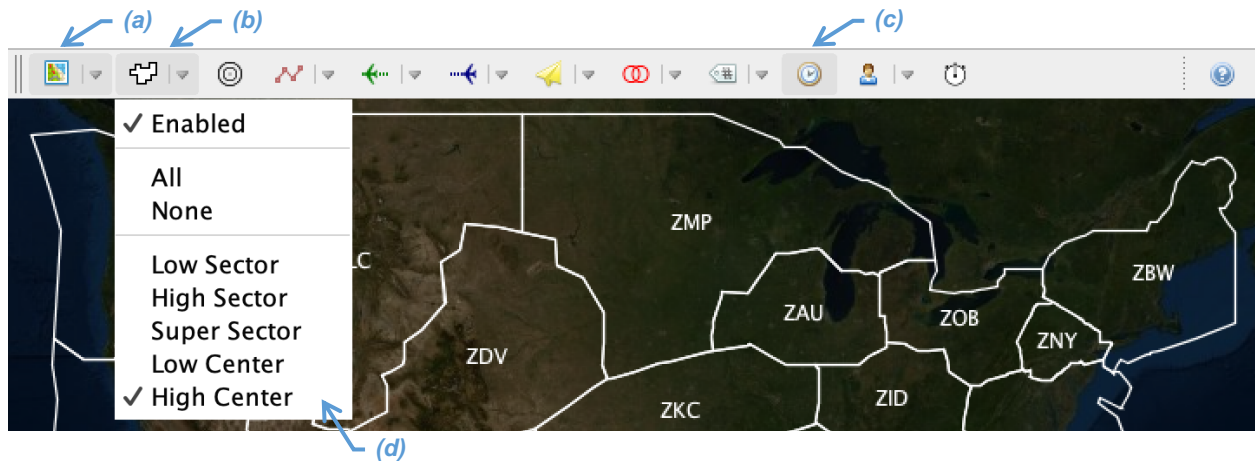


Figure 3.1. Toolbar with three enabled layers: (a) Map Tile, (b) Boundary, and (c) Clock; (d) is the drop-down menu of the Boundary layer.

The menu buttons can also be accessed by pressing a number key 1 (to select the first menu button) through key 0 (to select the last menu button) on the keyboard. Pressing the space key will toggle a menu toggle button or show a menu. Menu items and descriptions of each toolbar button will be described in Section 4.

### 3.2. Viewer Canvas

The viewer canvas component located at the center of the Traffic Viewer provides a rectangular region for layers to draw elements such as map tile images, airspace boundaries and vehicle icons. The top-left corner of the viewer canvas represents the anchor of the world map; thus, resizing the application frame to a higher or a lower dimension will not affect the anchor reference. The viewer canvas supports the following mouse interactions:

1. Panning the anchor by dragging (click on the left mouse button and then drag).
2. Zooming the view by scrolling the mouse wheel (rotate up/down to zoom in/out).
3. Tracking the world location by moving the mouse cursor.
4. Locking or unlocking a visual element that is closest to the cursor (double click on the left mouse button when the cursor mode is selected).

Whenever a user interacts with the viewer canvas using the mouse, the current geo-location at the cursor tip is indicated on the Location Information status bar (see Section 5.7), element keys (such as vehicle IDs) of the currently locked visual elements are displayed on the Cursor status panel (see Section 5.9).

The viewer canvas also has a registry for storing information extracted from messages received from the TestBed messaging middleware. The registry may be accessed by individual layers.

### 3.3. Status Bar

The status bar located at the bottom of the application frame provides status bar buttons and information for individual layers. Clicking on a status bar button will show a setting dialog. Figure 3.2 shows the status bar with the Map settings dialog after the map status bar button is clicked. Menu items and descriptions of each status bar button will be described in Section 4.

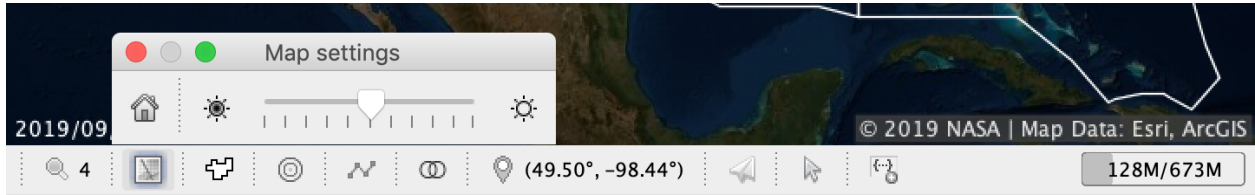


Figure 3.2. Status bar

## 4. Layers

In computer graphics, graphical elements of a complex drawing are rendered in separate layers in order to provide modularity and improve the performance of the application. Many graphical applications and visualization tools use layers to separate rendering elements [13, 14, 15]. Layers are used in Traffic Viewer to support the following features:

1. Same elements can be rendered on the same layer.
2. Individual layers can be enabled, disabled or reordered.
3. New layers can be added, and existing layers can be removed.
4. Separate elements can be handled in separate layers.

Figure 4.1 illustrates the concept of three enabled layers in Traffic Viewer: The bottommost map layer renders the map tile images, the middle boundary layer renders the airspace boundaries, and the uppermost track point layer renders the flown flight paths. Disabling the map layer will hide the map tile images while the airspace boundaries and the flown flight paths are continuing to show.



Figure 4.1. Layers in Traffic Viewer

Each layer must be a subclass of the abstract class `Layer`<sup>4</sup> and the subclass requires implementing the four abstract methods listed in Listing 4.1. A layer instance is initialized with a `config` parameter storing settings defined in the application configuration file. A toggle button or a split drop-down menu button to be shown on the toolbar is constructed by a layer menu. Whenever a change that might affect the draw operations is made by a user, the layer will be

<sup>4</sup> The fully qualified name of the class `Layer` is `gov.nasa.sntb.trafficviewer.ui.layer.Layer`.

notified by a `change type` parameter. Finally, the `paint` method is called to draw elements on the layer.

Listing 4.1. Abstract methods a layer subclass must implement.

```
public abstract class Layer {
    /**
     * Initializes this layer.
     *
     * @param config user configuration to be used.
     * @throws IllegalArgumentException if a property value has invalid format.
     * @throws IOException if initialization encounters an exception.
     */
    public abstract void initialize(Config config)
        throws IOException;

    /**
     * Creates layer menu for toolbar toggle button to enable or disable layer
     * related features.
     *
     * @return layer menu instance.
     */
    public abstract LayerMenu createLayerMenu();

    /**
     * Called when a change is triggered that would affect the layer drawing.
     * Subclasses may need to handle specific change types, so that shapes and
     * elements can be reconstructed in the next paint cycle.
     *
     * @param type enum value of change type to be handled.
     */
    public abstract void onChanged(ChangeType type);

    /**
     * Paints the elements in given graphics instance.
     *
     * @param g2 graphics to be updated.
     */
    public abstract void paint(Graphics2D g2);
}
```

Both geo-coordinates and screen-coordinates affine transformations are supported when drawing graphical elements in a two-dimensional space. The former one is used for rendering elements based on geo-coordinates (*latitude, longitude*) in degrees decimal form; the latter one is used for rendering elements based on screen-coordinates ( $x, y$ ) in pixels. Table 4.1 lists the affine matrices, where  $p_x$  and  $p_y$  are the pixel coordinates (see Section 4.1.1). The screen-coordinates affine transformation is used by default, and its affine matrix is an identity matrix.

Table 4.1. Affine matrix: (a) Geo-coordinates and (b) Screen-coordinates

$$\begin{bmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(b)

Listing 4.2 lists an example code snippet illustrating how to draw the International Civil Aviation Organization (ICAO) airport code “KSFO” at San Francisco International Airport using the geo-coordinates affine transformation.

Listing 4.2. Example of drawing to geo-coordinates space

```
public class AirportCodeLayer extends Layer {
    @Override
    public void paint(Graphics2D g2) {
        // use geo-coordinates affine transformation
        g2.setTransform(getGeoTransform());

        // find screen location of KSFO, and draw airport code
        Projection projection = getLayerParams().getProjection();
        double latitudeDeg = 37.615223;
        double longitudeDeg = -122.389977;
        GeoPoint geoPoint = new GeoPoint(latitudeDeg, longitudeDeg);
        Point screen = projection.toScreen(geoPoint);
        g2.drawString("KSFO", screen.x, screen.y);

        // restore to screen-coordinates affine transformation
        g2.setTransform(getScreenTransform());
    }
}
```

The fully qualified class names of the layers to be shown to the users need to be defined in the property `layer_class_list`. A layer is disabled by default unless its fully qualified class name is also defined in the property `layer_class_list_enabled`. Table 4.2 lists the properties that may be used to control the availability of the layers. Note that long property names are split into two lines, and the values are defined in project-specific configuration files.

Table 4.2. Layer properties

Name	Type	Description
<code>layer_class_list</code>	CSV <sup>5</sup>	List of fully qualified names of the layer classes to be used for draw operation. The list defines the draw sequence of the layers. The first value represents the fully qualified class name of the bottommost layer; the last value represents the fully qualified class name of the uppermost layer.
<code>layer_class_list_enabled</code>	CSV	List of fully qualified names of the layer classes to be initially enabled during startup.
<code>&lt;Layer.Class.Name&gt;. interfaceTypes</code>	CSV	List of interface types being supported for given layer class. A disabled layer will become enabled whenever the first such message is

<sup>5</sup> Comma-Separated Values (CSV) may be spanned into multiple lines by putting a backslash (\) character at the end of each line. Lines starting with the hash (#) symbol are considering comments.

		received. This property is useful for enabling a layer automatically whenever its first supported message is received.
viewer_canvas_repaint_period	long	Repaint period, in ms <sup>6</sup> , of the viewer canvas. Default: 1000


In order to avoid unnecessary computation whenever a repaint operation is performed, generated graphical elements and incoming messages can be stored in in-memory, time-based caches and registries. The properties specific to retention periods have the name suffix `retention_period`, e.g., `cache_retention_period`, and they are listed in Table 4.3.

Table 4.3. Registry retention properties

Name	Type	Description
cache_retention_period	long	Retention period, in ms, for caches.
conflict_registry_retention_period	long	Retention period, in ms, for Conflict registry.
flight_plan_registry_retention_period	long	Retention period, in ms, for Flight Plan registry.
resolution_registry_retention_period	long	Retention period, in ms, for Conflict Resolution registry.
track_registry_retention_period	long	Retention period, in ms, for Track registry.
trajectories_registry_retention_period	long	Retention period, in ms, for Trajectories registry.

Layers provided by Traffic Viewer are described in the following subsections.

#### 4.1. Map Tile Layer

The  Map Tile Layer displays a world map by drawing map tile images provided by map providers. This layer is the first, i.e., bottommost, layer on the viewer canvas in the default settings. The map tile images are square images with dimension 256 pixels in width and 256 pixels in height, i.e.,  $256\text{ px} \times 256\text{ px}$ <sup>7</sup>. Images are generated from sources using Web Mercator Projection [16] based on zoom levels. A zoom level is a non-negative integer determining the number of map tile images needed to cover the world map. At zoom level zero, only one map tile image is needed (see Figure 4.2), i.e., the entire world is rendered in a single map tile image.

---

<sup>6</sup> Millisecond (ms) is the unit of time used in Java programming language. One millisecond represents one-thousandth second, i.e.,  $1\text{ ms} = 1/1,000\text{ s}$ .

<sup>7</sup> Throughout this developer's guide, image dimensions are presented using pixels in width and pixels in height format, i.e.,  $\text{width px} \times \text{height px}$ .

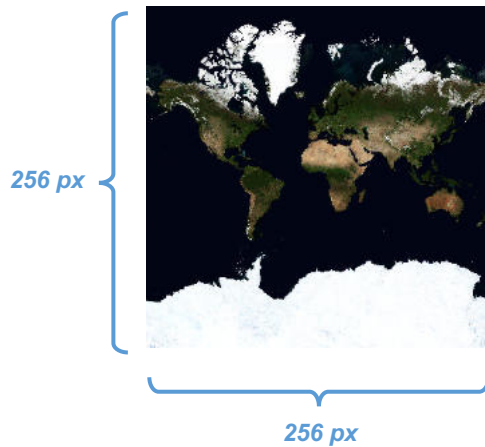


Figure 4.2. World map projected onto a single map tile image<sup>8</sup>

At each subsequent higher zoom level, both the width and the height dimensions of the world map are doubled. For example, at zoom level one, the world is composed of four (two by two) map tile images and each map tile image still has the dimension  $256\text{ px} \times 256\text{ px}$  (see Figure 4.3).

