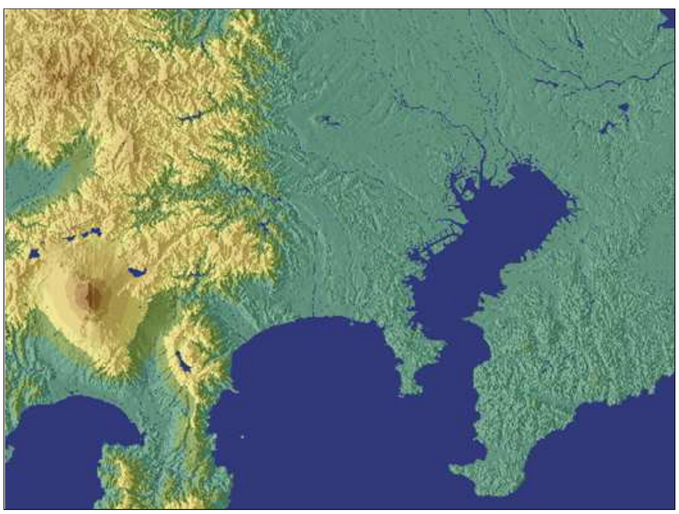
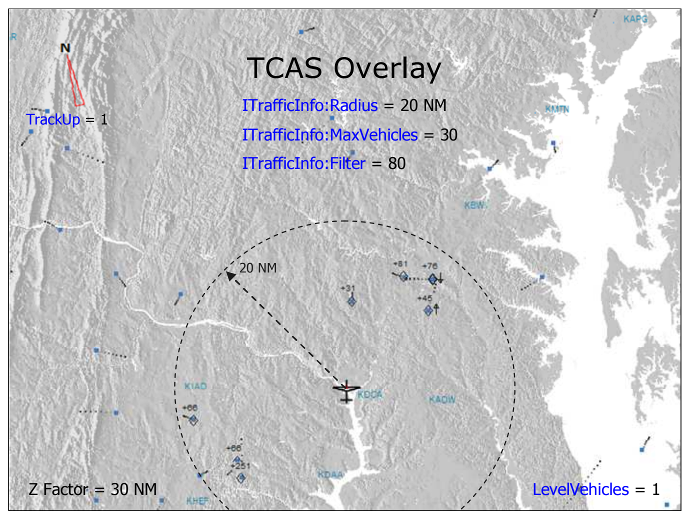
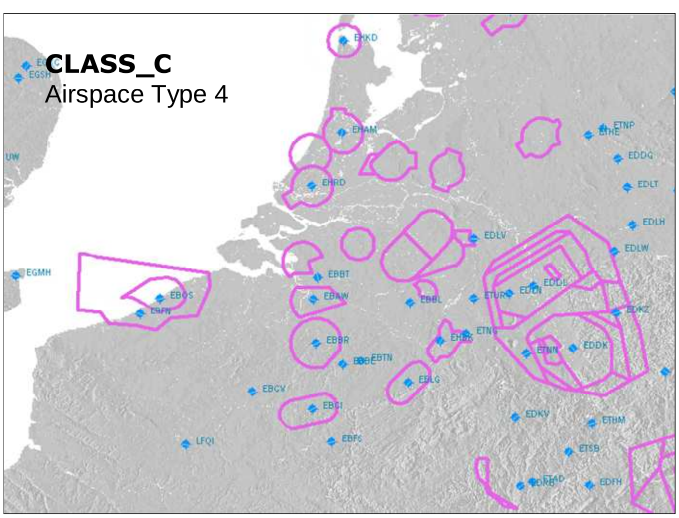
CustomDraw fs9gps:Map GUIDEBOOK v. 2.0.1

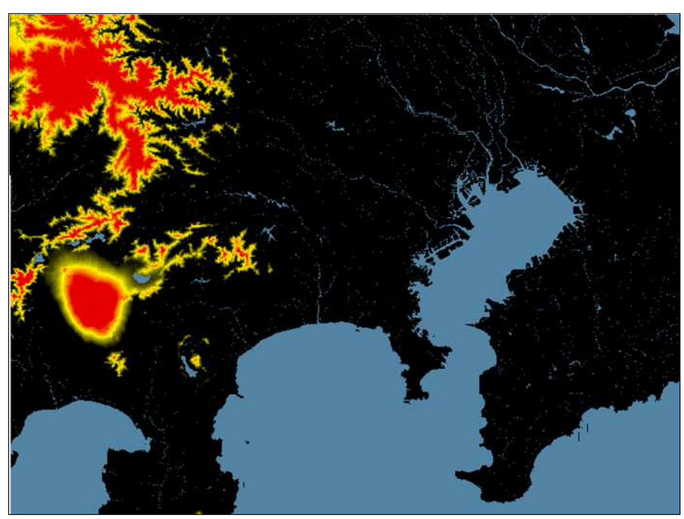
an empirical xml guide

Robert McElrath
July, 2015









Introduction

This is a guide for working with Flight Simulator's CustomDraw map function – the program that draws the map in the stock gps500 gauge. The purpose of the guidebook is to expand on Microsoft's SDK Moving Map documentation which is very brief so that inexperienced gauge programmers can get up and running more quickly.

The guidebook is written primarily with FSX in mind because FSX contains important additional mapping capabilities and related variables that are absent in FS9. However, an attempt has been made to note key differences between the two sims, for example, map projection scheme differences.

Almost all of the map variables are documented. However, there remain questions about a few and those are noted in the text. Additionally, as I have never used FSX race missions, there is no LayerRacePoints chapter yet.

In addition to the CustomDraw map variables, the following topics and map applications are discussed:

- ❑ XML gauge units vs. physical screen pixels
- ❑ Calibrating XML and CustomDraw map scales
- ❑ Transforming mouse X and Y into longitude and latitude
- ❑ Creating map overlays and coordinate rotation for TrackUp=1
- ❑ TCAS overlay using ITrafficInfo variables
- ❑ TAWS map
- ❑ Mouse click distance, bearing, latitude and longitude
- ❑ Nearest search centered on a mouse click point rather than aircraft position
- ❑ Adding a flight plan waypoint by mouse click
- ❑ Stationary Map vs. Moving Map

In my opinion, some interesting applications can be imagined when you calibrate XML and CustomDraw map scales and transform mouse X and Y into longitude and latitude.

I need to acknowledge the assistance of a few people; Tom Aguilo, and Robbie McElrath. Tom is the author of XMLVars (included in XMLTools), a variable handling module that I use to dynamically create XML variable arrays without which my rendition of TCAS is not possible. Robbie is the author of BlackBox and Logger, both of which were indispensable in the preparation of this guidebook. He also provided feedback on application of Affine transforms needed for coordinate rotation.

Finally, two fully functional XML gauges for use in FSX are available as download from the BlackBox website that demonstrate the applications mentioned above.

Bob McElrath
Bangkok, Thailand
July, 2015 (v.2.0.1)

Map Projections

Flight Simulator provides two different map systems having different projection schemes.

1. **Flight Planner Map and World Map:** FS9 and FSX Flight Planner and World maps both use the Equidistant Cylindrical, *Plate Carrée* projection (Flights → Flight Planner → Find Route and World → Map)
2. **CustomDraw fs9gps:Map:** FS9 – Sinusoidal Equal Area, Pseudocylindrical projection. FSX – Both Sinusoidal Equal Area and *Plate Carrée* projections. CustomDraw is the map engine for the moving map display used in the stock gps_500 and radar gauges and is the subject of this guidebook.

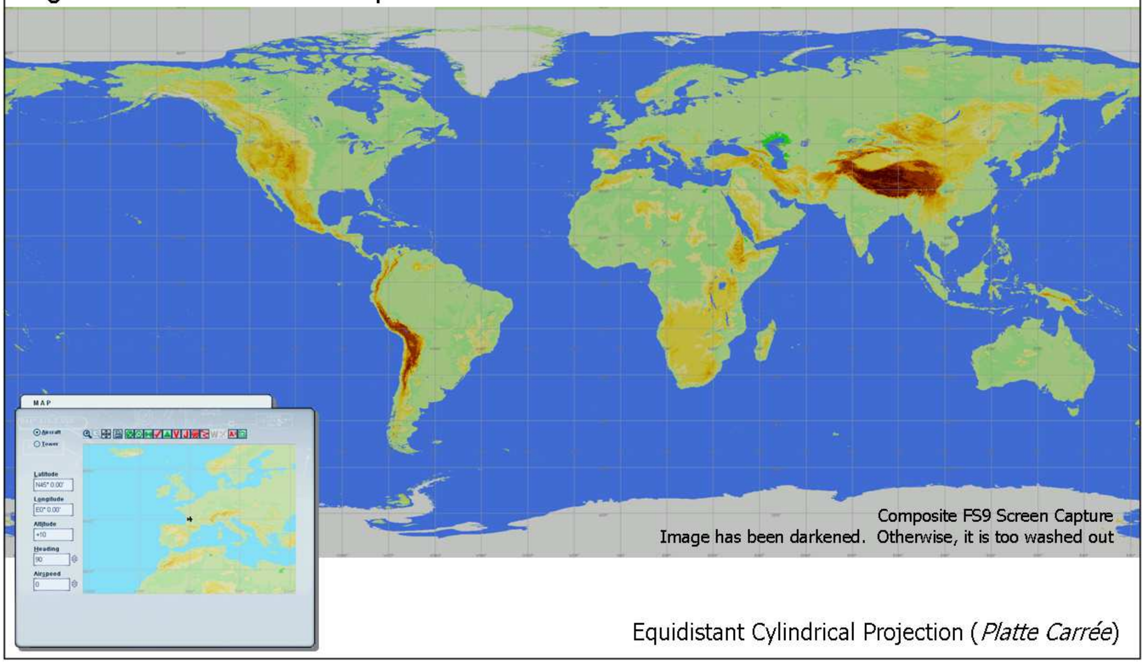
❑ Flight Planner and World Maps

Flight Planner and World maps of both FS9 and FSX incorporate an Equidistant Cylindrical, *Plate Carrée* projection. This projection is characterized by straight and orthogonal meridians (lines of constant longitude) and parallels (lines of constant latitude) producing square graticules (the lat-lon grid) and simple, computationally friendly equations. It is well suited for the easy panning around the globe and flight plan editing; North is Up, East is Right, and lat-lon position is simple to interpolate.

Its drawback is that east-west distances are progressively distorted as latitude increases toward the poles to the point where X-axis map scale becomes infinite at the poles. As shown on the next page, the high latitude distortion is very obvious when the map is zoomed out. Although useful and intuitive for general map reference, this projection system is poorly suited for navigation purposes required by a gps instrument because of the significant distance and angle distortions.

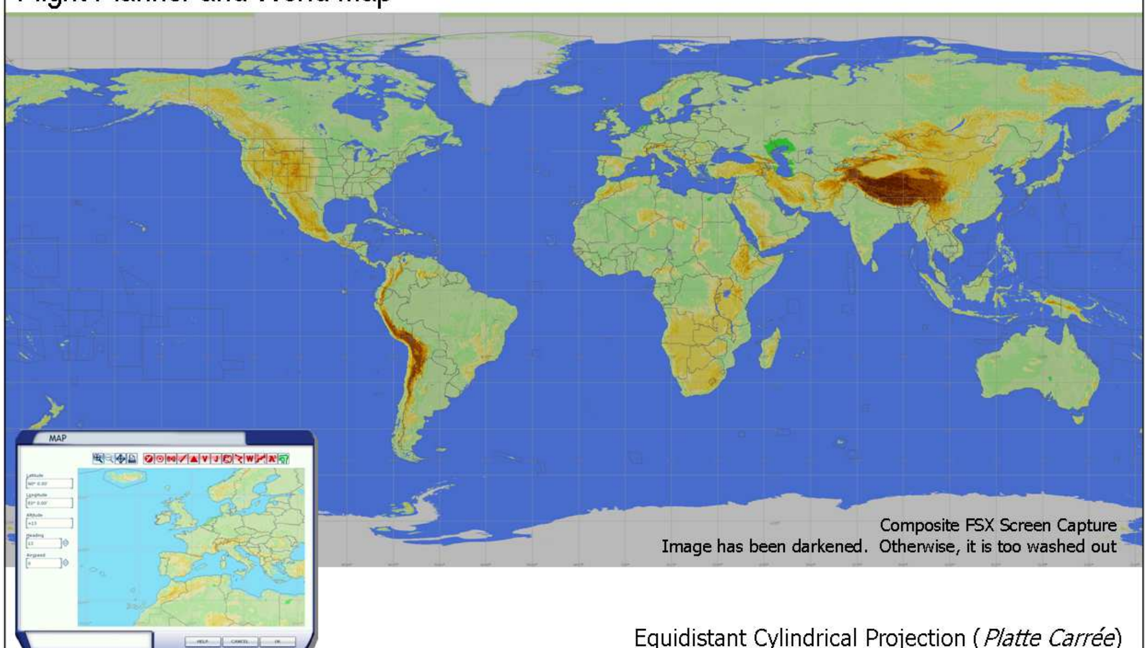
The images that follow are composite screen captures from the FS9 and FSX Flight Planner and World maps:

FS9
Flight Planner and World Map



Equidistant Cylindrical Projection (*Platte Carrée*)

FSX
Flight Planner and World Map



Equidistant Cylindrical Projection (*Platte Carrée*)

❑ CustomDraw fs9gps:Map

CustomDraw fs9gps:Map is Flight Simulators programmable map engine used in the stock gps gauges and in FSX radar applications. It is part of the gps.dll module. Map variables discussed in the SDK and this guide apply to the fs9gps:Map system.

In **FS9**, fs9gps:Map uses a Sinusoidal Projection scheme (a.k.a. Sansom-Flamsteed, Equal-Area Pseudocyclindrical, or Mercator Equal-Area Projection). Importantly, the Sinusoidal Projection is characterized by equal north-south and east-west map scales at all points on the globe. On the map as in reality, the length of each parallel is proportional to the cosine of the latitude, so real distance between meridians decreases toward the poles. The resulting shape of the earth is the region between two symmetric rotated cosine curves.

Sinusoidal Projections display shape correctly only along the central meridian and distort shape away from it. To mitigate this, the map can be "interrupted" by shifting the longitude of the central meridian and redrawing the map around the new central meridian. Flight Simulator incorporates interruption by *continuously* shifting the central meridian as the aircraft flies. The continuous shift is enabled when the <Longitude> variable is set to the aircraft longitude:

```
<Longitude> (A:PLANE LONGITUDE, radians) </Longitude>
```

This produces a very accurate map especially when zoomed in to the most common gps gauge operational ranges (200 NMiles or less).

Figure **A**, on the following page, is a composite screen shot of FS9's CustomDraw fs9gps:Map zoomed out to maximum Zoom. In this example, the central meridian is 90° West.

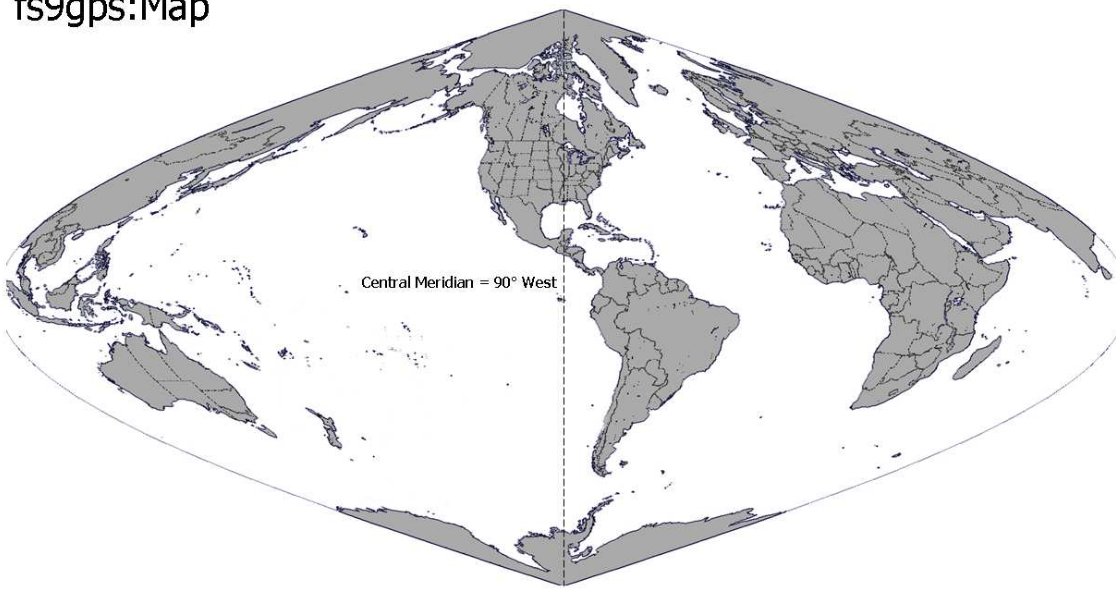
In **FSX**, the fs9gps:Map is a hybrid of Equal Area Sinusoidal and Equidistant Cylindrical projections that is a function of Zoom. At Ranges below 270 NM (Zoom less than 500,000 meters), FSX uses the Sinusoidal Projection like FS9. However, at Zoom >= 500,000 meters, it switches to the Equidistant Cylindrical projection as shown in the composite FSX fs9gps:Map screen capture in Figure **B**. A consequence of this switch is that the X-axis scale must be multiplied by the cosine of the latitude to yield correct east-west distances.

Distance distortion becomes so severe at high latitudes in this projection scheme that FSX reverts back to sinusoidal projection at latitudes greater than 70° North and South.

On Equidistant Cylindrical Projections, Range Rings are actually ellipses (except at the equator) because of the different X and Y axis scales. On Sinusoidal Projections, they are circles. Consequently, Range Rings (<LayerRangeRings>) should never be displayed in FSX at Zoom Factors of 270 NM and above.

FS9 CustomDraw
fs9gps:Map

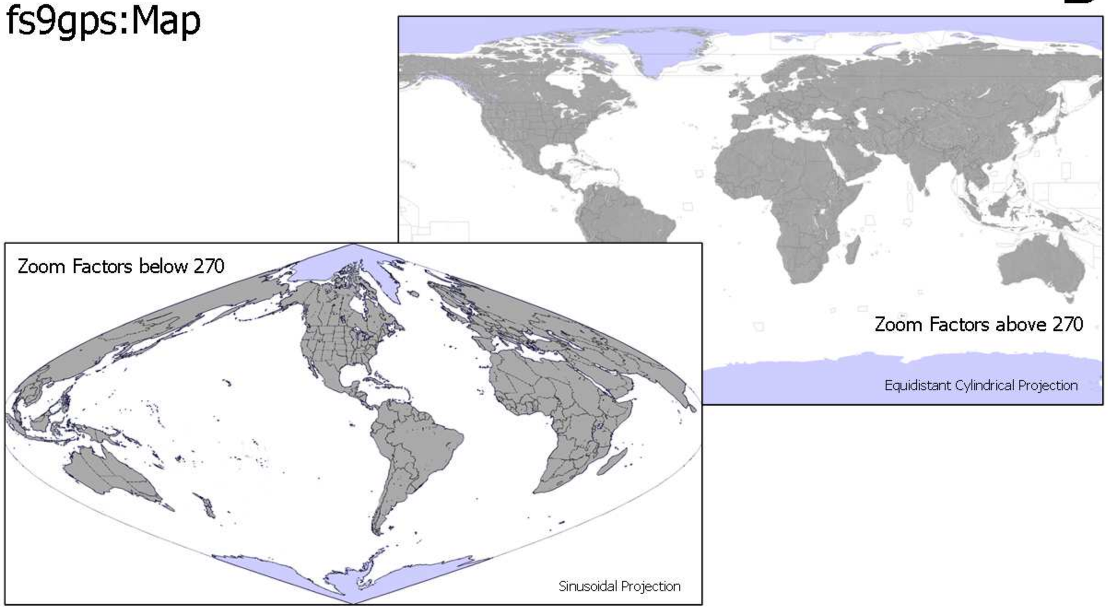
A



- ❑ Interrupted Sinusoidal Projection (*Equal Area Pseudocylindrical*): For all Zoom Factors

FSX CustomDraw
fs9gps:Map

B



- ❑ Equidistant Cylindrical Projection (*Platte Carrée*): Zoom Factors 270 NMiles (500 km) and above
- ❑ Interrupted Sinusoidal Projection similar to FS9: Zoom Factors below 270 NMiles (500 km)

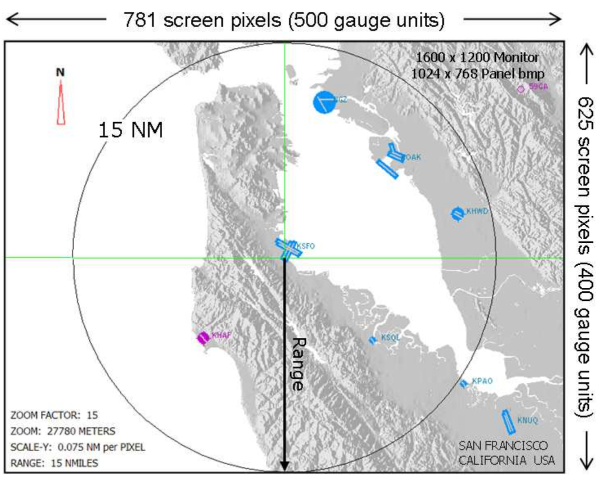
Range (Zoom Factor)

Zoom (meters) = ZoomFactor (NMiles) x 1852 (meters / NMile)
ZoomFactor (NMiles) = Range (NMiles)

Range is the radius of the biggest complete circle that can be drawn within the boundaries of the map, as demonstrated below. As an example, at a Zoom Factor of 15, or a Zoom of 15 x 1852 = 27780 meters, Range is 15 NMiles and the map covers 30 NMiles (2 times the Range) in the short direction.

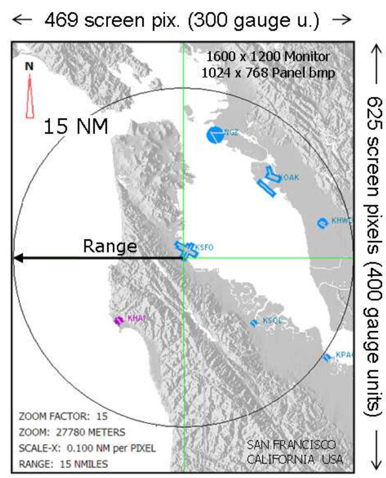
Note that Flight Simulator automatically draws the map to fit 2 times Range or Zoom Factor into the short side of the map display.

Technically, this is the shortest side as *measured in screen pixels*, not gauge units.




```
<CustomDraw Name="fs9gps:Map" X="500" Y="400">
<Zoom> (@g:map_ZoomFactor) 1852 * </Zoom>
```

ZOOM FACTOR: 15
ZOOM: 27780 METERS
SCALE-Y: 0.075 NM per GAUGE PIXEL-Y
RANGE: 15 NMILES




```
<CustomDraw Name="fs9gps:Map" X="300" Y="400">
<Zoom> (@g:map_ZoomFactor) 1852 * </Zoom>
```

ZOOM FACTOR: 15
ZOOM: 27780 METERS
SCALE-X: 0.100 NM per GAUGE PIXEL-X
RANGE: 15 NMILES

In the map on the left, the short, Y-axis scale is, by definition, 30 NMiles per 400 gauge units, or 0.075 NMiles per gauge unit-Y. For the right side map, the short, X-axis scale is, by definition, 30 NMiles per 300 gauge units, or 0.100 NMiles per gauge unit-X.

It is important to note that while a key property of Sinusoidal Projections is equal X and Y axis scales, when it comes to the screen display the long axis scale will <u>not necessarily</u> equal the short axis scale if distances have been measured using *gauge units*. The reason is that the shape of the gauge unit displayed is often rectangular rather than square - its aspect ratio is not 1:1. This has significant impact whenever the mouse or a movable cursor based on mouse (M:X), (M:Y) reference is used on fs9gps:Map for distance or location measurement, as discussed in the following section.

Screen Pixels vs. Gauge Units

Combining XML Objects with CustomDraw Map

If a gauge programmer wants to combine their own map applications such as a custom XML moving map overlay on the CustomDraw Map terrain base, or create the ability to click anywhere on the map to retrieve latitude and longitude, distance and bearing, then the difference between screen pixel and gauge measurement unit (gauge unit) aspect ratios a the transform function between the two must be understood and applied.

CustomDraw Map is measured in screen pixels, XML gauge applications are measured in gauge units, and the two are not the same.

The table below summarizes some of the things that can be accomplished using XML script with standard gps and CustomDraw variables.

Mouse Click Information	XML Overlay	Overlay Is Difficult
• Latitude and Longitude	• Symbols:	• Airspace
• Distance and Bearing	• Airports, Intersections	• Approach
• Add Flight Plan Waypoints	• VORs, NDBs, ILSs	• Borders
• Initiate Nearest Searches	• User Aircraft	• Grid (sinusoidal proj.)
• Frequencies	• Enroute Flight Plan	• Airways
• Runways	• TCAS Map – Air Traffic	
• Services	• User Defined Points	
• Facilities	• Stationary Map	

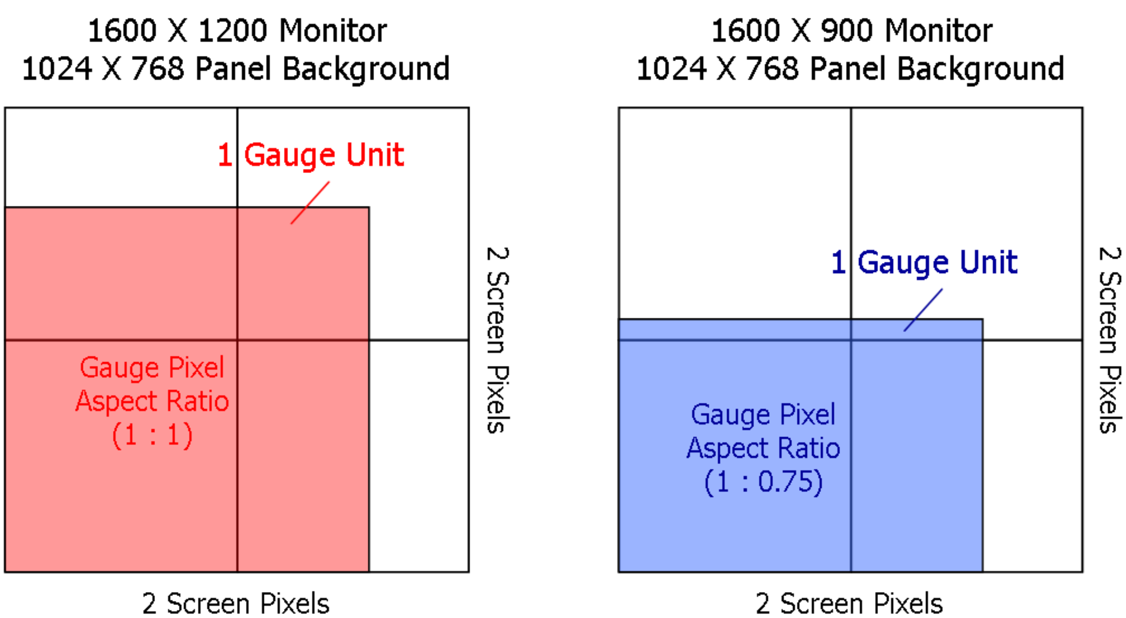
In general, moving map overlays of single point objects such as facilities or air traffic can be accomplished. Shapes like Airspace or line segments such as Airways that require data base access not available from a query of Flight Simulator's gps database are beyond the scope of this discussion.

While the applications above may be *possible* within the XML world, in my opinion it's not practical to replace most CustomDraw Map layers. A notable exception may be Traffic. The CustomDraw LayerVehicles was designed for an ATC Controller radar screen view, but it doesn't produce the best looking TCAS gauge display. However, even a TCAS II v7.1 system can be modeled using ITrafficInfo group variables, an overlay with custom symbols, XML script to identify the Traffic Alerts, and XML to replicate the v7.1 Resolution Advisories.

Screen pixels vs. Gauge units

Map Scale is the ratio of real distance to map display distance. For CustomDraw Map, the scale units are NMiles or meters per physical screen pixel. On the other hand, the XML Mouse parameters (M:X) and (M:Y), measure *gauge units*, <u>not</u> screen pixels. As demonstrated below, when a gauge unit is used to measure distance between two screen pixels, then:

- ❑ The number of gauge units will not necessarily equal the number of screen pixels – usually not, in fact, because the panel background image is usually not at the same aspect ratio as the screen. Panel background images of stock FS9 and FSX aircraft are 1024 X 768 pixels (4:3 ratio), but monitor screens vary: 1600 X 1200 (4:3), 1600 X 900 (4:2.25), 1920 X 1080 (4:2.25), 1280 X 1024 (4:3.2), etc.

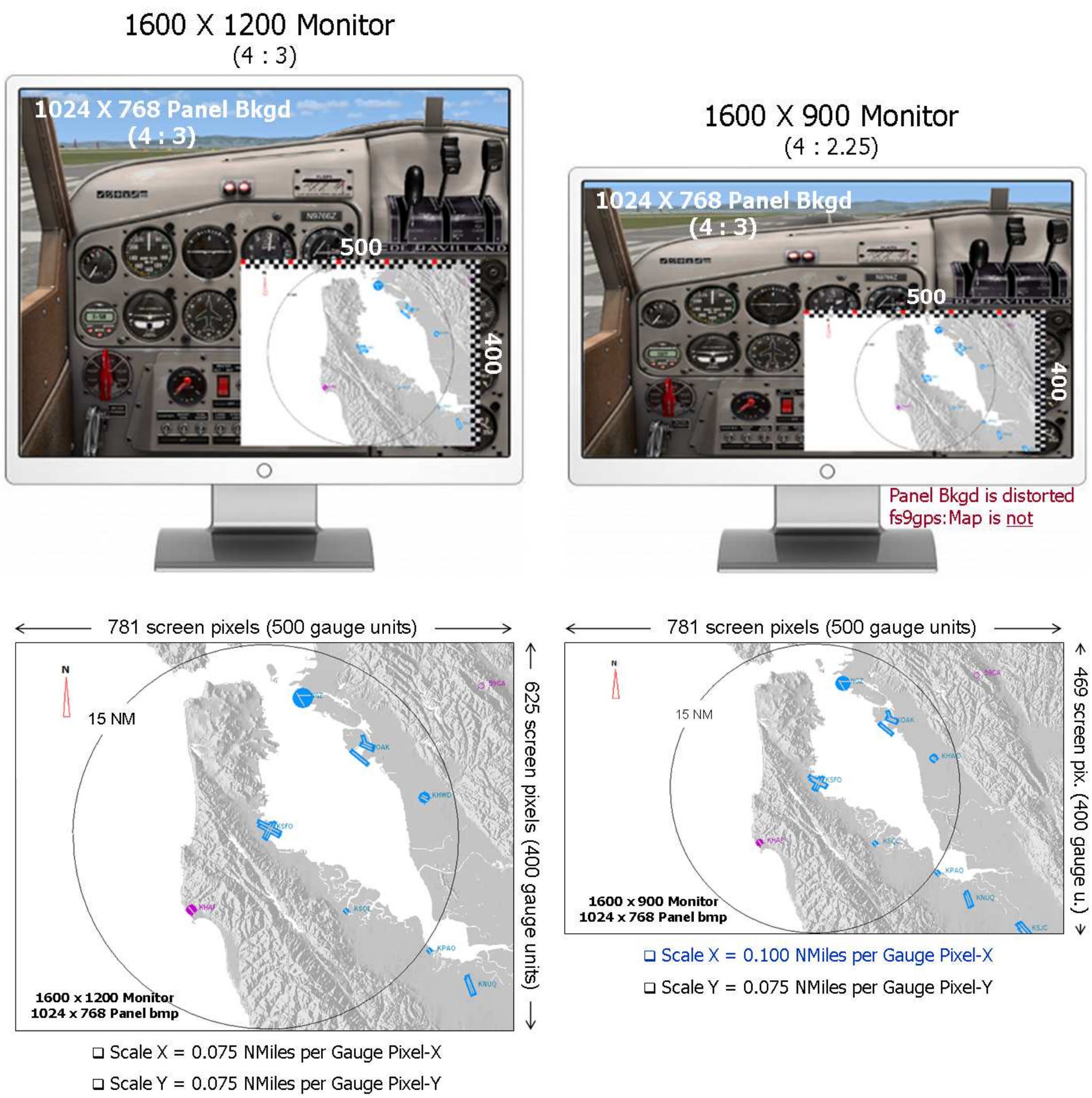


Panel Background Image may be distorted but CustomDraw Map is not

In the figures that follow, an FSX fs9gps:Map view of the San Francisco California, USA peninsula is shown as displayed on a 1600 x 1200 pixel screen and on a 1600 x 900 pixel screen. In both cases, the panel background bitmap image is the stock FS9 & FSX 1024 x 768 pixels, and the CustomDraw map size is 500 x 400 gauge units. The Zoom is low (Zoom less than 500 km) so the Sinusoidal projection is used:

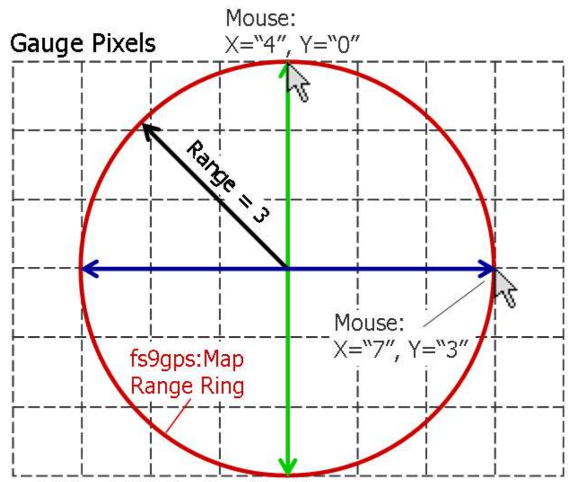
- ❑ The relative distances, angles, areas and shapes of the two map images rendered by CustomDraw are identical. There is no distortion of map elements between the two images. The 15 NM Range Ring is perfectly circular on both images. The panel background image and the *shape* of the 500 X 400 gauge unit map area may be stretched on different screens, but the map rendered by fs9gps:Map is never stretched or distorted.

- In other words, the fs9gps:Map engine is independent of both screen and panel background image resolutions, and it internally applies sinusoidal projection (zoom dependent in FSX) with equal X and Y axis scales. As rendered on the screen, all sinusoidal projection fs9gps:Maps have equal X and Y scales *as measured in screen pixels.*
- The only difference is that more map image is displayed in the east-west direction on the 1600 x 900 screen due to the different aspect ratio of that monitor.
- In each image, the short axis (Y axis) scale as <u>measured in gauge units</u> is the same. By definition, it is 30 NMiles per 400 gauge units (Zoom Factor = 15), or 0.075 NM per gauge unit-Y.
- The long axis (X axis) scale, <u>measured in gauge units-X</u>, is different between the two images. The reason is the aspect ratio of the screens, and consequently, the aspect ratios of the gauge units, are not the same.



❑ Three examples of Panel Background Image stretch

The following 3 figures demonstrate stretch of the panel background image on different monitors and in different view modes. Each produces a different gauge unit aspect ratio that must be accounted for if using the mouse to measure distance on the CustomDraw map or creating overlays for the CustomDraw map where the lat/lon of the point to be displayed must be correctly translated into gauge units.

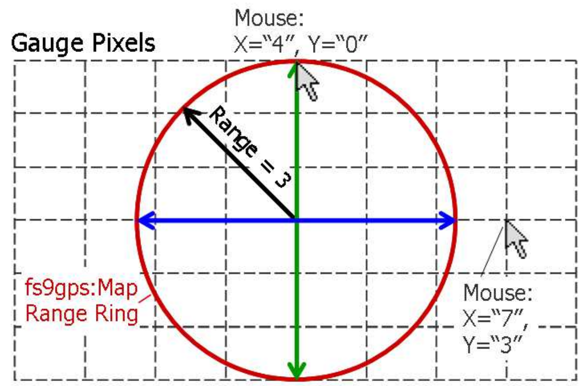


- ❑ 1600 x 1200 Monitor (4 : 3)
- ❑ 1024 x 768 Background (4 : 3)
- ❑ Full Screen View
- ❑ Square Gauge Pixels

1. 1600 x 1200 screen and 1024 x 768 panel background

In the figure above, both the screen and the panel background image have the same aspect ratio so there is no distortion of gauge units when the background panel bitmap image is enlarged to fill the screen.

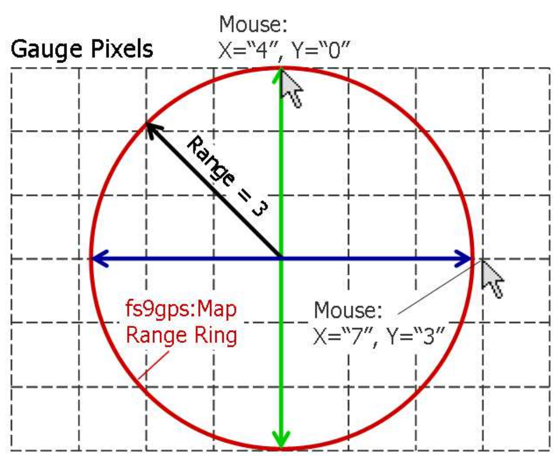
In this configuration, the short and long axis scales measured in gauge units are identical and, in the cartoon example, a mouse click at X="4", Y="0" is at a Range value of 3 and a mouse click at X="7", Y="3" is also at a Range value of 3. This is the simplest situation.



- ❑ 1600 x 900 Monitor (4 : 2.25)
- ❑ 1024 x 768 Background (4 : 3)
- ❑ Full Screen View
- ❑ Rectangular Gauge Pixels

2. 1600 x 900 screen and 1024 x 768 panel background

On the 1600 x 900 screen above, the panel background bitmap image is stretched to fill the screen and the gauge unit shape becomes elongated as demonstrated in the cartoon. A mouse click at X="4", Y="0" is still at a Range value of 3, but a mouse click at X="7", Y="3" is at a point on the map further than Range = 3. The short and long axis scales measured in gauge units are no longer equal; now a mouse click at X="6.25", Y="3" is at Range = 3.



- ❑ 1600 x 1200 Monitor (4 : 3)
- ❑ 1024 x 768 Background (4 : 3)
- ❑ Windowed View
- ❑ Slightly Rectangular Gauge Pixels

3. Windowed vs. Full Screen View mode

Even the subtle change of switching from Full Screen View to Windowed View (i.e., non-Full Screen View) affects map scales measured in gauge units because the background image is compressed to make room for the FS Menu, the Windows Task bar, and a one screen pixel black frame (in FS9). This changes the gauge unit aspect ratio which changes the XML map scales.

CustomDraw Map variables

CustomDraw Map Variables

Name, **X**, **Y**, and **Bright** must be placed in the CustomDraw start tag, they cannot be scripted as child elements like the rest of the fs9gps:Map variables. Name, X, and Y are mandatory. Bright is optional.

```
<CustomDraw Name="fs9gps:1:Map" X="275" Y="230" Bright="1">
```

- **Name="fs9gps:1:Map".** fs9gps:Map refers to the code that generates the map display. From the SDK: the ":1" is unnecessary if the panel in which the map is to appear has only one map. Otherwise use ":1", ":2" and so on to distinguish the different maps.

- **X="275" Y="230".** X and Y are the horizontal and vertical dimensions of the map display, measured in gauge units (gauge "pixels") not in monitor or screen pixels.

  As an example, the dimensions of the map display of the stock FS9 and FSX gps_500.xml gauges are 275 x 230 gauge units. Refer to line 759 of the FSX gps_500.xml gauge.

- **Bright="1".** Set to "1", "Yes", or "True" if the map remains at its normal brightness at dawn, dusk and night times of the day, otherwise it will be darkened.

The remaining fs9gps:Map variables discussed below as well as the Layer variables covered in subsequent chapters ***can all be scripted as child elements*** of the CustomDraw element.

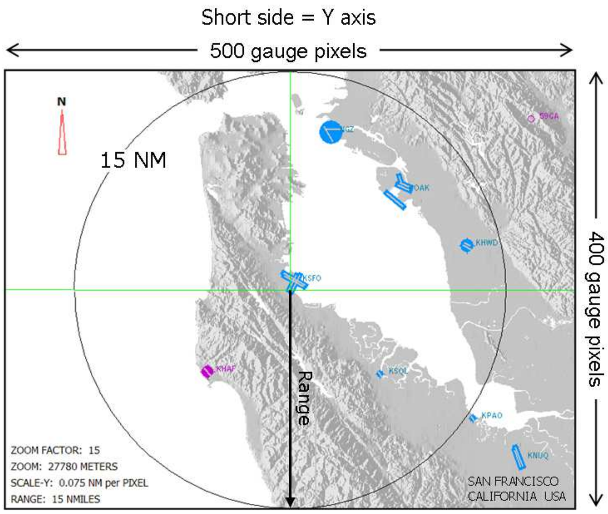
- **Zoom (meters, number)**

Zoom changes the apparent distance of the observer (pilot) to the ground surface shown in the map by changing map scales and area displayed as Zoom changes. It is a standard zoom definition - zoom in (smaller Zoom values) to see more detail, zoom out (larger Zoom values) to see more area.

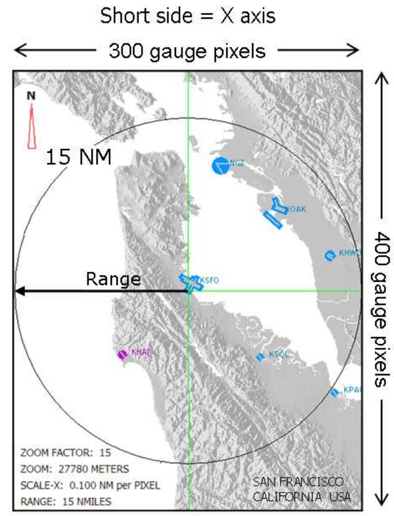
Zoom limits are 80 to 5,000,000 meters in FSX and 100 to 5,000,000 meters in FS9.

The terms Zoom and Range are sometimes used interchangeably, but the fs9gps:Map variable name is Zoom.

On the map, Zoom or Range represents one-half the distance of the short side of the map display as shown on the next page. Flight Simulator automatically draws the map to fit 2 times Range or Zoom into the short side of the map display.