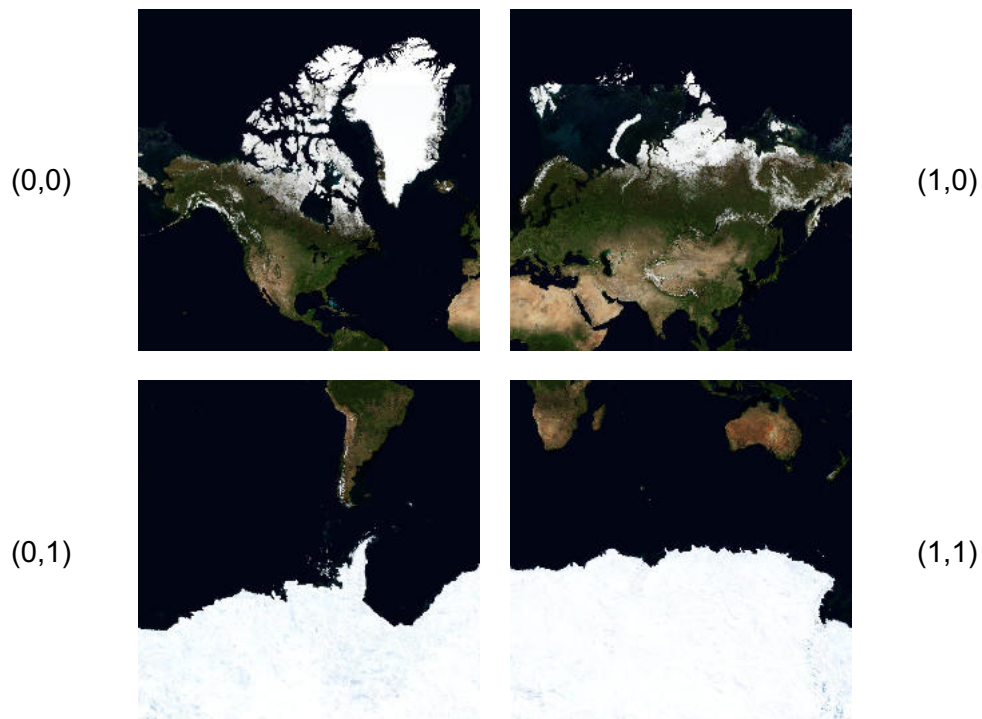


Figure 4.3. World map projected onto four map tile images<sup>9</sup>

---

<sup>8</sup> Source:

- <https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/0/0/0>

<sup>9</sup> Sources:

The higher the zoom level, the higher the number of map tile images the world map is composed of. In general, at zoom level  $z$ , the world map is covered by  $2^z \times 2^z = 2^{2z}$  number of map tile images. Since each map tile image has the dimension  $256 \text{ px} \times 256 \text{ px}$ , the world is covered by  $2^{2z} \times 256^2$  pixels. At each zoom level, map tile images can be referenced by tile coordinates  $(t_x, t_y)$ , where  $(0, 0)$  represents the tile at the top-left corner, and  $(1, 0)$  represents the second tile in the first row. Thus, a world map at zoom level  $z$  has tile coordinates ranging from  $(0, 0)$ , the top-left tile, to  $(2^z - 1, 2^z - 1)$ , the bottom-right tile. Without loss of generality, the map tile image coordinates can be represented by a tuple of three integers  $(z, t_x, t_y)$ , where  $z \geq 0$ , and  $t_x, t_y \in [0, 2^z - 1]$ .

#### 4.1.1. Web Mercator Projection

The Web Mercator Projection is a special form of Mercator Projection which projects the world onto a square image using four types of coordinates [17], as described in Table 4.4. Latitude values are between  $-90^\circ$  and  $+90^\circ$ , inclusive. Positive values are above the equator (N), and negative values are below the equator (S). Longitude values are between  $-180^\circ$  and  $+180^\circ$ , inclusive. Positive values are east of the prime meridian (E), and negative values are west of the prime meridian (W).

Table 4.4. Four types of coordinates in Map Mercator Projection

	Type	Coordinates	Units	Description
1.	Map	$(m_x, m_y)$	Degrees in World Geodetic System 1984 (WGS84) reference coordinate system	A geo-location point on the map where $m_x = \text{longitude deg}$ and $m_y = \text{latitude deg}$ .
2.	World	$(w_x, w_y)$	Pixels with double precision	Mercator projection of map coordinates onto a world map zoomed at zero level, i.e., a map with dimension $256 \text{ px} \times 256 \text{ px}$ .
3.	Pixel	$(p_x, p_y)$	Pixels with integer precision	Linear projection of world coordinates onto the world space. The scaling factor is $2^z$ , where $z$ is the zoom level.
4.	Tile	$(t_x, t_y)$	Pixels with integer precision	Linear projection of pixel coordinates onto the tile space.

Table 4.5 lists the projection functions used in Traffic Viewer to project a location from one coordinate system to another. One may apply these functions in series to find the screen coordinates  $(p_x, p_y)$  of specific map coordinates  $(m_x, m_y)$ .

- 
- <https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/1/0/0>
  - <https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/1/1/0>
  - <https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/1/0/1>
  - <https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/1/1/1>



Table 4.5. Projection functions implemented in Traffic Viewer

	From	To	Function
1.	Map	World	$w_x = \left(0.5 + \frac{m_x}{360}\right) \times 256,$ $w_y = \left[0.5 - \frac{\ln(1 + m_1) - \ln(1 - m_1)}{4\pi}\right] \times 256$ <p>where <math>m_1 = \max\left[-0.9999, \min(m_2, 0.9999)\right]</math>  and <math>m_2 = \sin\left(m_y \times \frac{\pi}{180}\right)</math></p>
2.	World	Pixel	$p_x = \lfloor 2^z \cdot w_x \rfloor$ $p_y = \lfloor 2^z \cdot w_y \rfloor$
3.	World	Tile	$t_x = \left\lfloor \frac{2^z \cdot w_x}{256} \right\rfloor$ $t_y = \left\lfloor \frac{2^z \cdot w_y}{256} \right\rfloor$

Inverse functions are needed in order to handle the user interactions such as querying the geo-location  $(m_x, m_y) = (\textit{latitude}, \textit{longitude})$  at the cursor location  $(p_x, p_y)$  on the application screen. Table 4.6 lists the inverse projection functions used in Traffic Viewer to un-project a location from the pixel coordinate system to the map coordinate system. Thus, given a cursor's location on the viewer canvas, the inverse functions can be applied to find the latitude and longitude location at the cursor's location.

Table 4.6. Inverse projection functions implemented in Traffic Viewer

	From	To	Inverse Function
1.	Pixel	Map	$m_x = \left(\frac{p_x}{2^z \cdot 256} - 0.5\right) \times 360$ $m_y = (\sin^{-1} p_1) \times \frac{180}{\pi}$ <p>where <math>p_1 = \max\left[-0.9999, \min(p_2, 0.9999)\right]</math>  and <math>p_2 = \frac{p_3 - 1}{p_3 + 1}</math>  and <math>p_3 = e^{\left(0.5 - \frac{p_y}{2^z \cdot 256}\right) \times 4\pi}</math></p>

The Web Mercator Projection can be applied to render custom geospatial data in map tile images. Figure 4.4 shows two examples of an extra weather map tile layer generated by a TestBed Weather Tile Image Server using Rapid Refresh [18] data from the NASA's ATM Sherlock ATM Data Warehouse. The filled contours, also known as isobands, of the weather data are created using triangular meshes with the Marching Squares algorithm [19]. Figure 4.4(a) shows the map tile layer using the wind magnitude calculated at an altitude of 30,000 ft; Figure 4.4(b) shows the map tile layer using the total cloud cover, by percentage.

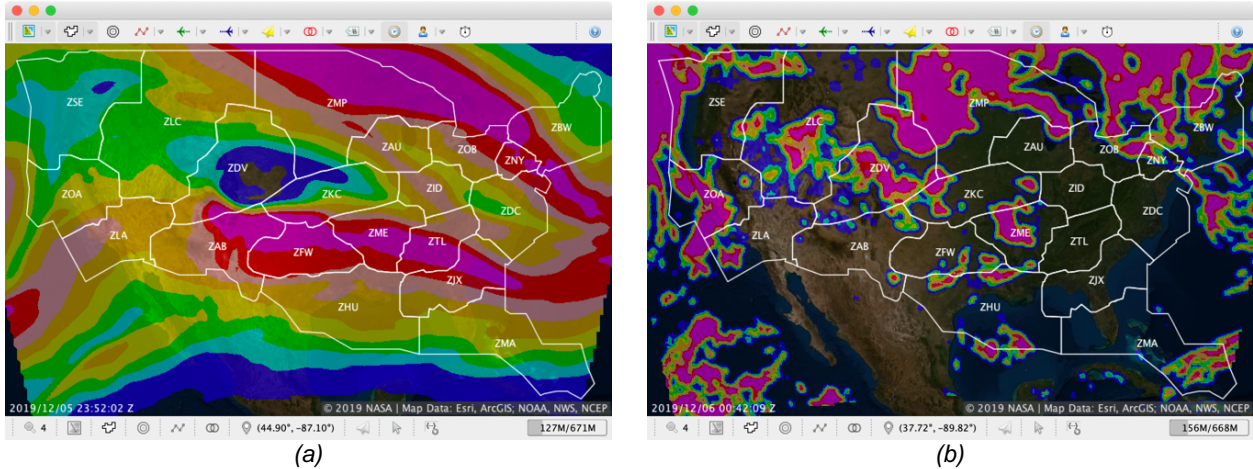


Figure 4.4. Map tile images of weather data: (a) Wind magnitude at 30,000 ft and (b) Total percentage of cloud cover (blue color indicates low value; red color indicates high value)

In order to reduce network data usage, a two-level caching scheme is employed: map tile images are cached in memory for short-term access and are cached on disk for medium-term access. The caching scheme is illustrated in Figure 4.5. For the file-based storage, the images are stored in one of the following folders based on the operating system:

1. Windows in `${user.home}\.sntb\TrafficViewer\cache\map_tile\`.
2. Linux in `~/.sntb/TrafficViewer/cache/map_tile/`.

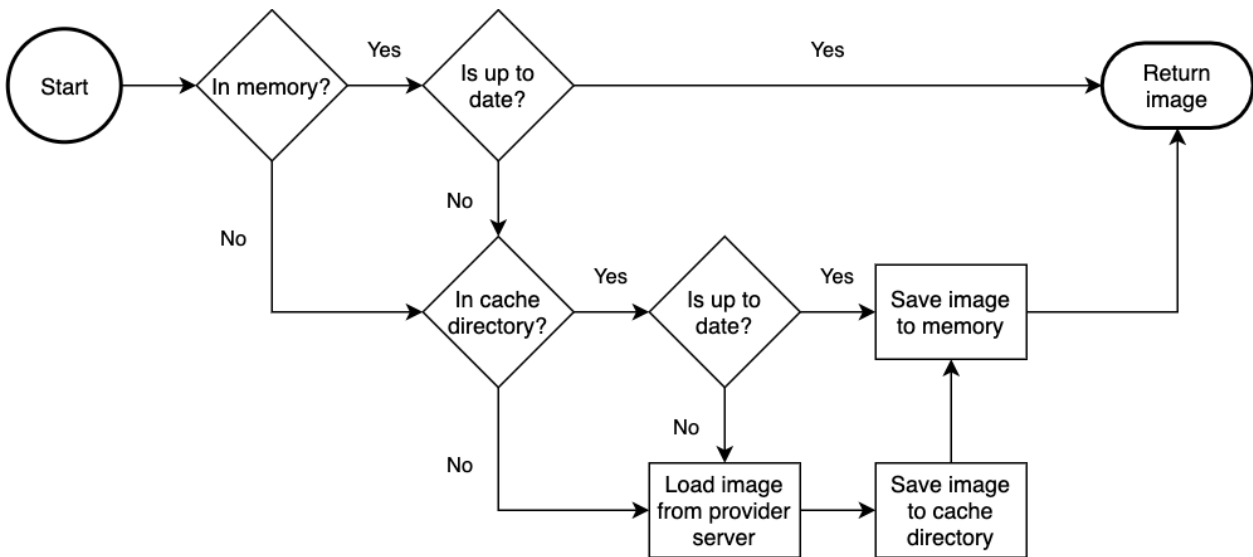


Figure 4.5. Two-level caching scheme to load a map tile image

<sup>10</sup> The macro `${user.home}` represents the user's home directory, e.g., `C:\Users\clai\`.

### 4.1.2. Map Tile Layer Properties

Table 4.7 lists the properties related to the map tile layer with the property name prefix `map_tile`. These properties apply to enabled map tile providers.

Table 4.7. Map tile properties

Name	Type	Description
<code>map_tile_cache_dir</code>	String	Directory for storing cached map tile files. Default value: <code>\${user.home}/.sntb/TrafficViewer/cache/map_tile/</code>
<code>map_tile_cache_size</code>	int	Maximum cached entries for map tile images in memory. Value is a non-negative integer. Default value: 1024
<code>map_tile_loader_timeout</code>	long	Time out for both network connection and read operation when loading an image Uniform Resource Locator (URL). Value is a non-negative long in milliseconds. A timeout of zero (0) is interpreted as an infinite timeout. Default value: 5000
<code>map_tile_preload_edge</code>	boolean <sup>11</sup>	Flag indicates whether to preload edge map tile images when the map canvas is being panned. Enabling this option will reduce a flickering effect on the viewer canvas edges while panning. Default value: true
<code>map_tile_preload_nextLevel</code>	boolean	Flag indicates whether to preload map tile images on the next (higher) zoom level. Enabling this option will reduce a flickering effect while zooming in the viewer canvas. Default value: false
<code>map_tile_preload_previousLevel</code>	boolean	Flag indicates whether to preload map tile images on the previous (lower) zoom level. Enabling this option will reduce a flickering effect while zooming out the viewer canvas. Default value: false

In addition, Table 4.8 lists the properties defined in the map tile layer with the name pattern `gov.nasa.sntb.trafficviewer.ui.layer.MapTileLayer.<Name>`. The `mapProviders` property specifies map tile image providers to be included in the layer.

Table 4.8. Map tile layer properties

<Name>	Type	Description
<code>alpha</code>	float	Image transparency factor, a value between 0 (fully transparent) and 1 (no transparency). Default value: 0.5

<sup>11</sup> Boolean values are either `true` or `false`.

fillColor	Color <sup>12</sup>	Background color of copyright region. Default value: (64, 64, 64, 192)
mapProviders	CSV	CSV of supported map provider names. Note that the NASA_ATM_Weather_THREDDS map provider is commented out by default, as it requires a TestBed Weather Image Server running on a local machine. Default value: ArcGIS_World_Imagery, \ ArcGIS_World_Boundaries_and_Places, \ # NASA_ATM_Weather_THREDDS,
textColor	Color	Color of copyright text. Default value: (224, 224, 224)

Each map image tile provider defined in the mapProviders property value must have the properties specified by the property key map\_provider.<mapProviderName>.<Name> as listed in Table 4.9.