<CustomDraw Name="fs9gps:Map" X="500" Y="400">
<Zoom> (@g:map_ZoomFactor) 1852 * </Zoom>
ZOOM FACTOR: 15
ZOOM: 27780 METERS
SCALE-Y: 0.075 NM per GAUGE PIXEL-Y
RANGE: 15 NMILES



<CustomDraw Name="fs9gps:Map" X="300" Y="400">
<Zoom> (@g:map_ZoomFactor) 1852 * </Zoom>
ZOOM FACTOR: 15
ZOOM: 27780 METERS
SCALE-X: 0.100 NM per GAUGE PIXEL-X
RANGE: 15 NMILES

The term Zoom Factor is defined in the stock gps_500 gauge to represent NMiles instead of the default units, meters. If the user wants a range of 15 NMiles, then the following XML can be used:

<Zoom> 27780 </Zoom> or

<Zoom> 15 1852.0 * </Zoom> or

<Zoom> (L:ZoomFactor, number) 1852.0 * </Zoom> or

<Zoom> (@g:map_ZoomFactor) 1852.0 * </Zoom>

where (L:ZoomFactor, number) and (@g:map_ZoomFactor) values equal 15. The constant, 1852.0, is the number of meters per Nautical Mile, and provides the conversion to NMiles.

- **Latitude**
- **Longitude (radians, number)**

Latitude and longitude of the center of the map, in radians. Usually, this is the aircraft position which can be defined as:

```
<Latitude> (A:GPS POSITION LAT, radians) </Latitude>
<Longitude> (A:GPS POSITION LON, radians) </Longitude>
```
or
```
<Latitude> (A:PLANE LATITUDE, radians) </Latitude>
<Longitude> (A:PLANE LONGITUDE, radians) </Longitude>
```

A:GPS POSITION LAT and LON are a good choice because they are updated every one second, consistent with other map related gps variables. They are the lat lon choice of the stock gps 500 XML gauge provided in Flight Simulator. In some applications such as a TCAS system, however, A:PLANE LATITUDE and LONGITUDE are preferred because these are updated every gauge update cycle.

In a Multiplayer ATC Controller session, the radar screen (map) can be centered on any fixed location. For the control tower at Johannesburg Intl. Airport (FAJS), Republic of South Africa (Lat: S26° 8.31093", Lon: E028° 15.08110"), the XML would be:

```
<Latitude> -26.138516 dgrd </Latitude>
<Longitude> 28.251352 dgrd </Longitude>
```
or
```
<Latitude> -0.4562032 </Latitude>
<Longitude> 0.4930791 </Longitude>
```

However, the normal practice is to select the airport in the Multiplayer set-up screen. FS will automatically load the control tower lat/lon.

- ❑ **Heading (radians)**

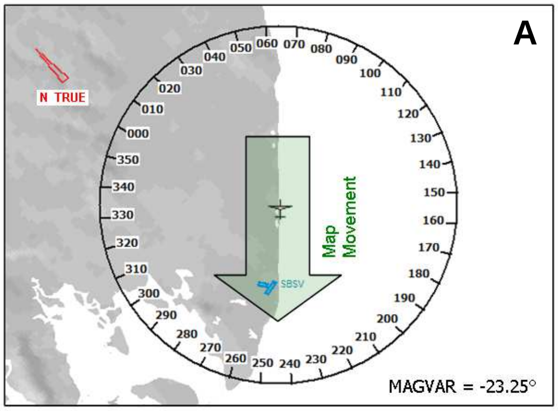
Heading determines the orientation and direction of movement of the map when the aircraft is in flight, and when TrackUp = 1.

Whether True or Magnetic Heading or even a fixed orientation is specified is a matter of preference. In actual Garmin GPS/GNS 400 and 500 Series and G1000 units, setup configurations accommodate True or Magnetic tracks or User defined orientation. In the stock Flight Simulator gps_500.xml gauge and G1000 MFD xml gauges, Heading is prescribed as **TRUE**, which is the usual preference, but could be changed if desired. Refer to line 764 in the gps_500.xml gauge or line 2706 in the MFD_Baron.xml gauge.

```
<Heading> (A:GPS GROUND TRUE TRACK, radians) </Heading>
```

(A:PLANE HEADING DEGREES TRUE, radians) is not a good variable to be used for Heading because in a cross-wind, the aircraft heading and ground track differ, but ground track is what is needed. The stock gps 500 XML gauge provided in Flight Simulator uses (A:GPS GROUND TRUE TRACK, radians).

Heading Examples



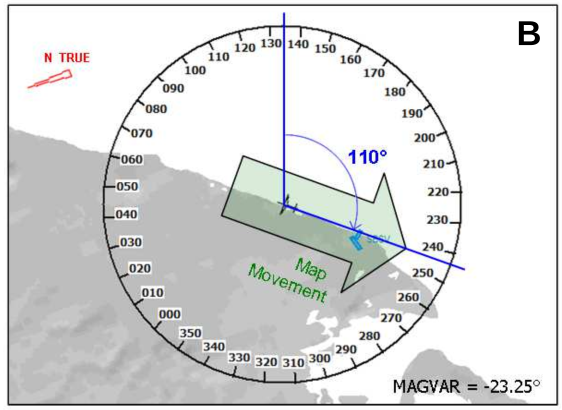
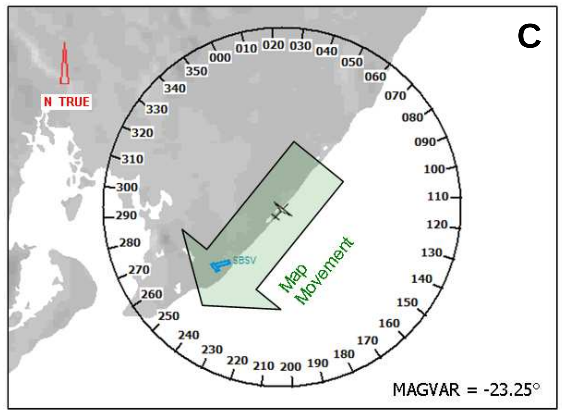


Fig **A**: Heading =
(A:GPS GROUND TRUE TRACK, radians)
TrackUp = 1

Fig **B**: Heading = 110 dgrd (constant)
TrackUp = 1

Fig **C**: Heading =
(A:GPS GROUND TRUE TRACK, radians)
TrackUp = 0



The figures above depict an aircraft flying northeast along the coast after departure from Deputado Luís Eduardo Magalhães International Airport (SBSV), Salvador da Bahia, Brazil. Map size is 50 x 62.5 NM.

In Figure **A**, Heading is set to the True ground track, and TrackUp is 1. The ground track that the aircraft is making always points up, to the top of the map. This is the normal configuration. Map movement is always 180 degrees from the ground track.

In Figure **B**, Heading is set to a constant 110 degrees:

```
<Heading> 110 dgrd </Heading>
```

In Figure **C**, TrackUp is not set to 1. In this event, the map is always oriented with the top, or up, towards true North regardless of the Heading value.

The compass rose of all three maps is oriented to magnetic North, consistent with the aircraft's DG or HSI compass.

Also note that in fs9gps:Map, the map surface moves and the aircraft cursor remains fixed. There is no capability using the Map variables for the aircraft to move across a fixed map view although this can be accomplished through use of an overlay as explained in the Example XML Maps chapter (and an XML gauge with this capability is available for download).

- ❑ **TrackUp (bool)**

TrackUp determines whether Heading (the aircraft ground track or other specified direction) or true North points up, toward the top of the map.

- TrackUp = 0. The top of the map, up, points toward true North
- TrackUp = any value other than zero. The direction determined by Heading points up, to the top of the map

- ❑ **CenterX**
- ❑ **CenterY (gauge units, number)**

CenterX and CenterY define the position on the map display where the map "center" is located. The map Center serves are the position of the aircraft or of the control tower in a ATC Controller session (using stock FSX radar.xml). CenterX and CenterY are gauge units measured from the upper left corner of the map.

- ❑ **SelectedVehicle (enum) FSX Only**

SelectedVehicle is a variable in the ITrafficInfo group that is useful when fs9gps:Map is set up as an ATC radar screen. It is the index pointer used to select a specific aircraft from the ITrafficInfo list in order to highlight its movement in contrast to all other aircraft on the radar screen, or to keep specific record of any flight variables associated with this aircraft. The aircraft must be included in the ITrafficInfo search results in order to be selected/highlighted. Only one aircraft can be Selected at a time.

Refer to the ITrafficInfo group chapter for further detail.

- ❑ **TagPosition (enum) FSX Only**

TagPosition is an index associated with the LayerVehicles group that controls placement of the aircraft flight status information label (TextDetailLayerVehicles) for the Selected aircraft on the radar screen. Its purpose is to help make the Selected aircraft information tag easier to see by moving its position relative to all the other aircraft information tags.

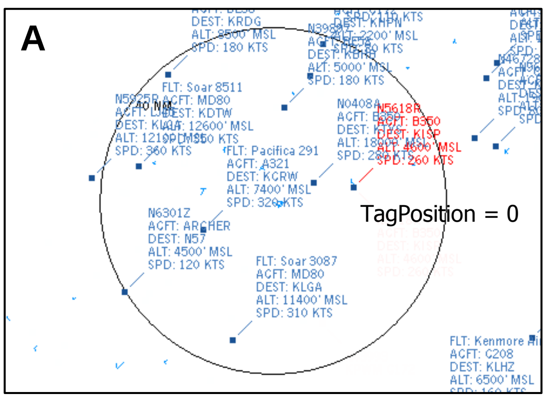
Tag placement relative to the aircraft symbol is as follows:

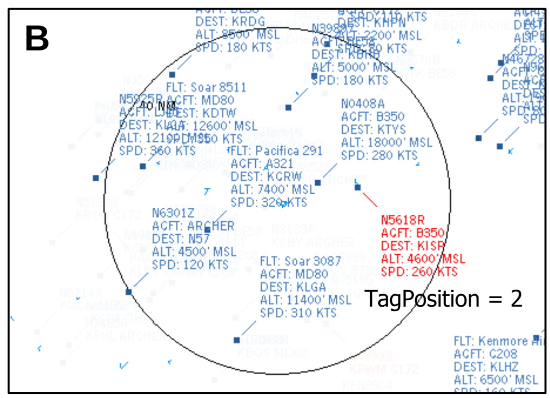
0 = UPPER_RIGHT (Default)	4 = LOWER_LEFT
1 = RIGHT	5 = LEFT
2 = LOWER_RIGHT	6 = UPPER_LEFT
3 = BOTTOM	7 = TOP

An aircraft must first be Selected and then a new TagPosition can be set if desired.

As an example, the radar screen images below show airborne AI traffic with flight information labels (TextDetailLayerVehicles = 2) displayed in the default Upper Right position (TagPosition = 0. See Fig **A**).

The table shows ITrafficInfo search results and N5618R, a very fast Beech King Air 350, has been Selected. Even though the Selected aircraft label always turns red, it is still a little difficult to read. Subsequently, its TagPosition was set to 2 = Lower Right, as shown in Figure **B,** and can be read more clearly in that position.





IDX	CALL	MODEL	DIST	VID	LAT	LON	ALT	VSI	GND	HDG	GSPD
0	N0408A	B350	10.9	144	40.725	-73.600	17997	-28	0	256	260
1	PAC291	A321	17.7	21	40.542	-74.167	7365	2515	0	265	300
2	N5618R	B350	18.9	137	40.707	-73.401	4559	-1610	0	109	264
3	N39897	BE58	22.0	77	41.014	-73.750	5001	1	0	61	179
4	N7727B	C172	30.2	12	41.133	-73.617	2243	-178	0	245	83
5	SOA8511	MD80	32.3	102	40.787	-74.491	12640	2733	0	298	331
6	SOA3087	MD80	33.5	24	40.116	-74.015	11370	-1872	0	45	323
7	N98956	C172	38.0	38	41.254	-73.554	5499	43	0	99	121
8	N21742	BE58	38.3	47	41.139	-74.345	8497	-18	0	252	166

The XML for this sequence is:

```
2 (>C:ITrafficInfo:SelectedVehicle)

<TagPosition> 2 </TagPosition>

(C:ITrafficInfo:SelectedVehicleID) (>C:fs9gps:SelectedVehicle)
```

Further discussion of the XML involved can be found in the ITrafficInfo group chapter.

❑ Map Object Color Syntax

Map object color must be specified using a composite hexadecimal number representing **B**lue-**G**reen-**R**ed shades, or the decimal equivalent of that composite hex. Example:

Blue: **107** Green: **27** Red: **137** BGR Hex: **0x6B1B89**

Blue shade = 107 ;	hex = **6B**	Composite BGR Hex: **0x6B1B89**
Green shade = 27 ;	hex = **1B**	Decimal equivalent = **7019401**
Red shade = 137 ;	hex = **89**	(6B1B89 hex = 7019401 decimal)

Alternatively, the decimal number **7019401** can be used in place of **0x6B1B89**.

Map object color differs from Text color which may be specified using either Windows color names***** like "blue" or "yellow" or a composite hex in the **RGB** form of "#891B6B".

* https://msdn.microsoft.com/en-us/library/aa358802(v=vs.85).aspx

❑ BackgroundColor (BGR hexadecimal) FSX Only

BackgroundColor is the background color of the map when LayerTerrain = 0. It is also the color of land when DetailLayerTerrain = 1 (Water Only). If BackgroundColor is omitted from the script, the default color is black.

In FS9, the background color cannot be selected. Whenever LayerTerrain = 0, the background color is always black in FS9.

❑ IceColor (BGR hexadecimal) FSX Only

IceColor is the color of the land surface when it is ice. If IceColor is omitted from the script, the default color is a light gray:

Blue: **222** Green: **222** Red: **222** BGR Hex: **0xDEDEDE**

In FS9, IceColor cannot be changed. It is always light gray **0xDEDEDE** like the FSX default.

❑ WaterColor (BGR hexadecimal) FSX Only

WaterColor is the color of water surfaces: oceans, lakes, and rivers. If WaterColor is omitted from the script, the default color is a light blue:

Blue: **247** Green: **222** Red: **132** BGR Hex: **0xF7DE84**

In FS9, WaterColor cannot be changed. It is always light blue **0xF7DE84** like the FSX default.

❑ ElevationXColor (BGR hexadecimal) FSX Only

ElevationXColor determines the terrain color applied between specified elevations.

The number in the variable name defines the **top** elevation on which the color will be applied. The units are feet and are not changed even if FS settings are set to metric (Options > Settings > General > Unit of measure > Metric).

For example, the xml:

```
<Elevation3000Color> 0x73C3C8 </Elevation3000Color>
```

produces a tan elevation color **0x73C3C8** between 2000 ft and 3000 ft elevation:

Blue: **115** Green: **195** Red: **200** BGR Hex: **0x73C3C8**

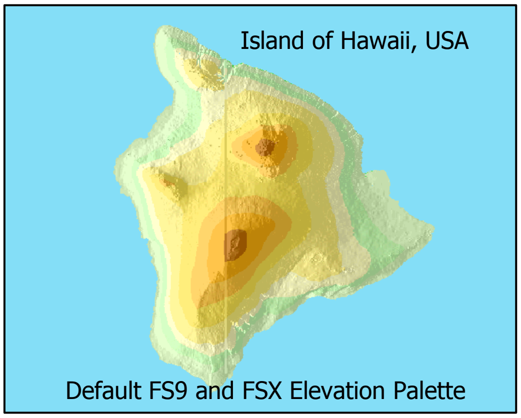
There 18 ElevationXColor variables representing 1000 ft elevation bands from -1000 ft to 17000 ft+ msl. The even 1000 ft interval is fixed and cannot be changed.

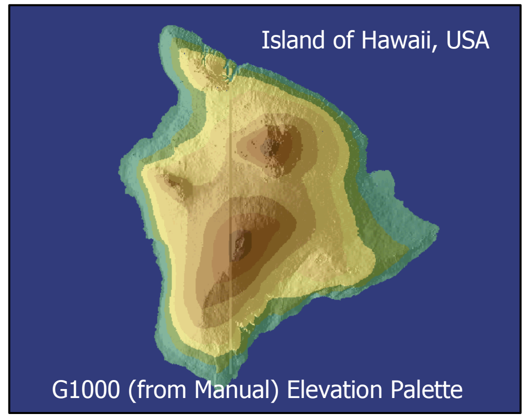
- Elevation0Color = Color between -1000 and 0 feet elevation
- Elevation1000Color = Color between 0 and 1000 feet elevation
- Elevation2000Color = Color between 1000 and 2000 feet elevation
- Elevation3000Color = Color between 2000 and 3000 feet elevation
- … et cetera …
- Elevation16000Color = Color between 15000 and 16000 feet elevation

The 1000 ft interval is similar for Elevation0Color through Elevation16000Color. Elevation17000Color is slightly different however:

- Elevation17000Color = Color 16000 feet and greater

The maps below show the default FS9 and FSX elevation colors on the left (no ElevationXColor variables used) and the Garmin 1000 MFD colors using ElevationXColor variables on the right. Refer to LayerTerrain chapter for additional discussion.





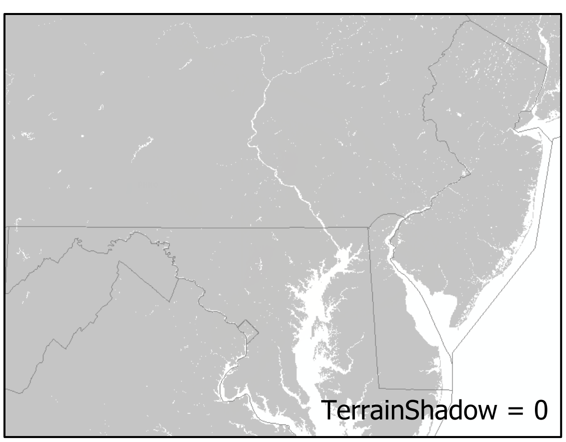
Note that TerrainShadow is enabled above which slightly changes color brightness due to illumination and shadow effects.

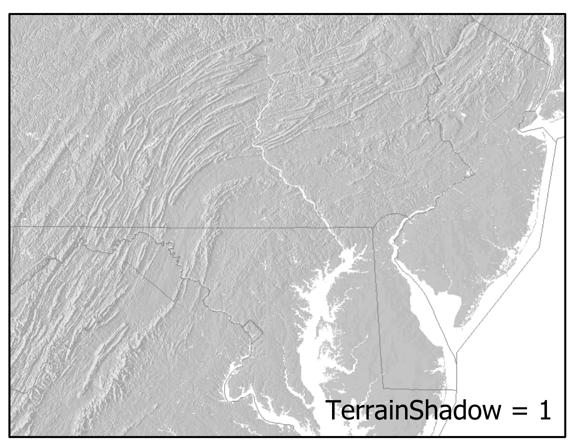
❑ TerrainShadow (bool) FSX Only

TerrainShadow highlights topography by illuminating it from compass direction (True) West. The illumination brightens colors on the west, and dims colors on the east side of terrain. Any value other that 0 enables TerrainShadow.

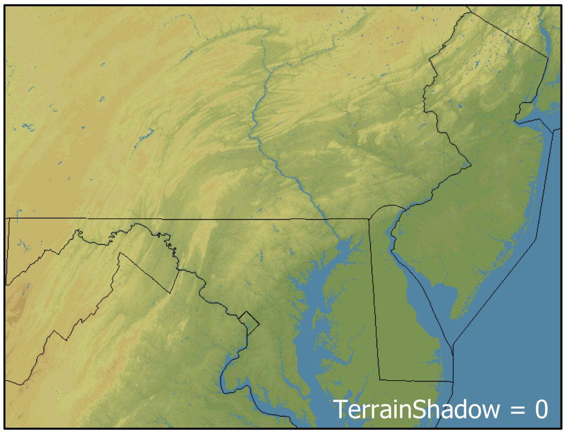
The maps below are from the eastern seaboard of the US in the Maryland, Pennsylvania, Delaware, New Jersey area. Map size is 250 x 200 NM.

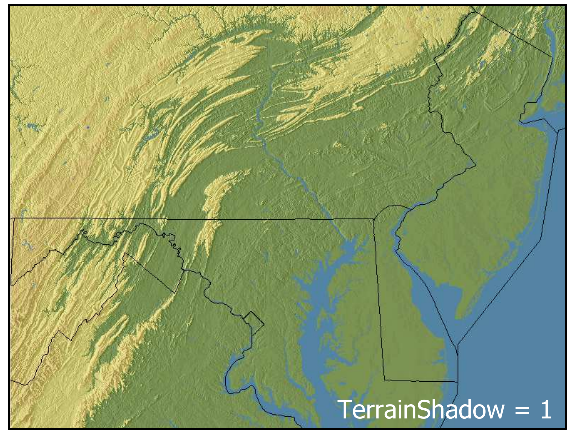
In the first pair of maps, all ElevationXColor variables are **0xC0C0C0**. Ridges of the Appalachian Mountains in the western and northern portions of the map are clearly illuminated when TerrainShadow is enabled.





The second pair of maps demonstrates TerrainShadow when the G1000 color palette is used with ElevationXColor variables.





TerrainShadow is available throughout all Zoom ranges in FSX.

In FS9, terrain shadowing is enabled by default for Zoom levels between 100 and 263,107 meters. At Zoom = 263,108 meters and higher, terrain shadow is disabled.

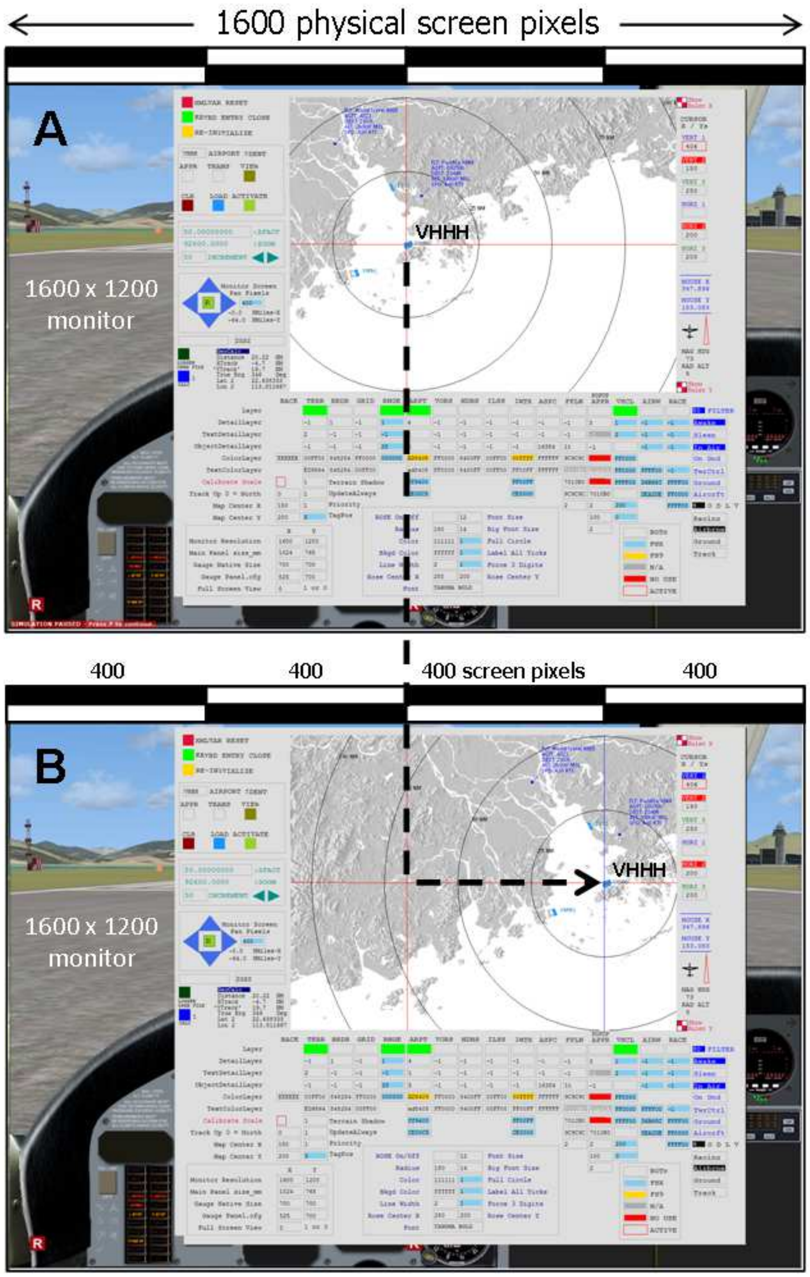
- ❑ **PanVertical**
- ❑ **PanHorizontal (physical screen pixels, enum) FSX Only**

PanVertical and PanHorizontal move the center of the map and are analogous to CenterX and CenterY except that the Pan variables are measured in physical screen pixels while CenterX and Y are measured in gauge pixels.

The screen captures on the next page show the pan effect. In Figure **A**, Hong Kong International Airport (VHHH) is at the center of the map, but the center has been adjusted using CenterX to cause the airport to be positioned at the horizontal mid point of the screen. The screen resolution is 1600 x 1200 pixels, so VHHH is at the 800 pixel position as shown by the red cross hairs.

Figure **B** is a screen shot after a pan to the left of 400 has been applied. Because the reference point (the red cross hairs) is fixed, panning to the left causes the map to move to the right. Now, VHHH has moved 400 physical screen pixels to the right as marked by the blue cross hair, and the red cross hair is centered on a location that was out of map view to the left before the pan.

Everything is shifted: Terrain, User's Aircraft, Range Rings, Traffic, etc.



The XML requires CustomDraw and Mouse entries. Within the CustomDraw element:

```
<PanVertical> (L:Pan_V) </PanVertical>
<PanHorizontal> (L:Pan_H) </PanHorizontal>
<PanReset> (L:Pan_Reset) </PanReset>
```

The PanVertical, Horizontal and Reset L:Vars can have any name and can be unitless.

Click spots are usually established to enable mouse control of panning. These compute the L:Pan_V, L:Pan_H, and L:Pan_Reset values. The Mouse XML below is associated with the pan controls ◀ ▶ ▲ ▼ and R of the gauge in the screen captures.

```
<Area Name="PanLEFT" Left="10" Top="275" Width="20" Height="30">
     <Cursor Type="LeftArrow"/>
     <Click Kind="LeftSingle">
          (L:Pan_H) (L:Pan_Pix, enum) - (>L:Pan_H)
     </Click>
</Area>

<Area Name="PanRIGHT" Left="60" Top="275" Width="20" Height="30">
     <Cursor Type="RightArrow"/>
     <Click Kind="LeftSingle">
          (L:Pan_H) (L:Pan_Pix, enum) + (>L:Pan_H)
     </Click>
</Area>

<Area Name="PanUP" Left="30" Top="255" Width="30" Height="20">
     <Cursor Type="DownArrow"/>
     <Click Kind="LeftSingle">
          (L:Pan_V) (L:Pan_Pix, enum) - (>L:Pan_V)
     </Click>
</Area>

<Area Name="PanDOWN" Left="30" Top="304" Width="30" Height="20">
     <Cursor Type="UpArrow"/>
     <Click Kind="LeftSingle">
          (L:Pan_V) (L:Pan_Pix, enum) + (>L:Pan_V)
     </Click>
</Area>

<Area Name="PanRESET" Left="37" Top="282" Width="15" Height="15">
     <Cursor Type="Hand"/>
     <Click Kind="LeftSingle">
          (L:Pan_Reset) ! (>L:Pan_Reset)
     </Click>
</Area>
```

In this example, the magnitude of the pan (number of screen pixels) is a variable, (L:Pan_Pix,_enum), that was previously set with a value of 400:

```
400 (>L:Pan_Pix, enum)
```

❑ PanReset (unitless) FSX Only

Toggling PanReset will reset the map to its center position prior to the application of PanVertical and/or PanHorizontal. Refer to the XML example above.

❑ Priority (bool)

From SDK: *Set to True to draw the map as a priority.*

❑ MapLoading (bool) FSX Only

From SDK: *Set to True if the map is currently being loaded.* This is a SET, not GET variable?

❑ UpdateAlways (bool) FSX Only

From SDK: *Set to True if the map should be updated every frame. Set to False if the map is only to be updated when positions have changed enough.*

❑ Priority, MapLoading, UpdateAlways

Unfortunately, it is difficult to be sure what these last 3 variables do. I have only a few observations:

- UpdateAlways will not by itself cause terrain to be refreshed (see TAWS chapter)
- LayerVehicles AI traffic positions do not update when UpdateAlways=0 *and* user aircraft is not moving. Therefore, in an ATC simulation, UpdateAlways must always be set to 1. Note that ITrafficInfo variables (other than ITrafficInfo:CurrentDistance) update every gauge cycle regardless of UpdateAlways setting.
- The map is "jittery" when UpdateAlways = 1.
- Priority=1 clearly speeds certain scripts such as the TAWS refresh - the terrain color refresh (see TAWS chapter)

I welcome specific feedback on the purpose and actions of Priority, MapLoading, and UpdateAlways.

Number Formats

Map variables require numerical input in the form of **HEXADECIMAL** or **DECIMAL** numbers. Unlike A:, E:, P:, and L:Vars, Units are not added.

❑ HEXADECIMAL NUMBERS

Hexadecimal values can be derived using a **Bit Table** or **Composite Hex** method.

1) Bit Table

Selection choices are organized by use of a bit table. In the LayerAirports example below, there are seven categories of Airports and selection of more than one category is allowed. If Soft Surface, Hard Surface and Non-Towered airports are wanted, then the bit table selections look like:

BIT TABLE

64	32	16	8	4	2	1	- Decimal equivalent
PRIVATE	HELIPORT	WATER	SOFT SURFACE	HARD SURFACE	NOT TOWERED	TOWERED	
6	5	4	3	2	1	0	- Bit number (Bit 0 thru Bit 6)
0	**0**	**0**	**1**	**1**	**1**	**0**	- ObjectDetailLayerAirports selections

The resulting Binary number is **0 0 0 1 1 1 0**. The Decimal integer equivalent is **14** and the Hexadecimal is **E**. The appropriate XML is therefore either:

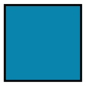
```
<ObjectDetailLayerAirports> 0xE </ObjectDetailLayerAirports>
```

or,

```
<ObjectDetailLayerAirports> 14 </ObjectDetailLayerAirports>
```

2) Composite Hexadecimal

Map object color should be specified using a composite hexadecimal number representing Blue-Green-Red shades, or the decimal equivalent of that composite hex. Example:



Blue: **173**   Green: **132**   Red: **8**   BGR Hex: **0xAD8408**

Blue shade	= 173 ;	hex = **AD**	Composite BGR Hex: **0xAD8408**
Green shade	= 132 ;	hex = **84**	Decimal equivalent = **11371528**
Red shade	= 8 ;	hex = **08**	(AD8408 hex = 11371528 decimal)

Alternatively, the decimal integer number **11371528** can be used in place of **0xAD8408**. The appropriate XML is therefore either:

```
<ColorLayerAirports> 0xAD8408 </ColorLayerAirports>
```

or,

```
<ColorLayerAirports> 11371528 </ColorLayerAirports>
```

- **DECIMAL NUMBERS**

Decimal numbers used with Map variables are **Integer**, **Bool**, or **Floating Point**, depending upon the requirements of the variable.

1) Integer – A whole number that does not include a fractional part. Single selection values such as DetailLayerVehicles (it's either -1, 0, 1, 2, or 3, but not a combination of more than one selection).

```
<DetailLayerVehicles> 2 </DetailLayerVehicles>
```

2) Bool – A subset of Integer, the variables using Bool expect either 0 or 1. An example is LayerTerrain which turns the Terrain layer On or Off. Any number other than 0 will display the layer. A zero results in no rendering.

```
<LayerTerrain> 1 </LayerTerrain>
```

3) Float - The number can include fractional values to the right of the decimal point; that is, it isn't necessarily an integer. An example is Latitude and Longitude which require input in Radians, and consequently need the precision of several digits to the right of the decimal point:

```
<Latitude> -0.4562032 </Latitude>
```

❑ USE OF EXPRESSIONS

Expressions (formulas) or other Variables are commonly used with Map variables.

Examples:

```
<Longitude> (A:PLANE LONGITUDE, radians) </Longitude>

<Latitude> -26.138516 dgrd </Latitude>

<Zoom> (L:ZoomFactor, number) 1852.0 * </Zoom>

<Zoom> (@g:map_ZoomFactor) 1852.0 * </Zoom>
```

❑ STRING VARIABLES

There are only a few string variables associated with fs9gps:Map:

CustomDraw Map

- ❑ **LayerApproachAirport (string)**

  LayerApproachAirport is the fs9gps ICAO identity of the approach airport.

  It is the full ICAO, not the Ident. Equivalent to WaypointAirportICAO.

CustomDraw Rose

- ❑ **Font (string)**

  Font used for the degrees markings, for example, Arial.

LayerTerrain

LayerTerrain draws terrain elevation and water background colors.

❑ LayerTerrain (bool)

LayerTerrain controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering. If LayerTerrain is zero, the background color of the map is determined by the BackgroundColor variable in FSX. In FS9, the background color is always black.

Example XML:

```
<LayerTerrain> 1 </LayerTerrain>
```

❑ DetailLayerTerrain (enum)

DetailLayerTerrain controls whether or not terrain elevation colors are displayed.

- DetailLayerTerrain = -1. Terrain elevation colors are displayed. In FS9, the default terrain elevation palette is applied to land and water. In FSX, the default terrain elevation palette is applied to land if no ElevationXColor variables are defined. Water color can be set using WaterColor.
- DetailLayerTerrain = 0. No terrain elevation colors are displayed. This has the same effect as LayerTerrain = 0. The map background color will be set by BackgroundColor in FSx, and will always be black in FS9.
- DetailLayerTerrain = 1. Water only. Land will show no elevation colors but will be the same color as the map background color. Water is a default dark blue in FS9, same in FSX, but can be reset using WaterColor in FSX.
- DetailLayerTerrain = 2+. Same as -1. Terrain elevation colors are displayed. In FS9, the default terrain elevation palette is applied to land and water. In FSX, the default terrain elevation palette is applied to land if no ElevationXColor variables are defined. Water color can be set using WaterColor.

Example Elevation Colors

The following examples demonstrate some of the options. The maps cover Tokyo Bay and Mt. Fuji and are centered on Atsugi Aero Base (RJTA), Kanagawa Prefecture, Japan. Map size 87.5 x 70 NM.

Figure **A**

VFR Sectional chart, US FAA format for comparison

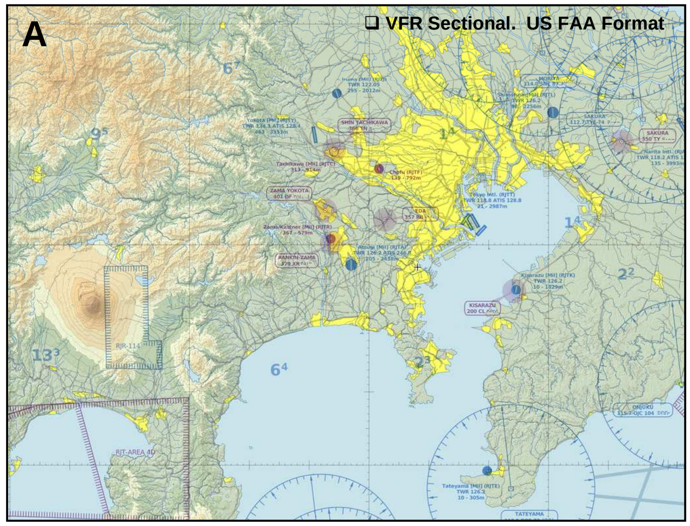


Figure **B**

DetailLayerTerrain = -1

Default FSX terrain color palette. No ElevationXColor variables are in the XML script

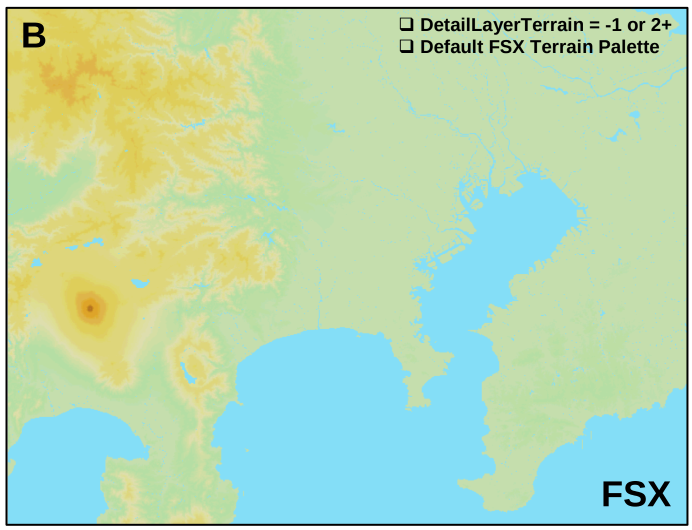


Figure **C**

DetailLayerTerrain = -1

Default FSX colors with Terrain Shadowing

Comparison with the US VFR Sectional which is the basis of fs9gps:Map format is apparent

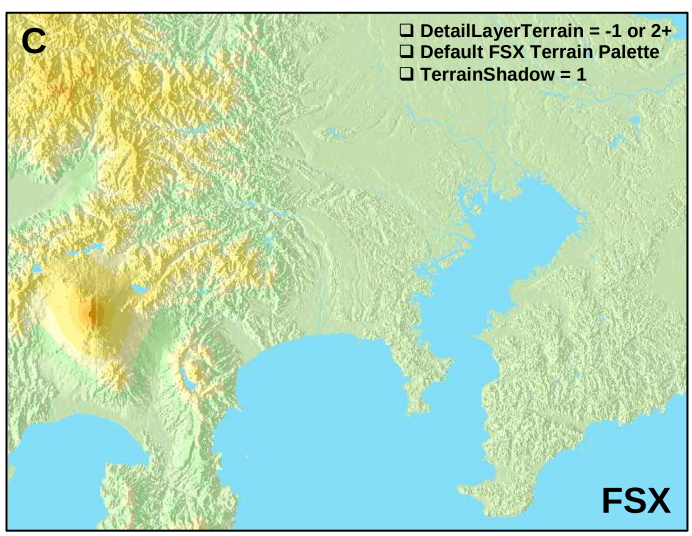


Figure **D**

DetailLayerTerrain = -1

WaterColor was changed to Garmin's G1000 water color. Provides better contrast with land

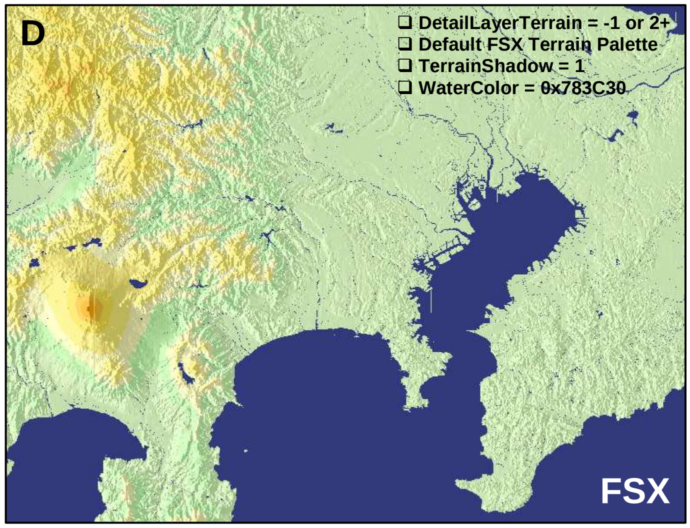


Figure **E**

DetailLayerTerrain = -1

Garmin terrain color palette (from Garmin G1000 Manual) was used with ElevationXColor variables

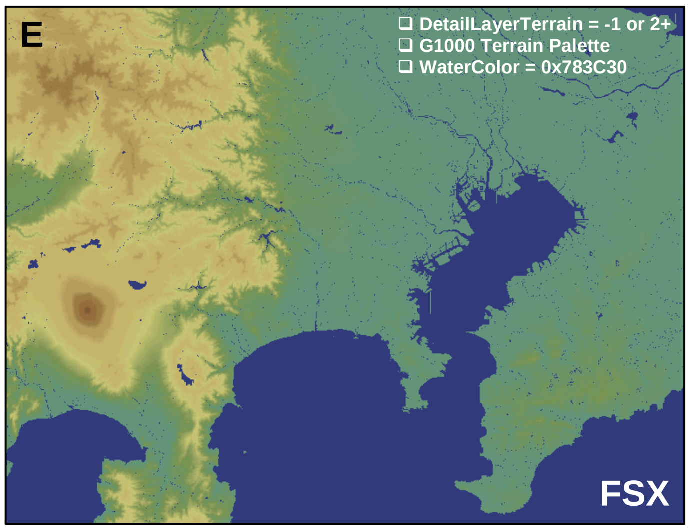


Figure **F**

DetailLayerTerrain = -1

Garmin G1000 terrain color palette was used with ElevationXColor variables

TerrainShadow = 1

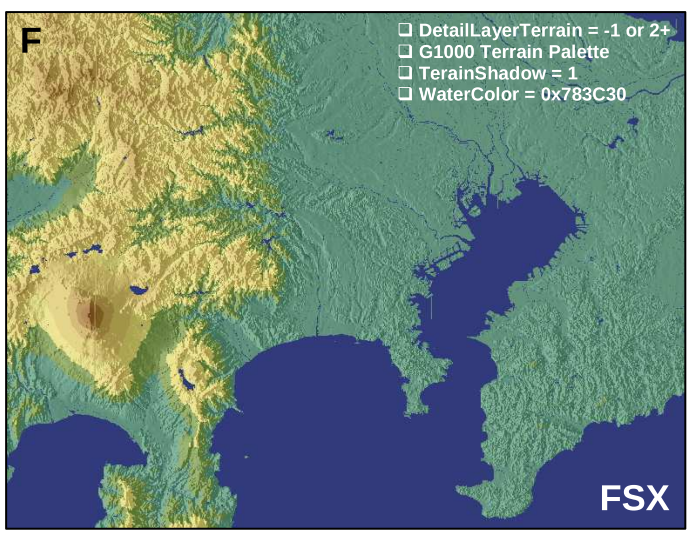


Figure **G**

DetailLayerTerrain = 1.

Water only.

When DetailLayerTerrain is set to 1, elevation colors are disabled, and BackgroundColor is used as the land color. In this example light green, 0x99FFCC

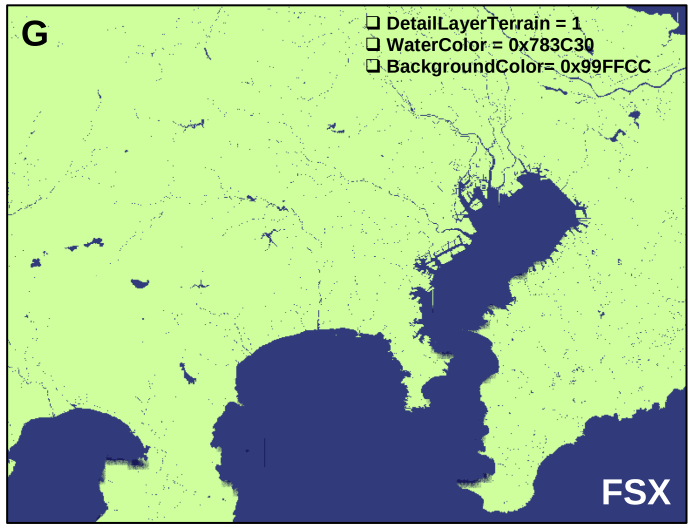


Figure **H**

DetailLayerTerrain = -1

TerrainShadow example. For this view, all ElevationxColor variables are 0xC0C0C0 and TerrainShadow = 1

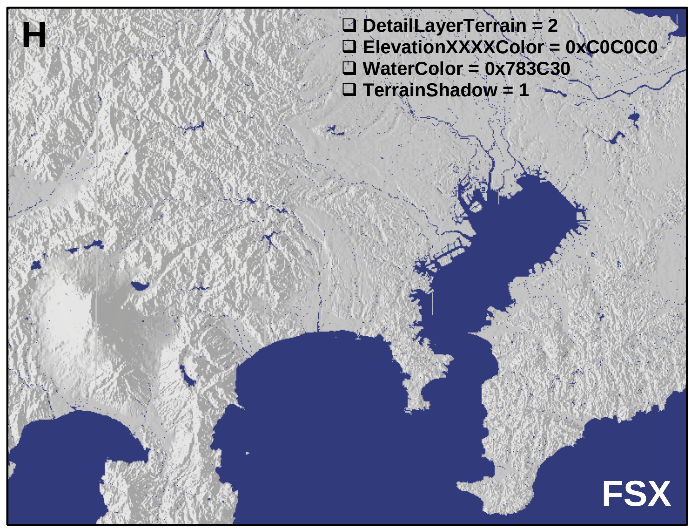


Figure **I**

DetailLayerTerrain = -1

TAWS Map approximation.

All ElevationXColor variables for altitudes above aircraft are red, ElevationXColor for the layer beneath aircraft is yellow, and below that, are all black. FSX stock WaterColor used to mask screen refresh flicker

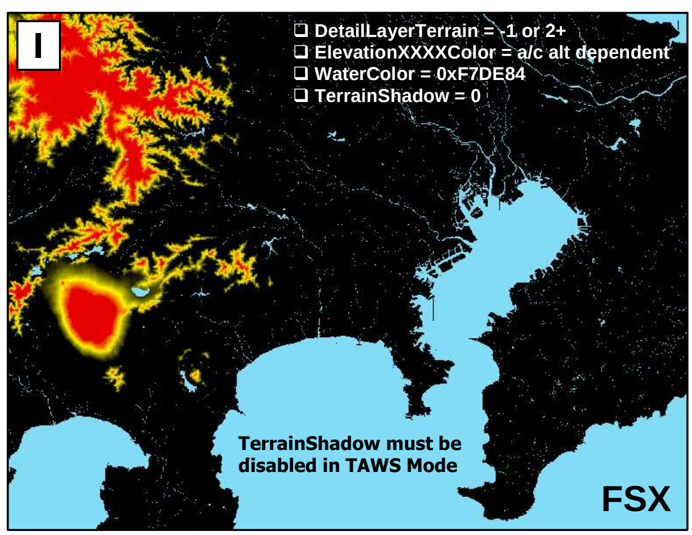


Figure **J**

DetailLayerTerrain = -1

FS9 Terrain color. This is the default, and only, terrain color scheme available.

In FS9, TerrainShadow is not a variable but is applied by default in low Zoom ranges

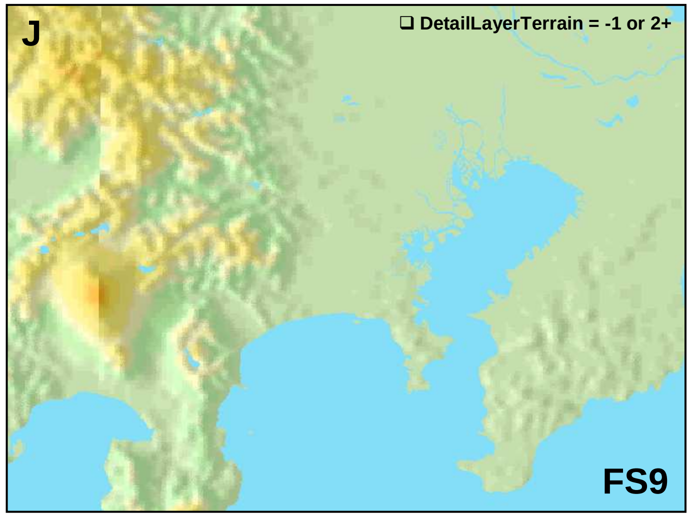
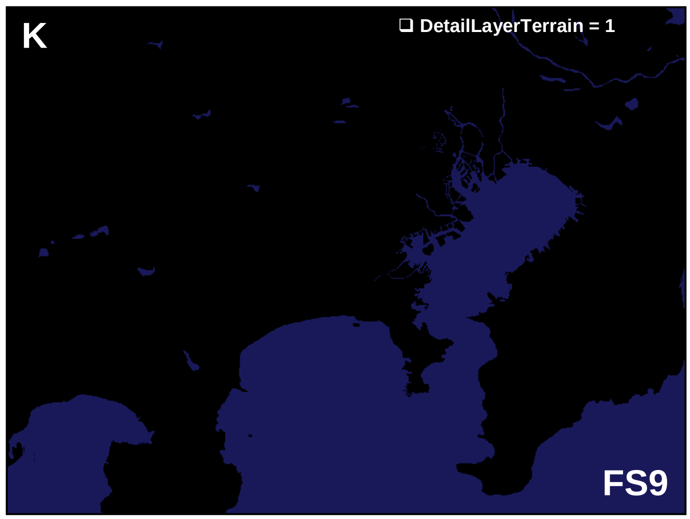


Figure **K**

DetailLayerTerrain = 1

Water only.

In FS9, there is no land or water color choice. Land color is always black, and black background is the only FS9 option. Very dark image.



⃠ TextDetailLayerTerrain

TextDetailLayerTerrain is not functional. There is no text in this layer.

❑ ObjectDetailLayerTerrain (bool)

ObjectDetailLayerTerrain is redundant with LayerTerrain and it is best to omit this variable from the XML script altogether. If ObjectDetailLayerTerrain is zero, then regardless of LayerTerrain or DetailLayerTerrain values, no elevation colors are displayed and the map background is determined by BackgroundColor in FSX or is black in FS9. Any number other than zero will enable DetailLayerTerrain to control elevation color parameters. In all cases, LayerTerrain is the parent variable and should be used to control display of terrain elevation colors.

ColorLayerTerrain

ColorLayerTerrain is not functional in either FS9 or FSX.

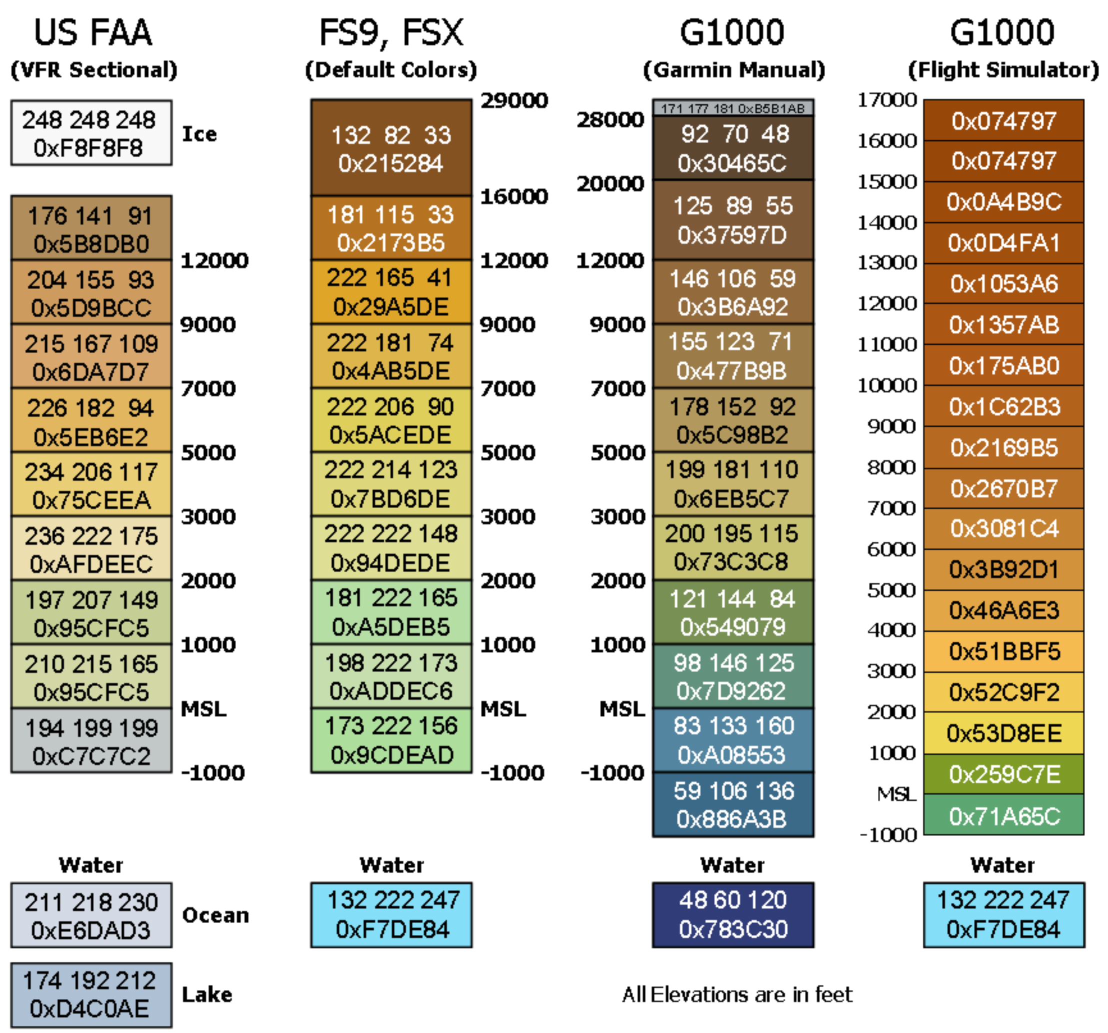
TextColorLayerTerrain

TextColorLayerTerrain is not functional. There is no text in this layer.

Elevation Color Palette Examples

Elevation color palettes of FS9, FSX and the real Garmin 1000 MFD within the limits of the color fidelity of screen captures and Pilot Guide pdf manuals are shown below.

The RGB decimal values are in normal **R G B** order whereas the hexadecimal is standard FS BGR hexadecimal.



Color Feathering (FSX)

When using either the default FSX color palette or custom ElevationXColor color variables without TerrainShadow, FSX feathers the colors. This is something to be aware of when developing a terrain awareness map.

The maps below demonstrate the effect on Elevation4000Color.

Figure **A** is a contour map of the Island of Hawaii, USA. The 2000′, 3000′, and 4000′ topographic contours from the FSX terrain data are displayed.

In Figure **B**, Elevation4000Color = 0x37597D (a chocolate brown color) and TerrainShadow = 1. The elevation color uniformly fills the interval from 4000 feet to 3000 feet as expected, and no color feathering occurs.

Figure **C** is the same map but with TerrainShadow = 0. This presents a few issues to deal with for a terrain awareness display. The area outlined by the dashed line is enlarged a little in Figure **D**.

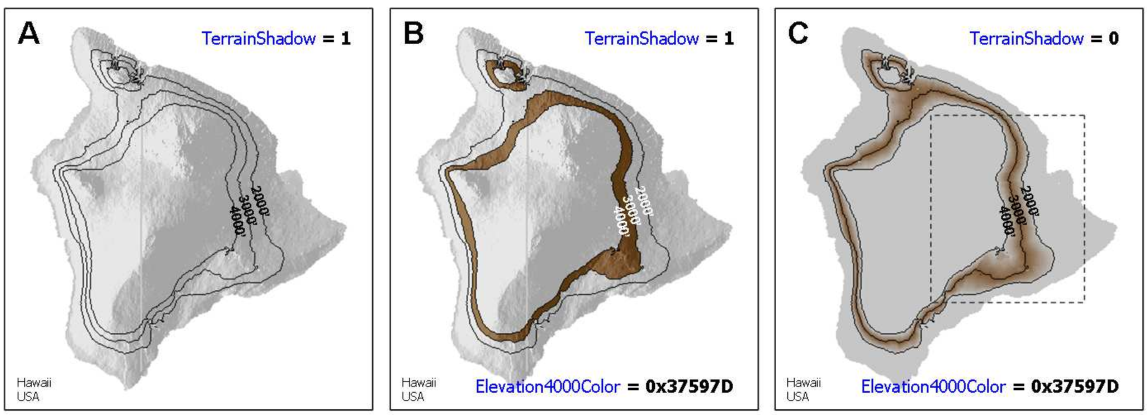
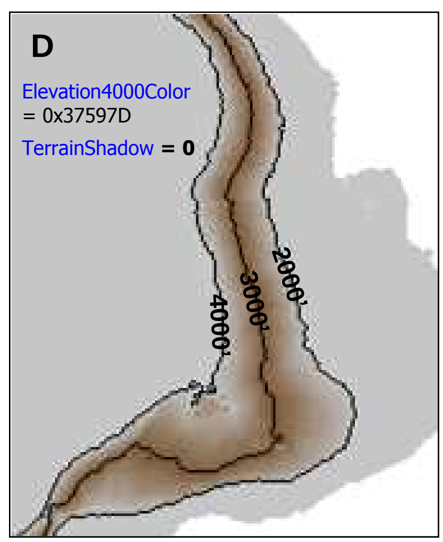


Figure **D** shows that the brown color band associated with Elevation4000Color is actually centered on the 3000′ elevation contour and feathers out in both directions for 1000 vertical feet.

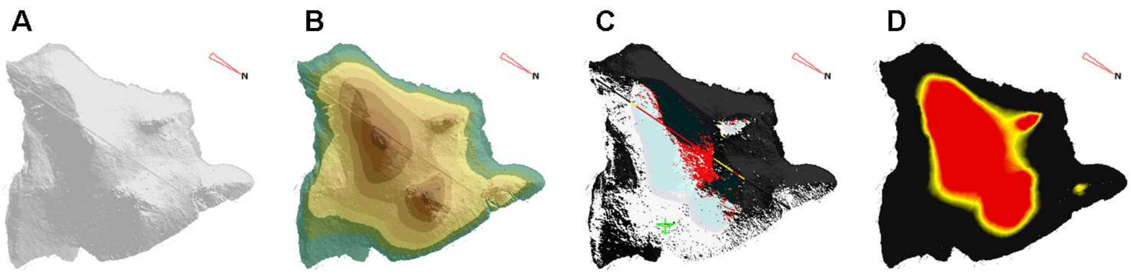


TerrainShadow Affect on TAWS Colors (FSX)

Certain colors display poorly when Terrain Shadow is enabled. Consequently, terrain palette ElevationXColor color values must be selected carefully, and special applications such as a TAWS terrain awareness map facsimile should be rendered without Terrain Shadow.

The following demonstrates the issue:

RELIEF AND TOPOGRAPHIC ELEVATION MAPS	TAWS TERRAIN AWARENESS MAPS



A	B	C	D
#C0C0C0 ElevationXColor	Garmin 1000 color pallette	Poor Red, Yellow, Black TAWS colors	Acceptable Red, Yellow, Black TAWS colors
TERRAIN SHADOW = 1	**TERRAIN SHADOW = 1**	**TERRAIN SHADOW = 1**	**TERRAIN SHADOW = 0**

Figures **A** and **B** demonstrate gray and colored palette terrain maps. The selected colors display well using either TerrainShadow = 0 or TerrainShadow = 1.

Figures **C** and **D** are TAWS terrain awareness maps scripted to show red, yellow, or black colors only as a function of A:RADIO HEIGHT. Figure **C** is displayed with TerrainShadow = 1 and is obviously not satisfactory.

Figure **D** shows the same Red, Yellow, Black ElevationXColor values as Figure **C**, but displayed without terrain shadow, that is, with TerrainShadow = 0. Color blending occurs when TerrainShadow = 0, but this image is clearly superior to Figure **C**.

Why this behavior with certain colors and TerrainShadow = 1, I haven't tried to figure out. The effect appears to be independent of graphics hardware and drivers.

The chapter TAWS Terrain Awareness Map in FSX discusses an approach to scripting a Terrain Awareness Map approximation.

LayerBorders

LayerBorders adds geopolitical boundary lines. Onshore, these are international boundaries as well as state boundaries in the United States. Offshore in FSX, they include coarsely digitized territorial water boundaries and large regional boundaries such as those recognized in the South Pacific. Offshore in FS9, coastlines are drawn. Boundaries drawn by LayerBorders do not correspond to ICAO Regions in all cases (e.g., FSX South Pacific, Australia).

In FSX only, LayerBorders includes some (although relatively few compared to a list of current day border disputes) disputed international boundaries.

❑ LayerBorders (bool)

LayerBorders controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerBorders> 1 </LayerBorders>
```

❑ DetailLayerBorders (bool)

DetailLayerBorders is redundant with LayerBorder. Any number other than 0 will display the layer. A zero results in no rendering. It is recommended to exclude DetailLayerBorders from the XML script and to control display of the layer using LayerBorders.

⃠ TextDetailLayerBorders

TextDetailLayerBorders is not functional. No names are displayed by this layer.

❑ ObjectDetailLayerBorders (decimal or hexadecimal)

FSX:

In **FSX** only, ObjectDetailLayerBorders controls whether disputed international boundaries, “non-disputed” international and state boundaries, or both, are displayed.

- ObjectDetailLayerBorders = -1. Default. Disputed and "Non-Disputed"
- ObjectDetailLayerBorders = 0. Nothing is drawn
- ObjectDetailLayerBorders = 1. Disputed boundaries are drawn
- ObjectDetailLayerBorders = 2. Disputed and "Non-Disputed" boundaries

Technically, FSX ObjectDetailLayerBorders is a *sort* of binary construction that can be represented by a hexadecimal number:

2	1	- Decimal equivalent
INTERNATIONAL AND REGIONAL BOUNDARY	DISPUTED INTERNATIONAL BOUNDARY	
1	0	- Bit number (Bit 0 and Bit1)
1	**0**	- ObjectDetailLayerBorders selections

However, Bit 0 (disputed boundaries) is always live and consequently, always drawn. Using the selection above, **0x2**, disputed boundaries are still displayed.

As far as I can tell, thirteen disputed boundaries are depicted by Flight Simulator (FSX):

Disputants (alphabetic)	Disputed Territory
Guyana - Venezuela	Zona en Reclamatión: Guyana Esequiba
Guyana - Suriname	New River Triangle
Israel - State of Palestine	Gaza Strip
Israel - State of Palestine	West Bank
Israel - ??	North District excluding Golan Heights
Cyprus - N. Cyprus	Turkish Cypriot Area (Northern Cyprus)
Morocco - Spain	Ceuta, Melilla, Peñón de Vélez de la Gomera, Peñón de Alhucemas islands, Islas Chafarinas, Isla de Alborán
Azerbaijan - Nagorno Karabakh	Nagorno-Karabakh territory
India - Pakistan	Azad Kashmir
India - Pakistan	Northern Areas, Siachen Glacier
China - India	Aksai Chin
China - Taiwan	Taiwan, Penghu Islands, Green and Orchid Islands
Morocco - Sahrawi Arab Dem. Rep.	Western Sahara

Disputed boundaries are drawn using a dashed, one screen pixel wide line. Non-disputed boundaries (according to FSX) use a solid 1 pixel line.

FS9:

In **FS9** only, ObjectDetailLayerBorders controls whether coastlines, international borders (plus State borders in the USA) or both, are displayed. The FS9 database does not include disputed territories.

A Decimal or Hexadecimal number is used that is in the form of a bit table filter similar to filters in Nearest searches (reference: GPS Guidebook page 62-63).

ObjectDetailLayerBorders (FS9)

Bit	Name	Bit	Name
0	Coastlines	1	Borders

2	1	- Decimal equivalent
BORDERS	COASTLINE	
1	0	- Bit number (Bit 0 and Bit 1)
1	**0**	- ObjectDetailLayerBorders selections

As an example, if borders but no coastline is desired, then Bit 1 is selected which results in the Binary number **`1 0`**. The decimal equivalent is 2 and the hexadecimal is 0x2. The appropriate XML:

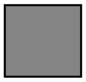
`<ObjectDetailLayerBorders> 2 </ObjectDetailLayerBorders>` or

`<ObjectDetailLayerBorders> 0x2 </ObjectDetailLayerBorders>`

Boundaries and Coastlines are drawn using a one screen pixel wide long dash, short dash line.

❑ ColorLayerBorders (BGR hexadecimal)

ColorLayerBorders controls the color of the boundary line. The default is a dark gray:



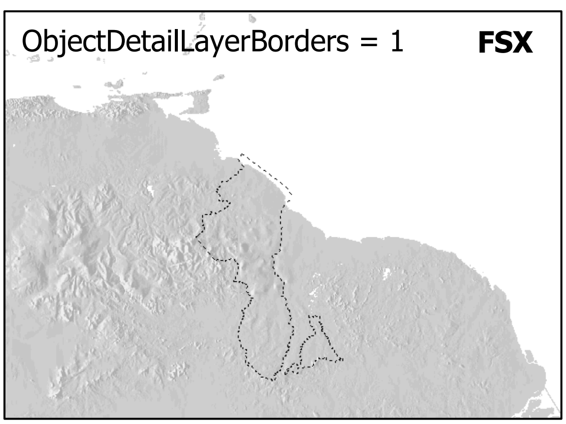
Blue: **132** Green: **132** Red: **132** BGR Hex: **0x848484**

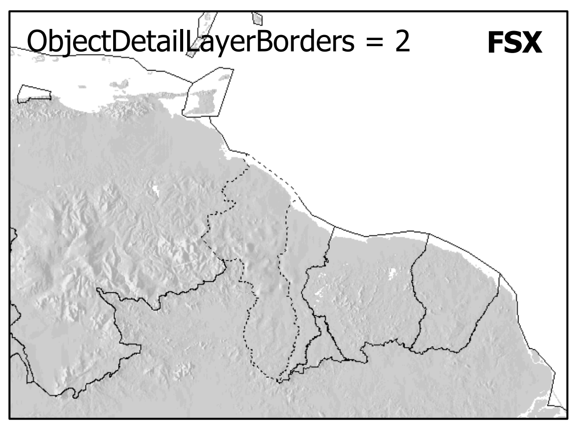
⃠ TextColorLayerBorders

TextColorLayerBorders is not functional.

Borders Example - FSX

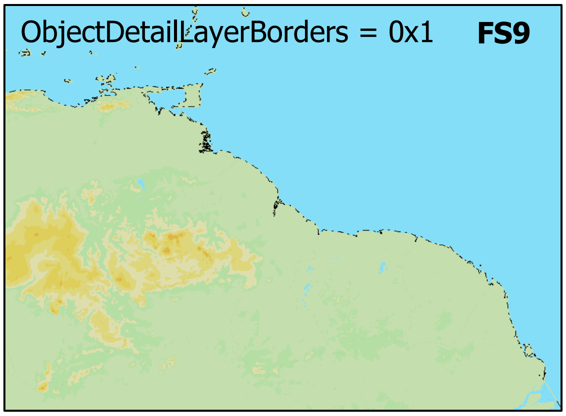
The maps below demonstrate FSX ObjectDetailLayerBorders in the Venezuela – Guyana – Suriname – French Guiana region of South America. The Guyaya Esequiba and Suriname New River Triangle disputed territories are shown by the dotted lines. Map size 1000 x 800 NM.

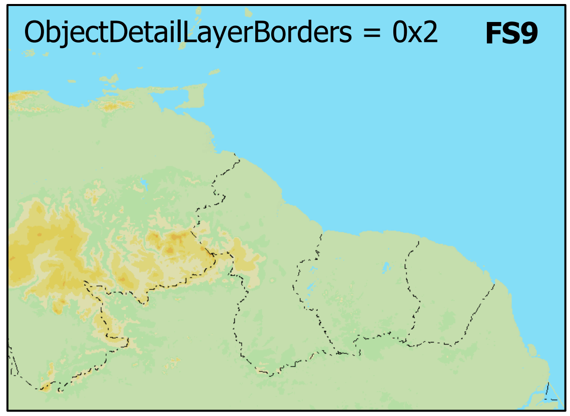


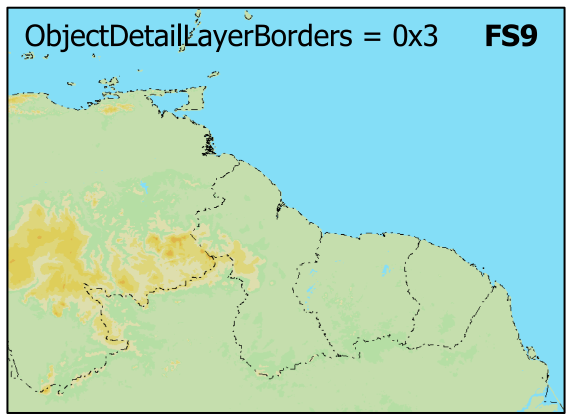


Borders Example – FS9

The maps below demonstrate FS9 ObjectDetailLayerBorders in the Venezuela – Guyana – Suriname – French Guiana region of South America. Map size 1000 x 800 NM.







LayerGridLines

FSX Only

LayerGridLines does not appear to be functional.

LayerRangeRings

FSX Only

LayerRangeRings adds range rings at prescribed intervals centered on the aircraft position A:PLANE LATITUDE and A:PLANE LONGITUDE. Range rings should be displayed <u>only</u> when TrackUp = 0 (top of the map is true North) and when Zoom is less than 500 km (270 NM).

❑ LayerRangeRings (bool)

LayerRangeRings controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML

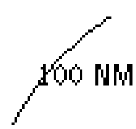
```
<LayerRangeRings> 1 </LayerRangeRings>
```

⃠ DetailLayerRangeRings

DetailLayerRangeRings is not functional.

⃠ TextDetailLayerRangeRings

TextDetailLayerRangeRings is not functional. Text labels of the range, or radius, of the circle are always drawn regardless of the value for TextDetailLayerRangeRings. The text label position, size, and units (NM) cannot be changed by the user. The label appears inside the ring, and its position is a function of ring interval and Zoom factor. It can appear in all four quadrants.



❑ ObjectDetailLayerRangeRings (nmiles enum)

ObjectDetailLayerRangeRings is the range increment between rings in Nautical Miles. Only integer values (enum) are accepted.

The default unit for range rings is Nautical Mile. It's impractical to try to achieve different range ring units such as KM because the text label always displays "NM" and fractional range values are not permitted. Even changing the measurement system to metric in Options > Settings > General will not change range ring units of NM.

Range Ring Center

Range rings are centered on the users aircraft position, **A:GPS POSITION LAT** and **A:GPS POSITION LON** (1), or A:PLANE LATITUDE and A:PLANE LONGITUDE, but not CustomDraw Latitude and Longitude which define the center of the map.

If the map center is changed using CenterX and CenterY or by panning with PanHorizontal and PanVertical, the range rings will still remain centered on A:GPS POSITION LAT and A:GPS POSITION LON / A:PLANE LATITUDE and LONGITUDE.

Range Ring Center in Multiplayer Air Traffic Controller Radar Gauge

In a multiplayer Air Traffic Controller session, Flight Simulator loads A:GPS POSITION LAT and LON (see radar.xml) with the coordinates of the Tower View (the latitude and longitude of the <SceneryObject> within <Tower>) found in the airport bgl file.

Note that the Tower coordinates can be displayed in a Nearest Traffic search by setting ITrafficInfo:Filter bits #4 and #1 and viewing (C:ItrafficInfo:C:PLANE LATITUDE, degrees) and LONGITUDE. Refer to ITrafficInfo chapter for information on Nearest Traffic searches. Tower View latitude and longitude are not the same as WaypointAirportLatitude and Longitude nor are the Control Tower coordinates retrievable as gps.dll variables.

Projection Change at 500,000 Meter Range

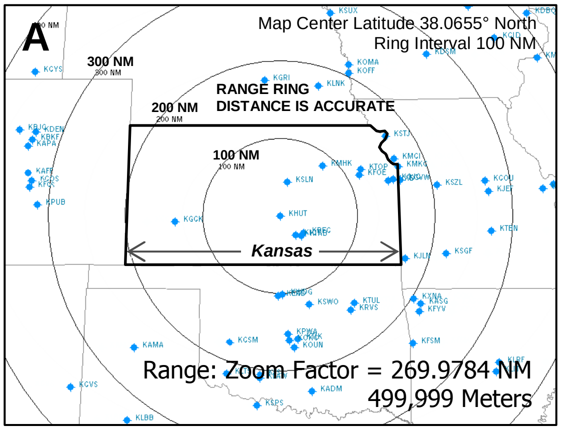
FSX changes map projections at 500,000 meter (~270 NMiles) Range.

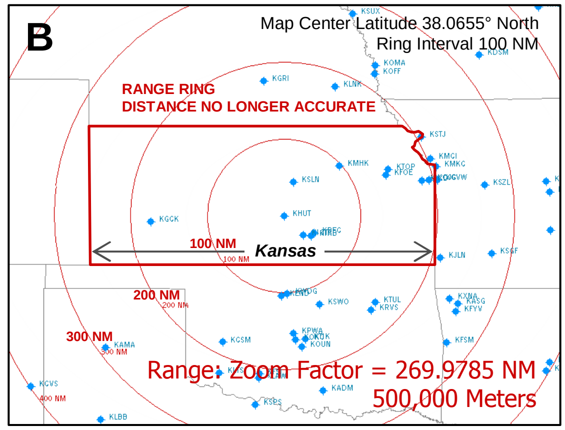
From 80 to 499,999 meters Range, FSX uses a Sinusoidal Projection that is characterized by equal north-south and east-west map scales at all points on the globe. This yields range rings that are circular as demonstrated in Figure **A** on the following page.

At 500,000+ meters Range, FSX uses the Equidistant Cylindrical Platte Carrée projection. In this projection, the north-south and east-west map scales are no longer equal except at the Equator and east-west map distances are progressively stretched as latitude increases. This results in the map view shown in Figure **B** (look at the width of the State of Kansas, USA). LayerRangeRings can only draw circular rings, however, so as a result, rings drawn when Zoom is 500,000 meters or greater (270 NM or greater) are accurate in the North-South direction only, but very inaccurate in the East-West direction (Fig. **B**). This gets worse as latitudes increase away from the Equator.

The elliptical rings in Figure **C** are accurate, but these had to be drawn using FSX Ellipse Objects, not LayerRangeRings.

LayerRangeRings should not be used when Zoom exceeds 500,000 meters (when Zoom Factor is 270 NM or greater).

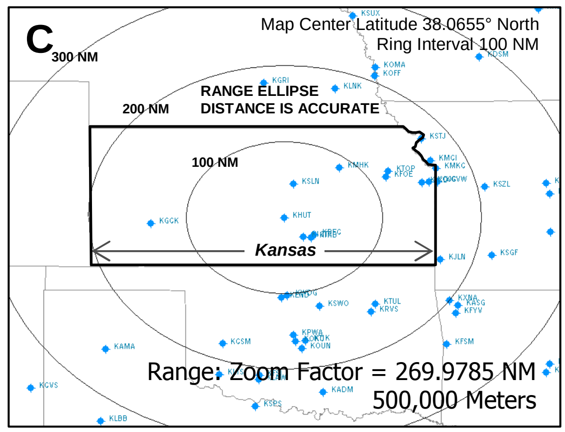




LayerRangeRings always draws circular rings as shown in Figures **A** and **B**.

The rings drawn in Figure **B** are inaccurate, however, because of the change to the *Platte Carrée* map projection where N-S and E-W map scales are no longer equal.

The elliptical rings in Figure **C** are accurate, but had to be drawn using FSX Ellipse Objects, not LayerRangeRings.



RangeRings: TrackUp=0 only

Range rings reflect accurate distance only when TrackUp=0 and Zoom is less than 500km. Range Rings are really an Air Traffic Controller radar gauge feature. ATC radar is normally configured with North up (TrackUp=0).

❑ ColorLayerRangeRings

ColorLayerRangeRings is the color applied to the range rings and associated text labels.

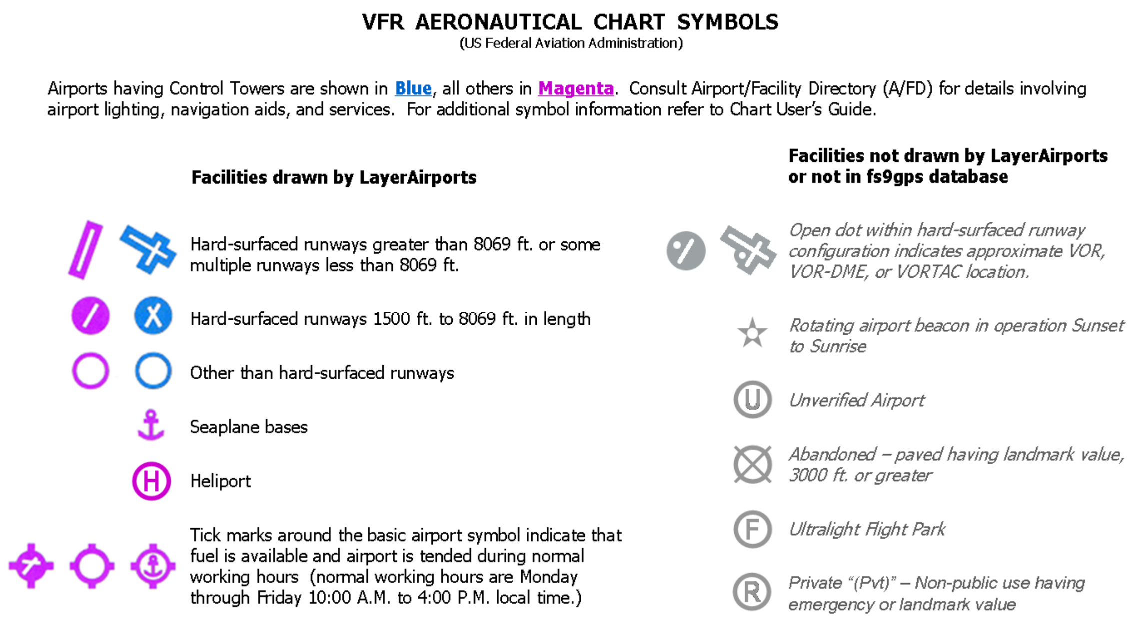
The default color is a lime green shade:

Blue: **0** Green: **197** Red: **0** BGR Hex: **0x00C500**

(1) A:GPS POSITION LAT and A:GPS POSITION LON are updated every one second. A:PLANE LATITUDE and A:PLANE LONGITUDE are updated every gauge update cycle. For most (but not all - TAWS) map purposes, A:GPS POSITION LAT, LON is sufficient.

LayerAirports

LayerAirports renders airport symbols at the location specified by WaypointAirportLatitude and WaypointAirportLongitude. CustomDraw map replicates the symbols used on U.S. VFR Aeronautical Charts as shown below:



The type of airport, symbol, text label, and colors can be controlled through specification of parameters in the LayerAirports group.

❑ LayerAirports (bool)

LayerAirports controls display of the layer. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerAirports> 1 </LayerAirports>
```

❑ DetailLayerAirports (enum)

DetailLayerAirports controls the type of symbol displayed. Only one index value can be selected at a time.

DetailLayerAirports Index

-1 = Default	1 = Dot	3 = Circle Rwy	5 = Runways
0 = Draw nothing	2 = Circle	4 = Block Rwy	

- -1 = Default. Flight Simulator automatically chooses the airport symbol type depending on Zoom setting as part of its default de-cluttering scheme:

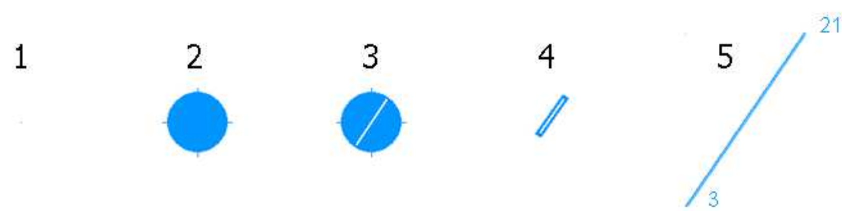
FSX	Index	Zoom range (m)		ZoomFactor range (NM)	
Runways	**5**	80	11,050	0.0432	5.9665
Block Rwy	**4**	11,051	55,250	5.9671	29.8326
Circle Rwy	**3**	55,251	110,500	29.8332	59.6652
Circle	**2**	110,501	552,500	59.6658	298.3261
Dot	**1**	552,501	2,965,000	298.3267	1600.9719
Nothing	**0**	2,965,001	5,000,000	1600.9725	2699.7840

- 0 = Draw nothing
- Index 1 through 5. The user specifies type of airport symbol to display

Example XML:

```
<DetailLayerAirports> 3 </DetailLayerAirports>
```

Airport Symbol Orientation



Airport symbol index 3, 4, and 5 (Circle Rwy, Block Rwy, and Runways) are oriented according to magnetic direction of the runway(s) as demonstrated above. The 1 screen pixel size symbol Index 1 Dot is not discernable in this screen shot reproduction.

❑ TextDetailLayerAirports (enum)

TextDetailLayerAirports controls the type of text labeling that is displayed. The text display is cumulative such that Index 2 = Index 1 plus Index 2, Index 3 = Index 1 plus Index 2 plus Index 3, and so forth.

TextDetailLayerAirports Index

-1 = Default	1 = Ident	3 = Elevation & Runway Length	5 = Runway Numbers
0 = Draw nothing	2 = Name	4 = Control and Advisory Freq	

- -1 = Default. Flight Simulator automatically chooses the airport text label depending on Zoom setting as part of its default de-cluttering scheme. The values below represent zoom ranges within which the corresponding text information is displayed. The default is affected also by the screen resolution as shown in the FSX examples. The numbers for one screen resolution compared to another are proportional, but I don't understand the details of why they are different.

DEFAULT TEXT DISPLAY ZOOM RANGES

	FSX: 1600 x 1200						FSX: 1600 x 900					
	Zoom range (m)			**ZoomFactor range (NM)**			**Zoom range (m)**			**ZoomFactor range (NM)**		
Runway Numbers	80	*to*	4,447	0.043	*to*	2.401	80	*to*	3,316	0.043	*to*	1.790
Frequencies	80	*to*	10,970	0.043	*to*	5.923	80	*to*	8,177	0.043	*to*	4.415
Elevation & Length	80	*to*	14,825	0.043	*to*	8.005	80	*to*	11,050	0.043	*to*	5.967
Name	80	*to*	22,237	0.043	*to*	12.007	80	*to*	16,575	0.043	*to*	8.950
Ident	80	*to*	148,250	0.043	*to*	80.049	80	*to*	110,500	0.043	*to*	59.665
Nothing	148,251	*to*	5,000,000	80.049	*to*	2699.784	110,501	*to*	5,000,000	59.666	*to*	2699.784

FSX: Permissible Zoom range for fs9gps:Map is 80 to 5,000,000 meters

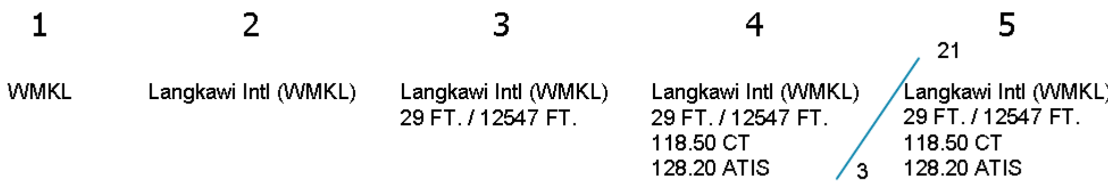
	FS9: 1600 x 1200					
	Zoom range (m)			**ZoomFactor range (NM)**		
Runway Numbers	100	*to*	8,745	0.054	*to*	4.722
Frequencies	100	*to*	21,862	0.054	*to*	11.805
Elevation & Length	100	*to*	43,725	0.054	*to*	23.610
Name	100	*to*	145,750	0.054	*to*	78.699
Ident	100	*to*	291,500	0.054	*to*	157.397
Nothing	291,501	*to*	5,000,000	157.398	*to*	2699.784

FS9: Permissible Zoom range for fs9gps:Map is 100 to 5,000,000 meters

- 0 = Draw nothing
- 1 = IDENT. The 3 to 4 character WaypointAirportIdent of the airport. The SDK refers to this as the ICAO.
- 2 = Airport Name WaypointAirportName plus Ident in parentheses.

- 3 = 2 plus airport elevation WaypointAirportElevation and length of the longest runway WaypointAirportRunwayLength. Elevation and length are reported in U.S. System (ft) or Metric (m) depending upon Flight Simulator Settings (Settings – General – International – Units of Measure).
- 4 = 3 plus Control and Advisory frequencies. The control (Control Tower or CTAF - Common Traffic Advisory Frequency) and advisory (ATIS - Automated Terminal Information Service, or AWOS - Automated Weather Observation System) frequencies are listed if they are available. In the case of multiple control or advisory frequencies available for an airport, the frequency with the lowest WaypointAirportFrequency Index is displayed.
- 5 = 4 plus Runway numbers displayed at the appropriate end of each runway.

TextDetailLayerAirports Example



❑ ObjectDetailLayerAirports (decimal or hexadecimal)

ObjectDetailLayerAirports controls the types of airports that are displayed. A Decimal or Hexadecimal number is used that is in the form of a bit table filter similar to filters in Nearest searches (reference: GPS Guidebook NearestIntersectionCurrentFilter, page 62-63).