Table 4.9. Map provider properties

<Name>	Type	Description
name	String	Name of the map provider, a text to be displayed in the drop-down menu.
attribution	String	Attribution text of map provider, a detail text to be shown in the “About Map Tiles” dialog (see Figure 4.6).
source	String	Source text of map provider, an abbreviation of the map provider to be shown on the bottom-right corner of the viewer canvas.
maxSouthEastTileUrl	URL	URL of the map tile image at the South-East corner at the maximum zoom level. The URL value usually has the format <code>https://&lt;host&gt;/&lt;type&gt;/tile/&lt;z1&gt;/&lt;ty&gt;&lt;/tx&gt;</code> .
maxZoomLevel	int	0-based maximum zoom level that is supported by the map provider.
retentionPeriod	String	Duration to retain the cached map tile images on disk. The value format is YY-MM-DD'T' hh:mm:ss, e.g., 0-0-30T0:0:0 represents 30 days.
enabled	boolean	Flag indicates whether the map provider is enabled (true) or disabled (false) by default.

Listing 4.3 lists an example list of the values for the ArcGIS World Imagery map provider (long lines are wrapped).

<sup>12</sup> Color values are strings in (rrr, ggg, bbb) or (rrr, ggg, bbb, aaa) format, where rrr, ggg, bbb, aaa are integer color component values—red, green, blue, and alpha—between 0 and 255, inclusive. If the alpha component value is omitted, its default value is 255.

East corner is 23, the maximum zoom level is set to 19 as indicated by the `maxZoomLevel` property value.

Listing 4.3. Example map provider properties

```
map_provider.ArcGIS_World_Imagery.name = ArcGIS: World Imagery
map_provider.ArcGIS_World_Imagery.attribution = Esri, DigitalGlobe, GeoEye,
  Earthstar Geographics, CNES/Airbus DS, USDA, USGS, AeroGRID, IGN, and the GIS
  User Community
map_provider.ArcGIS_World_Imagery.source = Esri, ArcGIS
map_provider.ArcGIS_World_Imagery.maxSouthEastTileUrl =
  https://services.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer/
  tile/23/8374868/8388607
map_provider.ArcGIS_World_Imagery.maxZoomLevel = 19
map_provider.ArcGIS_World_Imagery.retentionPeriod = 0-0-30T0:0:0
map_provider.ArcGIS_World_Imagery.enabled = true
```

Since map providers may have different maximum zoom levels, to ensure map tiles from all the map providers will be displayed, Traffic Viewer limits the maximum zoom level. Given a list of map providers,  $P = \{P_1, P_2, \dots, P_n\}$ , defined in the properties, the upper zoomable level supported in the map tile layer is the minimum value of the `maxZoomLevel` properties among the map providers regardless they are enabled or not, thus,

$$\text{Upper Zoom Level} = \min_{i \in [1..n]} \{\text{maxZoomLevel}(P_i)\}$$

#### 4.1.3. Map Tile Layer Menu

Table 4.10 lists the menu items associated with the map tile layer. Each menu item and its description in the drop-down menu will be listed in this section. A checkmark (✓) symbol on the first column indicates a check box menu item is initially selected, thus, the associated feature is initially enabled.

Table 4.10. Map tile layer menu

	Menu Item	Description
✓	Enabled	Enables/disables the map tile layer. This menu item allows a user to quickly enable or disable the layer without manually disabling all other features.
	About Map Tiles...	Shows a dialog listing the sources and attributions of individual map tile image providers (see Figure 4.6).
	All	Selects all the map tile image providers. Use this menu item to enable all the map providers.
	None	Clears selections of all the map tile image providers. Use this menu item to disable all the map providers.
✓	ArcGIS: World Imagery	Shows/hides ArcGIS World Imagery [20] map tile images (see Figure 4.7(a)).
	ArcGIS: World Boundaries and Places	Shows/hides ArcGIS World Boundaries and Places [21] map tile images (see Figure 4.7(b)).
	<Map Provider>	Shows/hides map tile images by additional map providers. Check marks are shown whenever the map providers are enabled.

The sources, attributions, and copyright information of the map tile image providers are shown in the “About Map Tiles” dialog (see Figure 4.6). The information is obtained by the map provider’s three properties: `name`, `attribution`, and `source`.

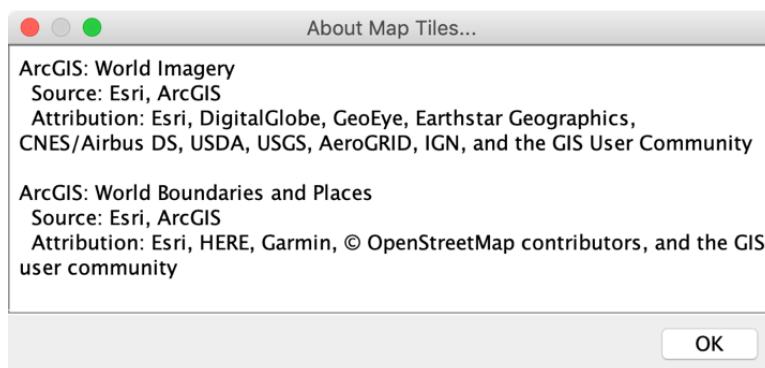


Figure 4.6. About Map Tiles dialog

Two map tile image providers by ArcGIS Online are included in Traffic Viewer. The two providers present high-resolution satellite imagery as well as world boundaries and places (see Figure 4.7). Additional map tile image providers can be included in the property `mapProviders` (see Section 4.1.2).

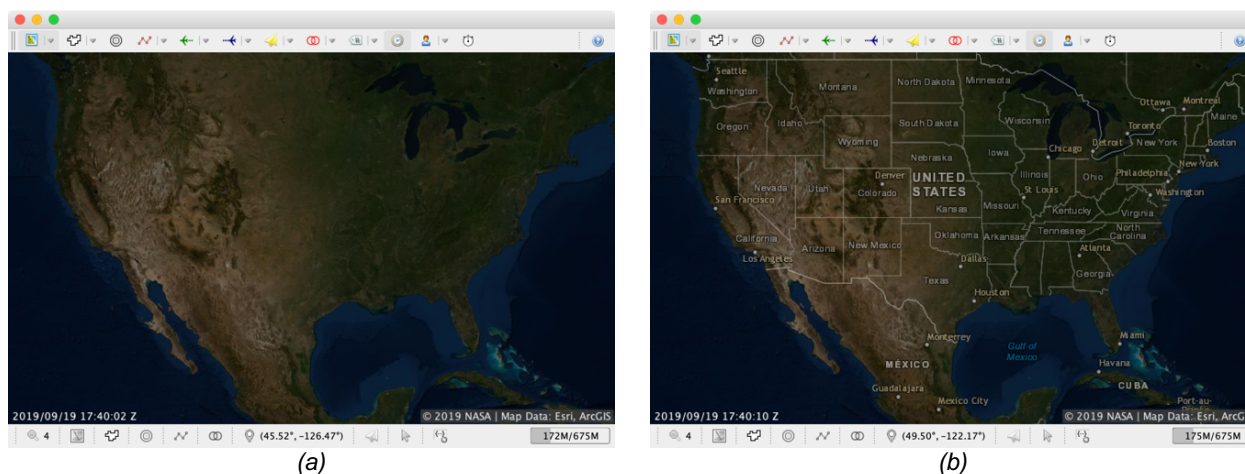


Figure 4.7. Map providers: (a) ArcGIS: World Imagery, and (b) ArcGIS: World Imagery, Boundaries and Places

#### 4.1.4. Map Tile Setting Menu

Setting menus on the status bar can be found in the following subsections:

5.1.	Zoom Level .....	46
5.2.	Map Settings .....	46
5.7.	Location Information .....	49
5.9.	Cursor Status .....	50

## 4.2. Boundary Layer

The Boundary Layer draws named boundaries of airspace such as Air Route Traffic Control Centers (ARTCCs), sectors, and subsectors using polygons and texts that are defined in boundary definition files. The polygons are represented in GeoJSON format [22] and an example snippet of a boundary definition is listed in Listing 4.4.

Listing 4.4. Example snippet of boundary definition

```

{
  "type": "GeometryCollection",
  "geometries": [
    {
      "coordinates": [
        [
          [ -105.34167, 36.716667 ],
          [ -105.0, 36.716667 ],
          [ -103.15, 37.308334 ], ...
          [ -105.34167, 36.716667 ]
        ]
      ],
      "type": "Polygon",
      "name": "ZAB"
    },
    { ... },
    { ... }
  ]
}

```

This layer is useful for showing boundaries and points of interest that are specific to simulation runs. Figure 4.8 shows boundaries of Oakland ARTCC (ZOA) with the high-altitude sectors (a) disabled or (b) enabled.

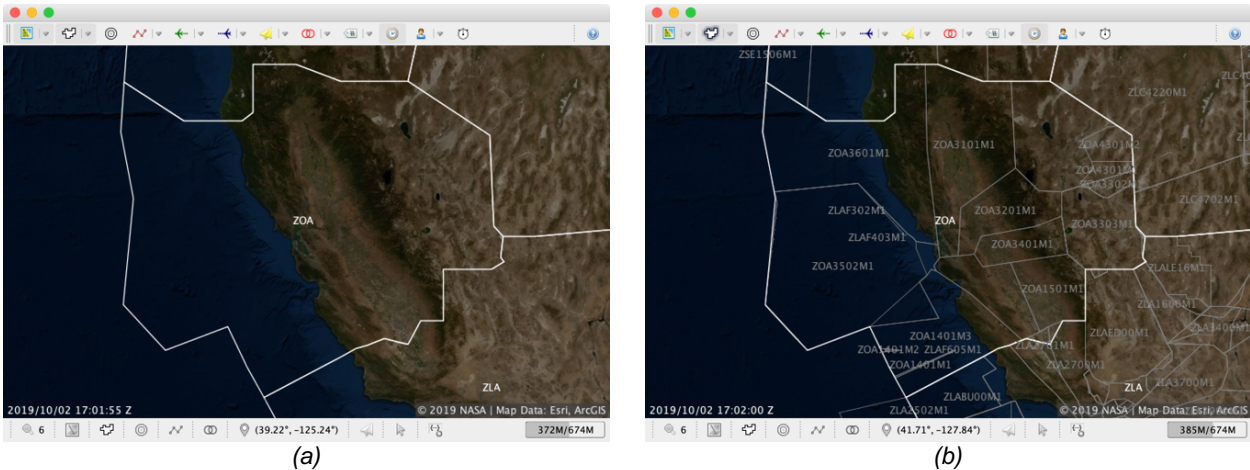


Figure 4.8. Oakland ARTCC boundary with high-altitude sectors (a) disabled and (b) enabled

Table 4.11 lists the three supported boundary definition file formats in the Boundary Layer. The plain text files with the file extension `.json` do not use compression and are good for development. Compressed files have smaller file sizes and they improve the performance whenever a Traffic Viewer instance is distributed to other parties during a simulation run.

Table 4.11. Supported boundary definition file formats

	File Extension	File Format	Use Case
1.	.json	Plain text file in GeoJSON format.	During development.
2.	.json.gz	GZIP [23] file of a plain text file.	Linux platform.
3.	.json.zip	ZIP [24] archive file of a plain text file.	Windows platform.

Users may show or hide a specific group of boundaries via the layer drop-down menu. For example, all the high-altitude sectors are initially hidden, and a user may change them to visible by selecting the “High Sector” menu item. In addition, specific boundaries may be shown or hidden by entering their names in the Boundary settings dialog (see Section 4.2.3).

#### 4.2.1. Boundary Layer Properties

Properties related to the boundary layer have the prefix `boundary_`. Table 4.12 lists the properties used in this layer.

Table 4.12. Map tile layer properties

Name	Type	Description
<code>boundary_home_min</code>	(latitude, longitude)	Geo-coordinates of top-left home region to be zoomed to. Setting either latitude or longitude to NaN <sup>13</sup> will use the visible boundary values. Default value: (NaN, NaN)
<code>boundary_home_max</code>	(latitude, longitude)	Geo-coordinates of bottom-right home region to be zoomed to. Setting either latitude or longitude to NaN will use the visible boundary values. Default value: (NaN, NaN)
<code>boundary_type_list</code>	CSV	List of boundary types to be included in the boundary layer menu. Note that underscores ( <code>_</code> ) will be replaced with spaces ( <code> </code> ) in the menu. Default value: <pre>Low_Sector, \ High_Sector, \ Super_Sector, \ Low_Center, \ High_Center,</pre>
<code>boundary_selected_type_list</code>	CSV	Selected list of boundary layers to be initially shown. Default value: <pre>High_Center,</pre>

Each boundary defined in the `boundary_type_list` property value must have the properties specified by the property key `boundary.<BoundaryType>.<Name>` as described in Table 4.13. These boundary properties define the location of boundary definition file, colors of the boundaries and names, and the width of the boundary lines.

Table 4.13. Boundary layer properties

<Name>	Type	Description
<code>file</code>	String	Path of the boundary definition file.
<code>lineColor</code>	Color	Color of the boundary lines.

<sup>13</sup> NaN represents a Not-a-Number value of type double. Its value is `java.lang.Double.NaN`.



lineWidth	float	Width, in px, of the boundary lines.
textColor	Color	Color of the boundary names.

#### 4.2.2. Boundary Layer Menu




Table 4.14 lists the menu items associated with the boundary layer. The menu items let a user know what boundaries are available and which are enabled. Boundaries can be individually enabled or disabled.

Table 4.14. Boundary layer menu


	Menu Item	Description
✓	Enabled	Enables/disables the boundaries layer. This menu item allows a user to quickly enable or disable the layer.
	All	Selects all the boundaries. Use this menu item to enable all the boundaries.
	None	Clears selections of all the boundaries. Use this menu item to disable all the boundaries.
	Low Sector	Shows/hides low-altitude sector boundaries.
	High Sector	Shows/hides high-altitude sector boundaries.
	Super Sector	Shows/hides superhigh-altitude sector boundaries.
	Low Center	Shows/hides low-altitude center boundaries.
✓	High Center	Shows/hides high-altitude center boundaries.
	<Boundary Name>	Shows/hides other boundaries. Check marks are shown whenever the boundaries are enabled.

#### 4.2.3. Boundary Setting Menu

Setting menu on the status bar can be found in the following subsection:

0.  Clicking on the  Darker button or moving the slider's knob towards the Darker 47 button will decrease the brightness of the map tiles (see Figure 5.3(a)). On the other hand, clicking on the  Brighter button or moving the slider's knob towards the Brighter button will increase the brightness of the map tiles (see Figure 5.3(b)).  
Boundary Settings .....

#### 4.3. Range Ring Layer

The  Range Ring Layer draws range rings on the canvas to measure the distance from a specific location. Clicking on the layer menu button will enable or disable the layer. Though the layer menu is a toggle button and does not provide additional menu items, a user may control the range rings in the Range Rings settings dialog (see Section 4.3.1), including the center location of the range rings, distance labels between two consecutive rings, number of rings, and locations of the distance labels. Figure 4.9 shows a screenshot of Traffic Viewer with ten rings, separated by 20 nmi, centered at the San Francisco International Airport, and the ring distance labels located at both the East (90°) and the South (180°) directions.

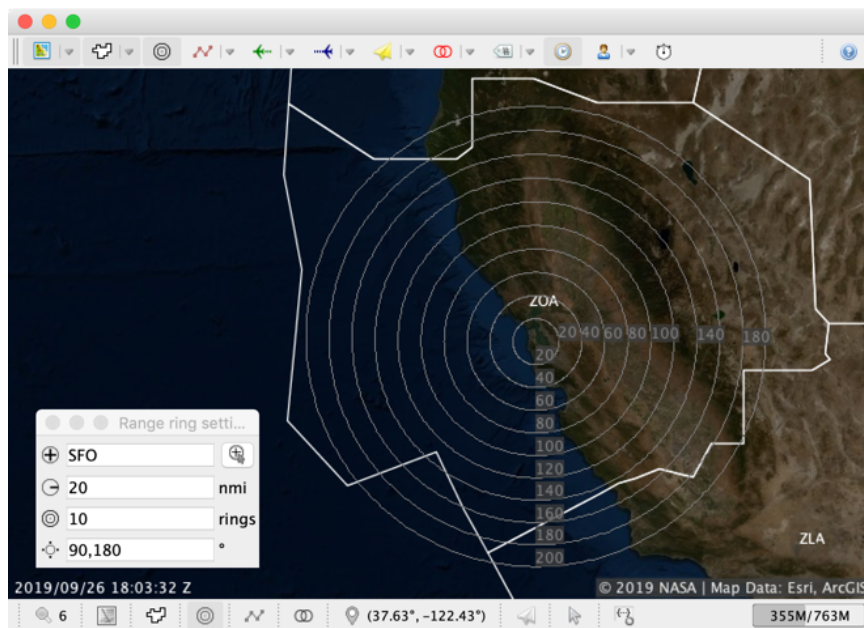


Figure 4.9. Range ring layer

Table 4.15 lists the properties defined in the range ring layer with the name pattern `gov.nasa.sntb.trafficviewer.ui.layer.RangeRingLayer.<Name>`. These range ring properties define the colors of the rings and the distance labels.

Table 4.15. Range ring layer properties

<Name>	Type	Description
ringColor	Color	Color of the rings. Default value: (128, 128, 128)
textFillColor	Color	Background fill color of the text regions. Default value: (64, 64, 64, 192)
stepDegrees	int	Heading change per step for drawing ring boundary. Value is a positive integer. Default value: 10

#### 4.3.1. Range Ring Setting Menu

Setting menu on the status bar can be found in the following subsection:

- 5.4. Range Ring Settings ..... 47

#### 4.4. Flight Plan Layer

The Flight Plan Layer draws flight plan routes and displays their associated waypoint names. Flight plan routes are defined in the Flight Plan Data Exchange Model messages. The routes are values defined in the “Route of Flight” field of the FAA Form 7233-1 [25]. A route of flight consists of a departure point, a series of waypoints, and a destination point. An example of the route of flight is listed in Listing 4.5.

Listing 4.5. Example route of flight in flight plan

```
MMMZ./ .VENUS044013. .HOTTT.PINNG1.KPHX
```

This layer uses the FAA’s Aeronautical Data (formerly called National Flight Data Center) [26] files to resolve the waypoint locations into geo-locations. In simulation experiments, the Aeronautical Data may not cover all the waypoints needed. For example, an experiment may want to use new airspace adaptation data or assign existing waypoints to alternate locations. To address these use cases, Traffic Viewer supports custom mappings of waypoint and geo-location. The resolution of a waypoint to a geo-location is based on three lookup sources—runtime settings, properties, and custom files:

1. The runtime settings are specified using the Flight Plan setting menu (see Section 5.5). This allows a user to dynamically specify mappings during a simulation experiment.
2. The properties are specified in the file located at `config/sntb.properties`. This allows a user to specify mappings while configuring a simulation experiment.
3. The custom files, in GeoJSON format, are specified by the property name `gov.nasa.sntb.trafficviewer.ui.layer.FlightPlanLayer.waypointFiles`. This allows a user to specify mappings for a simulation experiment.

Figure 4.10 shows the (a) flight plans and waypoints in cyan color and (b) setting menu that allows users to enter or override waypoint locations.

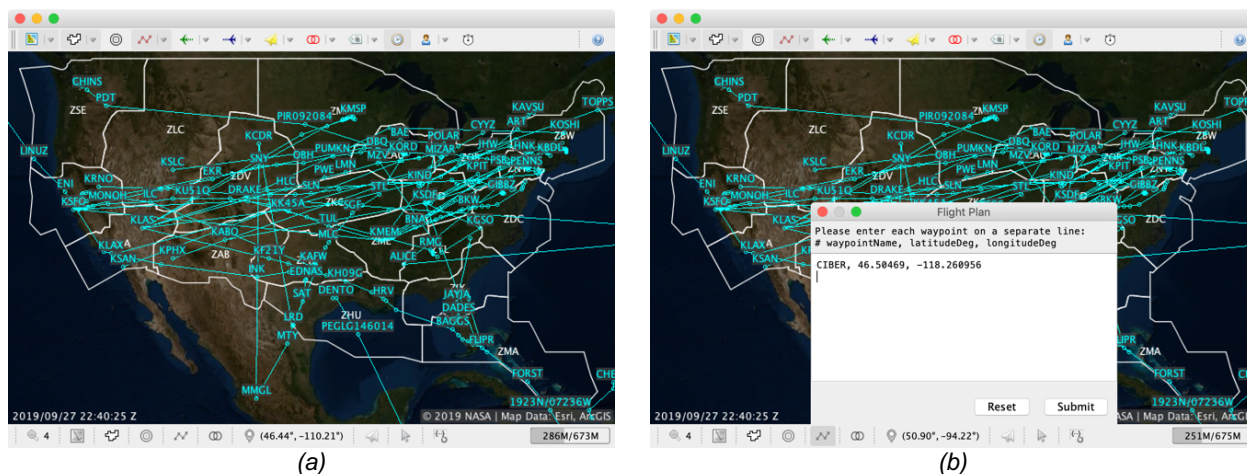


Figure 4.10. Flight plan layer: (a) flight plans and waypoints and (b) flight plan setting

#### 4.4.1. Flight Plan Layer Properties

Table 4.16 lists the properties defined in the flight plan layer with the name pattern `gov.nasa.sntb.trafficviewer.ui.layer.FlightPlanLayer.<Name>`. The properties define the colors of the flight plan routes and waypoint names, as well as the location of the custom waypoint files.

Table 4.16. Flight plan layer properties

<Name>	Type	Description
<code>fillColor</code>	Color	Background color of waypoint name region. Default value: (64, 64, 64, 192)
<code>routeColor</code>	Color	Color of flight route paths. Default value: (0, 255, 255)
<code>textColor</code>	Color	Color of waypoint names. Default value: (0, 255, 255)
<code>trafficStartTime</code>	String	Start time for the waypoint mapping database, in the Java format <code>YYYY-MM-DD 'T' hh:mm:ss 'Z'</code> .

		Default value: 2016-03-23T09:00:00.0Z
waypointFiles	CSV	Command-separated list of files storing user defined waypoints.

#### 4.4.2. Flight Plan Layer Menu

Table 4.17 lists the menu items associated with the flight plan layer. The menu items let a user show or hide waypoints as well as flight plans that are published from specific data sources. The “cursor” mode, when selected, hides flight plans and waypoints of all vehicles except the one that is closest to the cursor.

Table 4.17. Flight plan layer menu


	Menu Item	Description
✓	Enabled	Enables/disables the flight plan layer.
	Cursor	If selected, shows the flight plan of the vehicle that is closest to the cursor only.
✓	Waypoints	Shows/hides the names of the waypoints.
	All	Selects all the data sources. Use this menu item to enable all the data sources.
	None	Clears selections of all the data sources. Use this menu item to disable all the data sources.
✓	<Data Source>	Shows/hides flight plans that are associated with the data source.

#### 4.4.3. Flight Plan Setting Menu

Setting menu on the status bar can be found in the following subsection:

5.5.  Flight Plan Settings ..... 47

### 4.5. Flight Path Layer

The  Flight Path Layer draws flown paths of flights. Figure 4.11 shows a screenshot of the flight plan layer displaying flown paths of the flights. Traffic Viewer stores received track locations in the track registry and the track locations are ordered by their timestamps. A flown path is created by constructing a two-dimensional path based on the received track locations. The color of the flown path is based on the vehicle color (see Section 4.7.1). If the cursor mode is enabled on this layer via the layer menu, only the flight path of the flight that is closest to the cursor position will be shown.

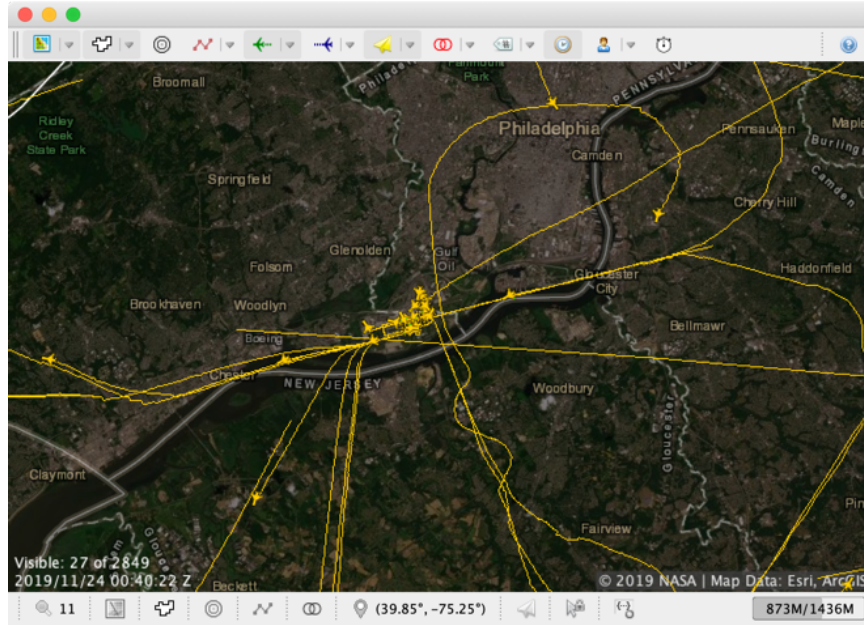


Figure 4.11. Flight path layer


#### 4.5.1. Flight Path Layer Menu

Table 4.18 lists the menu items associated with the flight path layer. The menu items let a user show or hide flown trajectories that are published from specific data sources. The “cursor” mode, when selected, hides flight paths of all vehicles except the one that is closest to the cursor.

Table 4.18. Flight path layer menu

	Menu Item	Description
✓	Enabled	Enables/disables the flight path layer.
	Cursor	If selected, shows the flight path of the vehicle that is closed to the cursor only.
	All	Selects all the data sources. Use this menu item to enable all the data sources.
	None	Clears selections of all the data sources. Use this menu item to disable all the data sources.
✓	<Data Source>	Shows/hides flight paths that are associated with the data source.

#### 4.6. Trajectory Layer

The  Trajectory Layer draws predicted paths of flights. Figure 4.12 shows a screenshot of the trajectory layer displaying a predicted path of a single flight currently in Fort Worth ARTCC (ZFW).

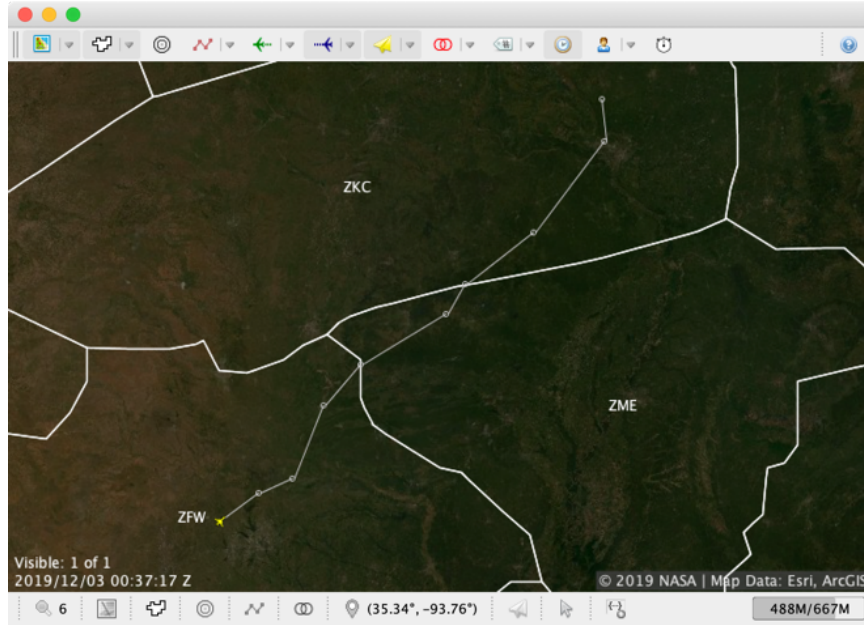


Figure 4.12. Trajectory layer

#### 4.6.1. Trajectory Layer Properties

Table 4.19 lists the properties defined in the trajectory layer with the name pattern `gov.nasa.sntb.trafficviewer.ui.layer.TrajectoryPlanLayer.<Name>`.