Bit	Name	Bit	Name	Bit	Name	Bit	Name
0	Towered	2	Hard Surface	4	Water	6	Private
1	Not Towered	3	Soft Surface	5	Heliport		

Combinations of airport types can be selected according to the following rules:

ObjectDetailLayerAirports Rules

1. Bit 0 or 1, or both (Towered or Not Towered) must <u>always</u> be selected
2. Bit 2 or 3, or both (Hard or Soft Surface) must be selected for LAND (non-Water or Heliport) airports

As an example, if all Water, Hard and Soft Surface, Towered and Non-Towered airports are to be displayed, then the selection in binary number format is:

64	32	16	8	4	2	1	- Decimal equivalent
PRIVATE	HELIPORT	WATER	SOFT SURFACE	HARD SURFACE	NOT TOWERED	TOWERED	
6	5	4	3	2	1	0	- Bit number (Bit 0 thru Bit 6)
0	**0**	**1**	**1**	**1**	**1**	**1**	- ObjectDetailLayerAirports selections

The Binary number is **0 0 1 1 1 1 1**. The Decimal equivalent is 31 and the Hexadecimal is 1F. The appropriate XML is therefore either:

```
<ObjectDetailLayerAirports> 31 </ObjectDetailLayerAirports>
```

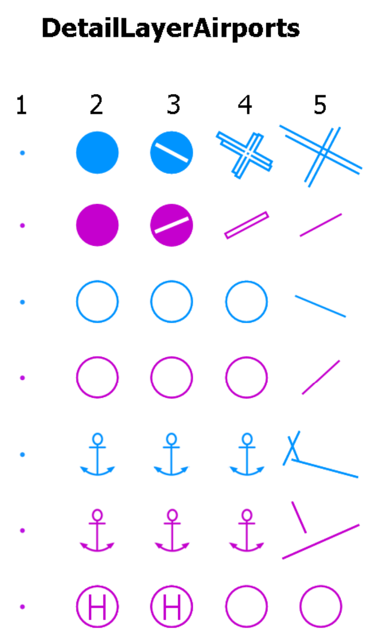
or,

```
<ObjectDetailLayerAirports> 0x1F </ObjectDetailLayerAirports>
```

ObjectDetailLayerAirports for individual airport types, together with DetailLayerAirports symbol options is shown in the diagram below:

ObjectDetailLayerAirports

Bit:	Private 6	Heli 5	Water 4	Soft 3	Hard 2	Non-Tower 1	Tower 0	Decimal	Hex		DetailLayerAirports
	0	0	0	0	1	0	1	= 5	5	Hard Surface, Towered	
	0	0	0	0	1	1	0	= 6	6	Hard Surface, Non-Towered	
	0	0	0	1	0	0	1	= 9	9	Soft Surface, Towered	
	0	0	0	1	0	1	0	= 10	A	Soft Surface, Non-Towered	
	0	0	1	0	0	0	1	= 17	11	Water, Towered	
	0	0	1	0	0	1	0	= 18	12	Water, Non-Towered	
	0	1	0	1	0	1	0	= 42	2A	Heliport, Non-Towered	
	0	1	0	1	0	0	1	= 41	29	Heliport, Towered	None in fs9gps database
	1	0	0	1	1	1	1	= 64+	40+	Private	None in fs9gps database



❑ TextColorLayerAirports (BGR hexadecimal)

TextColorLayerAirports controls color of the Towered airport text label. The text color for Untowered airports always matches the airport symbol color for Untowered airports. This is the same for both FS9 and FSX. Its format is hexadecimal Blue-Green-Red.

For both FS9 and FSX, the default TextColorLayerAirports color is the same as the airport symbol.

❑ ColorLayerAirports (BGR hexadecimal) (FS9 only)

FS9 ONLY: ColorLayerAirports controls color of Towered airport symbols only. It does not affect the default magenta color applied to Untowered airports. Its format is hexadecimal Red-Green-Blue and with the hex values concatenated from right to left (in other words, GBR).

Example XML:

```
<ColorLayerAirports> 0xFF9400 </ColorLayerAirports>
```

The default color for Towered Airports is a blue-green shade:

Blue: **173** Green: **132** Red: **8** BGR Hex: **0xAD8408**

The default, *and only*, color for Untowered airports is a magenta shade:

Blue: **206** Green: **0** Red: **197** BGR Hex: **0xCE00C5**

❑ ColorLayerAirportsTowered (BGR hexadecimal) (FSX only)

FSX ONLY: ColorLayerAirportsTowered controls color of Towered airport symbols. Its format is hexadecimal Red-Green-Blue and with the hex values concatenated from right to left (in other words, GBR).

Example XML:

```
<ColorLayerAirportsTowered> 0xFF9400 </ColorLayerAirportsTowered>
```

The default ColorLayerAirportsTowered color is a blue-green shade:

Blue: **173** Green: **132** Red: **8** BGR Hex: **0xAD8408**

❑ ColorLayerAirportsUntowered (BGR hexadecimal) (FSX only)

FSX ONLY: ColorLayerAirportsUntowered controls color of Untowered airport symbols. Its format is hexadecimal Red-Green-Blue and with the hex values concatenated from right to left (in other words, GBR).

Example XML:

```
<ColorLayerAirportsUntowered> 0xCE00C5 </ColorLayerAirportsUntowered>
```

The default ColorLayerAirportsUntowered color is a magenta shade:

Blue: **206** Green: **0** Red: **197** BGR Hex: **0xCE00C5**

LayerVORs

LayerVORs draws VORs and associated text labels at locations defined in the gps.dll database (WaypointVorLatitude and WaypoinyVorLongitude).

❑ LayerVORs (bool)

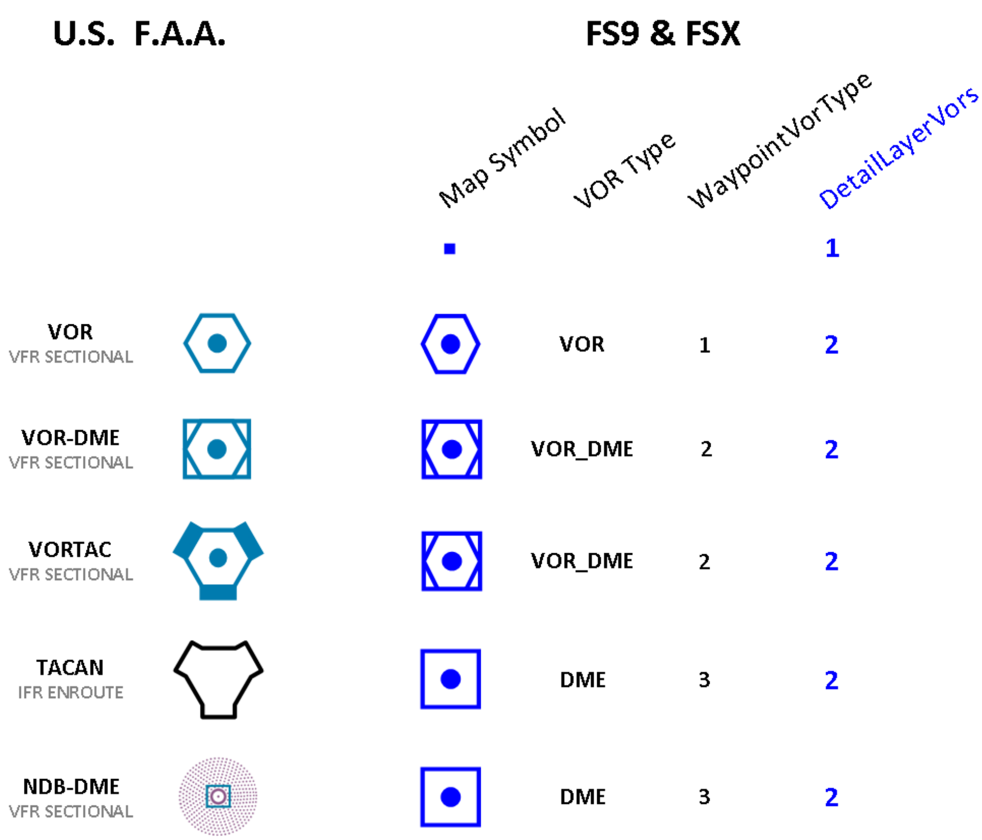
LayerVORs controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerVors> 1 </LayerVors>
```

❑ DetailLayerVORs (enum)

DetailLayerVORs controls the type of VOR symbol displayed. There is not much of a selection; it's either a one pixel dot or a VOR symbol, as shown in the figure below. Choose DetailLayerVORs = 1 for a dot, or 2 for a symbol. When 2 is chosen, then WaypointVorType automatically determines which symbol style is displayed.



The FSX SDK lists categories for the following VOR Types:

#	VOR Type	#	VOR Type
0	UNKNOWN	4	TACAN
1	VOR	5	VORTAC
2	VOR_DME	6	ILS
3	DME	7	VOT

http://msdn.microsoft.com/en-us/library/cc526954.aspx#VorType

However, not all categories are populated in the gps.dll database; only VOR Type 1, 2, and 3 are found. The remaining VOR Types are grouped in with Type 2 and 3 VORs or are not founds in the database:

- TACANs (Type 4) are included in the Type 3 DME category
- VORTACs (Type 5) are included in the Type 2 VOR_DME category
- ILS (Type 6) belong to the Airport Group
- VOT (Type 7) are not populated because VOTs are meaningless in Flight simulator

Consequently, when fs9gps:Map displays VORs, it can either use a single pixel dot, or one of three symbols that represent 5 different real-world VOR types.

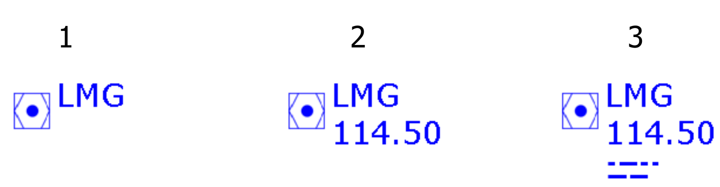
The default (no entry) DetailLayerVORs is 2.

❑ TextDetailLayerVORs (enum)

TextDetailLayerVORs is an integer index representing the type of text label to display:

-1 = Default	2 = Ident + Frequency
0 = Nothing	3 = Ident + Frequency + Ident Morse
1 = Ident	

The following is an example of the Limoges VOR/DME, Limoges, France:



The default TextDetailLayerVORs, -1 or no entry, results in a 1 = Ident label.

ObjectDetailLayerVORs (decimal or hexadecimal)

ObjectDetailLayerVORs has very little meaning. The SDK indicates that a Decimal or Hexadecimal number is used. This is in the form of a bit table filter similar to filters in Nearest searches (reference: GPS Guidebook NearestIntersectionCurrentFilter, page 62-63). In this case, however, there is only one valid selection, VOR.

ObjectDetailLayerVORs

		Bit # Name	Bit # Name
-1 DEFAULT	0 DRAW NOTHING	0 VOR	1 VOT

Although this is the simplest possible bit table case, to be thorough, the bit selection process is still demonstrated below. To select VOR, choose bit 0 as indicated:

2	1	- Decimal equivalent
VOT	VOR	
1	0	- Bit number (Bit 0 and Bit 1)
0	**1**	- ObjectDetailLayerVORs selections

The decimal equivalent of binary **0 1** is 1, and the hexadecimal is likewise equal to 1. The XML:

```
<ObjectDetailLayerVORs> 1 </ObjectDetailLayerVORs>
```

or,

```
<ObjectDetailLayerVORs> 0x1 </ObjectDetailLayerVORs>
```

VOT is not a valid choice principally because VOTs (VOR test transmitters) are not found in the gps database. Furthermore, because they are testing facilities for real aircraft VOR receivers, they are meaningless in Flight Simulator to begin with.

The default, -1 or no entry, results in VOR selected.

One other point, the Type of VORs displayed cannot be filtered in ObjectDetailLayerVORs as can be done in ObjectDetailLayerAirspaces, for example.

Bottom line, a zero and any other even number, positive or negative, results in no display. Any odd number, positive or negative, results in a display of the VOR layer.

❑ ColorLayerVORs (BGR hexadecimal)

ColorLayerVORs controls the color of the symbol, DetailLayerVORs. The default (no entry) color is blue:

Blue: **255** Green: **0** Red: **0** BGR Hex: **0xFF0000**

❑ TextColorLayerVORs (BGR hexadecimal)

TextColorLayerVORs controls the color of the symbol, TextDetailLayerVORs. Syntax for this variable is the same as for ColorLayerVORs.

If this variable is not included in the script, then the no entry default color is the same color as the VOR symbol.

LayerNDBs

LayerNDBs draws Non Directional Beacons and associated text labels at locations defined in the gps.dll database (WaypointNdbLatitude and WaypointNdbLongitude).

❑ LayerNDBs (bool)

LayerNDBs controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

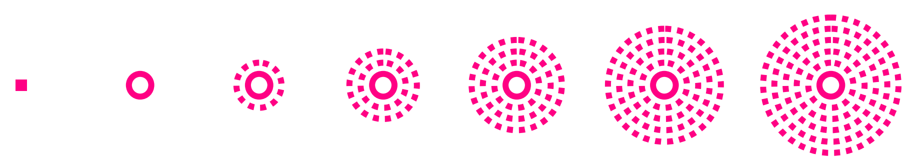
```
<LayerNDBs> 1 </LayerNDBs>
```

❑ DetailLayerNDBs (enum)

DetailLayerNDBs controls the type of NDB symbol displayed, either a one pixel dot or a NDB symbol. The number of rings displayed on the NDB is a function of Zoom. Other than the number of rings which is displayed automatically, the size cannot be changed.

All NDB Types (WaypointNdbType) are displayed with the same symbol as shown below. ILS Marker Beacons cannot be drawn by fs9gps:Map.

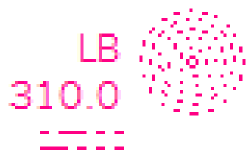
- DetailLayerNDBs = -1 or omitted. Default. NDB symbol is displayed. The number of rings is a function of Zoom
- DetailLayerNDBs = 0. Nothing is drawn
- DetailLayerNDBs = 1. Dot. A single pixel is drawn. Independent of Zoom
- DetailLayerNDBs = 2+. NDB symbol is displayed. The number of rings is a function of Zoom



	Dot = 1	NDB Symbol = -1 or 2+					
	1 Pixel	**0 Rings**	**1 Ring**	**2 Rings**	**3 Rings**	**4 Rings**	**5 Rings**
Zoom Min (m):	296,501	39,534	26,536	19,767	15,184	13,178	80
Zoom Max (m):	2,965,000	296,500	39,533	26,535	19,766	15,813	13,177
Zoom Max (NM):	1601.0	160.1	21.3	14.3	10.7	8.5	7.1

❑ TextDetailLayerNDBs (enum)

TextDetailLayerNDBs determines which text label to display. The label is displayed to the left of the NDB symbol and cannot be re-positioned.



- -1 or omitted. Default. Text displayed is a function of Zoom
- 0 = No text drawn
- 1 = NDB Ident only
- 2 = NDB Ident + Frequency (MHz)
- 3 = NDB Ident + Frequency (MHz) + Ident Morse Code

TextDetailLayerNDBs 1, 2, and 3 are independent of Zoom level. Text displayed using TextDetailLayerNDBs = -1 is a function of Zoom, however, as follows:

TextDetailLayerNDBs = -1

	No Text Label	Ident	Ident + Freq	Ident + Freq + Morse
Zoom Min (m):	296,501	59,301	14,826	80
Zoom Max (m):	5,000,000	296,500	59,300	14,825
Zoom Max (NM):	2699.9	160.1	32.0	8.0

❑ ObjectDetailLayerNDBs (bool)

ObjectDetailLayerNDBs is redundant with LayerNDBs. Any number other than 0 will display the layer. A zero results in no rendering. If ObjectDetailLayerNDBs is omitted, the default is to display the layer.

❑ ColorLayerNDBs (BGR hexadecimal)

ColorLayerNDBs controls the color of the NDB symbol and Morse Code. The default ColorLayerNDBs is a magenta shade:

Blue: **132** Green: **0** Red: **255** BGR Hex: **0x8400FF**

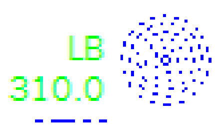
❑ TextColorLayerNDBs (BGR hexadecimal)

TextColorLayerNDBs controls the color of the Ident and Frequency text. The default color is a magenta shade:

Blue: **132** Green: **0** Red: **255** BGR Hex: **0x8400FF**

NDB Color Example

An example demonstrating ColorLayerNDBs and TextColorLayerNDBs:



- ColorLayerNDBs is 0xFF0000 (blue)
- TextColorLayerNDBs is 0x00FF00 (lime)

LayerILSs

LayerILSs draws Instrument Landing System components (localizer cone or localizer course line) for approaches utilizing a localizer (LDA, LOC, or ILS approach types). These correspond to FlightPlanApproachType = 10, 11, and 13. Microwave Landing System approaches, FlightPlanApproachType = 12, are absent in the stock gps database as far as I know, although there is limited MLS deployment in the real world.

FSX offers two ILS symbols (cone and course line) while FS9 is limited to cone only.

❑ LayerILSs (bool)

LayerILSs controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerILSs> 1 </LayerILSs>
```

❑ DetailLayerILSs (bool)

DetailLayerILSs is redundant with LayerILSs. Any number other than 0 will display the layer. A zero results in no rendering.

⃠ TextDetailLayerILSs (enum)

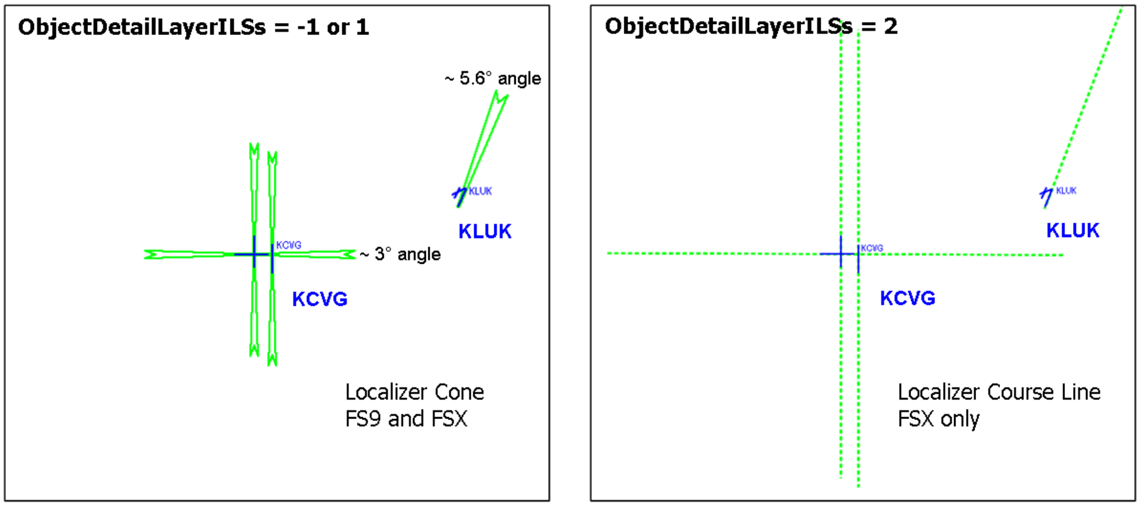
TextDetailLayerILSs is non-functional.

❑ ObjectDetailLayerILSs (enum)

ObjectDetailLayerILSs determines which ILS symbol is used, a localizer cone or a localizer course line.

- ObjectDetailLayerILSs = -1 or 1. Localizer Cone. This represents the shape of the localizer (horizontal guidance) beams.
- ObjectDetailLayerILSs = 2. Localizer Course Line. FSX only. Added as part of the ATC radar feature.

The figures on the following page show FS9 and FSX localizer symbols associated with ILS and LOC approaches at Cincinatti / Northern Kentucky International Airport (KCVG), USA.

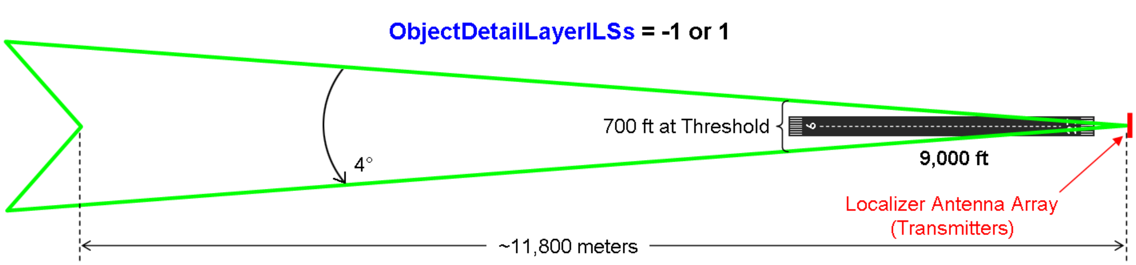


Localizer Cone Symbol Dimensions

Localizer cone width varies with runway length. In Flight Simulator as in the real world, localizer beams are 3 to 6 degrees wide. Localizer antennas are tuned for a beam width of 700 feet at the landing threshold of the runway they serve. Antenna arrays are predominantly located about 1010 feet or more from the stop end of the runway. The 1000+ ft gap is required to provide a runway safe area for aircraft takeoffs and landings that are a little too low. As far as I can tell, the stock FS database accounts for accurate coordinates of localizer array locations and resulting localizer cone angles.

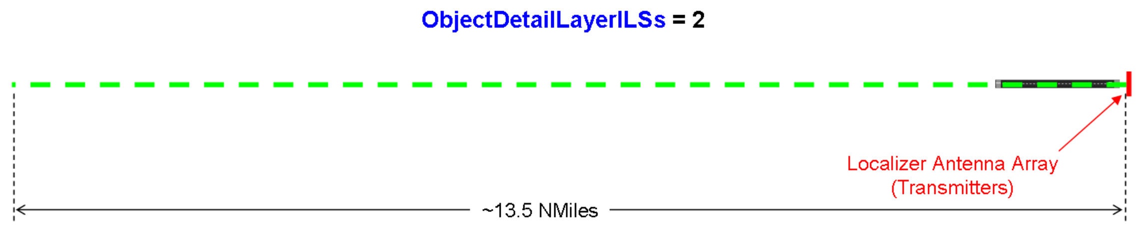
In the example below, the landing threshold is about 10,000 feet from the antenna array, producing a beam angle of 4 degrees. The longer the runway, the narrower the beam. Flight Simulator's localizer cone symbol reflects this width as shown by the difference between the symbols for KLUK ILS Rwy 21L (6101 foot runway) at ~ 5.6° and KCVG ILS Rwy 27 (12,000 foot runway) which is ~ 3°.

Length of the cone symbol drawn in fs9gps:Map is usually around 11,800 meters.



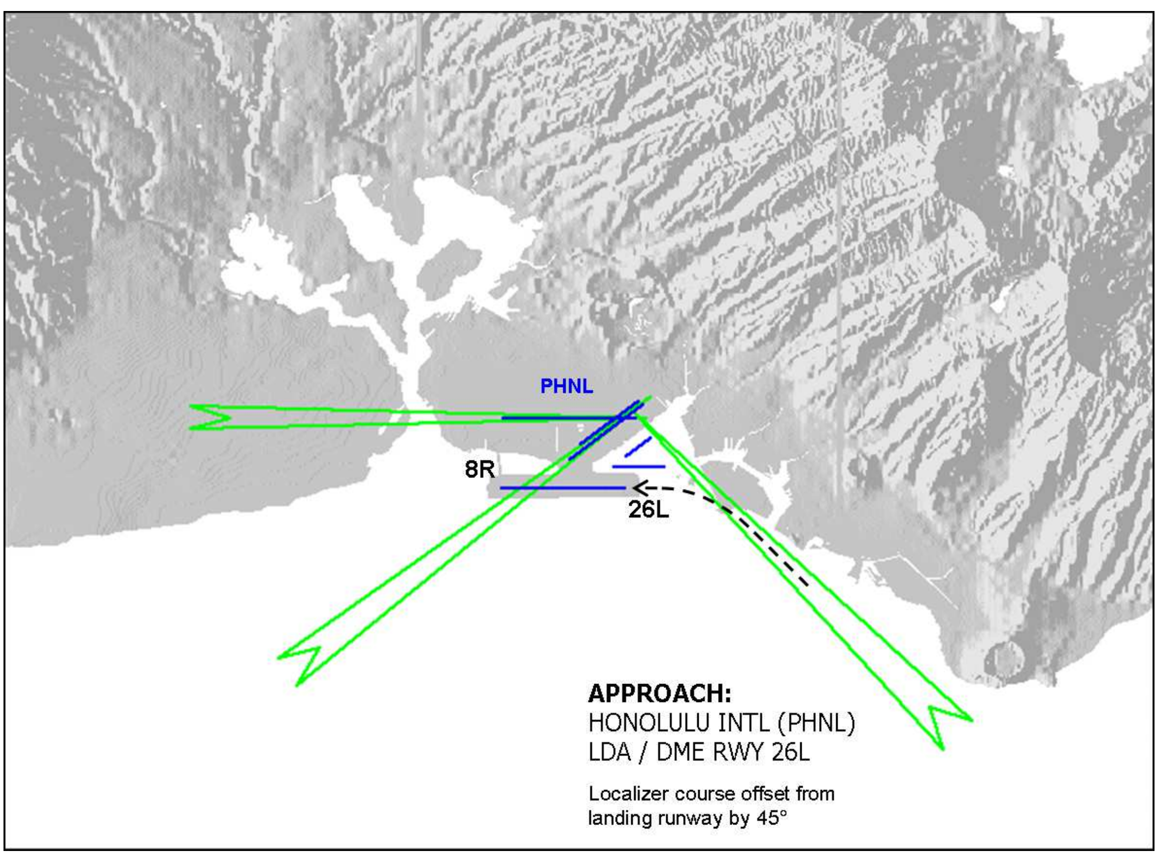
Localizer Course Line Symbol Dimensions

The localizer course line extends about 13.5 NMiles outward from the antenna array which is roughly half of the distance at which the localizer indicator becomes active in the aircraft in a straight-in approach.



Localizer Orientation

Except for irregular fs9gps database errors, course lines and cone orientation are co-linear with the runway centerline for ILS and LOC approaches and have an offset orientation for 'Offset' LOC and LDA approaches which, by definition, are at angles to the runway centerline.



❑ ColorLayerILSs (BGR hexadecimal)

ColorLayerILSs controls color of the ILS symbol. If ColorLayerILSs is omitted from the XML script, the default color is lime:

	Blue: **0**	Green: **255**	Red: **0**	BGR Hex: **0x00FF00**

⃠ TextColorLayerILSs (BGR hexadecimal)

TextColorLayerILSs is non-functional.

LayerIntersections

LayerIntersections draws Enroute and Terminal intersections at locations defined in the WaypointIntersections group.

❑ LayerIntersections (bool)

LayerIntersections controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerIntersections> 1 </LayerIntersections>
```

❑ DetailLayerIntersections (enum)

DetailLayerIntersections controls the style of intersection symbol, either a single pixel dot or a triangle.

- DetailLayerIntersections = -1 or omitted. Triangle
- DetailLayerIntersections = 0. Draw nothing
- DetailLayerIntersections = 1. Single pixel dot
- DetailLayerIntersections = Any number other than 0 or 1. Triangle

DetailLayerIntersections: 1 -1 or 2

❑ TextDetailLayerIntersections (bool)

TextDetailLayerIntersections controls the display of the intersection Ident. The text is positioned to the right of the intersection symbol at about the 4 o'clock position and cannot be moved. Any number other than zero will display the Ident; zero will hide Ident.

Draw nothing **0 =** Draw Nothing

-1 or 1 = Ident text label

❑ ObjectDetailLayerIntersections (bool)

ObjectDetailLayerIntersections is redundant with LayerIntersections. Any number other than 0 will display the layer. A zero results in no rendering.

❑ ColorLayerIntersections (BGR hexadecimal) FS9 Only

ColorLayerIntersections is a FS9 variable that controls the color of enroute intersections only. Terminal intersections and text labels in FS9 are always blue (0xCE0000). If ColorLayerIntersections is omitted from the XML script, the default color is magenta.

Blue: **255** Green: **0** Red: **255** BGR Hex: **0xFF00FF**

❑ ColorLayerIntersectionsEnroute (BGR hexadecimal) FSX Only

ColorLayerIntersectionsEnroute is an FSX Only variable that controls the color of enroute intersections and the Ident text label. If it is omitted from the XML script, the defauilt color is magenta.

Blue: **255** Green: **0** Red: **255** BGR Hex: **0xFF00FF**

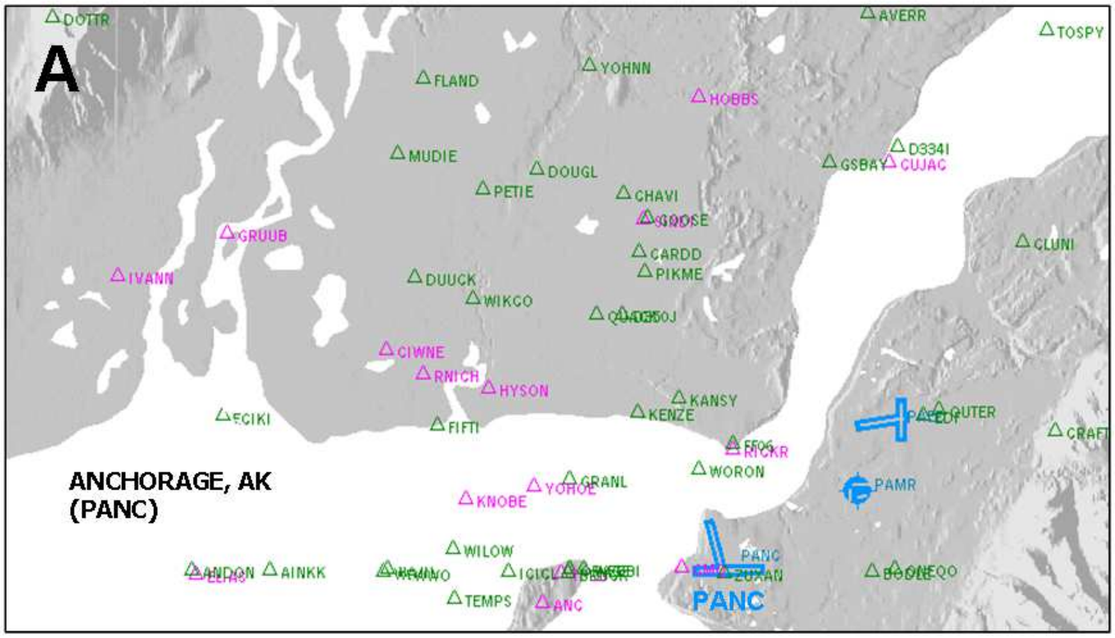
❑ ColorLayerIntersectionsTerminal (BGR hexadecimal) FSX Only

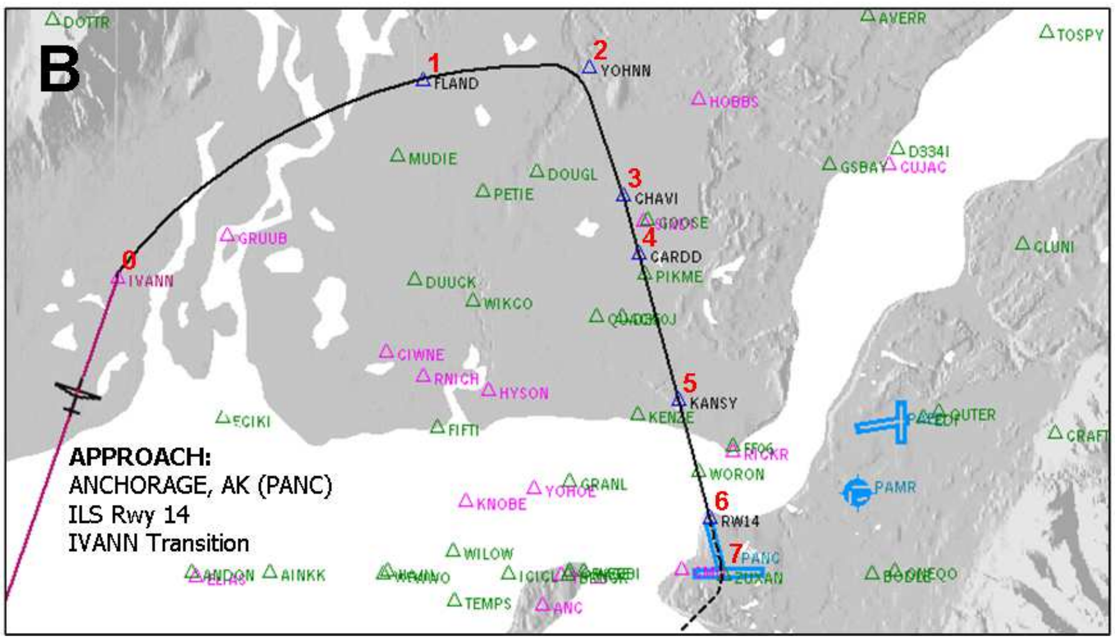
ColorLayerIntersectionsTerminal is an FSX Only variable that controls the color of terminal intersections and the Ident text label. If it is omitted from the XML script, the defauilt color is a deep blue shade.

Blue: **206** Green: **0** Red: **0** BGR Hex: **0xCE0000**

❑ TextColorLayerIntersections (BGR hexadecimal) FS9 and FSX

TextColorLayerIntersections controls the Ident text color for enroute intersections only. It does not affect terminal intersections. If it is omitted from the XML script, the default color the same as the enroute intersection symbol.





FLIGHT PLAN NEW APPROACH: PANC

10 :ApprWaypointsNumber	13 :FlightPlanApprType	9 :FlightPlanWaypointApproachIndex
ILS 14 :FlightPlanApprName	IVANN :FlightPlanApprTransName	0 :FlightPlanActiveApproachWaypoint
1 :FlightPlanIsActiveApproach	2 :FlightPlanApproachIndex	3 :FlightPlanApproachTransitionIndex

-- FlightPlanWaypointApproach --

Idx	ICAO 111 123456789012	Name	Type	Mode	Latitude (Deg Min)	Longitude (Deg Min)	Latitude (Degrees)	Longitude (Degrees)	Alt	Trgt	Leg Dist	Course (mag)
0	WPA IVANN	IVANN	1	1	61 19.9487	-150 41.8593	61.332478	-150.697656	0	0	40.11	-1.00
1	WPAPANCFLAND	FLAND	5	1	61 26.1380	-150 24.1759	61.435633	-150.402931	5000	0	10.75	71.76
2	WPAPANCYOHNN	YOHNN	5	1	61 27.0499	-150 12.7461	61.450831	-150.212435	4000	0	5.56	89.46
3	WPAPANCCHAVI	CHAVI	1	1	61 22.7010	-150 6.9209	61.378350	-150.115348	3300	0	5.66	140.00
4	WPAPANCCARDD	CARDD	1	1	61 20.7290	-150 5.7960	61.345483	-150.096600	3300	0	2.04	140.00
5	WPAPANCKANSY	KANSY	1	2	61 15.8652	-150 3.0515	61.264420	-150.050858	1600	0	5.04	140.00
6	RPAPANCRW14	RW14	1	2	61 11.9679	-150 0.8640	61.199465	-150.014399	202	0	4.04	140.00
7			9	3	61 10.5364	-149 59.9676	61.175607	-149.999460	800	800	1.50	140.00
8	WPA NAPTO	NAPTO	1	3	60 48.3610	-150 26.1460	60.806017	-150.435767	2000	0	26.41	173.00
9	WPA NAPTO	NAPTO	7	3	60 48.3610	-150 26.1460	60.806017	-150.435767	2000	0	14.34	173.00

Additional points

Figure **A** shows intersections displayed by LayerIntersections in the vicinity of Achorage, Alaska, USA.

- Enroute intersections are colored the default **magenta** △. These are intersections used for cross-country navigation purposes and are often part of Victor and Jet Airway routes. Enroute waypoints often serve as approach transition waypoints.
- Terminal waypoints are displayed in **green** △ in this example. These intersections are used by Terminal and approach procedures into and out of airports.

Figure **B** shows the flight path for the ILS Rwy 14 Approach, IVANN Transition into Anchorage International Airport (PANC).

- The Ident text color of terminal waypoints used by the approach (FLAND, YOHNN, CHAVI, CARDD, KANSKY) is the same color (black in this example) as the approach flight path. ColorLayerFlightPlan color selections override LayerIntersections color selections in display of Flight Plan and Approach elements, and the symbol color of terminal waypoints included in the approach revert to the default blue color.
- The ICAO of terminal intersections contains the Ident of the "owning airport" in ICAO character positions 4 through 7. In this example, PANC. The ICAO of enroute intersections does not contain the Ident of an "owning airport" (e.g., IVANN).
- Approach Waypoint Index 6 is a Runway Waypoint used by Flight Simulator to define the runway location. All approaches include such a waypoint. It is displayed when LayerFlightPlan is enabled and an approach is loaded, but absent otherwise.

LayerAirspaces

LayerAirspaces draws airspace boundaries. LayerAirspaces has more filtering capability (i.e., include only selected airspace types) but less color flexibility than most other layers.

❑ LayerAirspaces (bool)

LayerAirspaces controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerAirspaces> 1 </LayerAirspaces>
```

❑ DetailLayerAirspaces (bool)

DetailLayerAirspaces is entirely redundant with LayerAirspaces.

⃠ TextDetailLayerAirspaces

TextDetailLayerAirspaces is non-functional.

❑ ObjectDetailLayerAirspaces (decimal or hexadecimal)

ObjectDetailLayerAirspaces is a filter used to select which types of airspaces to draw. It is the same filter applied by NearestAirspaceQuery discussed in **FS9GPS Module Guidebook** (pp 81-82). Input to ObjectDetailLayerAirspaces can be decimal or hexadecimal format but not binary.

A check of airspaces found in the fs9gps database returned 19,261 unique airspace sectors as summarized in the table below.

The database scan was widespread and although probably not 100% comprehensive, it was close to it and I believe very representative of what is in the database. From the returns, a few observations can be made:

- Not all Airspace 'Types' actually exist in the database. TOWER, CLEARANCE, GROUND, DEPARTURE, APPROACH, NATIONAL_PARK, MODE_C, and RADAR Airspace Types are all absent.
- Not all Airspace Types populated in the database can, or should, be drawn by LayerAirspaces. CENTER, CLASS_A, CLASS_F, CLASS_G, and TRAINING Airspace Types are not drawn.

AIRSPACE NAME	TYPE	COUNT	DRAWN?	AIRSPACE NAME	TYPE	COUNT	DRAWN?
CENTER	1	4196	×	*APPROACH*	*13*	*0*	⦸
CLASS_A	2	307	×	**MOA**	**14**	**623**	✓
CLASS_B	**3**	**574**	✓	**RESTRICTED**	**15**	**3285**	✓
CLASS_C	**4**	**1687**	✓	**PROHIBITED**	**16**	**833**	✓
CLASS_D	**5**	**2573**	✓	**WARNING**	**17**	**391**	✓
CLASS_E	**6**	**1926**	✓	**ALERT**	**18**	**45**	✓
CLASS_F	7	9	×	**DANGER**	**19**	**2211**	✓
CLASS_G	8	74	×	*NATIONAL_PARK*	*20*	*0*	⦸
TOWER	*9*	*0*	⦸	*MODE_C*	*21*	*0*	⦸
CLEARANCE	*10*	*0*	⦸	*RADAR*	*22*	*0*	⦸
GROUND	*11*	*0*	⦸	TRAINING	23	527	×
DEPARTURE	*12*	*0*	⦸		Total:	19,261	

Airspace Definitions

Flight Simulator incorporates airspace boundaries defined by the air traffic authorities of various countries around the world. Most countries adopt the 1990 ICAO Airspace classification for the *type* of air traffic control, that is, the flight rules (IFR, SVFR, or VFR, ATC communication, speed, and separation protocol) applied within each airspace class, however, country-by-country adaptation and boundary definitions for the airspace classes vary widely. Countries are free to select and apply only those Airspace Classes that are suitable to their needs and to develop their own chart symbol styles as well.

As an example, some countries do not use Class **B** airspace. Others designate a blanket layer between certain altitudes (therefore lacking geographic boundaries that can be drawn on a map) as Class **B**, and yet others apply Class **B** to familiar 'upside down wedding cake' boundaries that are easily drawn on a map.

Flight Simulator's line styles and colors for each Class are consistent across all countries, but in the real world they are not.

Center Airspace

Center (Airspace Type 1) is not an Airspace, per se. Center airspace boundaries represent the areas of responsibility of individual enroute air traffic control centers (Air Route Traffic Control Centers - ARTCCs in the US, Area Control Centers in Europe). The boundaries of Centers determine where the Flight Simulator ATC function hands-off aircraft to the subsequent Center. In FS, only Center airspaces have a frequency and a frequency name, NearestAirspaceCurrentFrequency and CurrentFrequencyName. The CurrentFrequencyName is always "Center". Center altitude boundaries are defined as NearestAirspaceCurrentMinAltitude = 0 (surface) and CurrentMaxAltitude = 100000 meters, or edge of space. Center Airspace boundaries are not drawn by LayerAirspaces nor found on US Sectional (VFR) charts which are the basis for the LayerAirspaces format.

Air Traffic Control-Based Airspace Classes

ATC-based Airspace Classes include Class **A** through Class **G**. These correspond to Flight Simulator Airspace Types 2 – 8 (NearestAirspaceCurrentType). The first five, Classes **A** through **E**, are Controlled Airspaces, and where geographic boundaries for these airspaces exist in the database, Flight Simulator draws them.

However, geographic boundaries that can be drawn on a map do not always exist in FS or the real world. Class **A** Airspace is not used by all countries, but where it is used, such as in the United States and Russia, it is a blanket airspace defined by altitude limits but not geographic boundaries. Consequently, Class **A** Airspaces cannot be drawn by LayerAirspaces even though Class **A** airspaces are found in the database.

Additionally, US Class **E** Airspace generally exists everywhere below 18,000 ft there is not already Class **A**, **B**, **C**, or **D** or **G** Airspace, and everywhere above FL60. "Everywhere" lacks boundaries that can be drawn, so instead, FS draws Class **E** boundaries only where they extend to the ground surface.

Class **F** and **G** are Non-Controlled Airspaces and are not drawn in Flight Simulator.

Special Use Airspaces

Special Use Airspace's (SUA) purpose is to advise pilots of activities or areas that have special flight rules or may be hazardous at certain times. Seven SUA types are found in the Flight Simulator gps database and six of these are drawn by LayerAirspaces.