Table 4.19. Trajectory layer properties

<Name>	Type	Description
trajectoryColor	Color	Color of the trajectory lines. Default value: (192, 192, 192)

#### 4.6.2. Trajectory Layer Menu

Table 4.20 lists the menu items associated with the trajectory layer. The menu items let a user show or hide trajectories that are published from specific data sources. The “cursor” mode, when selected, hides trajectories of all vehicles except the one that is closest to the cursor.

Table 4.20. Trajectory layer menu

	Menu Item	Description
✓	Enabled	Enables/disables the trajectory layer.
	Cursor	If selected, shows the trajectory of the vehicle that is closed to the cursor only.
	All	Selects all the data sources. Use this menu item to enable all the data sources.
	None	Clears selections of all the data sources. Use this menu item to disable all the data sources.
✓	<Data Source>	Shows/hides trajectories that are associated with the data source.




## 4.7. Track Point Layer

The 🚩 Track Point Layer shows vehicles at their last known locations. Each vehicle is represented by an image icon and controlled by three vehicle-related properties: vehicle color, vehicle type, and vehicle retention period. Note that these properties have similar sets of property keys and the property value formats in the file `config/TrafficViewer.properties`.

### 4.7.1. Vehicle Icon

Vehicle icons are square images with various dimensions. They are scaled at each zoom level such that vehicles will look smaller at lower zoom levels and larger at higher zoom levels. The vehicle icons are grouped based on a vehicle type; the vehicle type is also the name of a resource folder in the directory `$SNTB_HOME/smart-nas/TestBedPlugins/TrafficViewer/src/main/resources/gov/nasa/sntb/trafficviewer/track/`. In the vehicle type folder, each icon filename has a naming pattern `<type>_<width>x<height>.png` consisting of three parts: a vehicle type, an icon width, and an icon height. Both width and height parts are integer values in pixels. For example, `aircraft_96x96.png` is an icon for the aircraft type with a dimension  $96px \times 96px$ .

Table 4.21. Supported vehicle types and icons

	Vehicle Type	Icon	Description
1.	aircraft		Default type for all the vehicles.
2.	drone		Quadcopter drone.
3.	vtol		Vertical take-off and landing.

Traffic Viewer application provides three groups of vehicle icons, as shown in Table 4.21. Vehicles represented by the icons are headed north, i.e., with a heading of  $360^\circ$ . The layer will rotate the icon based on the heading of a vehicle. In order to support image rotation, vehicle boundary in an icon needs to be rendered within a circle inscribed in the image square. Figure 4.13 shows the design of an aircraft icon image with dimension  $96px \times 96px$ :

- The green border represents the image boundary. The image may contain four kinds of colors: black, gray, white, and transparent. The black and gray colors are for the vehicle outline. The white color is for the region inside the aircraft outline such as the fuselage and the wings. The transparent selection is for the region outside the aircraft outline such as the airspace.
- In order to support image rotation and truncation, the aircraft outline needs to be within a circle inscribed in the image square. The circle is highlighted in dark blue color and marked with the label “supported” for illustration.
- The aircraft image is rotated by 45 degrees clockwise as indicated by the green border. The rotated image will be truncated to a square image with the dimension  $96px \times 96px$  as indicated by the orange border. Note that the circle inscribed in the image square can still be rendered in the Track Point layer using the truncated squared image bounded by the orange border.

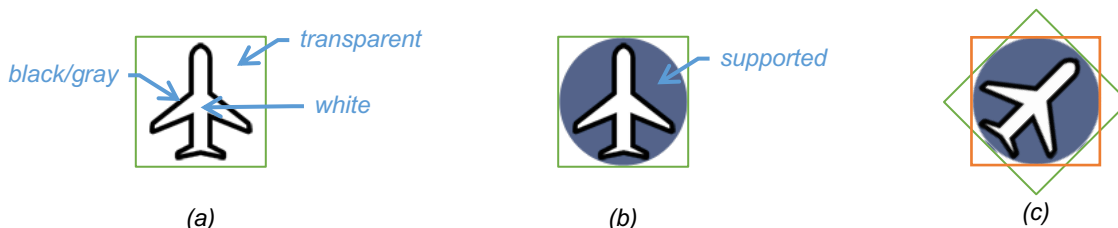


Figure 4.13. Vehicle images: (a) Sample aircraft icon, (b) Supported region, and (c) Heading of 45 degrees.

In general, vehicle icons are square images. Black, gray, white, and transparent colors are supported. The vehicle body needs to be headed north and within a circle inscribed in the image square.

For each vehicle type, Traffic Viewer provides ten images with the following dimensions: (1)  $16px \times 16px$ , (2)  $24px \times 24px$ , (3)  $32px \times 32px$ , (4)  $48px \times 48px$ , (5)  $64px \times 64px$ , (6)  $72px \times 72px$ , (7)  $96px \times 96px$ , (8)  $128px \times 128px$ , (9)  $256px \times 256px$ , and (10)  $512px \times 512px$ . The relationship between zoom levels and vehicle dimensions is listed in Table 4.22. Whenever an image with a given dimension is not available in the image set directory, the image will be scaled based on the previous and next available images. For example, at zoom level ten, the vehicle size is  $14px \times 14px$  and the image will be scaled based on the image with the dimension  $16px \times 16px$ . Vehicle icon sizes can be overridden by the user to a value between one and 512 pixels via the Track Settings (see Section 5.8).

Table 4.22. Zoom levels and vehicle dimensions in px

Zoom	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19+
Size	1	1	1	3	7	9	11	11	13	13	14	14	15	15	16	16	32	64	128	256

#### 4.7.2. Vehicle Color

Color of the vehicle icons are controlled by the vehicle color properties as shown in Table 4.23. The color values are used by layers to draw vehicle related elements such as vehicle icons, flown paths, and predicted trajectories.

Table 4.23. Vehicle color properties

<Name>	Type	Description
vehicle_color	Color	Default color of vehicle images. Default value: (255, 255, 0)
vehicle_color.<source>	Color	Color of vehicle images for individual data sources. Data sources may be component names with or without the dot delimiter, e.g., MACS.1 or MACS for the Multi-Aircraft Control System (MACS) adapter. Note that colons and spaces in a data source must be escaped, e.g., SWIM\:\ ASDE-X for the SWIM: ASDE-X adapter. If a data source is not defined, the color value specified by the property vehicle_color is used.
vehicle_color_callsign_prefixes	List	List of callsign prefixes to be used for vehicle color. Value format is a list of (R,G,B):[prefix1, prefix2, ..., prefixN]; where (R,G,B) is the color definition, e.g., (0, 128, 255), and [prefixX] is the callsign prefix, e.g., [AAL, SWA]. Here is an example value: vehicle_color_callsign_prefixes = \ (255, 0, 0):[AAL, DAL]; \ (0, 255, 0):[SWA]; \ (0, 0, 255):[UAL];

#### 4.7.3. Vehicle Type

Vehicle icons are controlled by the vehicle type properties as shown in Table 4.24. The type values are used by layers to draw vehicle icons.



Table 4.24. Vehicle type properties

<Name>	Type	Description
vehicle_type	String	Default type of vehicle images. Values are folder names in the resource directory \$SNTB_HOME/smart-nas/TestBedPlugins/TrafficViewer/src/main/resources/gov/nasa/sntb/trafficviewer/track/ such as aircraft, drone or vtol. Default value: aircraft
vehicle_type.<source>	String	Type of vehicle images for individual data sources. If a data source is not defined, the type value specified by the property vehicle_type is used.
vehicle_type_callsign_prefixes	List	List of callsign prefixes to be used for vehicle type. Value format is a list of [type]:[prefix1, prefix2, ..., prefixN]; where [type] is the vehicle type, e.g., aircraft, and [prefixX] is the callsign prefix, e.g., [AAL, SWA]. Here is an example value: vehicle_type_callsign_prefixes = \ aircraft:[AAL, SWA]; \ drone:[UTM]; \ vtol:[UAM];

#### 4.7.4. Vehicle Retention Period

Durations of the vehicle data to be kept in the application memory are controlled by the vehicle retention period properties as shown in Table 4.25. The duration values are used by layers to draw vehicle related elements such as vehicle icons, flown paths, and predicted trajectories.

Table 4.25. Vehicle type properties

<Name>	Type	Description
vehicle_retention	long	Default retention period, in ms, for the vehicle data to be stored in the application memory. If a vehicle has not been received for this specific time period, the vehicle data will be eligible for removal. Default value: 600000
vehicle_retention.<source>	long	Retention period, in ms, for individual data sources. If a data source is not defined, the retention period value specified by the property vehicle_retention is used.
vehicle_retention_callsign_prefixes	List	List of callsign prefixes to be used for vehicle retention period, in ms. Value format is [period]:[prefix1, prefix2, ..., prefixN]; where [period] is the vehicle retention period, in ms, and [prefixX] is the callsign prefix, e.g., [AAL, SWA]. Here is an example value: vehicle_retention_callsign_prefixes = \ 60000:[SWA]; \ 90000:[AAL, UAL];

#### 4.7.5. Track Point Layer Properties

Table 4.26 lists the property defined in the track point layer with the name pattern `gov.nasa.sntb.trafficviewer.ui.layer.TrackPointLayer.<Name>`. This sole property controls the text color of the vehicle count message located at the bottom-left corner of the viewer canvas.

Table 4.26. Track point layer properties

<Name>	Type	Description
textColor	Color	Color of vehicle count. Default value: (255, 255, 255)

#### 4.7.6. Track Point Layer Menu

Table 4.27 lists the menu items associated with the track point layer. The menu items let a user show or hide vehicle icons that are published from specific data sources. The “cursor” mode, when selected, highlights the vehicle that is closest to the cursor.

Table 4.27. Track point layer menu


	Menu Item	Description
✓	Enabled	Enables/disables the track point layer.
	Cursor	If selected, shows the track point of the vehicle that is closed to the cursor only.
	All	Selects all the data sources. Use this menu item to enable all the data sources.
	None	Clears selections of all the data sources. Use this menu item to disable all the data sources.
✓	<Data Source>	Shows/hides track points that are associated with the data source.

#### 4.7.7. Track Setting Menu

Setting menu on the status bar can be found in the following subsection:

5.8.  Track Settings ..... 49

### 4.8. Conflict Layer

The  Conflict Layer draws flight conflicts. A conflict involves two flights and each flight will have its own conflict boundary. Figure 4.14 shows a screenshot of the conflict layer demonstrating a conflict between two vehicles, NASA123 and NASA456. The boundaries of the vehicles are indicated by red circles and the conflict information is included.

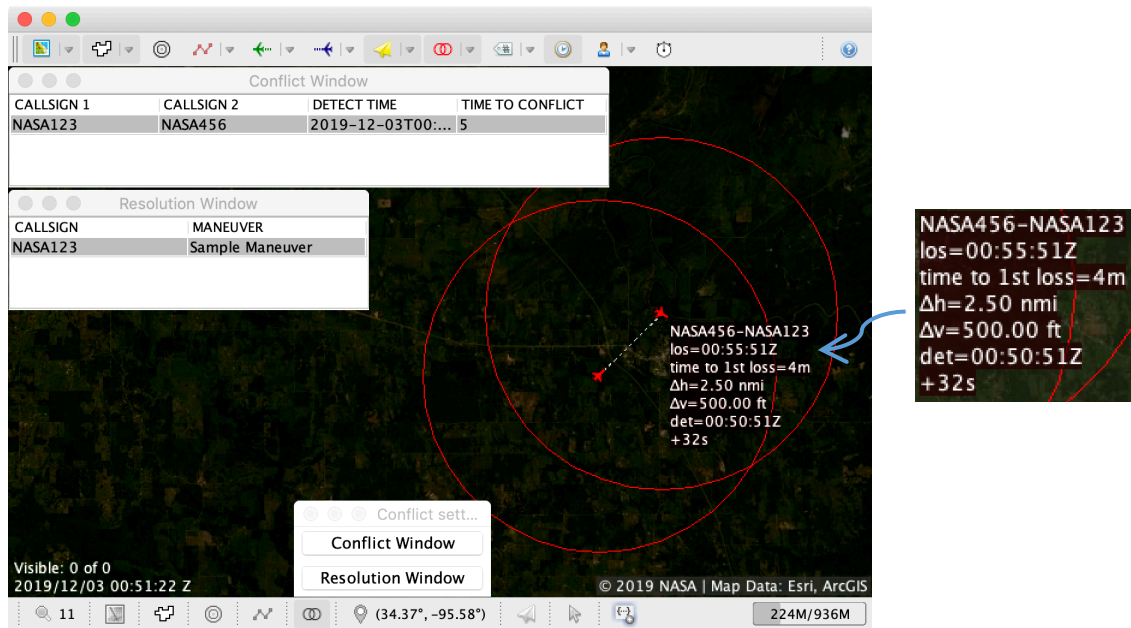


Figure 4.14. Conflict layer

#### 4.8.1. Conflict Layer Properties

Table 4.28 lists the properties defined in the conflict layer with the name pattern `gov.nasa.sntb.trafficviewer.ui.layer.ConflictLayer.<Name>`. The properties define the colors for the conflict boundaries and the information tags, as well as which information tags are displayed. By default, all information tags are displayed.

Table 4.28. Conflict layer properties

<Name>	Type	Description
<code>aircraftColor</code>	Color	Color of the aircraft in conflict. Default value: (255, 0, 0)
<code>fillColor</code>	Color	Background color of tag region. Default value: (32, 0, 0, 192)
<code>linkColor</code>	Color	Color of the link between two aircraft. Default value: (192, 192, 192)
<code>pathColor</code>	Color	Color of the boundary paths. Default value: (255, 0, 0)
<code>textColor</code>	Color	Color of the tag texts. Default value: (255, 255, 255)
<code>tag.Callsign</code>	boolean	Show (true) or hide (false) the tag displaying callsigns of the aircraft in conflict. Default value: true
<code>tag.Conflict_Time</code>	boolean	Show (true) or hide (false) the tag displaying detected conflict time. Default value: true
<code>tag.Time_to_Loss</code>	boolean	Show (true) or hide (false) the tag displaying time to first loss. Default value: true

tag.Horizontal_Distance	boolean	Show (true) or hide (false) the tag displaying horizontal distance between the two aircraft in conflict. Default value: true
tag.Vertical_Distance	boolean	Show (true) or hide (false) the tag displaying vertical distance between the two aircraft in conflict. Default value: true
tag.Detection_Time	boolean	Show (true) or hide (false) the tag displaying detection time of the conflict. Default value: true
tag.Elapsed_Time	boolean	Show (true) or hide (false) the tag displaying elapsed time of the conflict detection. Default value: true

#### 4.8.2. Conflict Layer Menu

Table 4.29 lists the menu items associated with the conflict layer. The menu items let a user show or hide conflict information. All conflict information will be displayed in the initial settings. Note that the order of the tag menu items maps to the order of the conflict tags being displayed.

Table 4.29. Conflict layer menu


	Menu Item	Description
✓	Enabled	Enables/disables the conflict layer.
	All	Selects all the data tags. Use this menu item to enable all the data tags.
	None	Clears selections of all the data tags. Use this menu item to disable all the data tags.
✓	Callsigns	Shows/hides the callsigns of the vehicles in conflict.
✓	Conflict Time	Shows/hides the conflict time of the first loss of separation.
✓	Time to 1st Loss	Shows/hides the time to the first loss of separation.
✓	Horizontal Distance	Shows/hides the horizontal distance, in nautical miles, at the first loss of separation.
✓	Vertical Distance	Shows/hides the vertical distance, in feet, at the first loss of separation.
✓	Detection Time	Shows/hides the time when the conflict is detected.
✓	Elapsed Time	Shows/hides the elapsed time since the detection time.

#### 4.8.3. Conflict Setting Menu

Setting menu on the status bar can be found in the following subsection:

5.6.  Conflict Settings .....	48
--	----

#### 4.9. Flight Tag Layer

The  Flight Tag Layer draws flight information such as callsign, altitude, groundspeed, and heading. Figure 4.15 shows a screenshot of the flight tag layer demonstrating the tags associated with individual flights. To avoid tag overlapping, the layer employs a quadtree data structure [27] to keep track of the tags being drawn.

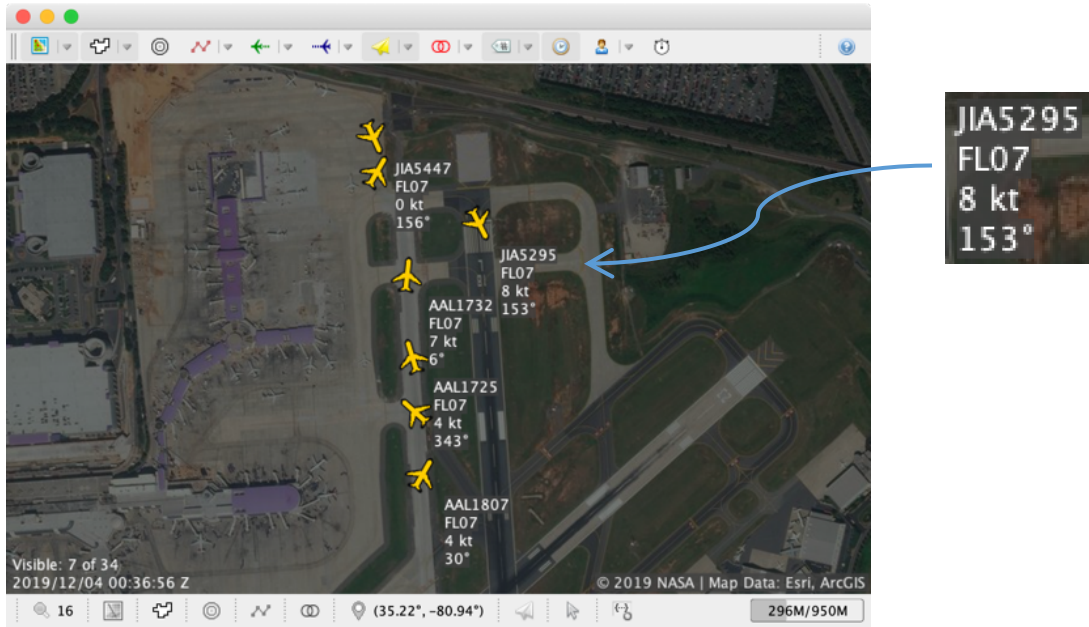


Figure 4.15. Flight tag layer

#### 4.9.1. Flight Tag Layer Menu

Table 4.30 lists the menu items associated with the flight tag layer. The menu items let a user show or hide tags. Callsign, altitude, groundspeed, and heading tags are initially selected. In the “always draw” mode, when selected, all the tags will be drawn on the layer regardless tag overlapping. Note that the order of the tag menu items maps to the order of the flight tags being displayed.

Table 4.30. Flight tag layer menu

	Menu Item	Description
✓	Enabled	Enables/disables the flight tag layer.
	Always Draw	If selected, flight information tags will always be visible regardless overlapping with the other flight information tags.
	All	Selects all the data tags. Use this menu item to enable all the data tags.
	None	Clears selections of all the data sources. Use this menu item to disable all the data tags.
✓	Callsign	Shows/hides the callsign tags.
	Source	Shows/hides the data source tags.
	Vehicle Type	Shows/hides the vehicle type tags.
✓	Altitude	Shows/hides the altitude tags.
✓	Groundspeed	Shows/hides the groundspeed tags.
	Vertical Speed	Shows/hides the vertical speed tags.
✓	Heading	Shows/hides the heading tags.
	Track Time	Shows/hides the track time tags.
	Elapsed Time	Shows/hides the elapsed time tags.

## 4.10. Clock Layer

The 🕒 Clock Layer shows the current simulation time at the bottom-left corner of the view canvas as shown in Figure 4.16. Clicking on the layer menu button will enable or disable the layer. The layer menu is a toggle button and it does not provide additional menu items.

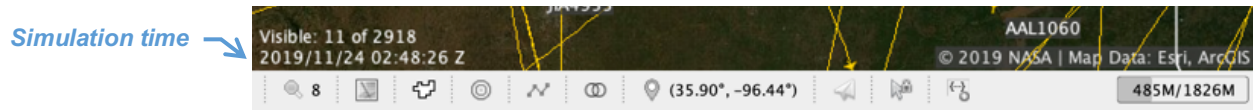


Figure 4.16. Clock layer

## 4.11. Remote Control Layer

The 👤 Remote Control Layer allows basic collaboration among different users within a simulation run, especially for simulations involving physically separated locations such as laboratories. Multiple Traffic Viewer instances may be run in a simulation. For example, instance A is running at Building A and instance B is running at Building B. The two instances can be communicated via the remote control layer.

### 4.11.1. Remote Control Layer Menu

Table 4.31 lists the menu items associated with the remote control layer. Whenever the layer is enabled (disabled by default), Traffic Viewer can be set to either client mode or host mode. A client mode (selected by default) allows the viewer canvas to be controlled by a host. Whenever a user interacts with the host, the viewer canvas publishes the current cursor location, zoom level and map anchor information to all the clients. Only one host is allowed.

Table 4.31. Remote control layer menu

	Menu Item	Description
	Enabled	Enables/disables the remote control layer.
✓	👤 Client	Changes to client mode.
	👤 Host	Host mode to control other traffic viewers.

## 4.12. Duration Layer

The ⌚ Duration Layer shows the durations of time spent on drawing each enabled layer. Clicking on the layer menu button will enable or disable the layer. Time required to draw each enabled layer is displayed at the top-right corner of the viewer canvas as shown in Figure 4.17. This is useful for identifying draw performance of individual layers so that code optimization can be applied in order to improve the application response time. The layer menu is a toggle button and it does not provide additional menu items.

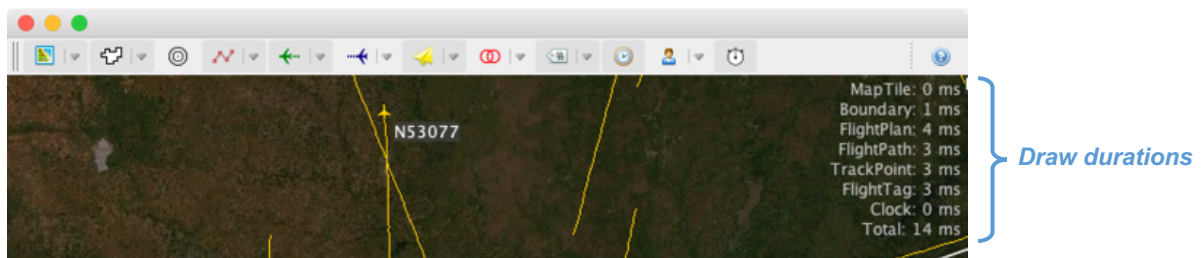


Figure 4.17. Duration layer

## 5. Status Bar

The Status Bar located at the bottom of the application frame provides information about Traffic Viewer. A status bar consists of a series of graphical user interface widgets as shown in Figure 5.1: (a) zoom level, (b) map settings, (c) boundary settings, (d) range ring settings, (e) flight plan settings, (f) conflict settings, (g) location information, (h) track settings, (i) cursor status, (j) user message, and (k) memory monitor. Clicking on any widget will show a detail panel.

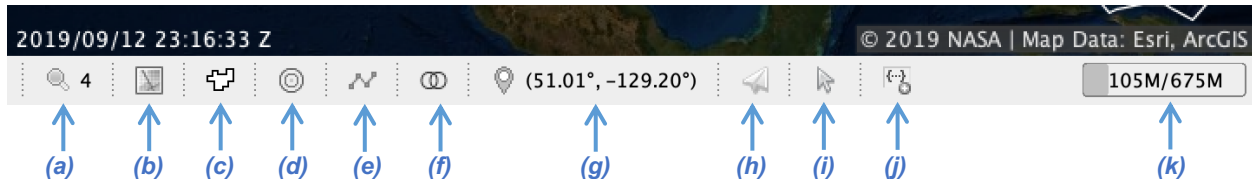


Figure 5.1. Status bar

Each status bar button must be a subclass of the abstract class `Status Bar Button`<sup>14</sup> and the subclass requires implementing the two abstract methods listed in Listing 5.1.

Listing 5.1. Abstract methods a status bar button subclass must implement.


```
public abstract class StatusBarButton extends ToggleButton {
    /**
     * Creates a new content component that will be included in the window.
     * This method is called only once.
     *
     * @return a new content component.
     * @see #createWindow()
     */
    protected abstract JComponent createContentComponent();



    /**
     * Updates content component. This method is called when the button is
     * clicked.
     *
     * @see #onClicked()
     */
    protected abstract void updateContentComponent();
}
```

The status bar buttons can be accessed by pressing and holding a shift key while pressing a number key 1 (to select the first button) through key 0 (to select the last button) on the keyboard. Pressing the space key will show a detail panel. The following sections describe each widget.

<sup>14</sup> The fully qualified name of the class Status Bar Button is `gov.nasa.sntb.trafficviewer.ui.status.StatusBarButton`.

### 5.1. Zoom Level

The  Zoom Settings panel allows a user to control the map zoom level. A vertical slider is provided for adjusting the zoom level. Move the slider's knob towards the bottom will increase the zoom level and zoom in the map. On the other hand, move the slider's knob towards the top will decrease the zoom level and zoom out the map. Figure 5.2(a) shows a zoom-in operation by adjusting the knob from zoom level four to zoom level five. Figure 5.2(b) shows a zoom-out operation by adjusting the knob from zoom level four to zoom level three.

In addition, the  Zoom with Mouse Wheel toggle button allows the user to enable or disable the zoom control via mouse wheel. By default, a user may rotate up/down the mouse wheel to zoom in/out. This feature can be disabled () by deselecting the Zoom with Mouse Wheel button.

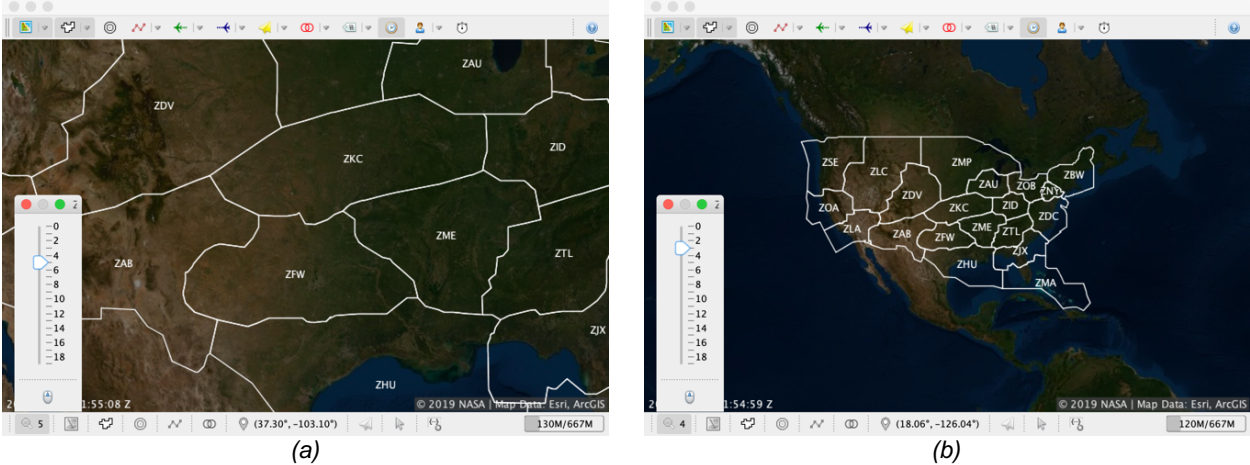




Figure 5.2. Zoom (a) in and (b) out

### 5.2. Map Settings

The  Map Settings panel allows a user to control map layer related settings. Clicks on the  Reset Zoom button resets the map to the initial boundary defined by the two properties boundary\_home\_min and boundary\_home\_max (see Section 4.2.1). In addition, two buttons and a horizontal slider are provided for adjusting the brightness of the map tile images.