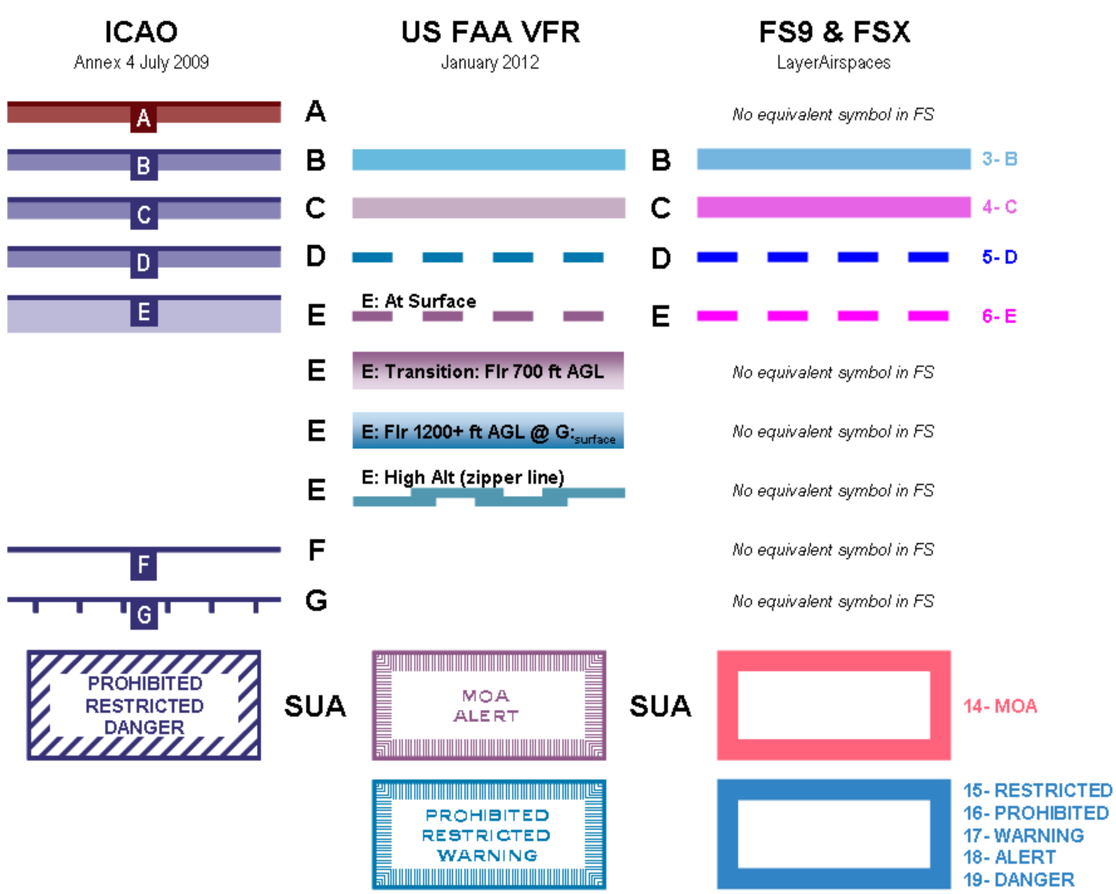
SUA Name	FS Airspace Type	FS Drawn?	Controlled Airspace?	
MOA	14	Yes	No	Military. USA. Purpose is to separate high-speed military traffic from IFR traffic. VFR also permitted but with caution
RESTRICTED	15	Yes	Yes	Not prohibited to fly, but unauthorized penetration not allowed and possibly dangerous at certain times (eg, live military firing, bombing ranges in US)
PROHIBITED	16	Yes	No*	Flight of aircraft is not permitted
WARNING	17	Yes	No	Advisory in nature. Airspace over domestic or international waters that extends from three NM beyond shore
ALERT	18	Yes	No	Training Area: US. No restrictions but use caution. Alert areas may contain a high volume of pilot training or unusual activity. All but 2 Alert areas are in US. Other 2 are in S Korea
DANGER	19	Yes	No	Military usually. Non-US. Unauthorized penetration not allowed and possibly dangerous at certain times. Most are military operations areas (high speed a/c, live firing, etc)
TRAINING	23	No	No	Training Area: Non-US.

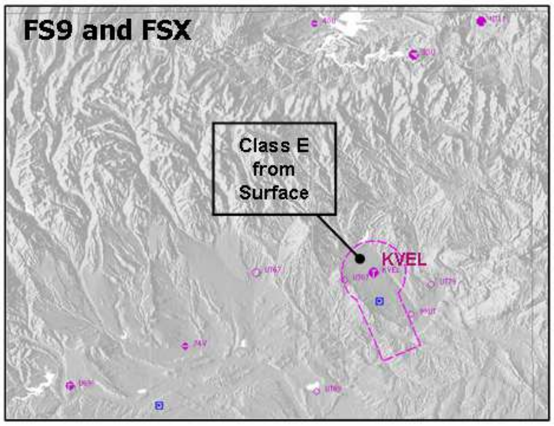
LayerAirspaces Line Format

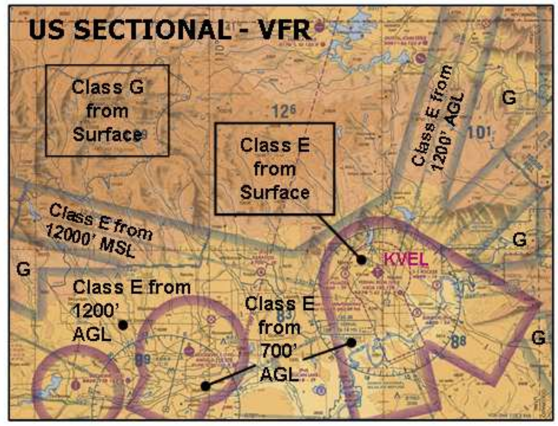
LayerAirspaces adopts the U.S. F.A.A. Sectional Chart (VFR) airspace symbol format.

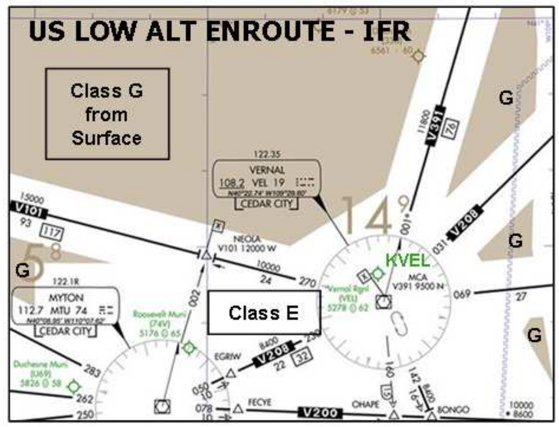
A comparison of ICAO (most of the non-US world), US F.A.A. Sectional, and Flight Simulator airspace boundary styles is shown below. Some points:

- The basis for LayerAirspaces line styles is the US F.A.A. VFR Sectional Chart, not High and Low Altitude Enroute charts which use different airspace symbol sets
- US F.A.A. Sectional charts depict Class **E** Surface, 700/1200 foot AGL Transition, **E** at **G**surface, and High Altitude MSL (zipper line) Areas. Within US Airspace, LayerAirspaces draws only the Class **E** Surface Airspace at ground surface
- FAA Sectionals use a magenta color for ALERT SUA whereas FS uses blue
- LayerAirspaces provides no text labeling of airspace name or altitude limits
- LayerAirspaces applies US FAA Sectional format world-wide, even in countries that use ICAO Standard symbol format or their own airspace boundary format
- The FS airspace database is not current. Regular updates occur in the real world





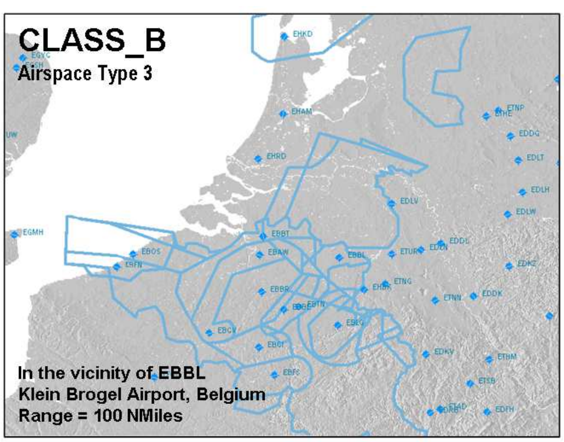


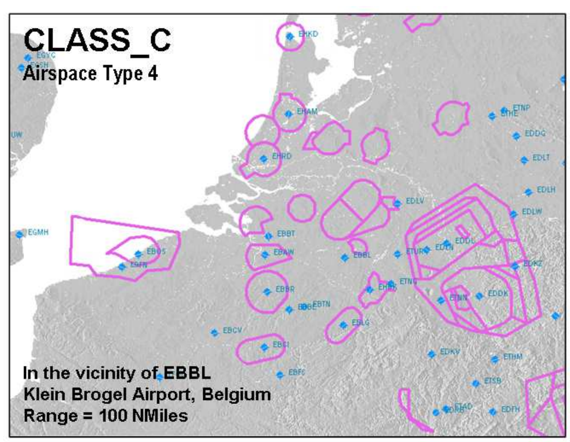


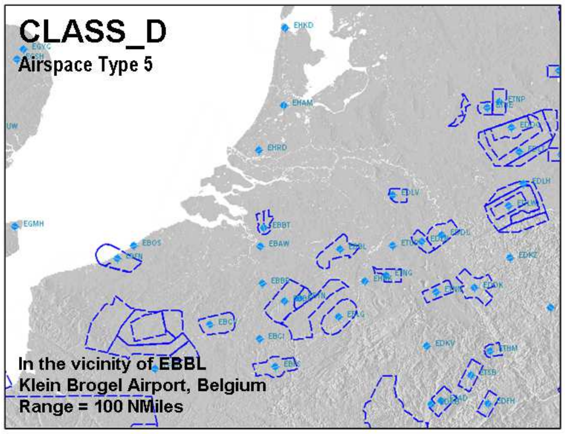
These figures show airspace below 14,500’ MSL down to the base of Class **E** or Class **G** and points out differences between FS and real charts. However, the lack of Class **E** detail has little significance in FS even in multiplayer controller simulations.

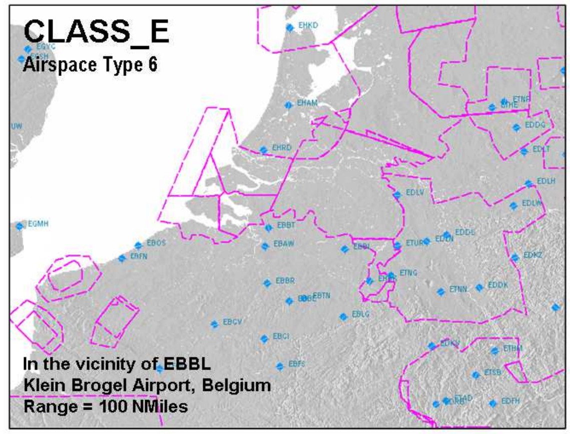
Line widths and colors (essentially) cannot be changed in LayerAirspaces. Additionally, line widths are constant and are not scaled according to Zoom.

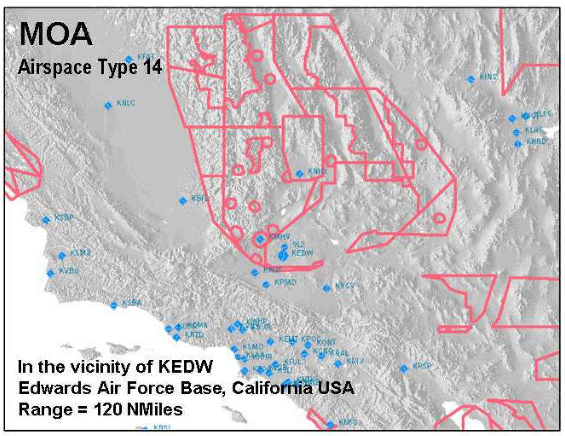
Examples of LayerAirspaces in FSX

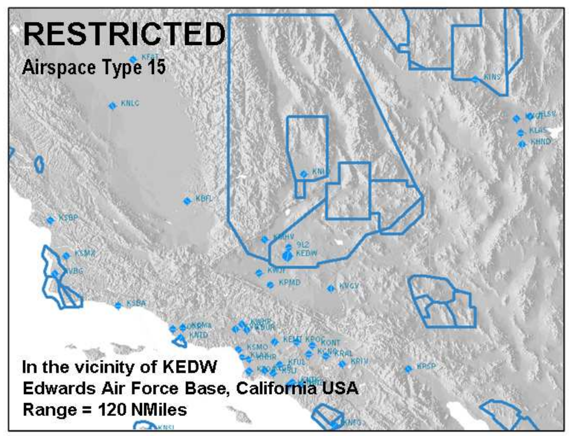


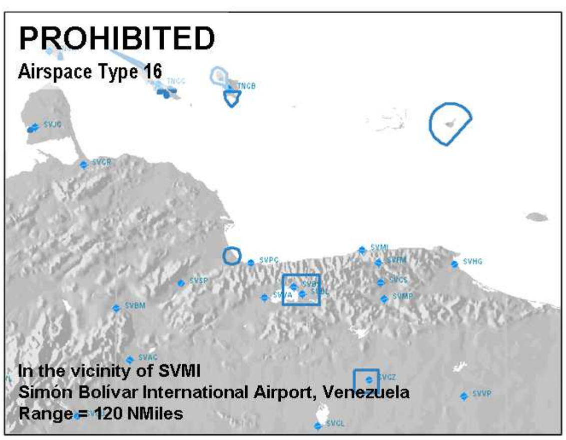


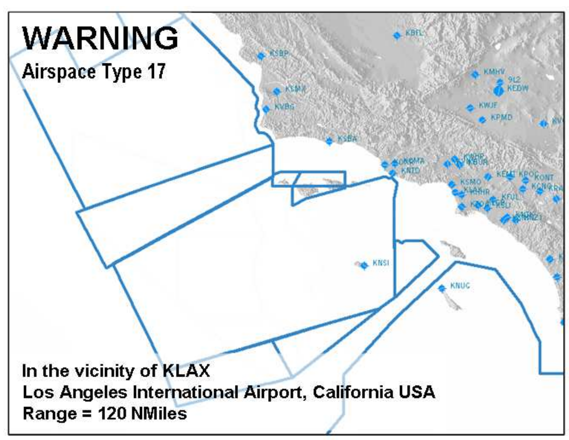

CLASS_D
Airspace Type 5
In the vicinity of EBBL
Klein Brogel Airport, Belgium
Range = 100 NMiles

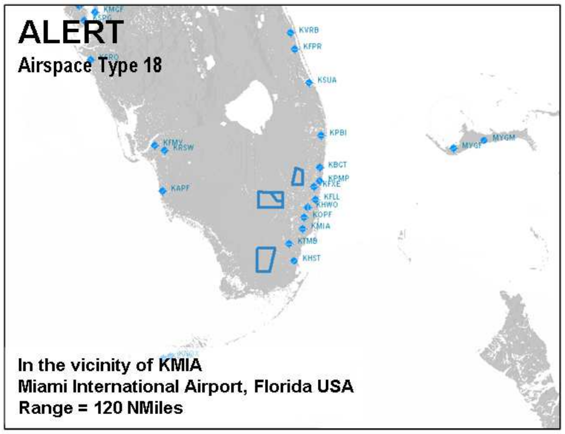

CLASS_E
Airspace Type 6
In the vicinity of EBBL
Klein Brogel Airport, Belgium
Range = 100 NMiles

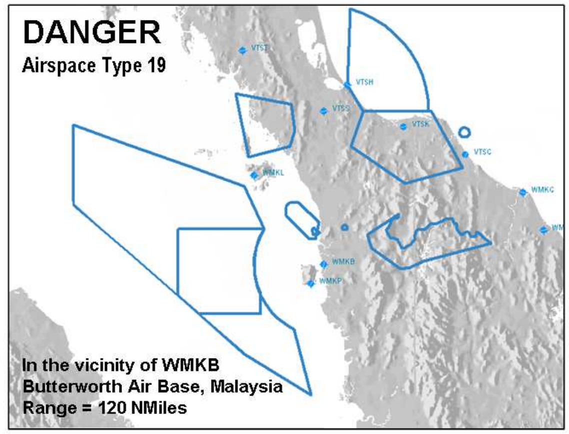

MOA
Airspace Type 14
In the vicinity of KEDW
Edwards Air Force Base, California USA
Range = 120 NMiles


RESTRICTED
Airspace Type 15
In the vicinity of KEDW
Edwards Air Force Base, California USA
Range = 120 NMiles


PROHIBITED
Airspace Type 16
In the vicinity of SVMI
Simón Bolívar International Airport, Venezuela
Range = 120 NMiles


WARNING
Airspace Type 17
In the vicinity of KLAX
Los Angeles International Airport, California USA
Range = 120 NMiles





⃠ ColorLayerAirspaces (BGR hexadecimal)

ColorLayerAirspaces is not really functional the way one would expect. Although ColorLayerAirspaces can be specified, it is applied to all airspace types, and most importantly, it has the effect of only dulling, or dimming the default color only, but not changing it to the specified color. Furthermore, all color choices for ColorLayerAirspaces darken the default color, never lighten, or brighten it.

Omitting ColorLayerAirspaces or using it with **0xFFFFFF** results in the default color, entering **0x000000** produces black airspace boundary lines.

⃠ TextColorLayerAirspaces

TextColorLayerAirspaces is not functional. Just as there is no TextDetailLayerAirspaces, there is no color choice either.

LayerFlightPlan

LayerFlightPlan draws a waypoint-to-waypoint path of the loaded Flight Plan. Approach procedures are also drawn after an Approach has been loaded (the Approach becomes part of the Flight Plan at that point).

❑ LayerFlightPlan (bool)

LayerFlightPlan controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

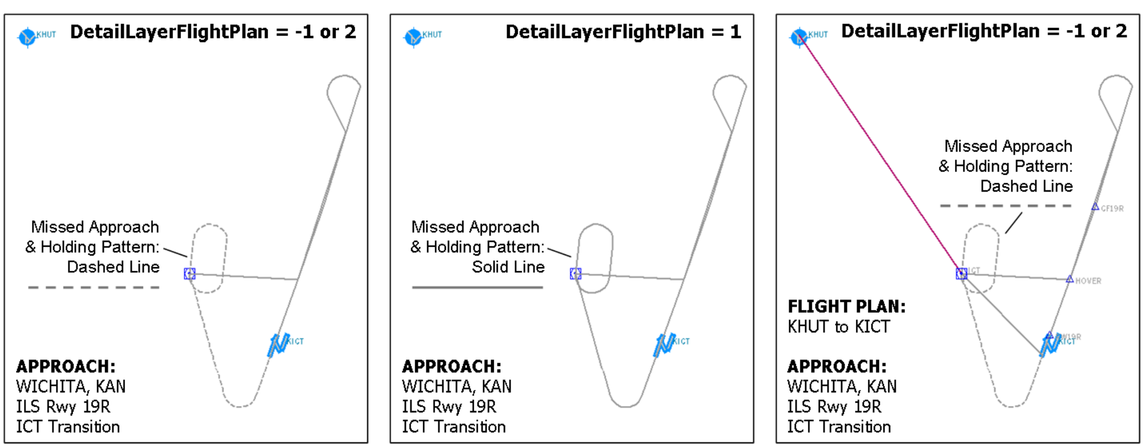
Example XML:

```
<LayerFlightPlan> 1 </LayerFlightPlan>
```

❑ DetailLayerFlightPlan (enum)

DetailLayerFlightPlan determines the line style of the Missed Approach path.

- **-1** = Default. Dashed Lines for Missed Approach (includes Holding Pattern). Solid lines for Enroute and Approach flight plan segments. Only Missed Approach and Holding Pattern can be dashed lines.
- **0** = Draw Nothing
- **1**= All Solid lines for Missed Approach (includes Holding Pattern), Enroute , and Approach flight plan segments
- **2** = Same as Default. Dashed Lines for Missed Approach (includes Holding pattern). Solid lines for Enroute and Approach flight plan segments



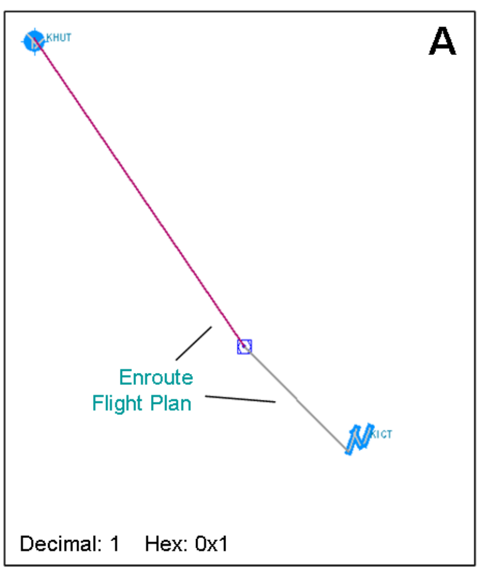
⃠ TextDetailLayerFlightPlan (enum)

TextDetailLayerFlightPlan is non-functional.

❑ ObjectDetailLayerFlightPlan (decimal or hexadecimal)

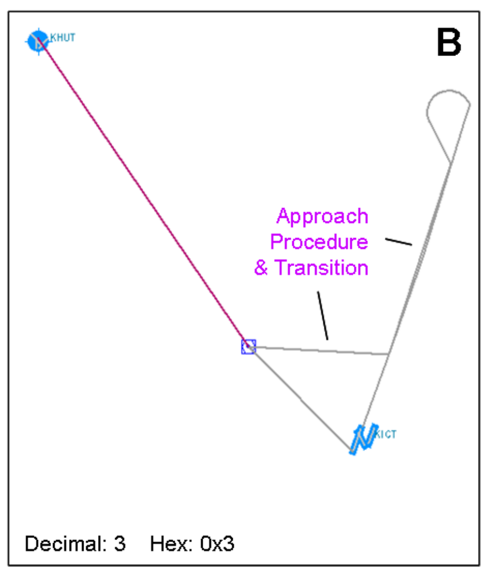
ObjectDetailLayerFlightPlan controls which Flight Plan and Approach segments are drawn. It is best thought of as a binary number that represents the choices as demonstrated below.

	8	4	2	1	- Decimal equivalent
	WAYPOINTS	MISSED APPROACH	APPROACH	ENROUTE	
	3	2	1	0	- Bit number (Bit 0 thru Bit 3)
Fig. A	**0**	**0**	**0**	**1**	- ObjectDetailLayerFlightPlan selections
Fig. B	**0**	**0**	**1**	**1**	- ObjectDetailLayerFlightPlan selections
Fig. C	**0**	**1**	**1**	**1**	- ObjectDetailLayerFlightPlan selections
Fig. D	**1**	**1**	**1**	**1**	- ObjectDetailLayerFlightPlan selections



ObjectDetailLayerFlightPlan:

0 0 0 1 Enroute



ObjectDetailLayerFlightPlan:

0 0 1 1 Approach + Enroute

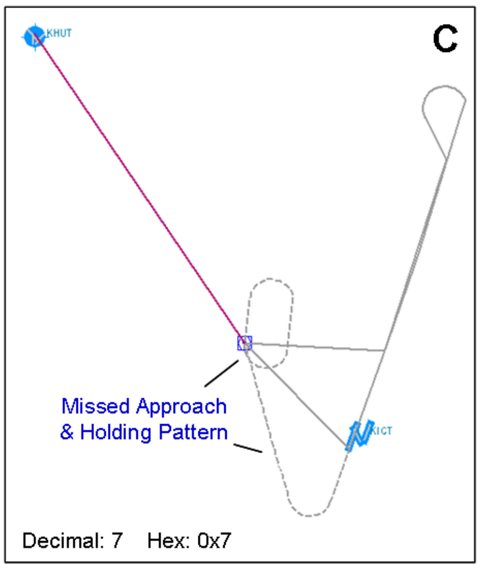
As an example, if the user wants to draw the Enroute Flight Plan and Approach Procedure (Figure **B**), the appropriate selection is bit 0 and bit 1. The resulting binary number is **0 0 1 1** whose decimal equivalent is 3. The hexadecimal equivalent is likewise 3.

Example XML:

```
<ObjectDetailLayerFlightPlan> 3 </ObjectDetailLayerFlightPlan>
```

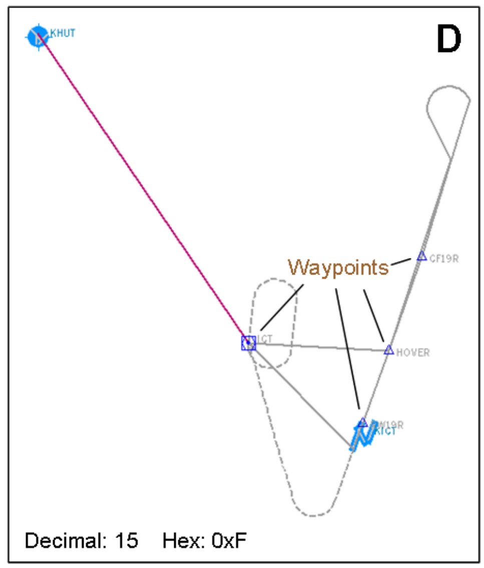
or,

```
<ObjectDetailLayerFlightPlan> 0x3 </ObjectDetailLayerFlightPlan>
```



ObjectDetailLayerFlightPlan:

0 1 1 1 Missed + Approach + Enroute



ObjectDetailLayerFlightPlan:

1 1 1 1 Waypoints + Missed + Approach + Enroute

The Missed Approach and Holding pattern is added in Figure **C**, and Waypoint designations are added in Figure **D**.

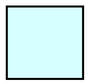
Selecting Waypoints adds the waypoint's (VFR) Aeronautical Chart symbol and the waypoint Ident text label to the right of the symbol. The text cannot be re-positioned.

There is no Zoom limit on the Flight Plan display – it is drawn at all Zoom levels, 80 to 5,000,000 meters for FSX, 100 to 5,000,000 meters for FS9.

❑ ColorLayerFlightPlan (BGR hexadecimal)

ColorLayerFlightPlan controls color of the non-Active Flight Plan leg. If ObjectDetailLayerFlightPlan Waypoints bit is set, then ColorLayerFlightPlan also controls the color of the Waypoint Ident text. Its format is hexadecimal Blue-Green-Red.

If ColorLayerFlightPlan is omitted from the XML script, the default color is a very pale blue shade which is suitable when terrain is showing:



Blue: **255** Green: **255** Red: **215** BGR Hex: **0xFFFFD7**

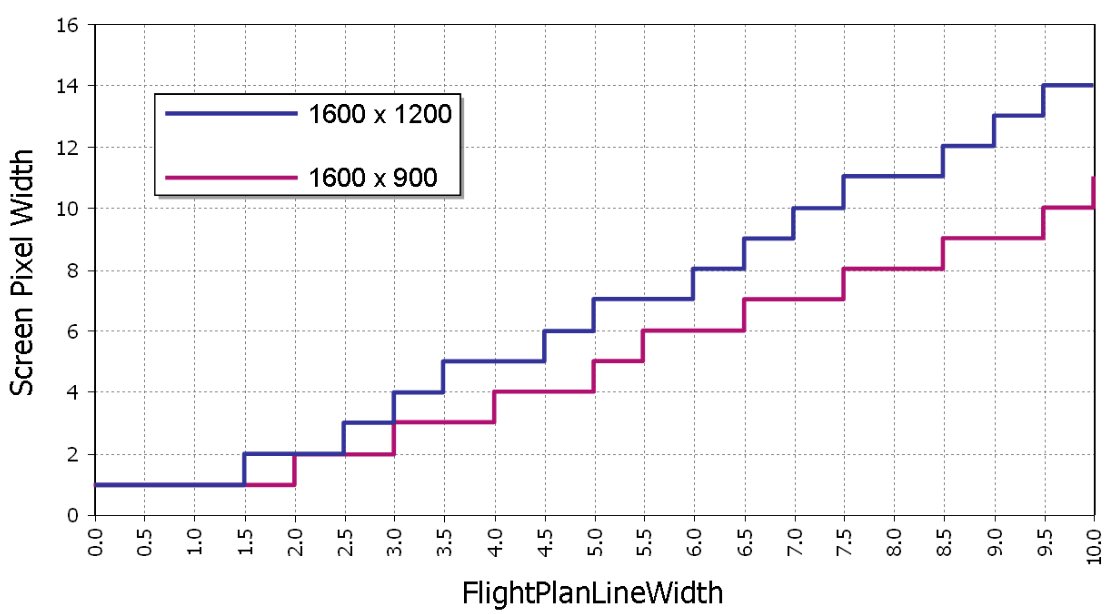
The stock gps_500.xml gauge uses a conditional statement within ColorLayerFlightPlan that sets the color to a medium gray, 0x808080, when no terrain background is showing, and white when it is.

⃠ TextColorLayerFlightPlan (BGR hexadecimal)

TextColorLayerFlightPlan is non-functional.

❑ FlightPlanLineWidth (number)

FlightPlanLineWidth controls the width of the Flight Plan line. It is approximately equal to screen pixel width rendered but can vary according to screen resolution and gauge configuration settings as demonstrated in the chart below. If FlightPlanLineWidth is omitted from the XML script or set equal to zero, a 1 screen pixel width line is drawn.



❑ ActiveColorLayerFlightPlan (BGR hexadecimal)

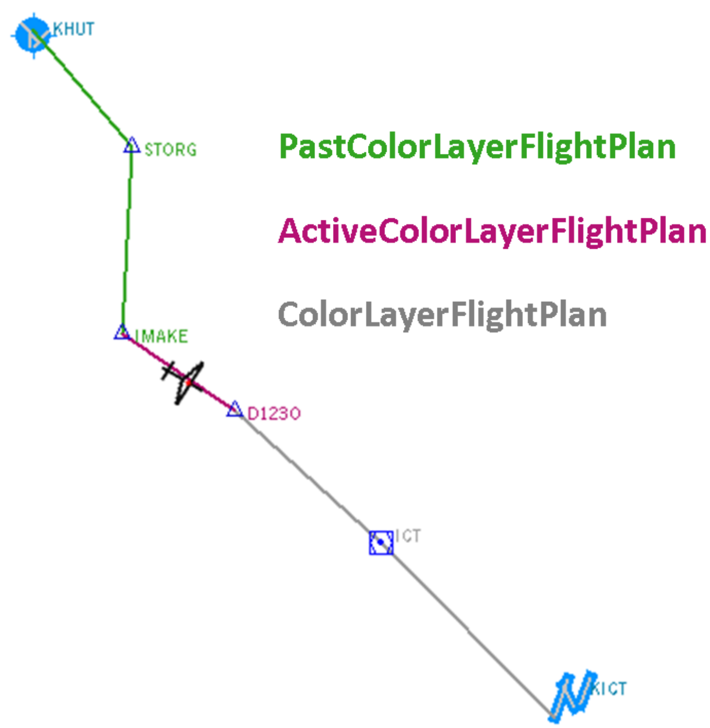
ActiveColorLayerFlightPlan is the color of the active Flight Plan or Approach segment. The default color if ActiveColorLayerFlightPlan is omitted from the XML script is a magenta shade:

Blue: **255** Green: **49** Red: **255** BGR Hex: **0xFF31FF**

If ObjectDetailLayerFlightPlan Waypoints bit is set, then ActiveColorLayerFlightPlan also controls the color of the active waypoint Ident text. ActiveColorLayerFlightPlan overrides ColorLayerFlightPlan for the active segment.

❑ PastColorLayerFlightPlan (BGR hexadecimal)

PastColorLayerFlightPlan is the color of all past, or completed, Flight Plan segments. As shown below, PastColorLayerFlightPlan also controls the color of past waypoints Ident text.



LayerApproach

LayerApproach draws a map of approach procedures identified by WaypointAirportICAO, WaypointAirportCurrentApproach and WaypointAirportCurrentTransition selections. This layer is limited to the approach procedure and does not include any of the enroute flight plan legs.

The screen capture on the right shows the FS9 and FSX Garmin GPS 500 Procedures Page after KICT ILS 19R Approach, ICT Transition has been selected. The insert map that displays this approach procedure uses variables of the LayerApproach group. In the stock gps_500 gauge, it is set up as a separate CustomDraw element (refer to lines 2720 through 2751 of the FSX gps_500.xml); it's not part of the main CustomDraw fs9gps:Map element (lines 756 through 783 of the FSX gps_500.xml).

After an approach has been loaded, it becomes part of the flight plan and will be rendered in the main map as part of the LayerFlightPlan group.

❑ LayerApproach (bool)

LayerApproach controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerApproach> 1 </LayerApproach>
```

❑ DetailLayerApproach (decimal or hexadecimal)

DetailLayerApproach controls the approach segments that are displayed. A Decimal or Hexadecimal number is used that is in the form of a bit table filter similar to filters in Nearest searches (reference: GPS Guidebook NearestIntersectionCurrentFilter, page 62-63).

Bit #	Name	Bit #	Name	Bit #	Name
0	Approach	1	Missed	2	Arrow Head

As an example, to draw the approach and missed approach segments, bits 0 and 1 are selected:

ARROW HEAD	MISSED	APPROACH	
4	2	1	- Decimal equivalent
2	1	0	- Bit number (Bit 0 thru Bit 2)
0	**1**	**1**	- DetailLayerApproach selections

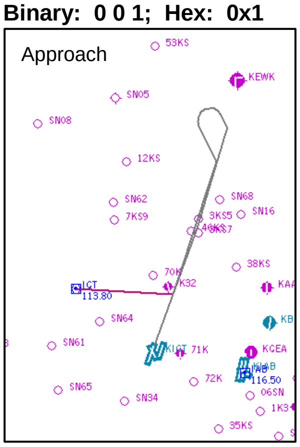
The decimal equivalent of binary **0 1 1** is 3. The hexadecimal value is likewise 3. The XML instruction is:

<DetailLayerApproach> 3 </DetailLayerApproach> or

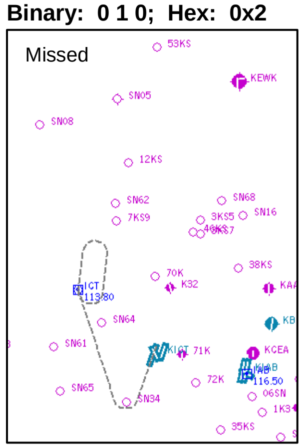
<DetailLayerApproach> 0x3 </DetailLayerApproach>

KICT: ILS 19R Approach, ICT Transition

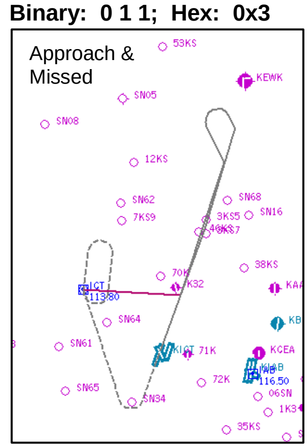
Binary: 0 0 1; Hex: 0x1



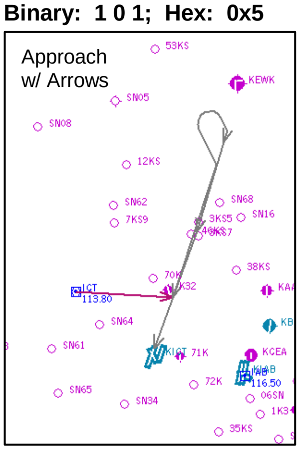
Binary: 0 1 0; Hex: 0x2



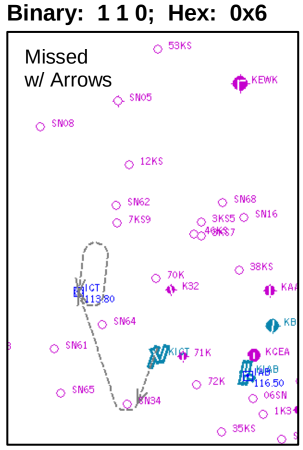
Binary: 0 1 1; Hex: 0x3



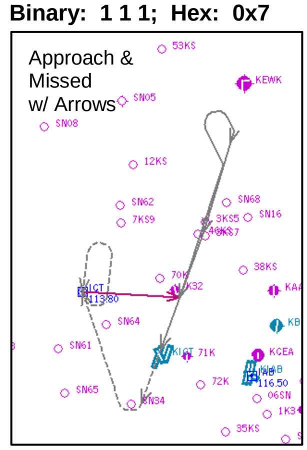
Binary: 1 0 1; Hex: 0x5



Binary: 1 1 0; Hex: 0x6



Binary: 1 1 1; Hex: 0x7



The default DetailLayerApproach value is 7, or 0x7.

⃠ TextDetailLayerApproach (enum)

Use of TextDetailLayerApproach has no effect in either FS9 or FSX. There is no text label associated with LayerApproach.

❑ ObjectDetailLayerApproach (bool)

Any number other than 0 will display the approach segments selected by DetailLayerApproach. A zero results in no rendering of the approach segments. This has the same effect as DetailLayerApproach = 0, and as such is of little use.

⃠ ~~ColorLayerApproach~~ (BGR hexadecimal)

Use of ColorLayerApproach appears to crash the approach map in both FS9 and FSX. This variable should not be used.

⃠ TextColorLayerApproach (BGR hexadecimal)

Use of TextColorLayerApproach has no effect in either FS9 or FSX. There is no text label associated with LayerApproach.

❑ LayerApproachAirport (string)

LayerApproachAirport is the fs9gps ICAO identity of the approach airport. It is the full ICAO, not the Ident. Equivalent to WaypointAirportICAO.

❑ LayerApproachAproach (enum)

LayerApproachAirport is the index pointer for the airport approach list. Equivalent to WaypointAirportCurrentApproach. This index pointer is used to select a specific approach procedure to display.

❑ LayerApproachTransition (enum)

LayerApproachTransition is the index pointer for the approach transitions list. Equivalent to WaypointAirportCurrentTransition. This index pointer is used to select a specific approach transition to display.

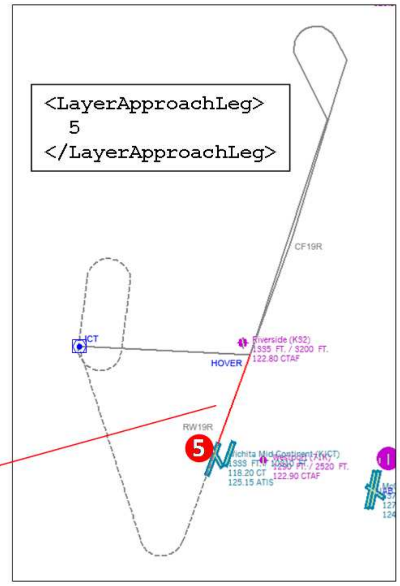
❑ LayerApproachLeg (enum)

LayerApproachLeg is an index pointer to the approach and missed approach segments. It's equivalent to FlightPlanWaypointApproachIndex and selecting an index number will cause the associated leg to be highlighted on the approach map. The KICT ILS 19R example below shows a 9 waypoint approach which results in 8 approach legs. Valid choices of LayerApproachLeg are 0 through 8. Leg 5 has been selected, resulting in display of that leg using LayerApproachLineActiveColor, which in this example, is red.