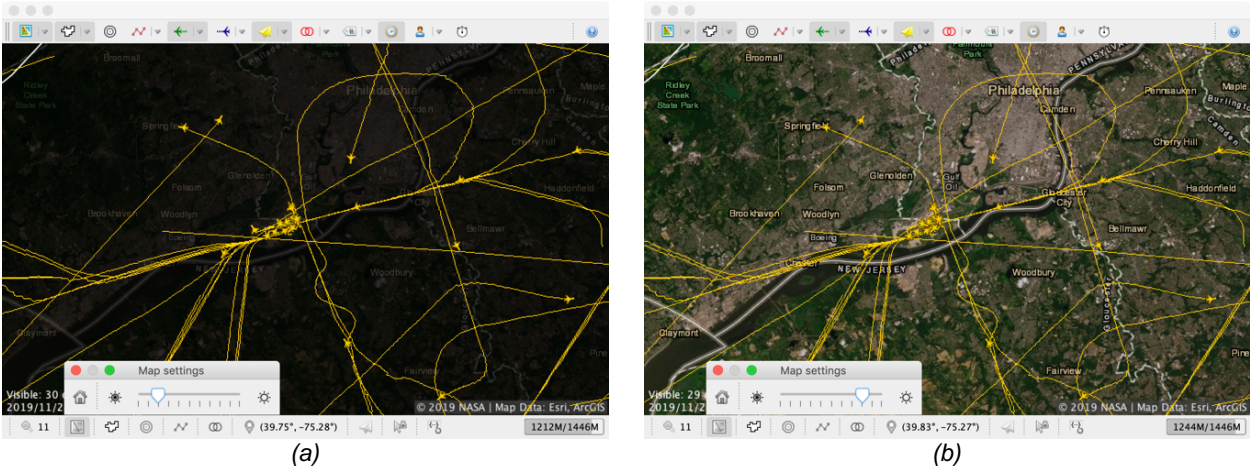


Figure 5.3. Map settings: (a) darker and (b) brighter



Clicking on the ☀ Darker button or moving the slider's knob towards the Darker button will decrease the brightness of the map tiles (see Figure 5.3(a)). On the other hand, clicking on the ☀ Brighter button or moving the slider's knob towards the Brighter button will increase the brightness of the map tiles (see Figure 5.3(b)).

### 5.3. Boundary Settings

The 🗨 Boundary Settings panel allows a user to control visibility of individual airspace regions. For example, the Oakland ARTCC boundary is made hidden in Figure 5.4(a). Clicking on the 🗨 Show All button will show all the airspace regions. On the other hand, clicking on the 🗨 Hide All button will hide all the airspace regions.

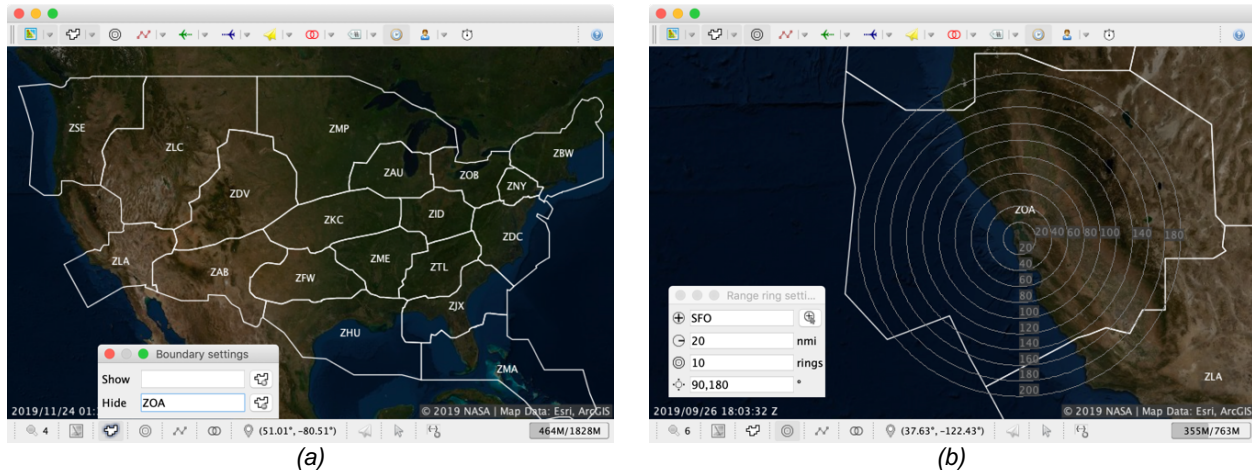


Figure 5.4. (a) Boundary settings and (b) Range ring settings

### 5.4. Range Ring Settings

The 🗨 Range Ring Settings panel allows a user to select and configure the center location of the range rings. Four settings are available:

1. The 🗨 Center location of the range rings setting supports values of latitude/longitude coordinates using a degree-minute-direction format (e.g., 4720N/12216W), a degree-minute-second-direction format (e.g., 472030N/1221645W), or a waypoint name (e.g., KSFO). In addition, a user may use the 🗨 Cursor Selection button to select the center location by clicking on the map.
2. The 🗨 Integer radius, in nmi, of the first ring from the center setting controls the separation distance of subsequent rings. Note that the radius value must be at least one (1) nmi.
3. The 🗨 Integer number of rings to be drawn controls how many rings will be shown. Note that the number of rings must be between one (1) and 1000, inclusive.
4. The 🗨 CSV of integer headings, in degrees, for ring radius texts setting controls the location of the distance labels, e.g., 0, 90, 180, 270.

Figure 5.4(b) shows ten rings, separated by 20 nmi, centered at the San Francisco International Airport, and the ring distance labels located at both the East (90°) and South (180°) directions.

### 5.5. Flight Plan Settings

The 🗨 Flight Plan Settings panel provides a capability for users to add or overwrite waypoint definitions at the application runtime. The panel (see Figure 5.5) has a text editor pane that allows a user enter waypoint definitions line by line. Each line is a CSV consisting of the following fields:

1. Waypoint name, e.g., CIBER. Though, by convention, waypoint names are in upper case, they are case-insensitive in Traffic Viewer.
2. Latitude in degrees, a floating-point value between  $-90^\circ$  and  $+90^\circ$ , inclusive. Positive values are above the equator (N), and negative values are below the equator (S).
3. Longitude in degrees, a floating-point value between  $-180^\circ$  and  $+180^\circ$ , inclusive. Positive values are east of the prime meridian (E), and negative values are west of the prime meridian (W).

Note that whenever multiple waypoints with the same (case insensitive) airspace name are defined, the last one is used. Empty lines or lines starting with hash (#) symbols are skipped.

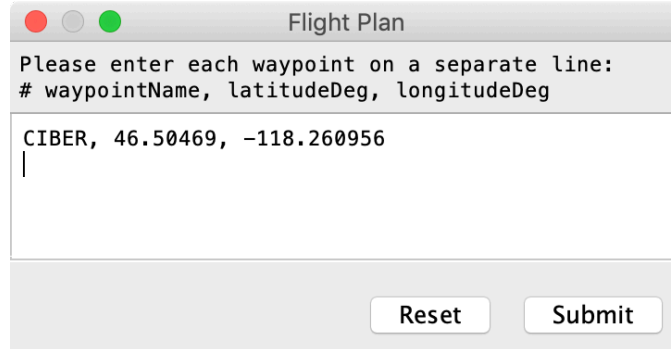



Figure 5.5. Flight plan setting panel

## 5.6. Conflict Settings

The  Conflict Settings panel allows a user to view conflicts and resolutions in separate windows. In general, conflict messages are published by conflict detection algorithms and resolution messages are published by conflict resolution algorithms. Figure 5.6(a) shows the conflict settings panel consisting of two buttons, Conflict Window and Resolution Window. Pressing the former button will display a Conflict Window, shown in Figure 5.6(b), listing basic information of the detected conflicts; pressing the latter button will display a Resolution Window, shown in Figure 5.6(c), listing basic information of the resolutions.

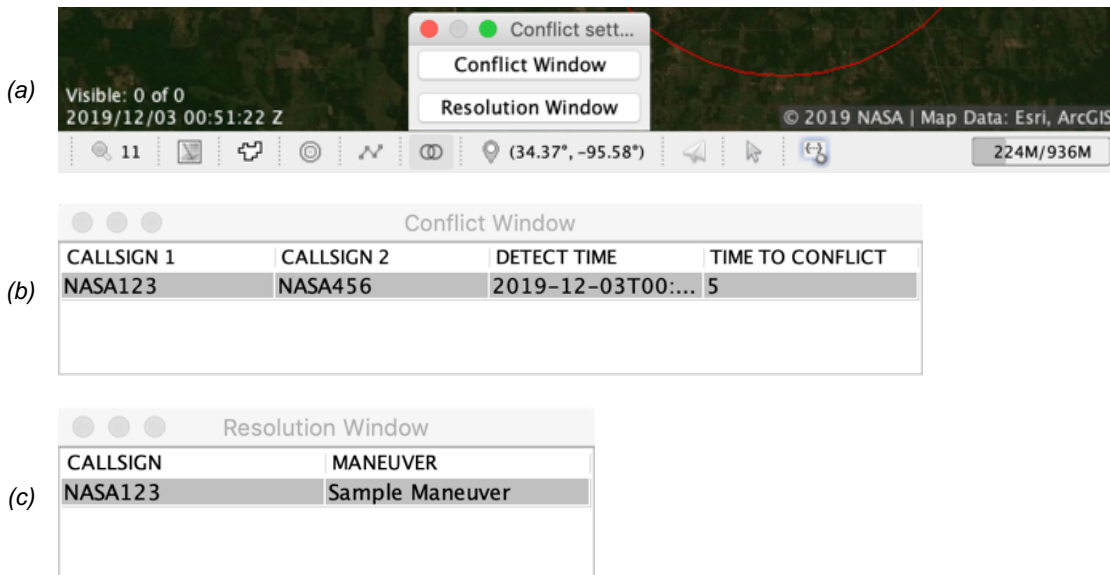









Figure 5.6. Conflict and resolution: (a) Conflict settings panel, (b) Conflict window, and (c) Resolution window

## 5.7. Location Information

The Location Information panel allows a user to view location specific information based on the current cursor location on the viewer canvas. Figure 5.7 shows the informative panel with seven labels:

- (a)  *Translate coordinates* indicate the map translation point  $(p_x, p_y)$  in the world space. This is the top-left corner position of the viewer canvas. The translate coordinates are affected whenever a drag or a zoom operation is performed on the viewer canvas. Moving the mouse without dragging will not affect the translate coordinates.
- (b)  *World coordinates* indicate the world point  $(w_x, w_y)$  in the world space.
- (c)  *Pixel coordinates* indicate the projected world coordinates  $(p_x, p_y)$  in the world space.
- (d)  *Tile coordinates* indicate the projected pixel coordinates  $(t_x, t_y)$  in the tile space.
- (e)  *Cursor coordinates* indicate the cursor point  $(c_x, c_y)$  in the screen space.
- (f)  *WGS84 coordinates (latitude, longitude)* indicate the geo-location at the cursor point in the degrees decimal form.
- (g)  *WGS84 coordinates (latitude, longitude)* indicate the geo-location at the cursor point in the degree-minute-second form.

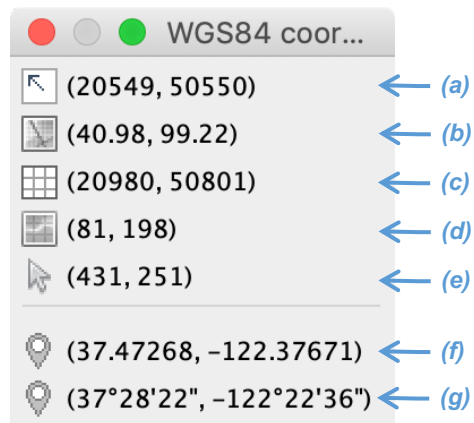



Figure 5.7. Location information panel: (a) *Translate coordinates*, (b) *World coordinates*, (c) *Pixel coordinates*, (d) *Tile coordinates*, (e) *Cursor coordinates*, (f) *WGS84 (decimal)*, and (g) *WGS84 (d°m's")*

## 5.8. Track Settings

The  Track Settings panel allows a user to control the size of the aircraft icons to be rendered on the viewer canvas via a horizontal slider. Dimension of track icon sizes range from one pixel to 512 pixels. This feature is used to enlarge the aircraft icons for better visibility. Figure 5.8 shows the track icons with the (a) auto-size mode and (b) 32-pixel size at zoom level seven.

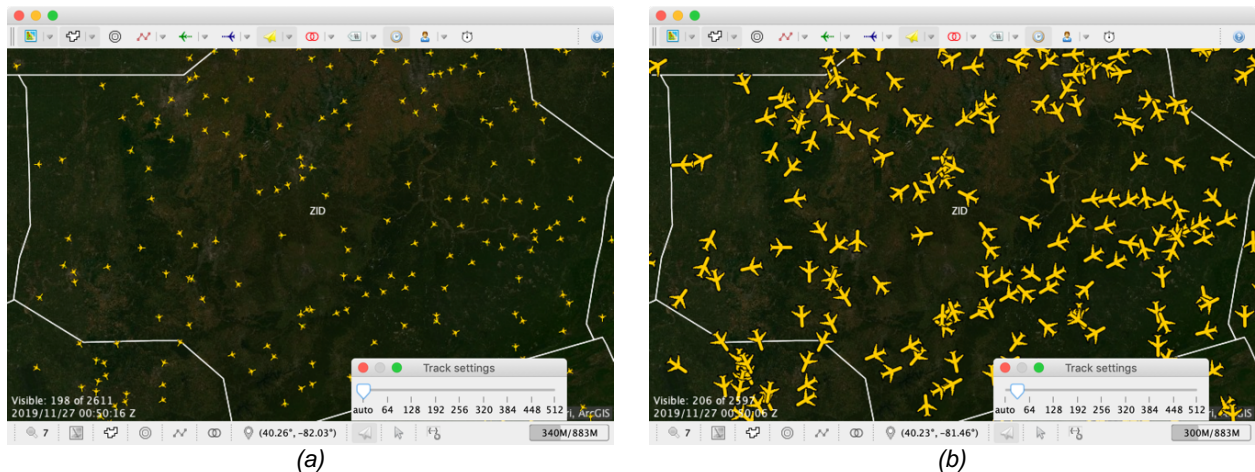


Figure 5.8. Track icon size settings: (a) auto-size mode and (b) 32 pixels

## 5.9. Cursor Status

The Cursor Status panel displays identifiers of the visual elements that are currently selected or locked. A visual element is selected if it is closest to the cursor and its layer has the “cursor” mode enabled. A visual element is locked if a double-click action is performed in a layer with the “cursor” mode enabled. Figure 5.9 shows a screenshot of Traffic Viewer with the cursor mode enabled on the three layers: Flight Plan layer (↗), Flight Path layer (←), and Track Point layer (✈). The panel also indicates that the flight with callsign N939FE is locked (🔒). Double clicking on the viewer canvas toggles the locked and unlocked status. In the locked status, the vehicle remains in the focus even if the cursor is moved. On the other hand, in the unlocked status, the vehicle that is closest to the cursor position is selected whenever the viewer canvas is repainted.

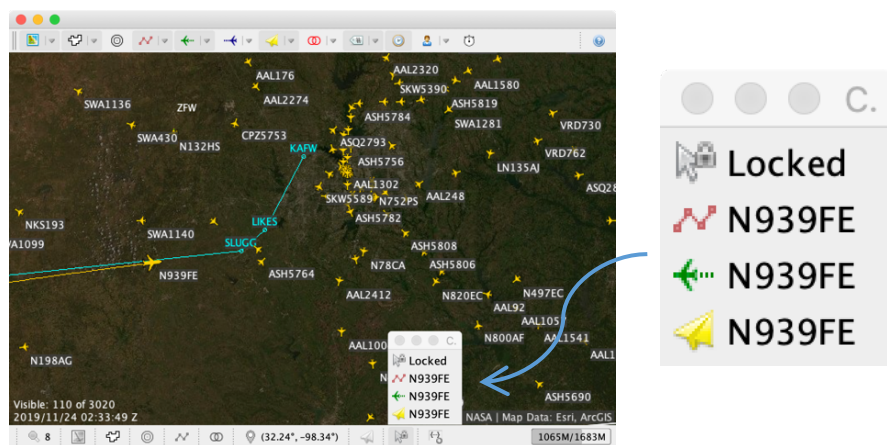



Figure 5.9. Cursor status panel

## 5.10. User Message

The  User Message panel allows a user to manually publish sample messages to the Traffic Viewer. This feature is useful for forcing the application to receive and handle the sample messages without creating an external TestBed adapter to publish the sample messages. The sample messages are defined in sample files located in the resource folder `$SNTB_HOME/src/main/resources/gov/nasa/sntb/trafficviewer/ui/usermessage/`.

Listing 5.2 lists the content of the sample file named `Track.sample`. Dynamic values, enclosed by pairs of macro symbols, `${` and `}`, can be set at the application runtime. For example, whenever publishing the sample track message, the following macros defined in the message will be replaced with the dynamic values:

1. `${latDeg}` and `${lonDeg}` represent the center geo-location of the viewer canvas.
2. `${time}` and `${tpub}` represent the current system clock time, in milliseconds.

*Listing 5.2. Default `Track.sample` file content*

```
{
  "vid": "NASA123",
  "latDeg": "${latDeg}",
  "lonDeg": "${lonDeg}",
  "altFt": 27000.0,
  "time": "${time}",
  "gsKt": 123.45,
  "crsDeg": 45.0,
  "vsFpm": -30.0,
  "meta": {
    "src": "Sample.1",
    "ver": "2.0a",
    "tpub": "${tpub}"
  }
}
```

Whenever a new message type is supported, please include a sample message file in the resources folder and update the method `MessageSampler.getSampleSndem(SndemType)`<sup>15</sup>.

## 5.11. Memory Monitor

The Memory Monitor mimics the heap status area in the Eclipse Integrated Development Environment for Java Developers [28] displaying the current memory usages. Figure 5.10(a) shows the memory monitor status panel in the Traffic Viewer application. Hovering over the status panel will display a tooltip window with yellow background indicated in Figure 5.10(b). The tooltip window lists the heap size and the current memory usages of the Java Virtual Machine (VM). Right clicking on the status panel will show a popup menu with four commands, shown in Figure 5.10(c), to interact with the memory monitor or to run Garbage Collector (GC):

1. Set Mark—marks the current heap usage by drawing a vertical line in the memory monitor area. The previous mark, if set, will be cleared. Left clicking on the memory monitor area will also perform this command.
2. Clear Mark—clears the previous mark, if set.
3. Show Max Heap—when selected (deselected by default), scales the width of the memory monitor area using the maximum heap size rather than the total heap size.
4. Run GC—runs the garbage collector in the Java VM.

---

<sup>15</sup> The fully qualified name of the class Message Sampler is `gov.nasa.sntb.trafficviewer.ui.usermessage.MessageSampler`.

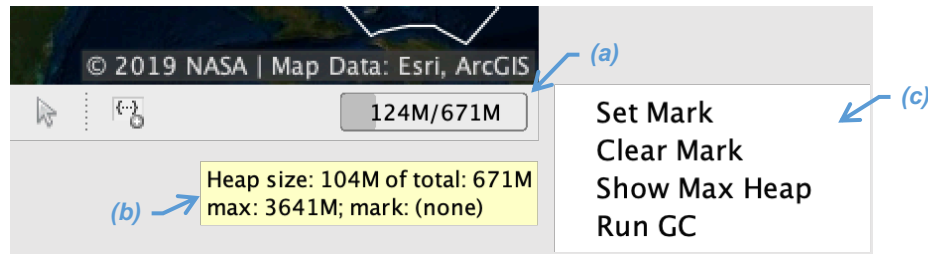


Figure 5.10. Memory monitor: (a) status panel, (b) tooltip window, and (c) popup menu

The memory monitor is useful for a user to see the memory usage of the application. If memory consumption continues to increase over time, it might indicate a memory leak in the application. In this case, it is important to identify and address the issue.

## 6. Shortcut Keys

Shortcut keys provide a faster way to perform action commands without accessing drop-down menus, pressing toolbar buttons, or clicking status bar buttons. Table 6.1 lists the supported shortcut keys in Traffic Viewer.

Table 6.1. Shortcut keys in Traffic Viewer

To Do This	Shortcut Keys	
	Mac OS X	Windows or Linux
Show/hide Help Dialog	?	Question Mark (?)
Close Dialog	Esc	Escape
Select a specific (the first to ninth) layer menu	1 through 9	1 through 9
Select the last layer menu	0	0
Make map tile images darker	[	Left square bracket ([)
Make map tile images brighter	]	Right square bracket (])
Move map north	↑	Up Arrow
Move map south	↓	Down Arrow
Move map west	←	Left Arrow
Move map east	→	Right Arrow
Move map north more	Page Up	Page Up
Move map south more	Page Down	Page Down
Move map west more	Home	Home
Move map east more	End	End
Select a specific (the first to ninth) status button	⇧ 1 through ⇧ 9	Shift + 1 through Shift + 9
Select the last status button	⇧ 0	Shift+0
Make track icons smaller	{	Left Curly Bracket ({)
Make track icons larger	}	Right Curly Bracket (})
Reset map to initial view	H	H
Zoom in map	=	Equal (=)
Zoom out map	-	Minus (-)

## 7. References

1. George, S., Satapathy, G., Manikonda, V., Palopo, K., Meyn, L., Lauderdale, T., Downs, M., Refai, M., and Dupee, R. (2011). "Build 8 of the Airspace Concept Evaluation System," AIAA Modeling and Simulation Technologies Conference, Portland, OR., AIAA-2011-6373.
2. Bilimoria, K. D., Sridhar, B., Grabbe, S. R., Chatterji, G. B., and Sheth, K. S. (2001) "FACET: Future ATM Concepts Evaluation Tool," Air Traffic Control Quarterly, Volume 9, Number 1.
3. Prevot, T. and Mercer, J. (2007) "MACS: A Simulation Platform for Today's and Tomorrow's Air Traffic Operations," AIAA-2007-6556, AIAA Modeling and Simulation Technologies (MST) Conference and Exhibit, Hilton Head, SC, 20-23 August 2007.
4. Chan, W., Barmore, B., Kibler, J., Lee, P., O'Connor, N., Palopo, K., Thippavong, D., and Zelinski, S., "Overview of NASA's Air Traffic Management - Exploration (ATM-X) Project," AIAA 2018-3363, 2018 AIAA Aviation Forum, AIAA Aviation Technology, Integration, and Operations Conference, Atlanta, GA, 25-29 June 2018.
5. Eshow, M. M., Lui, M., and Ranjan, S., (2014) "Architecture and Capabilities of a Data Warehouse for ATM Research," 2014 IEEE/AIAA 33<sup>rd</sup> Digital Avionics Systems Conference (DASC), Colorado Springs, CO, 2014, pp. 1E3-1-1E3-14.
6. "ActiveMQ" (Online). <https://activemq.apache.org/components/classic/download/>. Retrieved 2019/10/02.
7. "Git" (Online). <https://git-scm.com/>. Retrieved 2019/08/29.
8. "Gradle Build Tool" (Online). <https://gradle.org/>. Retrieved 2019/08/29.
9. "Java SE - Downloads | Oracle Technology Network | Oracle" (Online). <https://www.oracle.com/technetwork/java/javase/downloads/>. Oracle. Retrieved 2019/08/29.
10. "Introduction to the Standard Directory Layout" (Online). <https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html>. Retrieved 2019/11/27.
11. Lai, C. F. (2019), "Air Traffic Management TestBed Simulation Architect: User's Guide," NASA/TM-2019-220196.
12. "RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format" (Online). <https://tools.ietf.org/html/rfc8259>. Internet Engineering Task Force. Retrieved 2019/11/23.
13. "NASA WorldWind" (Online). <https://worldwind.arc.nasa.gov/>. National Aeronautics and Space Administration. Retrieved 2019/08/27.
14. "CesiumJS" (Online). <https://cesium.com/cesiumjs/>. Cesium. Retrieved 2019/12/18.
15. "Learning layer basics in Photoshop" (Online). <https://helpx.adobe.com/photoshop/using/layer-basics.html>. Adobe Inc. Retrieved 2019/08/27.
16. "WGS 84 and the Web Mercator Projection NGA Office of Geomatics" (Online). [http://earth-info.nga.mil/GandG/wgs84/web\\_mercator/%28U%29%20NGA\\_SIG\\_0011\\_1.0.0\\_WEBMERC.pdf](http://earth-info.nga.mil/GandG/wgs84/web_mercator/%28U%29%20NGA_SIG_0011_1.0.0_WEBMERC.pdf). National Geospatial Intelligence Agency. Retrieved 2019/08/26.
17. "Showing Pixels and Tile Coordinates" (Online). <https://developers.google.com/maps/documentation/javascript/examples/map-coordinates>. Google. Retrieved 2019/08/26.
18. Benjamin, S. G., Brown, J. M., Brundage, K. J., Schwartz, B. E., Smirnova, T. G., Smith, T. L., and Morone, L. L., "NWS Technical Procedures Bulletin 448; RUC-2-The Rapid Update Cycle Version 2," National Weather Service, Office of Meteorology, Silver Spring, MD, 1998.
19. "Contouring triangle meshes" (Online). [https://en.wikipedia.org/wiki/Marching\\_squares#Contouring\\_triangle\\_meshes](https://en.wikipedia.org/wiki/Marching_squares#Contouring_triangle_meshes). Wikipedia. Retrieved 2019/12/05.
20. "World Imagery (MapServer)" (Online). [https://services.arcgisonline.com/ArcGIS/rest/services/World\\_Imagery/MapServer](https://services.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer). ArcGIS Online. Retrieved 2019/12/26.

21. "Reference/World\_Boundaries\_and\_Places (MapServer)" (Online). [https://services.arcgisonline.com/ArcGIS/rest/services/Reference/World\\_Boundaries\\_and\\_Places/MapServer](https://services.arcgisonline.com/ArcGIS/rest/services/Reference/World_Boundaries_and_Places/MapServer). ArcGIS Online. Retrieved 2019/12/26.
22. "RFC 7946 - The GeoJSON Format" (Online). <https://tools.ietf.org/html/rfc7946>. Internet Engineering Task Force. Retrieved 2019/08/27.
23. "RFC 1952 - GZIP file format specification version 4.3" (Online). <https://tools.ietf.org/html/rfc1952>. Internet Engineering Task Force. Retrieved 2019/10/09.
24. "APPNOTE.TXT - .ZIP File Format Specification" (Online). <https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT>. Retrieved 2019/10/09.
25. "FAA Form 7233-1" (Online). [https://www.faa.gov/documentLibrary/media/Form/FAA\\_Form\\_7233-1\\_7\\_31\\_17.pdf](https://www.faa.gov/documentLibrary/media/Form/FAA_Form_7233-1_7_31_17.pdf). Federal Aviation Administration. Retrieved 2019/10/09.
26. "Aeronautical Data" (Online). [https://www.faa.gov/air\\_traffic/flight\\_info/aeronav/aero\\_data/](https://www.faa.gov/air_traffic/flight_info/aeronav/aero_data/). Federal Aviation Administration. Retrieved 2019/10/30.
27. Finkel, R. and Bentley, J. L., (1974) "Quad Trees: A Data Structure for Retrieval on Composite Keys," Acta Informatica. 4 (1): 1–9. doi:10.1007/BF00288933
28. "Eclipse IDE for Java Developers" (Online). <https://www.eclipse.org/downloads/packages/release/2019-09/r/eclipse-ide-java-developers>. Eclipse Foundation. Retrieved 2019/12/05.