```
FLIGHT PLAN WAYPOINT APPROACH

KICT :FlightPlanApproachAirportIdent
  13 :FltPlnApprType   9 :FltPlnApprWptsNum
  ILS 19R :FltPlnApprName      ICT :FPTransName

--------- FlightPlanWaypointApproach ---------
    ICAO      111
Idx 123456789012   Name  Type Mode Course   Dist
0   VK3    ICT      ICT   1     1    -1.0    0.0
1   WK3KICTHOVER   HOVER  1     1    93.1    8.8
2   WK3KICTHOVER   HOVER  3     1   328.0   21.3
3   WK3KICTCF19R   CF19R  1     2   197.0    8.9
4   WK3KICTHOVER   HOVER  1     2   193.0    6.3
5   RK3KICTRW19R   RW19R  1     2   193.0    4.8
6                         9     3   193.0    5.3
7   VK3    ICT      ICT   1     3   344.7   13.7
8   VK3    ICT      ICT   6     3   180.0   14.3
```



The default LayerApproachLeg value is 0, which is not associated with an approach leg, so nothing is highlighted except in the case of a Vectors transition. When a Vectors transition is selected, a 5 nm leg is added to the Final Approach Fix and highlighted.

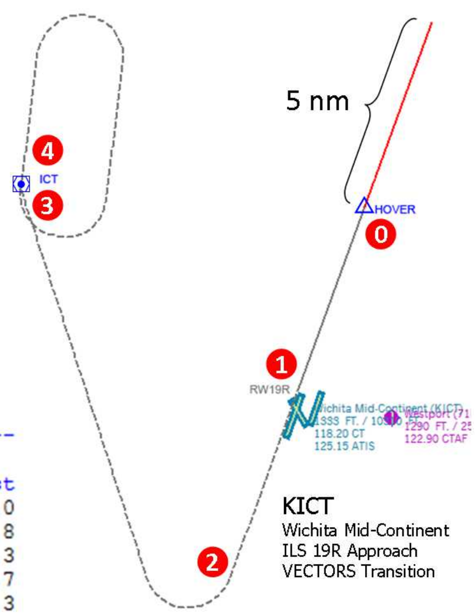
Vectors Transition

<LayerApproachLeg>
0
</LayerApproachLeg>

```
FLIGHT PLAN WAYPOINT APPROACH

KICT :FlightPlanApproachAirportIdent
  13 :FltPlnApprType   5 :FltPlnApprWptsNum
 ILS 19R :FltPlnApprName
 VECTORS :FPTransName

--------- FlightPlanWaypointApproach ---------
    ICAO      111
Idx 123456789012   Name  Type Mode Course  Dist
0   WK3KICTHOVER   HOVER  11    2   193.0   5.0
1   RK3KICTRW19R   RW19R  1     2   193.0   4.8
2                         9     3   193.0   5.3
3   VK3    ICT      ICT   1     3   344.7  13.7
4   VK3    ICT      ICT   6     3   180.0  14.3
```



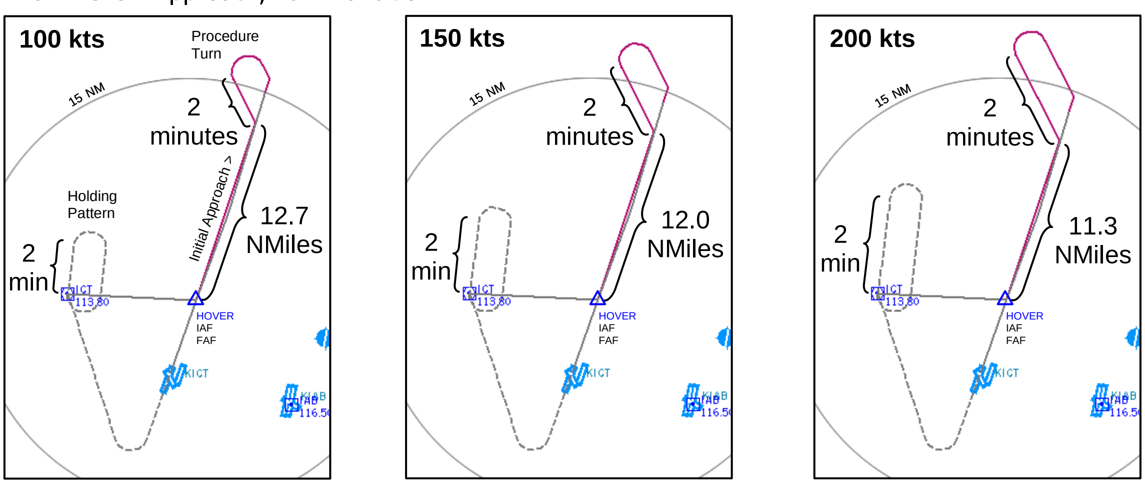
❑ LayerApproachAircraftSpeed (number, knots)

An aircraft's approach groundspeed affects the length of several approach segments depicted on the approach map. In the example that follows, the length of the outbound leg of the Initial Approach, entry into the 45° Procedure Turn (PT), and length of the Holding Pattern legs are all a functions of groundspeed and are rendered according to LayerApproachAircraftSpeed. LayerApproach assumptions include:

- A two minute 45° straight segment prior to initiating the 180° turn. The standard for this timed sub-segment is 1 minute for Category A and B aircraft and 1.25 min for Category C, D, and E, so Flight Simulator's choice is excessive.
- Two minute legs in the Holding Pattern
- Variable length outbound Initial Approach leg segment. This leg must be shortened as speed increases in order complete the PT within the 15 NM maneuvering limit from the PT Fix (HOVER intersection). Flight Simulator comes close, but does not quite accomplish this. Distances of 17.0, 17.5 and 18.3 NM for LayerApproachAircraftSpeed 100, 150 and 200 knots are rendered. The exaggerated duration of the 45° straight segment has a lot to do with this.
- The default LayerApproachAircraftSpeed is 1.3 times Flaps_Up_Stall_Speed found in the aircraft.cfg file. 1.3 times Full_Flaps_Stall_Speed would have been a more logical choice because reference approach speed is defined as 1.3 V_{SO}.

LayerApproachAircraftSpeed – Flight Simulator Assumptions

KICT ILS19R Approach, ICT Transition

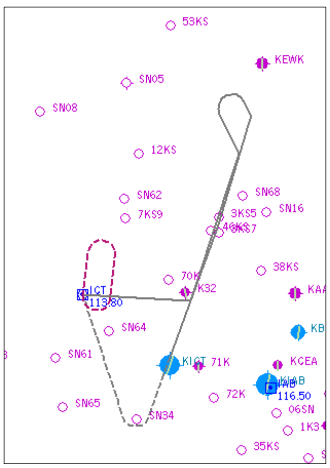


❑ LayerApproachLineActiveColor (BGR hexadecimal)

LayerApproachLineActiveColor is the color applied to the LayerApproachLeg segment. The default LayerApproachLineActiveColor is a dark magenta shade:

Blue: **107** Green: **27** Red: **137** BGR Hex: **0x6B1B89**

An example of LayerApproachLeg and LayerApproachLineActiveColor:



XML for this map:

```
<DetailLayerApproach>
  0x3
</DetailLayerApproach>

<LayerApproachLeg>
  8
</LayerApproachLeg>

<LayerApproachLineActiveColor>
  0x7010B0
</LayerApproachLineActiveColor>
```

⃠ ~~LayerApproachLineColor~~ (BGR hexadecimal)

Use of LayerApproachLineColor appears to crash the approach map in both FS9 and FSX. This variable should not be used.

The default, and <u>uneditable</u> approach line color is a very pale gray:

Blue: **247** Green: **247** Red: **247** BGR Hex: **0xF7F7F7**

❑ LayerApproachLineWidth (enum)

Screen pixel line width of the approach and missed approach segments. The default is 1 pixel.

Other LayerApproach Observations

To replicate the approach select function of the GPS 500 gauge, the Approach layer should be set up as a separate CustomDraw element apart from the main map. All variables needed to render a map such as BackgroundColor, Zoom, Latitude and Longitude as well as other desired layers like VORs and NDBs need to be included in this element.

There are a few unique considerations for the Approach map:

- BackgroundColor must be dark. Other than LayerApproachLineActiveColor, LayerApproach renders approach segments in a near-white color only (RGB 247, 247, 247; 0xF7F7F7). Although LayerApproachLineColor seems to be the logical color choice for the rest of the segments, its use crashes the map. *(Note: Colors of the approach maps in this section were edited to eliminate dark print images)*
- Latitude and Longitude should be set appropriate for the approach rather than the usual aircraft lat/lon. The stock gps_500.xml provides a good example, WaypointAirportApproachTransitionLatitude and Longitude.
- LayerRangeRings should not be used because range rings are always centered around the user aircraft position (A:PLANE LATITUDE, radians) and (A:PLANE LONGITUDE, radians), not ApproachTransitionLatitude and Longitude.
- TrackUp should be set to 0, that is, to True North.

LayerVehicles

FSX Only

LayerVehicles draws User, AI and Multiplayer on-ground and airborne *aircraft* traffic targets. Its primary function is to replicate an Air Traffic Control radar screen for use with the Flight Simulator Tower feature available in FSX Deluxe.

In LayerVehicles, 'vehicles' means aircraft, not ground vehicles like trucks or leisure boats, although it *may* be possible that LayerVehicles can draw commercial boat (Ships and Ferries) targets; the SDK indicates that **Traffic Tools** can view and customize AI ("artificial intelligence" or computer-controlled) aircraft and Ships and Ferries boat traffic. But, I have no knowledge about a connection between Traffic Tools and LayerVehicles nor have I seen a Ship or Ferry boat target ever painted by LayerVehicles.

A review of LayerVehicles itself is reasonably simple and straightforward. The related ITrafficInfo group, on the other hand, is of greater interest and provides much more insight into what can be done with traffic information.

❑ LayerVehicles (bool)

Layervehicles controls display of the layer. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

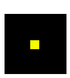
```
<LayerVehicles> 1 </LayerVehicles>
```

❑ DetailLayerVehicles (enum)

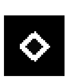
DetailLayerVehicles determines the style of aircraft symbol displayed.

-1 = Default. No symbol is drawn

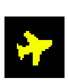
0 = Draw Nothing. No symbol is drawn



1 = ATC Symbol. Color can be changed

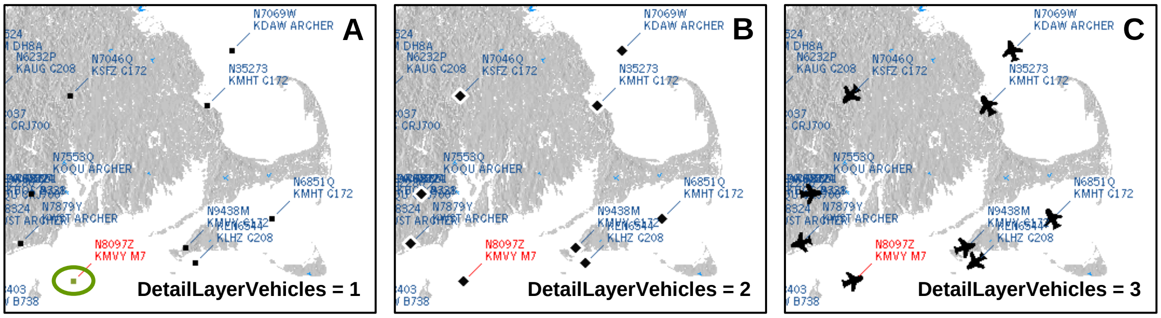


2 = TCAS Symbol. No color choice with this symbol. Always a white diamond with black fill



3 = Realistic Symbol. But not too realistic looking. Color can be changed

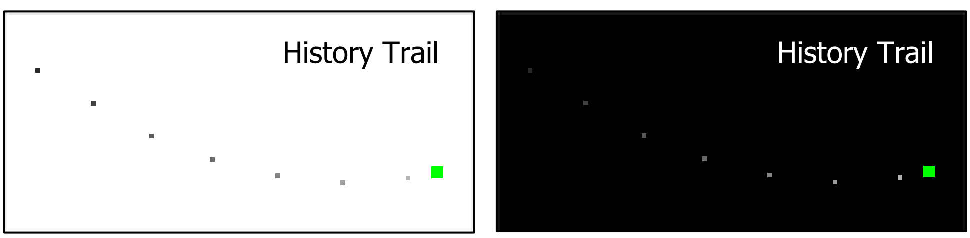
The figures below demonstrate DetailLayerVehicles styles when displayed on the map.



A few observations:

- **Figure A.** DetailLayerVehicles = 1. A good choice as it produces the cleanest looking display. Although the Selected aircraft symbol in the lower left corner is colored olive green (ColorLayerVehiclesSelected), the label color of the Selected aircraft is always red. Size of the aircraft symbol cannot be changed.

  DetailLayerVehicles = 1 includes a History Trail as shown below. Every few seconds, the aircraft leaves 'breadcrumbs' showing where it has been.



History Trail is part of DetailLayerVehicles = 1 and cannot be turned off.

- **Figure B.** DetailLayerVehicles = 2. Always a white diamond with black fill. This symbol appears best on a dark background. No History Trail, but Track Line can be displayed for this symbol. Size of the aircraft symbol cannot be changed.
- **Figure C.** DetailLayerVehicles = 3. Can produce a congested looking display. The color of the Selected aircraft remains the same color as the rest of the aircraft. No History Trail. Size of the aircraft symbol cannot be changed.

If DetailLayerVehicles is not included in the XML code, no aircraft symbol will be drawn.

❑ TextDetailLayerVehicles (enum)

TextDetailLayerVehicles controls the format of aircraft flight status information displayed in the text label for each aircraft.

Draw nothing — **-1 = Default.** No text label is drawn

Draw nothing — **0 = Draw Nothing.** No text label is drawn

1 = Realistic. Five items of information are displayed on two lines of text that alternate back and forth about every two seconds.

2 = Detailed. Five items of information are displayed on five lines.

The flight status information consists of:

1. **Aircraft Call Sign.** In this example, N2678Q
2. **Aircraft Model.** For example, LJ45
3. **Destination Airport.** This is the Ident of the destination waypoint of the AI or Multiplayer aircraft. In this example, PAWG, Wrangell Airport, Wrangell Alaska.
4. **Altitude.** In Realistic Format, it is Altitude (MSL) in 100s of feet. In this example, 319 = 31,900 feet. Output is in US - Imperial units (feet, knots) even if simulation settings are metric. In real life, this is the Mode C standard pressure altitude reported in hundreds of feet by the aircraft transponder.
5. **True Airspeed.** In Realistic Format, True Airspeed is represented in 10s of knots. In this example, 44 = 440 knots. Output is in US - Imperial units even if the sim settings are metric. In real life, this is **Groundspeed** of course. Why does FS use True Airspeed when ITrafficInfo can access A:GROUND VELOCITY?

If TextDetailLayerVehicles is not included in the XML code, no text label will be drawn.

❑ ObjectDetailLayerVehicles (decimal or hexadecimal)

ObjectDetailLayerVehicles controls what is drawn. Usually, this is "Airborne' and 'Ground' vehicles (aircraft on the ground), and the bit selection is:

8	4	2	1	- Decimal equivalent
RACING VEHICLES	AIRBORNE VEHICLES	GROUND VEHICLES	TRACK LINE	
3	2	1	0	- Bit number (Bit 0 thru Bit 3)
0	**1**	**1**	**0**	- ObjectDetailLayervehicles selections

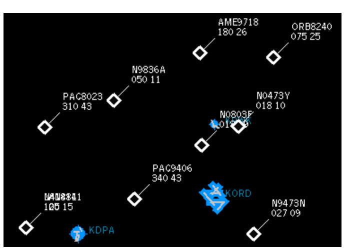
The resulting binary number is **0 1 1 0** whose decimal equivalent is 6 and hexadecimal equivalent is also 6. The appropriate XML is either:

```
<ObjectLayerDetailVehicles> 6 </ObjectLayerDetailVehicles>
```

or

```
<ObjectLayerDetailVehicles> 0x6 </ObjectLayerDetailVehicles>
```

- **Racing Vehicles:** Presumed that aircraft involved in a Race Mission are drawn.
- **Airborne Vehicles:** All Airborne AI or Multiplayer aircraft within the map boundaries are drawn.
- **Ground Vehicles:** All Awake Ground AI or Multiplayer aircraft within the map boundaries are drawn.
- **Track Line:** Displays a short track line indicating current A:PLANE HEADING DEGREES MAGNETIC for each AI or Multiplayer airborne aircraft. This is available only with DetailLayerVehicles = 2, TCAS. The track line is always white like the TCAS symbol border, consequently, a non-white background is necessary in order to see Track Line. Length of the Track Line is proportional to True Airspeed and it points in direction the aircraft is heading to unlike History Trails that show where the aircraft has come from.

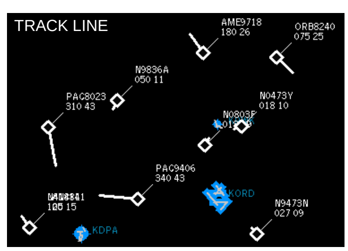


ObjectDetailLayerVehicles = 4

1 = Airborne Vehicles

0 = Ground Vehicles

0 = Track Line

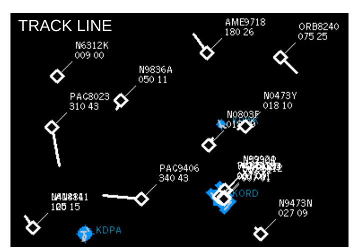


ObjectDetailLayerVehicles = 5

1 = Airborne Vehicles

0 = Ground Vehicles

1 = Track Line



ObjectDetailLayerVehicles = 7

1 = Airborne Vehicles

1 = Ground Vehicles

1 = Track Line

❑ ColorLayerVehicles (BGR hexadecimal)

ColorLayerVehicles controls color of the "ATC" and "Realistic" aircraft symbol (DetailLayerVehicles 1 and 3). Its format is hexadecimal Red-Green-Blue.

In the event that ColorLayerVehicles is not included the XML script, the default color is yellow:

Blue: **0** Green: **247** Red: **247** BGR Hex: **0x00F7F7**

❑ ColorLayerVehiclesSelected (BGR hexadecimal)

ColorLayerVehiclesSelected controls color of the Selected vehicle. This applies only to the "ATC" aircraft symbol (DetailLayerVehicles 1). The "TCAS" and "Realistic" symbols do not change color if Selected. Its format is hexadecimal Red-Green-Blue.

In the event that ColorLayerVehiclesSelected is not included the XML script, the default color is yellow:

Blue: **0** Green: **247** Red: **247** BGR Hex: **0x00F7F7**

❑ TextColorLayerVehicles (BGR hexadecimal)

TextColorLayerVehicles controls the color of the text label for all three aircraft symbol types. Applies to non-Selected aircraft only. Its format is hexadecimal Red-Green-Blue.

In the event that TextColorLayerVehicles is not included the XML script, the default color is magenta:

Blue: **255** Green: **0** Red: **255** BGR Hex: **0xFF00FF**

The text color of the Selected aircraft label is always red, regardless of the color of the Selected aircraft symbol itself.

ITrafficInfo: Nearest Traffic Group

FSX Only

ITrafficInfo is a Nearest search group analogous to other fs9gps Nearest groups such as NearestAirport. It returns AI or multiplayer aircraft traffic nearest the search origin that is normally defined as the user's aircraft or control tower position. It sorts data by ascending distance. Like all Nearest searches, ITrafficInfo returns an indexed list and a current line number (or pointer, index) must be supplied to obtain data about a specific aircraft. An ITrafficInfo search can return User's aircraft, multiplayer aircraft, AI aircraft, and AI ground vehicles such as airport trucks, ships and boats.

The ITrafficInfo variables retrieve flight and communication data of individual AI or multiplayer aircraft (i.e., 'Vehicles'). As referenced in the SDK, by using XML instructions a large number of Simulation Variables (A:Vars) can be retrieved each update cycle for every aircraft. The table below is a non-exhaustive sample.

- **ENGINE DATA**
  NUMBER OF ENGINES
  ENGINE TYPE
  PROP1 RPM
  TURB ENG1 N1
- **POSITION AND SPEED DATA**
  GROUND VELOCITY
  PLANE ALT ABOVE GROUND
  PLANE LATITUDE
  PLANE LONGITUDE
  PLANE ALTITUDE
  PLANE PITCH DEGREES
  PLANE BANK DEGREES
  PLANE HEADING DEGREES TRUE
  PLANE HEADING DEGREES MAGNETIC
- **FLIGHT INSTRUMENTATION DATA**
  AIRSPEED TRUE
  VERTICAL SPEED
  ATTITUDE INDICATOR PITCH DEGREES
  ATTITUDE INDICATOR BANK DEGREES
- **AVIONICS DATA**
  COM1 TRANSMIT
  COM1 ACTIVE FREQUENCY
  COM1 STANDBY FREQUENCY
  NAV1 ACTIVE FREQUENCY
  NAV1 AVAILABLE
  ADF1 ACTIVE FREQUENCY
  TRANSPONDER1 CODE
- **CONTROLS DATA**
  RUDDER POSITION
  ELEVATOR POSITION
  AILERON POSITION
  IS GEAR RETRACTABLE
  AILERON LEFT DEFLECTION
  AILERON RIGHT DEFLECTION
- **MISCELLANEOUS SYSTEMS DATA**
  ELECTRICAL MASTER BATTERY
  CIRCUIT AVIONICS ON
- **MISCELLANEOUS DATA**
  DESIGN SPEED VS0
  EMPTY WEIGHT
  SIM ON GROUND
- **STRING DATA**
  ATC TYPE
  ATC MODEL
  ATC ID
  ATC AIRLINE
  ATC FLIGHT NUMBER

Some variables, however, such as Fuel data (stock AI aircraft never run out of fuel or experience emergencies), the A:GPS variables, and Autopilot data are not retrievable from ITrafficInfo. Experiment to see which variables are retrievable.

- ❑ **ITrafficInfo:Latitude**
- ❑ **ITrafficInfo:Longitude (degrees or radians) [Get, Set]**

Latitude and Longitude of the reference point, usually the aircraft or control tower. Default is A:PLANE LATITUDE and LONGITUDE.

- ❑ **ITrafficInfo:MaxVehicles (enum) [Get, Set]**

The limit of the number of aircraft returned by the search. The larger the number, the longer it takes for the ITrafficInfo search to complete. It's good practice to keep ITrafficInfo:MaxVehicles to an appropriate size for the application. As an example, in a TCAS gauge, a maximum of 30 is a proper choice for ITrafficInfo:MaxVehicles. Default is 200 vehicles. Maximum is approximately 250 to 260. Any value set larger than this will likely crash the simulation – a memory issue, I think.

- ❑ **ITrafficInfo:Radius (meters, NMiles) [Get, Set]**

Maximum search radius. AI aircraft beyond ITrafficInfo:Radius will still be *displayed* by LayerVehicles on the map, but only those aircraft returned in the ITrafficInfo nearest traffic search will have accessible information. Default is 43 NMiles. AI aircraft are generated up to a maximum distance of 100 NM, so when working with AI traffic, there is no need to set Radius larger than 100 NM.

- ❑ **ITrafficInfo:Filter (enum or hexadecimal) [Get, Set]**

ITrafficInfo:Filter filters the Nearest Traffic search to include or exclude certain types and categories of aircraft or vehicles according to seven filter criteria:

- AWAKE. Bit #6. Awake are active ground or airborne aircraft. Setting this filter will include Awake aircraft in the search results. Only 'Awake' aircraft can be displayed on the map (i.e., radar screen). Awake and active does not necessarily mean that the aircraft is moving. It can be holding short, for example. Filters AI but not Multiplayer searches.
- SLEEPING. Bit #5. 'Sleeping' are ground AI aircraft that have been generated (i.e., spawned) by Flight Simulator but are not yet an active participant in the simulation. They have an aircraft Call Sign consisting of ATC Airline and Flight Number or ATC ID (e.g., SOA7192), Model (e.g., A321), a two waypoint Flight Plan (Departure and Destination airport) and a unique VehicleID. Variables associated with sleeping aircraft can be listed, but the aircraft symbol will not display on the map until it is awakened by Flight Simulator. Sleeping aircraft are initially positioned at airport gates and parking ramps as demonstrated in the figures that follow.

  Note that this filter *adds* 'Sleeping' ground aircraft. The list of aircraft returned always includes 'Awake' aircraft. Filters AI but not Multiplayer searches.

- IN_AIR. Bit #4. Airborne aircraft are included in the search results. By definition, these will also be Awake aircraft. The search condition is the same as (A:SIM ON GROUND, bool) = 0. This filter operates in AI as well as Multiplayer traffic searches.
- ON_GROUND. Bit #3. Ground aircraft/vehicles will be included in the search. The search condition is the same as (A:SIM ON GROUND, bool) = 1. This filter operates in AI as well as Multiplayer traffic searches.

Either IN_AIR or ON_GROUND (Bit #4 or Bit #3), or both, must <u>always</u> be selected in order for the Nearest Traffic search to function.

- TOWER_CONTROLLERS. Bit #2. I am not sure of the function of this bit. In my experience, it appears to have no effect in either single player free flight mode or multiplayer mode, so its function remains a mystery to me. If anyone knows what this does, please shoot me an email.
- GROUND_VEHICLES. Bit #1. This Bit enables AI ground vehicles *other than aircraft* to be included in the Nearest Traffic search. These include airport vehicles (fire trucks, bag tractors, etc.), road vehicles, ships and ferries and leisure boats. They can be either stationary or moving.

  Ground <u>aircraft</u> are included in the traffic search whenever Bit #3 ON_GROUND is selected, regardless of whether this Bit #1, GROUND_VEHICLES, is selected or not. As a consequence, it is not possible to isolate non-aircraft ground vehicles like trucks or boats by selecting Bits 3 and 1. ON_GROUND, Bit #3, must also be enabled whenever Bit #1 is selected.

  Note of interest: In multiplayer mode, the Air Traffic Controller (SimObjects\Misc\ControlTower) is returned in the traffic search when Bit #4 IN_AIR and Bit #1 GROUND_VEHICLE are both selected. As far as Flight Simulator is concerned, the Tower is both on the ground and in the air (control tower is elevated above the airport surface) and the height of the control tower can be displayed using (C:ITrafficInfo:C:PLANE ALT ABOVE GROUND, feet). This is also one way to see the lat/lon coordinates of the control tower.
- AIRCRAFT. Bit #0. I cannot determine the function of this variable. Aircraft are <u>always</u> included in a Nearest Traffic search if this bit is 1 or 0.

If ITrafficInfo:Filter is not included in the xml script, the default is decimal 89, equivalent to binary **1 0 1 1 0 0 1** ('Awake', 'In_Air', 'On_Ground', 'Aircraft')**.**

ITrafficInfo:Filter can be changed by user input at any time during the sim to alter the Nearest Traffic results "on the fly".

Lastly, note that AI traffic is not possible in Multiplayer mode.

Designating the Filter Value

As an example, if Airborne traffic is to be included in the nearest search, an appropriate selection is bit #6 and bit #4, 'Awake' and 'In_Air' as follows:

64	32	16	8	4	2	1	- Decimal equivalent
AWAKE	SLEEPING	IN_AIR	ON_GROUND	TOWER CONTROLLERS	GROUND VEHICLES	AIRCRAFT	
6	5	4	3	2	1	0	- Bit number (Bit 0 thru Bit 6)
1	**0**	**1**	**0**	**0**	**0**	**0**	- ITrafficInfo:Filter selections

The resulting binary number is **1 0 1 0 0 0 0**. Its decimal equivalent is 80 and hexadecimal equivalent is 50. The appropriate XML is therefore either:

`80 (>C:ITrafficInfo:Filter)` or

`0x50 (>C:ITrafficInfo:Filter)`

Sleep State

Sleep state has no influence on 'In_Air' aircraft; airborne vehicles are all 'Awake' by definition. Consequently, all of the following yield the same search results:

64	32	16	8	4	2	1	- Decimal equivalent
AWAKE	SLEEPING	IN_AIR	ON_GROUND	TOWER CONTROLLERS	GROUND VEHICLES	AIRCRAFT	
6	5	4	3	2	1	0	- Bit number (Bit 0 thru Bit 6)
0	**0**	**1**	**0**	**0**	**0**	**0**	- ITrafficInfo:Filter selections
0	**1**	**1**	**0**	**0**	**0**	**0**	- ITrafficInfo:Filter selections
1	**0**	**1**	**0**	**0**	**0**	**0**	- ITrafficInfo:Filter selections
1	**1**	**1**	**0**	**0**	**0**	**0**	- ITrafficInfo:Filter selections

Sleep state <u>does</u> affect the search of Ground aircraft, however. If all 'Awake' 'Ground' aircraft are to be included in the search, the selection would be:

64	32	16	8	4	2	1	- Decimal equivalent
AWAKE	SLEEPING	IN_AIR	ON_GROUND	TOWER CONTROLLERS	GROUND VEHICLES	AIRCRAFT	
6	5	4	3	2	1	0	- Bit number (Bit 0 thru Bit 6)
1	**0**	**0**	**1**	**0**	**0**	**0**	- ITrafficInfo:Filter selections

Example XML:

`72 (>C:ITrafficInfo:Filter)` or

`0x48 (>C:ITrafficInfo:Filter)`

Selecting 'Sleep' *adds* sleeping Ground aircraft to the search results. 'Awake' is the default sleep state and is always included for both 'In_Air' and 'Ground' searches. There is no way to isolate just 'Sleeping' 'On_Ground' aircraft. Both of the following yield the same search results, namely, 'Awake' plus 'Sleeping' 'On_Ground' aircraft:

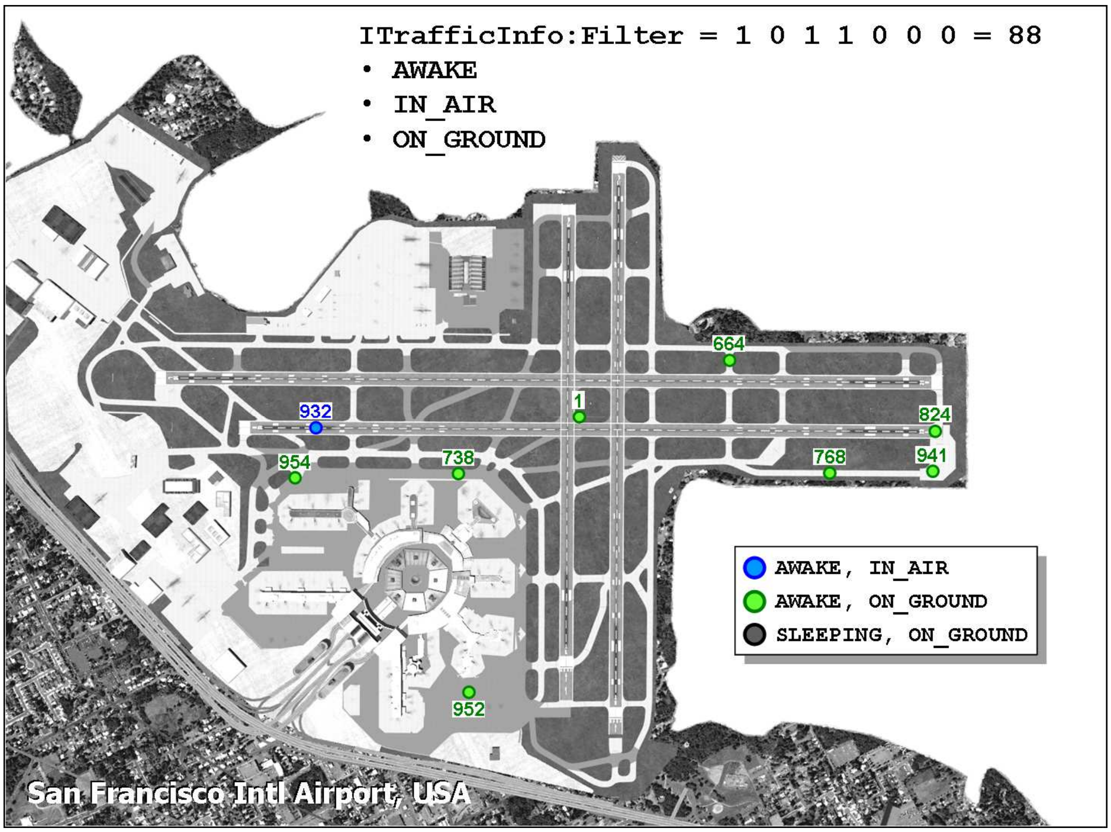
64	32	16	8	4	2	1	- Decimal equivalent
AWAKE	SLEEPING	IN_AIR	ON_GROUND	TOWER CONTROLLERS	GROUND VEHICLES	AIRCRAFT	
6	5	4	3	2	1	0	- Bit number (Bit 0 thru Bit 6)
1	**0**	**0**	**1**	**0**	**0**	**0**	- ITrafficInfo:Filter selections

Nearest Traffic Search Example

Next is an example of a Nearest Traffic search result for AI aircraft at San Francisco International Airport. It shows some of the types of data that can be retrieved for every aircraft during each gauge update cycle. The search radius was two NMiles so aircraft at nearby airports were not included. In this particular case, the search center is my user aircraft, N3968G, Vehicle ID #1, a Cessna 421 parked at the center of the airport facility. Alternatively, a Control Tower can be established as search center (the typical multiplayer setup for a Traffic Controller).

FILTER: Awake + Air + Ground **Binary 1 0 1 1 0 0 0; Decimal 88; Hexadecimal 0x58**

CUR IDX	CALL	MODEL	DIST	VID	LATITUDE	LONGITUDE	ALT	VSI	ON GND	MAG HDG	GND SPD	FLIGHT PLAN	
0	N3968G	C421	0.0	1	37.619200	-122.374750	18	0	1	13	0	KSFO, SUNOL	AWAKE
1	PAC3055	A321	0.3	738	37.619594	-122.382035	23	0	1	100	6	KSFO, KLAX	AWAKE
2	SOA6449	MD80	0.4	664	37.618443	-122.366445	21	0	1	13	6	KLAS, KSFO	AWAKE
3	WOR4124	B744	0.6	932	37.623854	-122.387304	443	1897	0	283	251	KSFO, KIAH	AWAKE
4	ORB9619	A321	0.6	768	37.612382	-122.364654	23	0	1	103	20	KSFO, KSLC	AWAKE
5	SOA2617	MD80	0.7	954	37.622370	-122.389393	21	0	1	62	9	KSFO, KBGM	AWAKE
6	ORB3479	A321	0.7	952	37.611284	-122.386258	23	0	1	85	5	KSFO, KCVG	AWAKE
7	AIR9811	MD80	0.9	824	37.611863	-122.358774	21	0	1	292	4	KSFO, KLAX	AWAKE
8	ORB3961	B744	0.9	941	37.610529	-122.359595	31	0	1	68	7	KSFO, KMIA	AWAKE

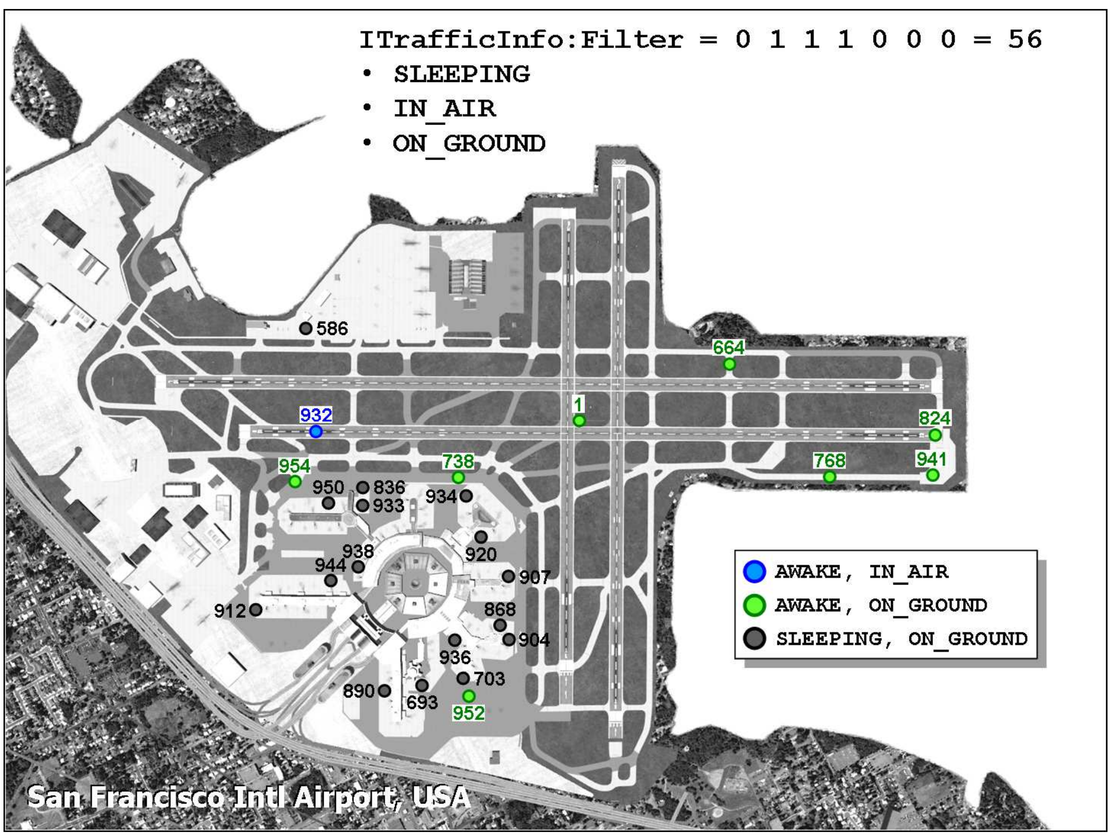


Cur Idx (Current Index), Dist (Distance), VID (Vehicle ID), and Flight Plan are SDK "documented" ITrafficInfo variables and are discussed later. As well, the simple XML required to retrieve other variables such as Latitude, Longitude, Altitude, etc is also addressed later.

Ground Vehicle ID 664 is a little noteworthy. It's an MD80 from Las Vegas McCarran that just landed Rwy 28R and is already on taxiway Papa, less than 2700' from the touch down zone. AI aircraft land hard and stop quickly in Flight Simulator.

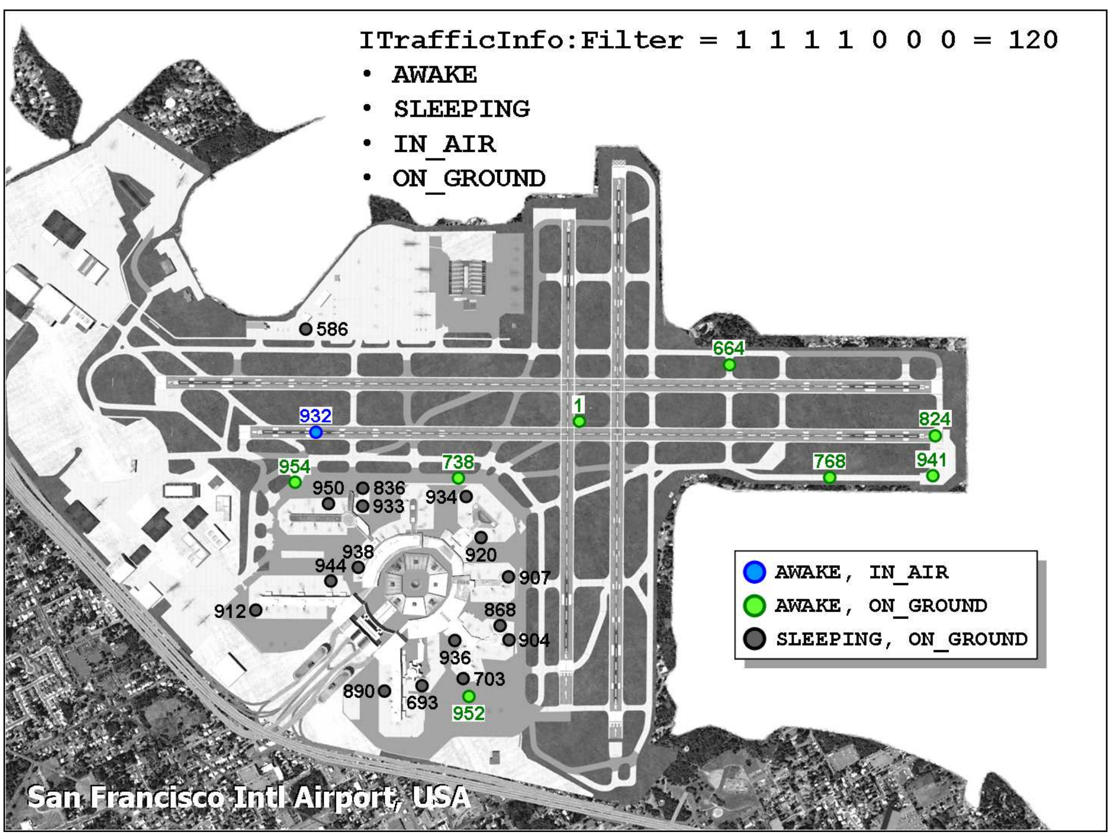
FILTER: Sleep + Air + Ground **Binary 0 1 1 1 0 0 0; Decimal 56; Hexadecimal 0x38**

CUR IDX	CALL	MODEL	DIST	VID	LATITUDE	LONGITUDE	ALT	VSI	ON GND	MAG HDG	GND SPD	FLIGHT PLAN	
0	N3968G	C421	0.0	1	37.619200	-122.374750	18	0	1	13	0	KSFO, SUNOL	AWAKE
1	ORB1123	B744	0.3	934	37.618599	-122.381857	31	0	1	169	0	KSFO, KDTW	SLEEPING
2	PAC3055	A321	0.3	738	37.619594	-122.382035	23	0	1	100	6	KSFO, KLAX	AWAKE
3	WOR1123	B744	0.4	920	37.616768	-122.382089	31	0	1	10	0	KSFO, CYUL	SLEEPING
4	SOA6449	MD80	0.4	664	37.618443	-122.366445	21	0	1	13	6	KLAS, KSFO	AWAKE
5	AIR1123	MD80	0.4	907	37.614761	-122.381686	21	0	1	269	0	KSFO, CYVR	SLEEPING
6	WOR	B738	0.5	868	37.613151	-122.383379	23	0	1	228	0	KSFO, KBUR	SLEEPING
7	WOR4124	B744	0.6	932	37.623854	-122.387304	443	1897	0	283	251	KSFO, KIAH	AWAKE
8	ORB	A321	0.6	836	37.620889	-122.386445	23	0	1	285	0	KSFO, KLAS	SLEEPING
9	PAC	A321	0.6	904	37.612537	-122.383202	23	0	1	284	0	KSFO, KALB	SLEEPING
10	ORB	B738	0.6	933	37.620574	-122.386610	23	0	1	287	0	KSFO, KMCI	SLEEPING
11	WOR	B738	0.6	936	37.613446	-122.385601	23	0	1	96	0	KSFO, KMCI	SLEEPING
12	ORB9619	A321	0.6	768	37.612382	-122.364654	23	0	1	103	20	KSFO, KSLC	AWAKE
13	WOR1123	B744	0.7	950	37.620978	-122.388331	31	0	1	193	0	KSFO, KATL	SLEEPING
14	SOA1123	MD80	0.7	938	37.618021	-122.388543	21	0	1	105	0	KSFO, KTUS	SLEEPING
15	ORB	CRJ700	0.7	703	37.612006	-122.386222	21	0	1	74	0	KSFO, KBUR	SLEEPING
16	AME1123	DH8A	0.7	586	37.627851	-122.385239	20	0	1	86	0	KSFO, KSCK	SLEEPING
17	SOA2617	MD80	0.7	954	37.622370	-122.389393	21	0	1	62	9	KSFO, KBGM	AWAKE
18	ORB3479	A321	0.7	952	37.611284	-122.386258	23	0	1	85	5	KSFO, KCVG	AWAKE
19	ORB	A321	0.7	944	37.617905	-122.390208	23	0	1	194	0	KSFO, CYVR	SLEEPING
20	PAC	CRJ700	0.8	693	37.612560	-122.388354	21	0	1	349	0	KSFO, KRDD	SLEEPING
21	ORB	B738	0.8	890	37.612739	-122.390091	23	0	1	97	0	KSFO, KONT	SLEEPING
22	AIR9811	MD80	0.9	824	37.611863	-122.358774	21	0	1	292	4	KSFO, KLAX	AWAKE
23	ORB3961	B744	0.9	941	37.610529	-122.359595	31	0	1	68	7	KSFO, KMIA	AWAKE
24	WOR	B738	0.9	912	37.618451	-122.394017	23	0	1	95	0	KSFO, KBWI	SLEEPING



FILTER: Awake + Sleep + Air + Ground **Binary 1 1 1 1 0 0 0; Decimal 120; Hexadecimal 0x78**

CUR IDX	CALL	MODEL	DIST	VID	LATITUDE	LONGITUDE	ALT	VSI	ON GND	MAG HDG	GND SPD	FLIGHT PLAN	
0	N3968G	C421	0.0	1	37.619200	-122.374750	18	0	1	13	0	KSFO, SUNOL	AWAKE
1	ORB1123	B744	0.3	934	37.618599	-122.381857	31	0	1	169	0	KSFO, KDTW	SLEEPING
2	PAC3055	A321	0.3	738	37.619594	-122.382035	23	0	1	100	6	KSFO, KLAX	AWAKE
3	WOR1123	B744	0.4	920	37.616768	-122.382089	31	0	1	10	0	KSFO, CYUL	SLEEPING
4	SOA6449	MD80	0.4	664	37.618443	-122.366445	21	0	1	13	6	KLAS, KSFO	AWAKE
5	AIR1123	MD80	0.4	907	37.614761	-122.381686	21	0	1	269	0	KSFO, CYVR	SLEEPING
6	WOR	B738	0.5	868	37.613151	-122.383379	23	0	1	228	0	KSFO, KBUR	SLEEPING
7	WOR4124	B744	0.6	932	37.623854	-122.387304	443	1897	0	283	251	KSFO, KIAH	AWAKE
8	ORB	A321	0.6	836	37.620889	-122.386445	23	0	1	285	0	KSFO, KLAS	SLEEPING
9	PAC	A321	0.6	904	37.612537	-122.383202	23	0	1	284	0	KSFO, KALB	SLEEPING
10	ORB	B738	0.6	933	37.620574	-122.386610	23	0	1	287	0	KSFO, KMCI	SLEEPING
11	WOR	B738	0.6	936	37.613446	-122.385601	23	0	1	96	0	KSFO, KMCI	SLEEPING
12	ORB9619	A321	0.6	768	37.612382	-122.364654	23	0	1	103	20	KSFO, KSLC	AWAKE
13	WOR1123	B744	0.7	950	37.620978	-122.388331	31	0	1	193	0	KSFO, KATL	SLEEPING
14	SOA1123	MD80	0.7	938	37.618021	-122.388543	21	0	1	105	0	KSFO, KTUS	SLEEPING
15	ORB	CRJ700	0.7	703	37.612006	-122.386222	21	0	1	74	0	KSFO, KBUR	SLEEPING
16	AME1123	DH8A	0.7	586	37.627851	-122.385239	20	0	1	86	0	KSFO, KSCK	SLEEPING
17	SOA2617	MD80	0.7	954	37.622370	-122.389393	21	0	1	62	9	KSFO, KBGM	AWAKE
18	ORB3479	A321	0.7	952	37.611284	-122.386258	23	0	1	85	5	KSFO, KCVG	AWAKE
19	ORB	A321	0.7	944	37.617905	-122.390208	23	0	1	194	0	KSFO, CYVR	SLEEPING
20	PAC	CRJ700	0.8	693	37.612560	-122.388354	21	0	1	349	0	KSFO, KRDD	SLEEPING
21	ORB	B738	0.8	890	37.612739	-122.390091	23	0	1	97	0	KSFO, KONT	SLEEPING
22	AIR9811	MD80	0.9	824	37.611863	-122.358774	21	0	1	292	4	KSFO, KLAX	AWAKE
23	ORB3961	B744	0.9	941	37.610529	-122.359595	31	0	1	68	7	KSFO, KMIA	AWAKE
24	WOR	B738	0.9	912	37.618451	-122.394017	23	0	1	95	0	KSFO, KBWI	SLEEPING



❑ ITrafficInfo:SortOrder

This variable is not implemented according to the SDK. Having said that, however, ITrafficInfo is a Nearest search and traffic returned by the search are ordered in ascending distance from the search origin (ITrafficInfo:Latitude and Longitude).

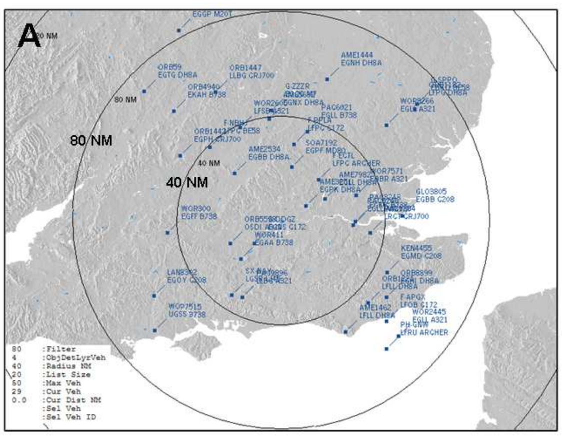
❑ ITrafficInfo:CurrentVehicle (enum) [Get, Set]

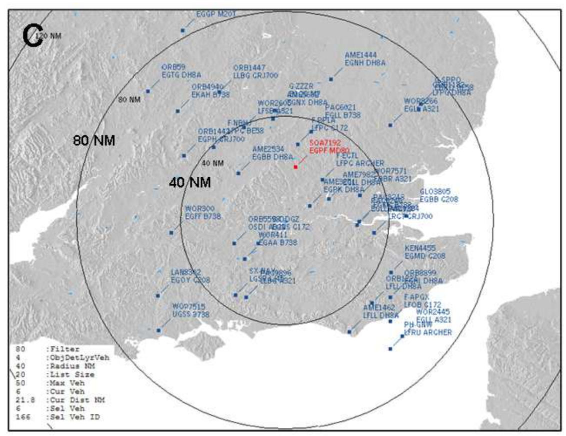
ITrafficInfo:CurrentVehicle is the index pointer for the nearest traffic search list. The first aircraft in the list is ITrafficInfo:CurrentVehicle 0. Example XML:

```
0 (>C:ITrafficInfo:CurrentVehicle)
```

❑ ITrafficInfo:SelectedVehicle (enum) [Get, Set]

ITrafficInfo:SelectedVehicle is an index pointer used to select a specific aircraft from the ITrafficInfo list in order to highlight its movement in contrast to all other aircraft on the radar screen. The aircraft must be included in the ITrafficInfo search results in order to be selected/highlighted. Only one aircraft can be Selected at a time.





B

IDX	CALL	MODEL	DIST	VID	LAT	LON	ALT	VSI	GND	HDG	GSPD
0	AME3251	DH8A	11.3	174	51.575	-0.207	7567	1669	0	329	205
1	G-DDGZ	C172	13.4	268	51.333	-0.734	6506	-172	0	322	111
2	AME7982	DH8A	19.0	167	51.618	-0.008	6338	-155	0	276	223
3	F-ECTL	ARCHER	21.4	275	51.739	-0.075	5455	149	0	148	120
4	WOR411	B738	21.7	190	51.233	-0.876	23990	232	0	320	416
5	ORB5598	A321	21.8	173	51.330	-0.981	25012	-146	0	111	448
6	SOA7192	MD80	21.8	166	51.823	-0.352	31933	217	0	331	411
7	AME2534	DH8A	25.8	221	51.781	-0.943	15610	-1489	0	349	262
8	PAC5246	A321	27.8	237	51.449	0.277	9120	-69	0	260	280
9	PAC3248	B738	28.6	200	51.471	0.301	8937	286	0	257	274
10	F-PPLA	C172	29.8	277	51.966	-0.329	5491	-251	0	148	116
11	WOR7571	A321	31.0	222	51.644	0.310	31007	-161	0	106	468
12	PAC9896	A321	33.1	192	50.992	-0.860	29008	-152	0	114	459
13	SX-NAY	LJ45	34.4	267	51.003	-0.962	27405	2307	0	120	402
14	PAC1304	CRJ700	34.4	202	51.397	0.447	13888	3062	0	100	363
15	PAC923	CRJ700	34.4	205	51.397	0.447	13873	3063	0	100	363
16	PAC6021	B738	35.6	144	52.045	-0.188	12051	369	0	233	343
17	WOR2605	A321	39.4	171	52.075	-0.891	15824	3143	0	128	380
18	AME5692	DH8A	39.4	165	52.129	-0.586	14021	-116	0	329	243
19	F-NBHJ	BE58	39.5	265	51.949	-1.197	7504	-90	0	141	184

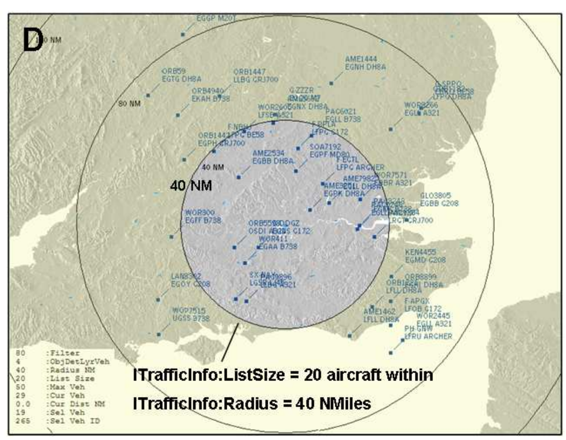


Figure **A** above is a traffic radar image around London Heathrow Airport. The Range is 80 NM. A nearest traffic search was enabled with a Filter value of 80 ('Awake' and 'In_Air'), a search radius of 40 NM, and maximum vehicle limit of 50. Additionally, LayerRangeRings, LayerTerrain and TerrrainShadow were enabled.

Figure **B** is a list of the 20 aircraft returned in the nearest Traffic search. If CurrentVehicle Index 6 is subsequently chosen, then the 'Selected' aircraft, **SOA7192**, can be highlighted in a different color as shown in Figure **C**. Until a new selection is made, SOA7192 will remain highlighted as its flight continues.

Figure **D** demonstrates that the radar screen will display all AI or multiplayer aircraft within the map boundary. Which aircraft are *displayed* on the map is controlled by ObjectDetailLayerVehicles (airborne and/or ground), not by ITrafficInfo:Filter. However, only the 20 aircraft returned from the nearest traffic search, that is, the aircraft that are within the 40 NM search radius, can be Selected or interrogated for various real-time data as shown in Table **B**.

Incidentally, a count of aircraft displayed on the map within the 40 NM search radius apparently results in 19, but the ITrafficInfo list contains 20. Note that Current Index 14 and 15 is essentially a duplicate aircraft. A common AI gen bug.

- **ITrafficInfo:SelectedVehicleID (enum) [Get]**

ITrafficInfo:SelectedVehicleID is a unique identification number automatically assigned by the traffic module to AI or multiplayer aircraft in order to enable selection/highlight of specific aircraft. The CurrentVehicle index pointer is not suitable for this purpose because it represents relative distances from the search origin at the point in time the search was made. As aircraft move around, the relative distance order constantly changes and the CurrentVehicle index of a particular aircraft may be 2 now, but could be a different number the next update cycle. On the other hand, SelectedVehicleID remains with the aircraft regardless of relative distance position until it is retired from the simulation by the traffic module.

Example XML script at the end of this section demonstrates the vehicle selection process.

- **ITrafficInfo:ListSize (enum) [Get]**

ITrafficInfo:ListSize is the number of aircraft returned by the nearest aircraft search. It is analogous to the Items number from other Nearest searches, for example, NearestAirportItemsNumber.

❑ ITrafficInfo:CurrentDistance (NMiles or km) [Get]

ITrafficInfo:CurrentDistance is the slant distance of each aircraft retrieved in the nearest traffic search from the search origin. Note that this is slant distance, not horizontal distance like GeoCalcDistance. ITrafficInfo:CurrentDistance incorporates the relative altitude difference between the search origin which is usually the user's aircraft, and the traffic aircraft.

A Note on Update Frequency

ITrafficInfo:CurrentDistance is updated every 2 seconds only by Flight Simulator.

However, other AI or multiplayer system variables are updated every gauge update cycle. For example:

- C:ITrafficInfo:C:PLANE LATITUDE
- C:ITrafficInfo:C:PLANE LONGITUDE
- C:ITrafficInfo:C:PLANE ALTITUDE
- C:ITrafficInfo:S:PLANE LATITUDE
- C:ITrafficInfo:S:PLANE LONGITUDE
- C:ITrafficInfo:S:PLANE ALTITUDE
- C:ITrafficInfo:C:AILERON LEFT DEFLECTION
- C:ITrafficInfo:C:PLANE ALT ABOVE GROUND, etc.

are all updated every gauge update cycle.

❑ ITrafficInfo:SelectedFlightPlan (String) [Get]

ITrafficInfo:SelectedFlightPlan is a list of Waypoint Idents of the flight plan for the selected aircraft. For AI aircraft, it consists only of the departure airport Ident (not ICAO as stated in the SDK) and the destination airport Ident. The SDK states that flight plans longer than two waypoints will be listed in comma separated format, however it appears that ITrafficInfo:SelectedFlightPlan will return the Idents of first two waypoints only of any flight plan.

In Multiplayer mode, ITrafficInfo:SelectedFlightPlan returns the Flight Plan string for the User's aircraft only, provided a Flight Plan is loaded. It will not return the Flight Plan for other players. Consequently, the ATC Controller function in Multiplayer will not

ITrafficInfo XML Script Examples

Example 1. Displaying a List of AI Aircraft Information

The first example demonstrates the set-up of the Nearest Traffic search and an example of an Element display for the list of aircraft retrieved in that search:

```
1  <Macro Name="CurrentCallsign">
2    (C:ITrafficInfo:C:ATC AIRLINE, string) d slen 0 &gt;
3      if{ 0 3 ssub uc (C:ITrafficInfo:C:ATC FLIGHT NUMBER, string) scat }
4      els{ (C:ITrafficInfo:C:ATC ID) d slen 0 == if{ (C:ITrafficInfo:CurrentPlayerName) } }
5  </Macro>
6
7  <Update>
8    (A:PLANE LATITUDE, radians) (>C:ITrafficInfo:Latitude, radians)
9    (A:PLANE LONGITUDE, radians) (>C:ITrafficInfo:Longitude, radians)
10   30 (>C:ItrafficInfo:MaxVehicles)
11   40 (>C:ItrafficInfo:Radius, nmiles)
12   0x50 (>C:ItrafficInfo:Filter)  <!-- AWAKE and IN AIR -->
13 </Update>
14
15 <Element Name="ITrafficInfo Nearest Traffic Search Display">
16 <Position X="10" Y="10"/>
17   <FormattedText X="500" Y="600" Font="courier new" FontSize="9" LineSpacing="9" Color="Blue"
18     BackgroundColor="white" Bright="Yes" Align="Right">
19   <Color Value="#111111"/>
20     <String>
21       \{clr2}
22       %CUR                                                                    ON MAG  GND\n
23       %IDX  CALL      MODEL   DIST   VID   LATITUDE   LONGITUDE    ALT    VSI  GND HDG  SPD FLIGHT PLAN\n
24       \{clr}
25         %((C:ItrafficInfo:ListSize) s2 0 !=)
26         %{if}
27           %(0 sp1)
28           %{loop}
29             %(l1 (>C:ITrafficInfo:CurrentVehicle))
30             %((C:ITrafficInfo:CurrentVehicle) (>C:ITrafficInfo:SelectedVehicle))
31             %((C:ITrafficInfo:CurrentVehicle))%!-5d!
32             %( @CurrentCallsign )%!-10s!
33             %((C:ITrafficInfo:C:ATC MODEL, string))%!-8s!
34             %((C:ITrafficInfo:CurrentDistance, nmiles))%!4.1f!
35             %((C:ITrafficInfo:SelectedVehicleID))%!6d!
36             %((C:ITrafficInfo:C:PLANE LATITUDE, degrees))%!11.6f!
37             %((C:ITrafficInfo:C:PLANE LONGITUDE, degrees))%!12.6f!
38             %((C:ITrafficInfo:C:PLANE ALTITUDE, feet))%!7d!
39             %((C:ITrafficInfo:C:VERTICAL SPEED, feet per minute))%!7d!
40             %((C:ITrafficInfo:C:SIM ON GROUND, bool))%!4d!
41             %((C:ITrafficInfo:C:PLANE HEADING DEGREES MAGNETIC, degrees))%!5d!
42             %((C:ITrafficInfo:C:GROUND VELOCITY, knots))%!5d!
43             %((C:ITrafficInfo:SelectedFlightPlan))%!12s!\n
44             %(l1 ++ s1 l2 &lt;)
45           %{next}
46         %{end}
47     </String>
48   </FormattedText>
49 </Element>
```

- **Lines 1 – 5:** A macro that generates the aircraft Call Sign from the AI ATC Airline Name plus Flight Number or CurrentPlayerName in the case of a multiplayer aircraft. The SDK explanation of **ssub** is incorrect. Corrected documentation for the **ssub** operator can be found in the FSDeveloper Wiki:

http://www.fsdeveloper.com/wiki/index.php?title=XML:_Displaying_only_part_of_a_string

- **Lines 8 - 12:** This is the standard Nearest Search setup – 1) search origin, 2) maximum items, 3) search radius, and 4) search filter. As soon as these statements are executed, the default ITrafficInfo search setup will be re-set to the new values in lines 8 through 12. Default ITrafficInfo setup values are applied whenever the user does not include them (Latitude and Longitide = A:PLANE LATITUDE and LONGITUDE, Max Vehicles = 200, search radius = 43 NM, Filter = decimal 89).

  In the example above, the setup instructions are executed every update cycle, however they need to be executed one time only in order to re-set existing setup values. Consequently, a better place for these lines of code might be within a Click section of a mouse area, or if left in the update section, limited to one execution cycle only by use of a conditional if{ } statement.

- **Line 18:** Note the use of **BackgroundColor** in the text format. This will nicely mitigate the objectionable anti-aliasing applied to text in FSX.
- **Line 25:** This statement will prevent the display of the list until the nearest traffic search is complete, as evidenced by ListSize being greater than zero. This is a standard approach for fs9gps nearest searches.
- **Line 27:** The value zero is stored into Register #1. "0" is always the value of the first index line.
- **Line 28:** The display loop begins. Variables for an individual traffic aircraft are displayed one aircraft at a time, one line at a time based on the current Index pointer, the value in Register #1.
- **Line 29:** Register #1 is loaded into the CurrentVehicle index pointer.
- **Line 30:** Two of the desired outputs for this particular list are the unique Vehicle ID and AI Flight Plan for each traffic aircraft. Unfortunately, these two variables can be retrieved only from the Selected aircraft. The XML to *Select* an aircraft involves passing a pointer value (in this case, the CurrentVehicle pointer value) to the Selected index pointer. The XML is straightforward:

```
(C:ITrafficInfo:CurrentVehicle) (>C:ITrafficInfo:SelectedVehicle)
```

  The result is that during each pass through the display loop, Line 30 causes the Current aircraft to also become the Selected aircraft, enabling retrieval and display of SelectedVehicleID and SelectedFlightPlan for each aircraft retrieved in the search.

- **Lines 36 – 42:** Note the special use syntax. As explained in the SDK, The **C:** following ITrafficInfo stands for **C**urrent, and values retrieved every update cycle by these code lines are the respective A:Var Simulation Variable values for the Current aircraft. Similarly, an **S:** can be used and the values retrieved will be for the **S**elected aircraft (although in this example, Line 30 already made the Current aircraft and the Selected aircraft one and the same).

This is very useful. Among other things, it satisfies the traffic data requirements to build a **TCAS** gauge. Plotting intruder aircraft on a TCAS moving map gauge or as an overlay to fs9gps:Map can be done in XML using map scale methods covered in the Map Projections chapter.

Two additional notes. First, the SDK states that to set the Current or Selected aircraft, use a statement such as

```
>C:ITrafficInfo:CurrentVehicle N
```

where N is a value between 0 and ListSize -1. However, it appears that the correct syntax is

```
N (>C:ITrafficInfo:CurrentVehicle)
```

Secondly, the SDK advises that for the units, the Simulation Variables are all treated as **number**, except for certain string variables. This could be a little misleading, and the use of standard Flight Simulator units as shown in Lines 36 thru 42 is encouraged. As always, FS will make internal conversions for any of its standard units.

- **Line 44:** The "incrementer". After each line of traffic aircraft information is retrieved and displayed, Register #1 is incremented by 1 and Register #2 is checked to see if all of the aircraft have been displayed.

The preceding code generated the following list of 14 AI aircraft retrieved in the search. It's a real-time display, with numbers and relative aircraft positions continuously changing.

CUR VHC	CALL	MODEL	DIST	VID	LATITUDE	LONGITUDE	ALT	VSI	ON GND	MAG HDG	GND SPD	FLIGHT PLAN
0	N04361	C172	3.9	1822	41.939203	-87.931752	4494	-30	0	0	107	KJFX, KMWC
1	N5283V	BE58	11.1	1801	41.817777	-87.949079	4421	963	0	195	168	KORD, KMVN
2	N0390H	LJ45	12.9	1815	42.168875	-87.767878	29405	-369	0	138	425	PAWG, KGSO
3	N2163C	LJ45	13.0	1813	42.168310	-87.767048	31431	-354	0	138	434	PAWG, KGSO
4	ORB9005	A321	14.8	1760	41.779304	-88.059500	6058	2557	0	215	302	KORD, KHRL
5	N7441F	ARCHER	17.7	1809	42.184758	-88.226678	4479	94	0	314	114	KORD, KFAR
6	ORB3699	B738	17.8	1523	41.871984	-88.271580	6940	-214	0	174	283	KBUR, KORD
7	N2149G	C172	19.3	1807	42.043217	-88.343248	5493	-39	0	106	108	KDBQ, 5G2
8	AME231	DH8A	21.1	1759	41.735382	-87.603956	3605	1698	0	119	204	KMDW, KPHF
9	N6226J	C172	22.6	1789	41.815989	-88.354474	2536	-342	0	241	108	KORD, KARR
10	ORB7009	B744	23.6	1749	42.381295	-87.812428	26415	643	0	95	478	KSFO, KDTW
11	AME8698	DH8A	26.1	1764	41.670726	-87.542150	14028	-158	0	293	247	KAOH, KFSD
12	WOR4596	A321	34.1	1725	41.448500	-87.773027	29025	-102	0	92	449	KDEN, KCLE
13	N79539	C172	34.3	1783	42.271971	-88.593538	6493	-45	0	311	108	KMDW, KFCM
14	SOA8460	MD80	37.4	1742	42.186135	-88.712667	18261	-2429	0	41	353	KDFW, KMKE
15	ORB7715	A321	37.5	1739	41.942966	-88.748174	24022	-100	0	317	427	KMYR, KMSP
16	PAC9103	B738	37.5	1719	41.735292	-87.163122	25966	377	0	350	441	MPTO, KGRB
17	WOR2045	A321	39.6	1744	42.118748	-88.783705	20401	-2686	0	41	413	KDFW, KMKE
18	ORB9628	CRJ700	39.8	1711	41.714269	-87.116660	12631	-1884	0	324	321	KOFP, KORD

In this example, the aircraft that is 'Selected' is also constantly changing (Line 30) as the display loop progresses through the nearest traffic search results. With this code it impossible to Select a particular aircraft and watch its flight progress on the map, as in Figure **C** above. Therefore, this script, specifically, Lines 30, 35, and 43 cannot be used if you want to be able to select a specific aircraft and follow its flight on the map or radar screen.

Example 2. Displaying the Selected Aircraft on the Map

This example shows code required to highlight and display the Selected aircraft on the map.

```
1  <ColorLayerVehiclesSelected> 0x0000FF </ColorLayerVehiclesSelected>
2  <TagPosition> 5 </TagPosition>
3
4  <!-- EXAMPLE: Users choice of CurrentVehicle Index passed to SelectedVehicle Index -->
5  3 (>C:ITrafficInfo:CurrentVehicle)
6  (C:ITrafficInfo:CurrentVehicle) (>C:ITrafficInfo:SelectedVehicle)
7
8  3 (>C:ITrafficInfo:SelectedVehicle)
9
10 (C:ITrafficInfo:SelectedVehicleID) (>C:fs9gps:SelectedVehicle)
```

- **Line 1:** If it is to stand out, the ColorLayerVehiclesSelected variable must be set to a different color than the other vehicle symbols, ColorLayerVehicles. In this example, the Selected aircraft will be displayed with a red symbol. By default, the Selected aircraft's text label will be red.
- **Line 2:** Additionally, the position (TagPosition) of the Selected aircraft's label can be changed to help alleviate label congestion. TagPosition operates only on the Selected vehicle, not all vehicles, so its an aircraft-by-aircraft process to reposition all tags. The revised TagPosition remains with the aircraft even when another is subsequently Selected. The default location (TagPosition 0) is upper right. In this example, the label, or tag position is set to 5, to the left of the Selected aircraft symbol.
- **Line 4 through 9:** The ability to Select an aircraft requires that first, a nearest traffic search has been completed. The nearest search returns an indexed list of aircraft traffic, and the Selected aircraft is then chosen from that list. In order to do that, the desired index pointer of the nearest traffic search list, CurrentVehicle, needs to be identified by the user and passed to CurrentVehicleSelected. That thought process is reflected by Lines 5 and 6, but it is more efficient to simply code Line 9.
- **Line 11:** The last step. In order for CustomDraw to accept the Selected aircraft for map display, this instruction must be included. Note also that it is not necessary to pass the SelectedVehicle index number to SelectedVehicleID:

  That is, the following is <u>not</u> necessary:

```
(C:ITrafficInfo:SelectedVehicle)

(>C:ITrafficInfo:SelectedVehicleID)
```

- **ITrafficInfo:CurrentPlayerName (string) [Get, Set]**

The MultiPlayer player name.

- **ITrafficInfo:SelectedPlayerName (string) [Get, Set]**

The selected MultiPlayer aircraft.

LayerAirways

FSX Only

LayerAirways draws Low Altitude Victor and High Altitude Jet Airway centerlines.

❑ LayerAirways (bool)

LayerAirways controls whether or not the layer is displayed. Any number other than 0 will display the layer. A zero results in no rendering.

Example XML:

```
<LayerAirways> 1 </LayerAirways>
```

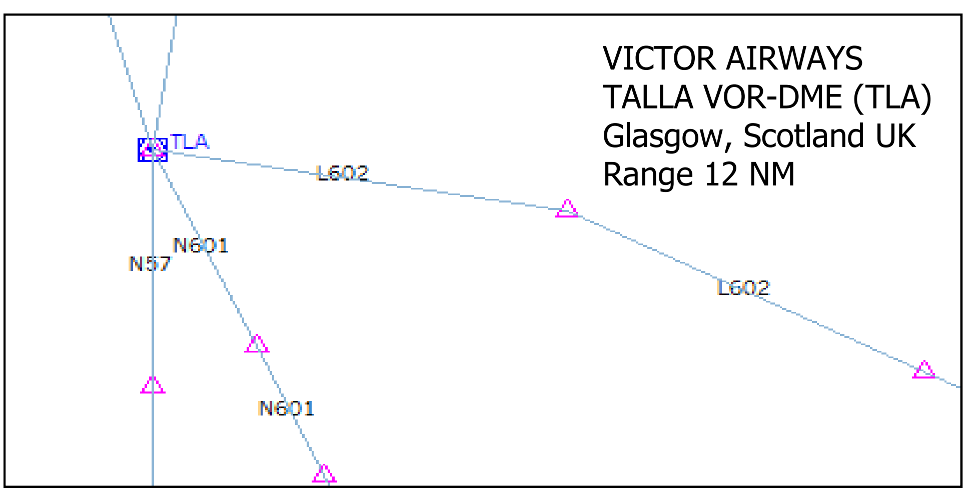
❑ DetailLayerAirways (enum)

DetailLayerAirways controls the line thickness.

- DetailLayerAirways = -1. Default. A 1 screen pixel wide line is drawn
- DetailLayerAirways = 0. Nothing is drawn
- DetailLayerAirways = 1. Thin Lines. A 1 screen pixel wide line is drawn
- DetailLayerAirways = 2. Thick Lines. A 3 screen pixel wide line is drawn

❑ TextDetailLayerAirways (bool)

TextDetailLayerAirways controls labeling of the Airway name. Any number other than 0 will display the name. Airway names are often, but not always, placed between enroute intersections that define airway segments.



❑ ObjectDetailLayerAirways (enum)

ObjectDetailLayerAirways determines whether Victor Airways, Jet Airways, or both are displayed.

- ObjectDetailLayerAirways = -1 or omitted. Default. Both Victor and Jet
- ObjectDetailLayerAirways = 0. Nothing displayed
- ObjectDetailLayerAirways = 1. Victor Airways displayed
- ObjectDetailLayerAirways = 2. Jet Airways displayed
- ObjectDetailLayerAirways = 3. Both Victor and Jet displayed

❑ ColorLayerAirwaysVictor (BGR hexadecimal)

ColorLayerAirwaysVictor is a BGR Hex number representing the color of Victor Airways. If ColorLayerAirwaysVictor is omitted, the default color is a light blue shade:

Blue: **214** Green: **181** Red: **140** BGR Hex: **0xD6B58C**

❑ ColorLayerAirwaysJet (BGR hexadecimal)

ColorLayerAirwaysJet is a BGR Hex number representing the color of Victor Airways. If ColorLayerAirwaysJet is omitted, the default color is a light magenta shade:

Blue: **222** Green: **164** Red: **222** BGR Hex: **0xDEA4DE**

❑ TextColorLayerAirways (BGR hexadecimal)

TextColorLayerAirways is a BGR Hex number representing the color of the name label of Victor Airways only. It is not applied to Jet Airways. If TextColorLayerAirways is omitted, the default color is cyan:

Blue: **255** Green: **255** Red: **0** BGR Hex: **0xFFFF00**

The default, and only color available for the name label of Jet Airways is a purple shade:

Blue: **148** Green: **90** Red: **148** BGR Hex: **0x945A94**

TAWS

Terrain Awareness Map in FSX

TAWS = GPWS + FLTA

Terrain Avoidance Warning Systems are a combination of a Ground Proximity Warning System and Forward Looking Terrain Avoidance. FS9 and FSX provide capability to model GPWS Modes 1 through 6, but not FLTA – that is, FS has no XML gauge capacity to measure terrain ahead of the aircraft and issue FLTA alerts as appropriate. The table below summarizes terrain avoidance modeling capability in Flight Simulator using stock FS variables and XML. Aural alerts require a third party sound module and sound files:

TAWS SYSTEM REQUIREMENTS	**TAWS Class A**	**TAWS Class B**	**Can Be Modeled By Flight Simulator?**		**Key Flight Simulator Variables**
Radar Altimeter	Required	Not Required	Yes	FS9 and FSX	A:RADIO HEIGHT
Airdata and Computer	Required	Not Required	Yes	FS9 and FSX	Various system variables (A:Vars)
Gear State Input	Required	Not Required	Yes	FS9 and FSX	A:GEAR HANDLE POSITION
Flaps State Input	Required	Required	Yes	FS9 and FSX	A:FLAPS HANDLE PERCENT
Supplemental Type Certification	Required	Not Required	N/A	Not Applicable	Not Applicable
Terrain Awareness Map	Required	Not Required	Approx	FSX only	ElevationXColor variables and A:PLANE ALTITUDE
Fully Autonomous GPWS	Required	Not Required	Yes	FS9 and FSX	A:Vars plus GPS module variables provides redundancy
GPWS ALERTS	**GPWS Mode**	**Acronym**	**Can Be Modeled By Flight Simulator?**		**Key Flight Simulator Variables**
Excessive Rate of Descent	Mode 1	ERD	Yes	FS9 and FSX	A:PLANE ALTITUDE and A:RADIO HEIGHT
Excessive Terrain Closure Rate	Mode 2	ECRT	Yes	FS9 and FSX	A:RADIO HEIGHT
Negative Climb Rate After Takeoff	Mode 3	NCAT	Yes	FS9 and FSX	A:PLANE ALTITUDE, A:RADIO HEIGHT and A:VERTICAL SPEED
Flight Into Terrain Not In Landing Configuration	Mode 4	FITNL	Yes	FS9 and FSX	A:RADIO HEIGHT, A:GEAR HANDLE POSITION, and A:FLAPS HANDLE PERCENT
Excessive Deviation Below Glideslope	Mode 5	EDGSD	Yes	FS9 and FSX	A:RADIO HEIGHT, A:NAV1 GSI
Excessive Bank Angle	Mode 6	EBA	Yes	FS9 and FSX	A:ATTITUDE INDICATOR BANK DEGREES
Altitude Callout	Mode 6	VC	Yes	FS9 and FSX	A:RADIO HEIGHT
Windshear Protection	Mode 7	WS	Doubtful	FS9 and FSX	A:AMBIENT WIND X and Z, but vertical wind speed (AMBIENT WIND Y) not available
FLTA ALERTS		**Acronym**	**Can Be Modeled By Flight Simulator?**		**Key Flight Simulator Variables**
Forward Looking Terrain Avoidance		FLTA	No	None	Beyond the capability of the GPS Module.
Premature Descent Alert		PDA	Yes	FS9 and FSX	A:PLANE ALTITUDE and GPS Module variables

Given the FLTA limitation, however, a crude Class A TAWS system can be built for FSX using XML script because CustomDraw Map can be configured to produce a Terrain Awareness Map facsimile. This chapter focuses on the Terrain Awareness Map.

❑ Terrain Awareness Map

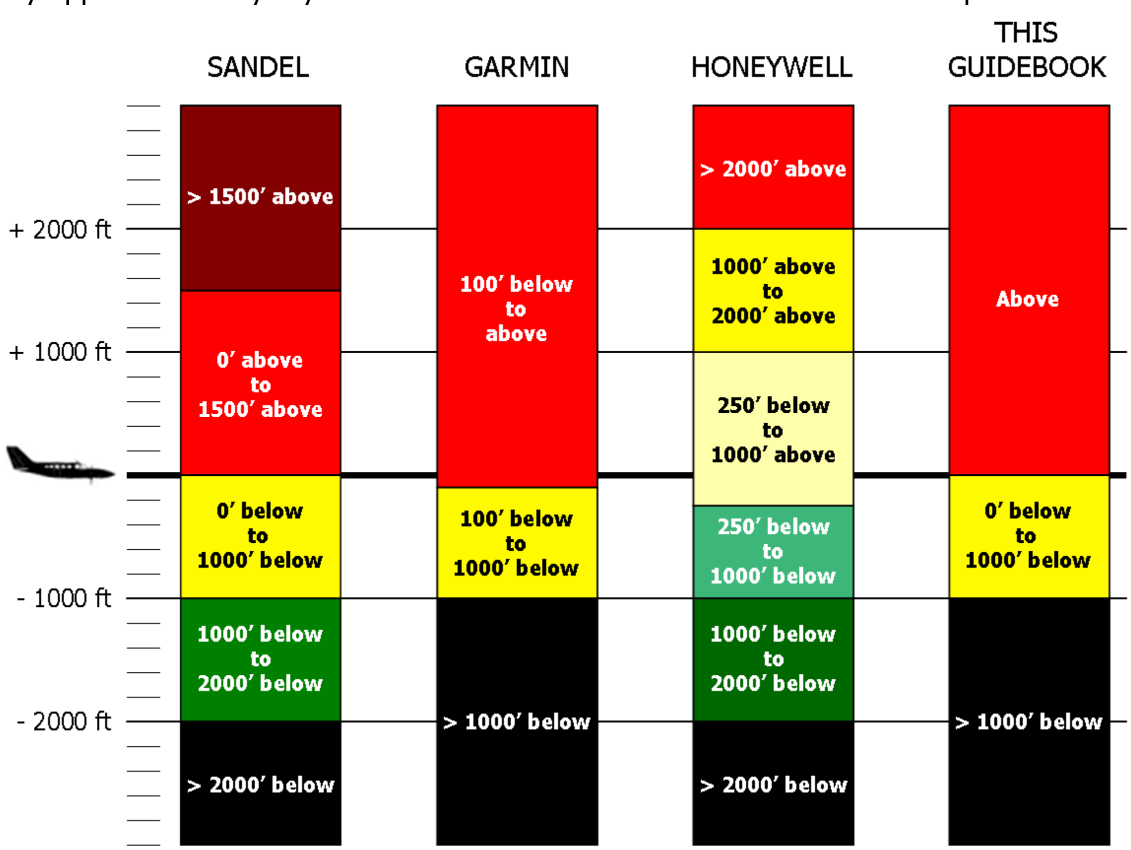
Using ElevationXColor variables and aircraft altitude, it is possible to create an approximate TAWS terrain awareness display. It is "approximate" at best because the coarse 1000 foot ElevationXColor interval combined with significant color feathering produces an inaccurate terrain awareness map.

Two issues must be addressed for the TAWS terrain awareness display in FSX:

- ❑ **Elevation color selection** that is a function of aircraft height above terrain
- ❑ **Terrain Refresh** needed due to aircraft altitude change

❑ Elevation Color Selection

The chart below shows Terrain Awareness Map colors used in a few TAWS systems that can be researched online. Of these, the Garmin 500W Series - G1000 TAWS color scheme is the simplest and makes the most sense for an FSX implementation that is only approximate anyway. The fewer colors the better in an FSX TAWS map.



The XML approach is to incorporate A:PLANE ALTITUDE conditions into the ElevationXColor expression. The following is an example of the Elevation4000Color (3000 ft to 4000 ft elevation layer) expression:

```
1  <Elevation4000Color>
2  (L:TAWS_Mode, bool) 0 == (A:SIM ON GROUND, bool) or
3     if{
4        0x6EB5C7  <!--  THE NON-TAWS ELEVATION COLOR  -->
5     }
6     els{
7        (A:PLANE ALTITUDE, feet) 5000 &gt;
8           if{
9              0x101010  <!--  BLACK  -->
10          }
11          els{
12             (A:PLANE ALTITUDE, feet) 4000 &gt;
13                if{
14                   0x00F6FF  <!--  YELLOW  -->
15                }
16                els{
17                   0x0202E3  <!--  RED  -->
18                }
19          }
20    }
21 </Elevation4000Color>
```

- **Lines 2 – 7.** If the TAWS switch is OFF or the aircraft is on the ground, then the standard non-TAWS elevation color is used. In this case, it is **0x6EB5C7** which is from the G1000 manual.
- **Line 6.** TAWS Mode. The TAWS switch is ON and the aircraft is in the air.
- **Lines 7 – 10.** If the aircraft is at an altitude greater than 5000 feet (line 7), then the top of the Elevation4000Color layer, which is 4000 feet, is more than 1000 feet below the aircraft, and according to the TAWS color palette the layer should be colored **BLACK** (line 9).
- **Line 11.** If the aircraft altitude is not greater than 5000 feet, then …
- **Lines 12 – 15.** If the aircraft is at an altitude greater than 4000 feet (line 12), then the top of the Elevation4000Color layer is between 0 feet and 1000 feet below the aircraft, and according to the TAWS color palette the layer should be colored **YELLOW** (line 14).
- **Line 16.** If the aircraft altitude is not greater than 4000 feet, then …
- **Line 17.** The top of the terrain layer (4000 feet) is at or above the aircraft, and according to the TAWS color palette, the layer should be colored **RED**.

Yellow Band Must be 1000′ (or Multiples of 1000′)

Because of the 1000 foot ElevationXColor interval, the difference between altitudes in line 7 and 12 must be 1000 feet, or multiples of 1000 feet. If not, then as the aircraft climbs or descends past each thousand foot altitude level, the yellow band will either disappear or double its width for a while.

If, for example, line 7 has 5000′ but line 12 has 4200′ instead of 4000′, then the yellow band will disappear when A:PLANE ALTITUDE is between 4000′ and 4200′. This will repeat for the other ElevationXColor layers if the line 7, line 12 elevations are expressed in a similar manner (i.e., not multiples of 1000′ difference).

Color Feathering

TerrainShadow must be disabled in TAWS mode or the TAWS colors will not display in a satisfactory manner. However, when TerrainShadow = 0, significant color feathering occurs over a 2000 foot elevation interval, and as well, the central color band is centered 1000 feet below the value expressed in the ElevationXColor variable name.

The maps below demonstrate the effect on Elevation4000Color.

Figure **A** is a contour map of the Island of Hawaii, USA. The 2000′, 3000′, and 4000′ topographic contours from the FSX terrain data are displayed (a Photoshop manipulation from ElevationXColor, not a direct extraction from the terrain database).

In Figure **B**, Elevation4000Color = 0x37597D (a chocolate brown color) and TerrainShadow = 1. The elevation color uniformly fills the interval from 4000 feet to 3000 feet as expected. But unfortunately, TAWS colors (black, red, yellow) will not display satisfactorily when Terrain Shadow is enabled.

Figure **C** is the same map but with TerrainShadow = 0 as required for TAWS Mode. This obviously presents a few issues to deal with for a terrain awareness display. The area outlined by the dashed line is enlarged in Figure **D**.

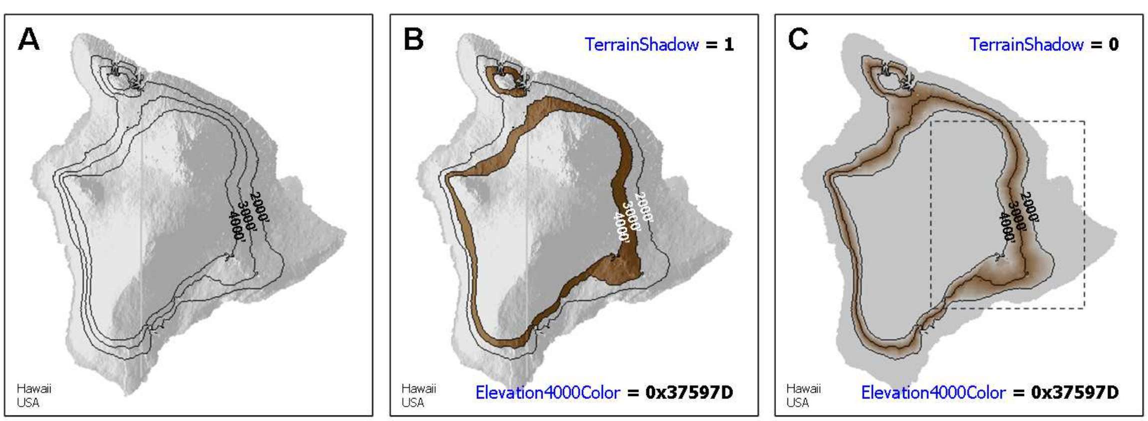
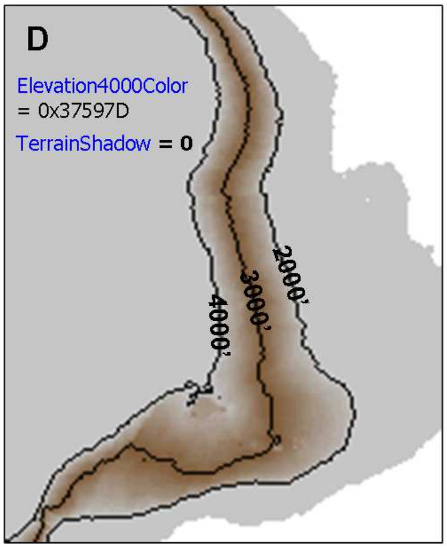
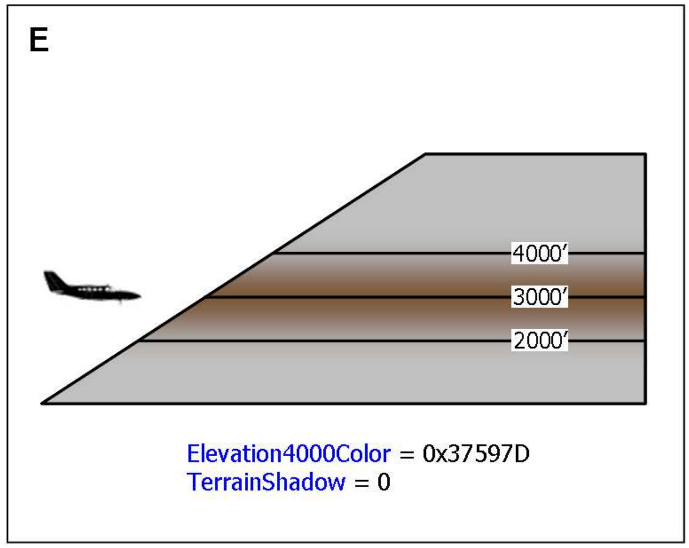
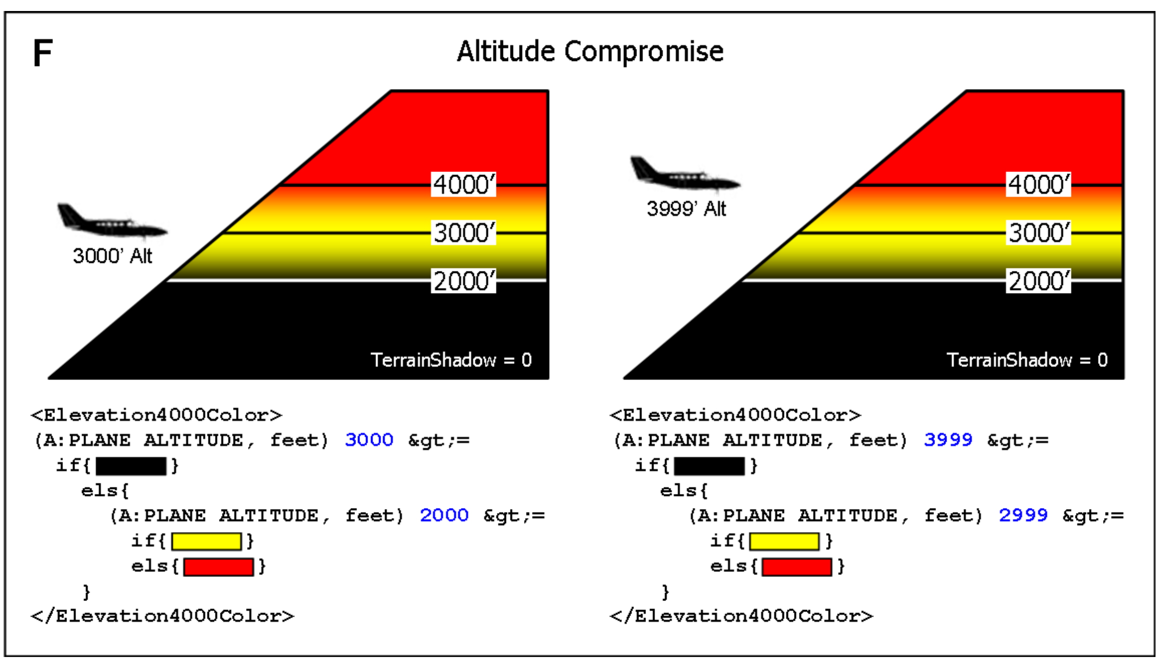


Figure **D** shows that the brown color band associated with Elevation4000Color is actually centered on the 3000′ elevation contour and feathers out in both directions for 1000 vertical feet. Figure **E** is a cross-sectional view.





For an aircraft flying at 3000′ altitude, the bottom of the yellow band should be at 2000′ elevation, as shown in Figure **F**. But ElevationXColor variables are available only at 1000′ intervals, so the same color band applies for an aircraft flying at 3999′ altitude – the TAWS map colors cannot change until the aircraft reaches 4000′ altitude.



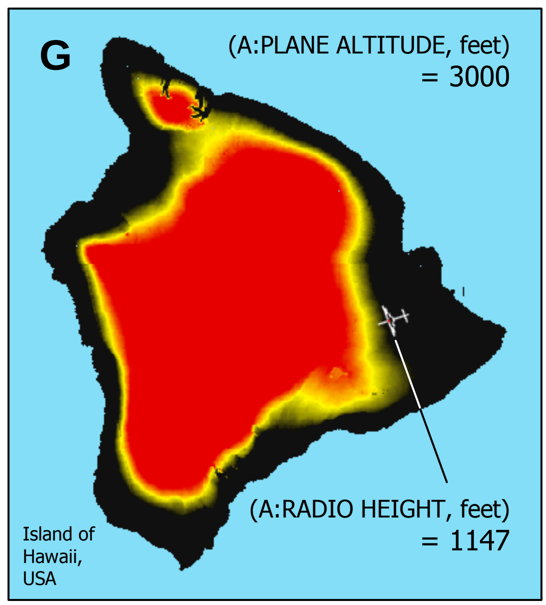
As a consequence of the coarse 1000′ color interval, a 500 foot altitude compromise between the two ElevationXColor expressions in Figure **F** is:

```
<Elevation4000Color>
 (A:PLANE ALTITUDE, feet) 3500 &gt;=
   if{        }
     els{
        (A:PLANE ALTITUDE, feet) 2500 &gt;=
          if{        }
          els{        }
     }
</Elevation4000Color>
```

The TAWS map display is therefore only "approximate". The coarse 1000′ color interval limits accuracy of the display and the color edges (i.e., black to yellow) are feathered rather than crisp, but yellow provides such a contrast to black that the bottom of the yellow color band is still quite apparent.

Figure **G** shows the terrain awareness map display using corresponding ElevationXColor expressions for all the elevation color variables. The aircraft is flying in a SW direction at 3000′ altitude. Its radar altimeter reads 1147 feet meaning that terrain clearance is in the Black TAWS color band. 1147 feet is close to the 1000 foot threshold for Yellow TAWS color and on the map, the aircraft is close to the bottom of the Yellow band. So in this particular snapshot, the terrain awareness map seems to be reasonably accurate.

DetailLayerTerrain = -1
TerrainShadow = 0



Radar Altimeter ElevationXColor Adjustment

About half of the time when close to terrain, the altitude compromise ends up being too liberal and the terrain awareness map shows the aircraft to be in the Black when radio altitude is less than 1000′. To help mitigate this, I prefer to incorporate the following radar altimeter condition as a final adjustment of the TAWS display colors:

```
<Elevation4000Color>
     (L:TAWS_Mode, bool) 0 == (A:SIM ON GROUND,bool) or
          if{ 0x6eb5c7 } //  NON-TAWS ELEVATION COLOR
          els{ (A:RADIO HEIGHT, feet) 1000 &lt;
               if{
                    (A:PLANE ALTITUDE,feet) 4000 &gt;=
                    if{ 0x101010 } //  BLACK
                    els{ (A:PLANE ALTITUDE, feet) 3000 &gt;=
                         if{ 0x00f6ff } //  YELLOW
                         els{ 0x0202e3 } //   RED
                    }
               }
               els{
                    (A:PLANE ALTITUDE,feet) 3500 &gt;=
                    if{ 0x101010 } //  BLACK
                    els{ (A:PLANE ALTITUDE, feet) 2500 &gt;=
                         if{ 0x00f6ff } //  YELLOW
                         els{ 0x0202e3 } //   RED
                    }
               }
          }
</Elevation4000Color>
```

The download XML gauges contain a complete list of ElevationXColor expressions.

❑ Terrain Refresh

After the terrain layer is rendered, CustomDraw Map will not regularly re-evaluate ElevationXColor expressions and re-draw the map as aircraft altitude changes would otherwise dictate. This is true even when UpdateAlways = 1, or "True". Therefore, in TAWS mode, the user must force terrain re-fresh.

This TAWS map refresh approach consists of two parts:

- Refreshing the terrain elevation colors
- Timing of the re-fresh

LayerTerrain Refresh

Refreshing the terrain elevation colors requires initiating the computation of a new, different, terrain display. One way to trigger this is to briefly change DetailLayerTerrain to 1 (Water Only). The other way, which I recommend, is to momentarily change Zoom. Refreshing any other layer or even toggling LayerTerrain, will not trigger the re-evaluation of terrain elevation colors needed for the TAWS display.

The new terrain does not need to be fully displayed, just initiated, before re-setting Zoom. There is an unavoidable but momentary dropout of the map display while this happens.

Example XML <Mouse> section – turn TAWS Mode On and Off:

```
1  <Area Name="TAWS MODE" Left="110" Top="430" Width="35" Height="13">
2     <Cursor Type="Hand" />
3     <Click Kind="LeftSingle">
4        (L:TAWS_Mode, bool) ! (>L:TAWS_Mode, bool)
5        (L:TAWS_Mode, bool)
6           if{
7              (L:ZFactor, number) (>L:ZFactorOrig, number)
8              (L:Background_Color, enum) (>L:BkgdColorOrig, enum)
9              (L:Terrain_Shadow, bool) (>L:Terrain_ShadowOrig, bool)
10             0 (>L:Terrain_Shadow, bool)
11             0 (>L:Update_Always, bool)
12             1 (>L:Map_Priority, bool)
13             1 (>L:Map_Loading, bool)
14             65973 (>L:Background_Color, enum)
15             1 (>L:AC_Cursor_Lime, bool)
16             1 (>L:Terrain_Refresh, bool)
17             0 (>L:TCAS_Mode, bool)
18          }
19       (L:TAWS_Mode, bool) !
20          if{
21             @TAWSClose
22          }
23       0 (>L:TAWS_Counter, enum)
24    </Click>
25 </Area>
```

- **Line 4.** TAWS Mode toggle ON and OFF.
- **Line 5 - 18.** Init sequence when TAWS mode is turned ON
- **Line 7 - 9.** The original settings of key display variables are stored for reference when TAWS is turned OFF and the terrain display returns to normal.
- **Line 10.** TerrainShadow must be 0 for TAWS Mode. Certain colors display extremely poorly when terrain shadow is enabled, among them, unfortunately, red and yellow, and black.
- **Line 11.** UpdateAlways = 0. Actually, this is just a preference, I prefer it to always be 0 otherwise the map noticeably "dances".
- **Line 12.** Priority = 1. Priority = 1 will significantly speed up terrain elevation color refresh.
- **Line 14.** BackgroundColor = 65973. This is the decimal equivalent of 0x010101, Black. When terrain elevation colors refresh, the terrain will usually disappear

momentarily and only the BackgroundColor will remain. It is preferable to have a "flash" of Black than say, BackgroundColor = 16711935 = 0xFF00FF = Magenta.

- **Line 15.** The aircraft cursor symbol color is changed to lime. To be seen in TAWS Mode, the cursor needs to be a color that contrasts with Black, Yellow, Red and water Blue. Lime or white are good a choices. User preference.
- **Line 16.** Terrain_Refresh is enabled. Terrain_Refresh is the code that momentarily changes Zoom which causes FSX to refresh terrain.
- **Line 17.** In this example gauge, TAWS and TCAS share the same screen, so TCAS mode is disabled when TAWS map is showing. Not a real-world condition.
- **Line 19.** Init sequence when TAWS mode is turned OFF. See TAWSClose macro below
- **Line 23.** The cycle skip counter required during the terrain refresh step is set to zero.

TAWSClose macro

```
1  <Macro Name="TAWSClose">
2     (L:ZFactorOrig, number) (>L:ZFactor, number)
3     (L:Terrain_ShadowOrig, bool) (>L:Terrain_Shadow, bool)
4     (L:BkgdColorOrig, enum) (>L:Background_Color, enum)
5     0 (>L:AC_Cursor_Lime, bool)
6     0 (>L:TAWS_Mode, bool)
7     1 (>L:Terrain_Refresh, bool)
8  </Macro>
```

- **Line 2 - 5.** Original values of key display settings are returned to pre-TAWS mode state
- **Line 6.** TAWS_Mode is turned off
- **Line 7.** An additional terrain refresh is performed. TAWS_Mode=0, so the standard terrain palette will be used.

Example XML <Update> section – Terrain Refresh:

The following is the terrain elevation color refresh script, placed in the Update section:

```
1  (L:Terrain_Refresh, bool)
2     if{
3        (L:ZFactor, number) (>L:ZFactorTemp, number)
4        2699 (>L:ZFactor, number)
5        (L:TAWS_Counter, enum) ++ (>L:TAWS_Counter, enum)
6        (L:TAWS_Counter, enum) 3 ==
7           if{
8              (L:ZFactorTemp, number) (>L:ZFactor, number)
9              0 (>L:Terrain_Refresh, bool)
10             0 (>L:TAWS_Counter, enum)
11          }
12    }
```

- **Line 3.** The current Zoom factor is stored as L:ZFactorTemp
- **Line 4.** Zoom factor is set to 2699 NM, the largest allowable Zoom. This will trigger a re-computation of the ElevationXColor variables which is the goal.
- **Line 5 – 6.** Cycle Skipping. Terrain color calculation *appears* to be a multi-cycle process. Lines 5 and 6 create a delay to allow sufficient time for the process to sufficiently progress before resetting zoom back to normal (line 8). In my experience, only a one cycle delay has been required, but experimentation with line 6 may be needed if the TAWS Mode colors do not appear (i.e., set the value to 3 or more).
- **Line 8.** The pre-refresh zoom factor is restored. In a similar manner with line 4, this triggers a re-computation of the terrain but this time with ElevationXColor values that are updated by current aircraft altitude.
- **Lines 9 - 10.** Terrain_Refresh flag is re-set to zero, as is the cycle skip counter.

Example XML <Update> section – Timing of the Terrain Refresh:

The refresh timing script, also in the Update section:

```
1  (L:Alt500, enum) (>L:Alt500_Old, enum)
2  (A:PLANE ALTITUDE, feet) 100 + (A:RADIO HEIGHT, feet) 1000 &lt;
3     if{ 500 } els{ 1000 } / int (>L:Alt500, enum)
4  (L:Alt500_Old, enum) (L:Alt500, enum) != (L:TAWS_Mode, bool) and
5     if{
6        1 (>L:Terrain_Refresh, bool)
7        0 (>L:TAWS_Counter, enum)
8     }
```

- **Line 1.** The value of L:Alt500 is stored into L:Atl500_Old.
- **Line 2 - 3.** This creates an aircraft altitude index so that with every 500 foot change in aircraft altitude, a terrain refresh can be initiated.
    - **(A:RADIO HEIGHT, feet) 1000 < if{ 500 } els{ 1000 } / int** creates an altitude index at 500' or 1000' intervals depending upon radar altitude. Even though ElevationXColor variables are limited to 1000' intervals and a 1000' altitude change index might at first seem sufficient, a 500' index is necessary at low radar altitude because my ElevationXColor variables change elevation color on the 1000 foot mark if RADIO HEIGHT is less than 1000' or on the 500 foot mark if it is not (this is the Radar Altimeter ElevationXColor Adjustment).
    - **100 +** is used to prevent the index from triggering a refresh right at the 500' or 1000' altitude increments, where aircraft typically level off. If this is omitted, constant minor changes in cruise altitude will force new TAWS colors. The 100 is added (+) which will create altitude index marks (and terrain refreshes) at the 400' and 900' level so that TAWS colors will be refreshed just before the aircraft reaches normal cruising altitudes.
- **Line 4.** If the altitude index has changed and TAWS Mode is enabled, then a terrain refresh will be triggered.
- **Line 6 - 7.** The cycle skip counter required during the terrain refresh step is set to zero and the Terrain Refresh flag is set to 1.

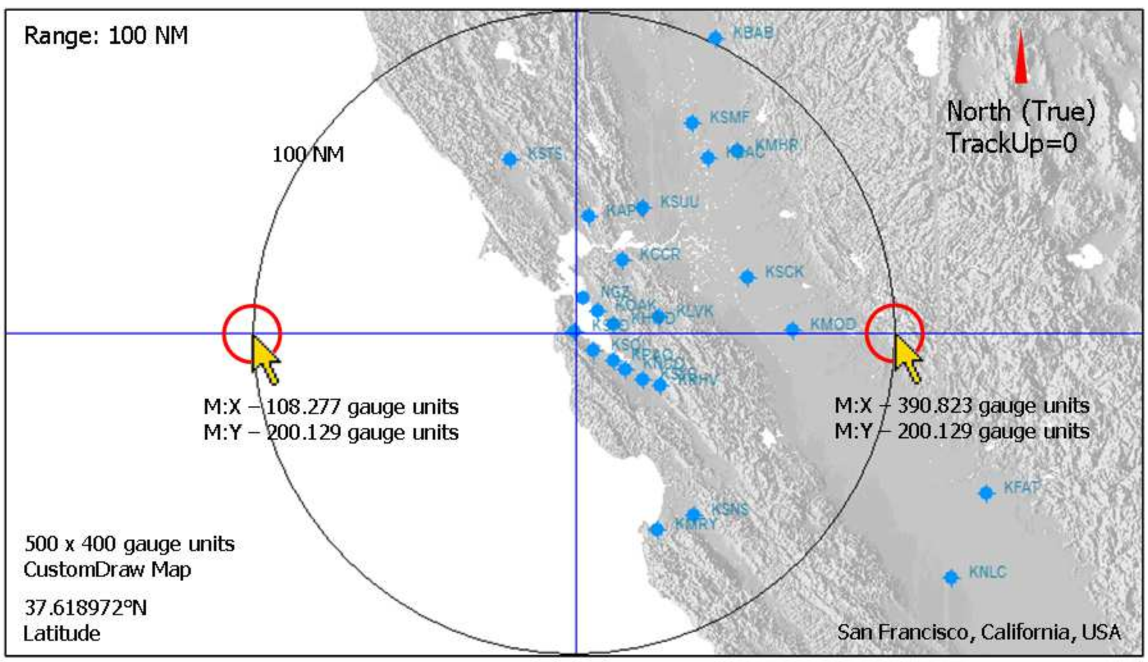
Map Scale Calibration for Overlays

XML and CustomDraw

XML overlays and mouse movements are measured in gauge units but CustomDraw Map renders screen pixels. The scales are not the same. To overlay XML gauge objects at the correct coordinates relative to the underlying CustomDraw map or to use the mouse on it, map scale measured in gauge units must be determined first. Both the X-axis and Y-axis scales (meters per gauge unit X and Y) are needed because they are often different. XML overlays and mouse use can add useful and very cool functionality to CustomDraw Map, but calibrating the scales needs to be done accurately.

Scale Calibration: FSX
Scale = Meters / (∆Gauge Units x Zoom Factor)

Calibration can be achieved by clicking the map at locations with known earth coordinates or distances, recording the mouse X and Y gauge unit position, and computing the X and Y scale functions that translate between the two.



Short Axis			**Long Axis**		
Z Factor (Range):	100.000	Nautical Miles	Known Distance:	100.000	Nautical Miles
			Mouse 1:	108.277	Gauge Units
Map Size:	400	Gauge Units	Mouse 2:	390.823	Gauge Units
Map Scale:	926.000	Meters / Gauge U.	Map Scale:	1310.940	Meters / Gauge U.

(Scale) / (Z Factor)	**9.26000**	Short Axis - XML Map Scale as a function of Z Factor
(Scale) / (Z Factor)	**13.10940**	Long Axis - XML Map Scale as a function of Z Factor

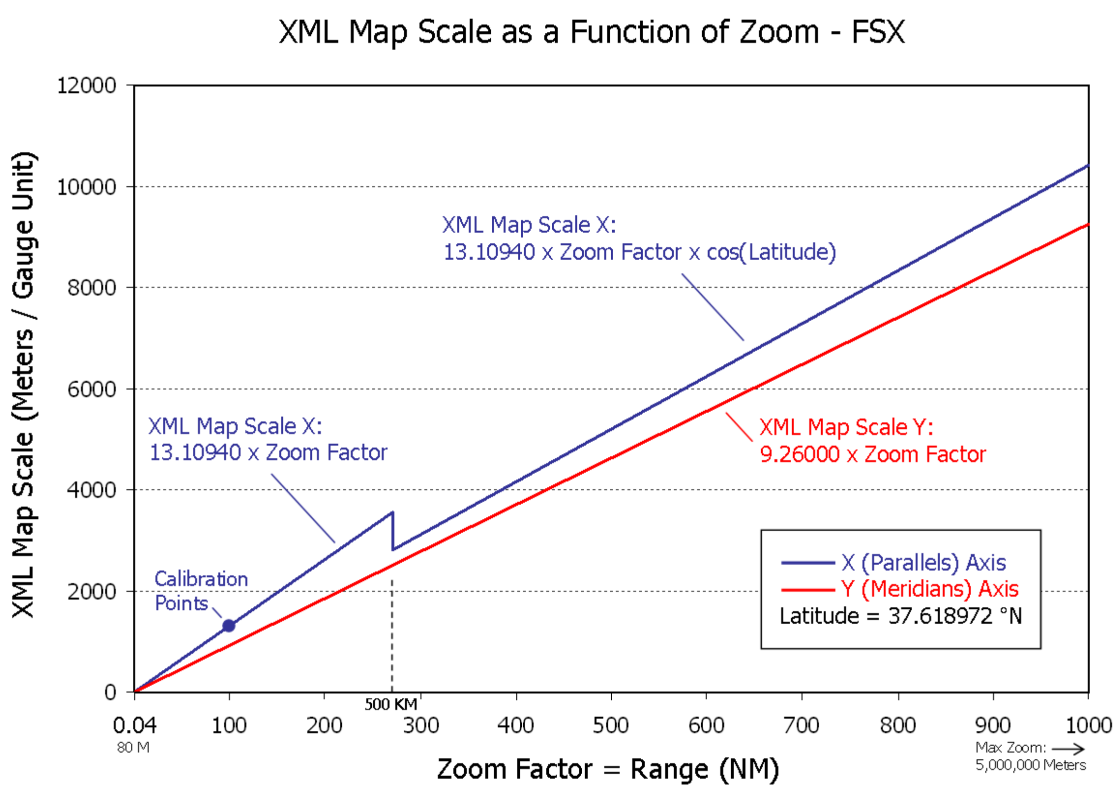
The manual calibration technique in FSX uses Range Rings to establish distance and involves two mouse clicks on the long axis. The short axis is always calibrated because Flight Simulator always fills the short axis with 2 x Range.

Calibration should be done at Zoom Factors (Range) less that 269 NM (Zooms less than 500 KM). The calibration sequence involves the following:

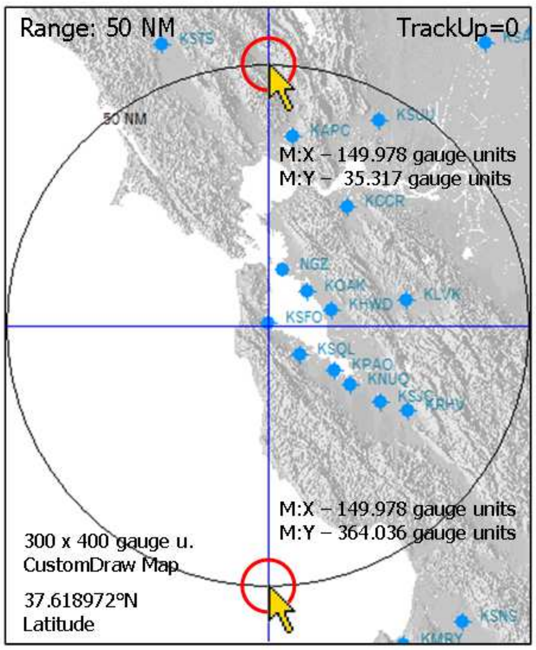
- TrackUp = 0
- LayerRangeRings = 1
- ObjectDetailLayerRangeRings = Z Factor (Range)
- Add Polyline Elements through the map center, CenterX and CenterY, as shown with the blue cursor lines on the map
- Click on the intersections of the range ring and the long axis polyline as shown above.

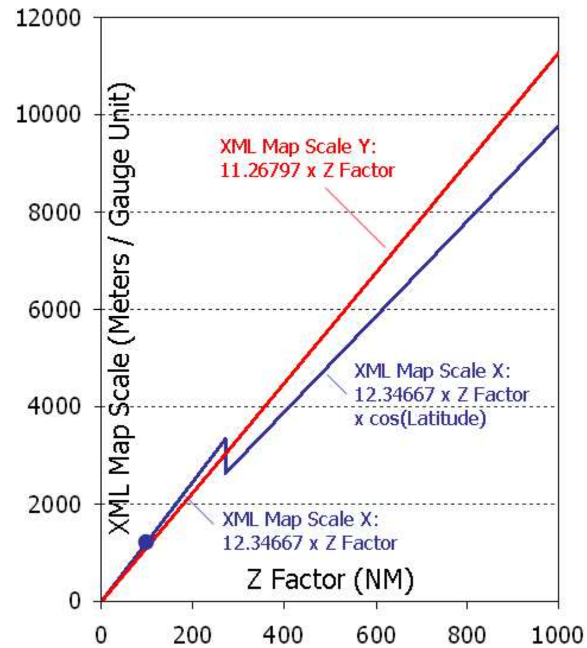
Two mouse clicks determine range ring diameter measured in gauge units for the long axis. The XML map scale is a simple calculation for each axis after that. Additionally, thse scale functions can be permanently stored as L:Vars so that calibration is no longer required unless the aspect ratio of the map changes.

The XML map scale functions calculated from the calibration example is shown below. Note that cosine correction of the X-axis scale at zooms below 500 KM is absent:



An example of short axis = X-axis:





Short Axis			Long Axis		
Z Factor (Range):	50.000	Nautical Miles	Known Distance:	50.000	Nautical Miles
			Mouse 1:	35.317	Gauge Units
Map Size:	300	Gauge Units	Mouse 2:	364.036	Gauge Units
Map Scale:	617.333	Meters / Gauge U.	Map Scale:	563.399	Meters / Gauge U.
(Scale) / (Z Factor)	**12.34667**	Short Axis - XML Map Scale as a function of Z Factor			
(Scale) / (Z Factor)	**11.26798**	Long Axis - XML Map Scale as a function of Z Factor			

Equations for Scale vs. Zoom functions:

	A	B	C	D	E	F
1		**Short Axis**			**Long Axis**	
2	Z Factor (Range):	50.000	Nautical Miles	Known Distance:	=B2	Nautical Miles
3				Mouse 1:	35.317	Gauge Units
4	Map Size:	300	Gauge Units	Mouse 2:	364.036	Gauge Units
5	Map Scale:	=(B2*1852)/(B4/2)	Meters / Gauge U.	Map Scale:	=E2*1852*2/ABS(E4-E3)	Meters / Gauge U.
6	(Scale) / (Z Factor)	**=B5/B2**	Short Axis - XML Map Scale as a function of Z Factor (1 / 1852 Gauge Unit)			
7	(Scale) / (Z Factor)	**=E5/B2**	Long Axis - XML Map Scale as a function of Z Factor (1 / 1852 Gauge Unit)			

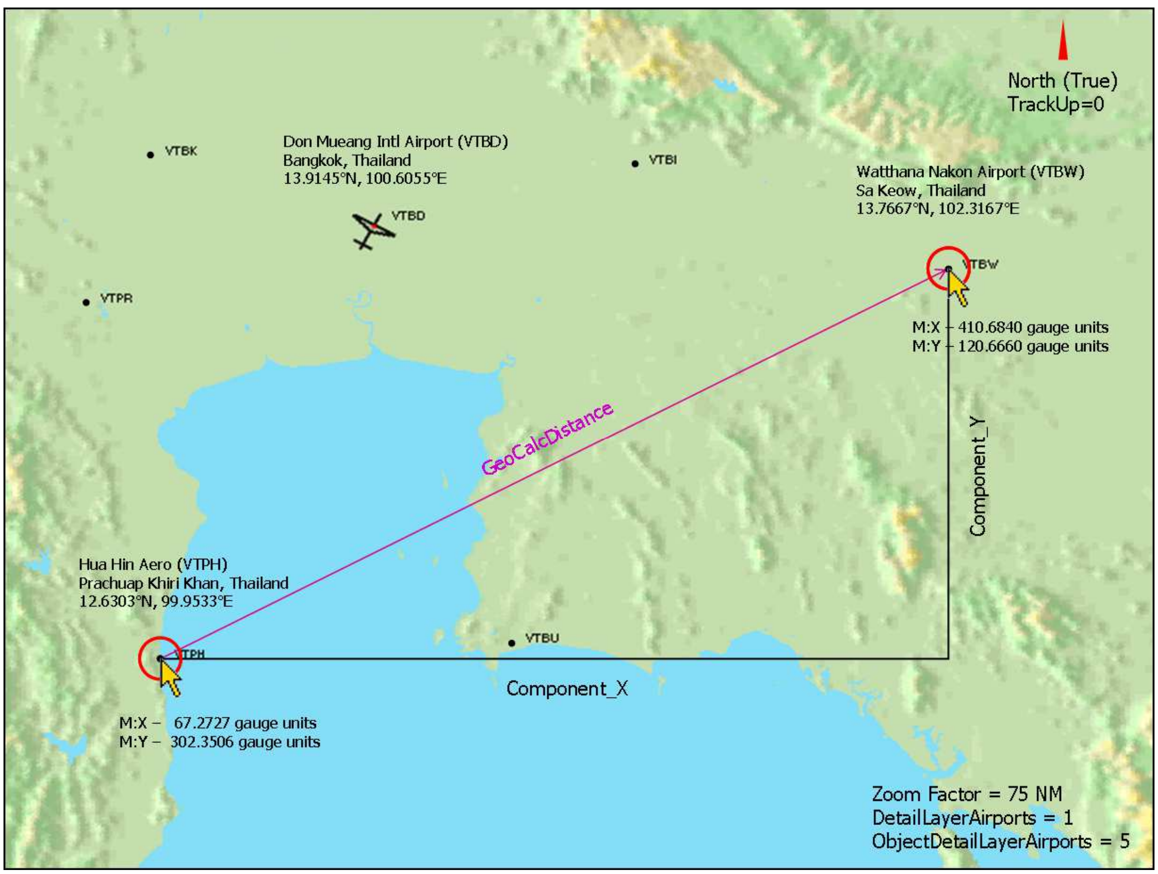
A functional XML example of this calibration technique is included in the download section of the BlackBox/CustomDraw website.

Scale Calibration: FS9

The idea behind scale calibration in FS9 is the same as for FSX, but range rings are not available so single point objects from the gps data base (airports, NDBs, or VORs) are substituted for range rings and GeoCalcDistance is used to establish known distance on the long axis.

Using airports as an example, the calibration sequence involves the following:

- TrackUp = 0
- DetailLayerAirports = 1. Point symbol
- ICAO or Ident to access WaypointAirportLatitude and Longitude
- Click the airport symbol to determine gauge X, Y coordinates
- GeoCalcDistance variable for distance between the two airports
- Geometric calculations to compute the X and Y components of the distance

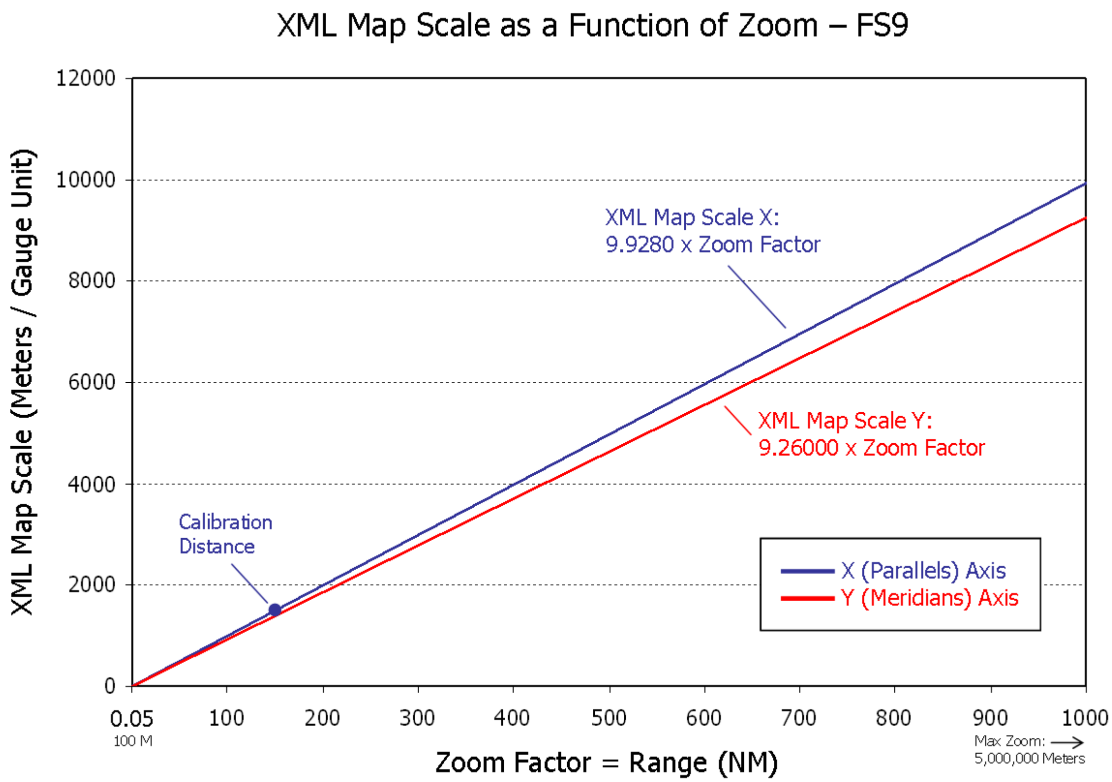


Equations for Scale vs. Zoom functions:

Zoom Factor:	75	NM	Zoom Factor = Range
Map Size X:	500	Gauge Units	
Map Size Y:	400	Gauge Units	
Mouse 1 X:	67.2727	Gauge Units	Calibration Point 1 X
Mouse 1 Y:	302.3506	Gauge Units	Calibration Point 1 Y
Mouse 2 X:	410.6840	Gauge Units	Calibration Point 2 X
Mouse 2 Y:	120.6660	Gauge Units	Calibration Point 2 Y
Delta Mouse X:	343.4113	Gauge Units	Point 2 - Point 1 X
Delta Mouse Y:	181.6846	Gauge Units	Point 2 - Point 1 Y
GeoCalcLatitude 1:	12.6303	Degrees	Calibration Point 1 Lat
GeoCalcLongitude 1:	99.9533	Degrees	Calibration Point 1 Lon
GeoCalcLatitude 2:	13.7667	Degrees	Calibration Point 2 Lat
GeoCalcLongitude 2:	102.3167	Degrees	Calibration Point 2 Lon
GeoCalcDistance:	153.9876	NM	
Delta Latitude:	1.1364	Degrees	
Delta Latitude:	68.1833	NM	Component_Y
Long Axis Distance:	138.0697	NM	Component_X
Long Axis Distance:	255705.0	Meters	Component_X
Short Axis XML Map Scale:	694.5000	Meters per Gauge Unit	
Long Axis XML Map Scale:	744.6029	Meters per Gauge Unit	
(Scale) / (Z Factor):	**9.2600**	Short Axis Scale as function of Z Factor	
(Scale) / (Z Factor):	**9.9280**	Long Axis Scale as function of Z Factor	

	A	B	C	D
1	Zoom Factor:	75	NM	Zoom Factor = Range
2	Map Size X:	500	Gauge Units	
3	Map Size Y:	400	Gauge Units	
4				
5	Mouse 1 X:	67.2727	Gauge Units	Calibration Point 1 X
6	Mouse 1 Y:	302.3506	Gauge Units	Calibration Point 1 Y
7	Mouse 2 X:	410.6840	Gauge Units	Calibration Point 2 X
8	Mouse 2 Y:	120.6660	Gauge Units	Calibration Point 2 Y
9	Delta Mouse X:	=ABS(B7-B5)	Gauge Units	Point 2 - Point 1 X
10	Delta Mouse Y:	=ABS(B8-B6)	Gauge Units	Point 2 - Point 1 Y
11				
12	GeoCalcLatitude 1:	12.6303	Degrees	Calibration Point 1 Lat
13	GeoCalcLongitude 1:	99.9533	Degrees	Calibration Point 1 Lon
14	GeoCalcLatitude 2:	13.7667	Degrees	Calibration Point 2 Lat
15	GeoCalcLongitude 2:	102.3167	Degrees	Calibration Point 2 Lon
16				
17	GeoCalcDistance:	153.9876	NM	
18	Delta Latitude:	=B14-B12	Degrees	
19	Delta Latitude:	=B18*60	NM	Component_Y
20	Long Axis Distance:	=SQRT(B17^2-B19^2)	NM	Component_X
21	Long Axis Distance:	=B20*1852	Meters	Component_X
22	Short Axis XML Map Scale:	=2*B1*1852/B3	Meters per Gauge Unit	
23	Long Axis XML Map Scale:	=B21/B9	Meters per Gauge Unit	
24				
25	(Scale) / (Z Factor):	**=B22/B1**	Short Axis Scale as function of Z Factor	
26	(Scale) / (Z Factor):	**=B23/B1**	Long Axis Scale as function of Z Factor	

The XML map scale – Zoom functions calculated from the FS9 calibration example is shown below:



Manual Calibration Summary Points

Accuracy is improved if two points are clicked for measurement by the mouse rather than using map CenterX and CenterY as one of the points.

XML examples for all calibration techniques are included in the download section of the BlackBox/CustomDraw website.

Transforming Lat/Lon Coordinates to Gauge Units

And Vice Versa

Making XML overlays for the CustomDraw map involves transforming latitude and longitude of overlay objects you wish to display into gauge units in order to accurately position them with respect to the underlying CustomDraw moving map. Using the mouse to identify coordinates or distances on the map involves the reverse – transforming XML gauge units into latitude and longitude. Several very cool applications are possible using these transforms.

The transform process is described by these simple relationships:

Meters_X = Scale_X x Gauge Units_X x Zoom Factor

Meters_Y = Scale_Y x Gauge Units_Y x Zoom Factor

where

- Meters: The real earth East-West ("X") and North-South ("Y") distance from the reference point, normally the users aircraft position, to the point of interest
- Scale: The Scale_X and Scale_Y functions derived during map calibration
- Gauge Units: The gauge unit difference (DeltaGU_X and DeltaGU_Y) between the reference point and the point of interest
- Zoom Factor: The map zoom setting where Zoom = Zoom Factor x 1852

❑ Transforming Lat/Lon Coordinates to Gauge Units: Creating Map Overlays

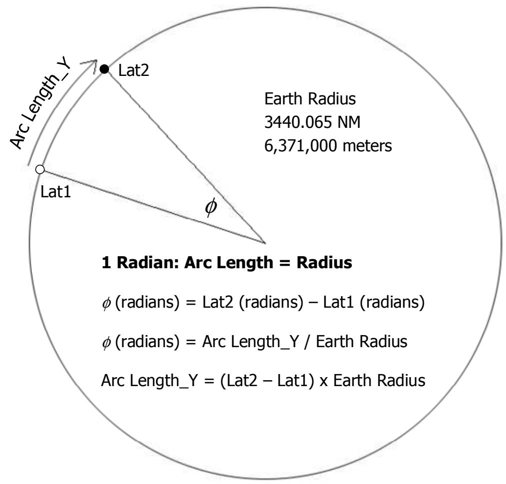
This example demonstrates the coordinate transform step for making a TCAS overlay from traffic coordinates returned by ITrafficInfo variables. The task is to determine the gauge unit position of the intruder aircraft given its latitude and longitude so it can be displayed using XML gauge units. When TrackUp = 0 (top of the map is True North), it is a straightforward two-step process of determining X and Y distance from the lat/lon pairs and then converting the X,Y distance into gauge units.

Distance calculation is separated into N-S and E-W components using spherical geometry assumptions shown on the following page. The North-South "Y" distance, or arc length, is:

Arc Length_Y = (Latitude2 – Latitude1) x Earth Radius where Latitude1 and 2 are expressed in radians, not degrees. I use Earth Radius = 3440.065 NM or 6371000 meters.

The East-West "X" component has a similar approach but Arc Length_X must be corrected for latitude:

Arc Length_X = (Longitude2 – Longitude1) x (Earth Radius x cos(Lat2)) where Longitude1 and 2 are expressed in radians.



Determination of distance should also account for the special case where the user aircraft and intruder aircraft are on opposite sides of the equator and/or prime meridian.

The second step, converting Arc Length_X and Arc Length_Y into gauge units, involves application of the map scales derived during calibration. The equations

DeltaGU_X = Arc Length_X / (Scale_X x Zoom Factor)

DeltaGU_Y = Arc Length_Y / (Scale_Y x Zoom Factor)

yield gauge units that are measured relative to the users aircraft position, CenterX and CenterY. Therefore, the final display location, Gauge_X and Gauge_Y, is the sum of the relative gauge units plus aircraft position, i.e., Gauge_X = DeltaGU_X + CenterX.

Example XML for FSX used for air traffic in a TCAS display:

```
(L:ZFactor, number) 1852 * (>L:Zoom, number)
(L:Reference Latitude, radians) (C:ITrafficInfo:C:PLANE LATITUDE, radians) - (>L:TCAS_DeltaLat, radians)
(C:ITrafficInfo:C:PLANE LONGITUDE, radians) (L:Reference Longitude, radians) - (>L:TCAS_DeltaLon, radians)
(C:ITrafficInfo:C:PLANE LATITUDE, radians) cos (>L:TCAS_CosLat2, number)
(L:TCAS_DeltaLat, radians) (L:EarthRadius, meters) * (>L:TCAS_ArcLen_Y, meters)
(L:TCAS_DeltaLon, radians) (L:EarthRadius, meters) * (L:TCAS_CosLat2, number) * (>L:TCAS_ArcLen_X, meters)
(L:TrackUp, bool) 0 ==
  if{
     (L:TCAS_ArcLen_Y, meters) (L:Scale_Y, number) / (L:ZFactor, number) / (>L:TCAS_DeltaGU_Y, number)
     (L:Zoom, number) 500000 &lt;
        if{ (L:TCAS_ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) /
           (>L:TCAS_DeltaGU_X, number) }
        els{ (L:TCAS_ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) /
           (L:TCAS_CosLat2, number) / (>L:TCAS_DeltaGU_X, number) }
     (L:TCAS_DeltaGU_Y, number) (L:CenterY, number) + (>L:TCAS_Gauge_Y, number)
     (L:TCAS_DeltaGU_X, number) (L:CenterX, number) + (>L:TCAS_Gauge_X, number)
  }
```

Lines 12 applies a cos(Lat2) correction to compensate for the projection change.

In FS9, the line 9 through 12 equivalent would reduce to 2 lines:

```
9  (L:ArcLen_Y, meters) (L:Scale_Y, number) / (L:ZFactor, number) / (>L:DeltaGU_Y, number)
11 (L:ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) / (>L:DeltaGU_X, number)
```

❑ **TrackUp = 1**

TrackUp = 1 is the normal map configuration for an aircraft gps or MFD display. In this mode, the ground track of the aircraft determines the direction to which the top of the map points. The map continuously rotates as the user aircraft changes ground path direction during flight. To display overlay objects such as other air traffic in the correct position with respect to the underlying CustomDraw map, the gauge units of the overlay object must also be rotated consistent with the base map.

There are two approaches to accomplish this. The first utilizes a coordinate rotation transform. For simplicity, I prefer a Euclidean transform applied to the real earth X and Y distances of the overlay object relative to user aircraft and subsequently converting the rotated X, Y into "rotated" gauge units for the XML display. The second is a vector solution in which a rotated, pseudo Lat/Lon is computed given distance and bearing. In this case, distance is the distance to the overlay object and bearing is the true bearing from user aircraft to the overlay object minus the rotation angle, the aircraft ground track direction. Flight Sim's built in GeoCalc variables are well suited for this solution. The gauge units of the pseudo Lat/Lon are then used to display the overlay object with XML. Of the two methods, I prefer Euclidean coordinate rotation; the code is simpler and the results are slightly more accurate.

Euclidean Coordinate Rotation

A two-dimensional coordinate rotation (2-d Affine transform) applies the following matrix multiplication:

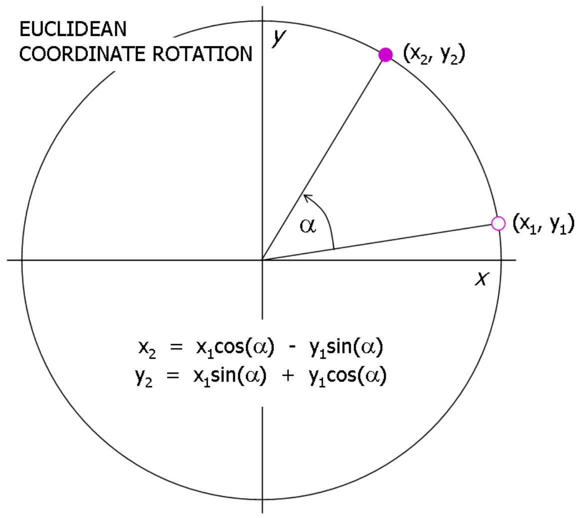
$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

where (x_2, y_2) are the coordinates of point (x_1, y_1) after rotation of angle α around the origin – normally the users aircraft. Expanding the matrix produces:

$$x_2 = x_1\cos(\alpha) - y_1\sin(\alpha)$$

$$y_2 = x_1\sin(\alpha) + y_1\cos(\alpha)$$

Note that the rotation must be applied to real earth X and Y distances, not the gauge units. Following rotation the new point, (x_2, y_2), is converted into gauge units for XML overlay display.



The FSX XML:

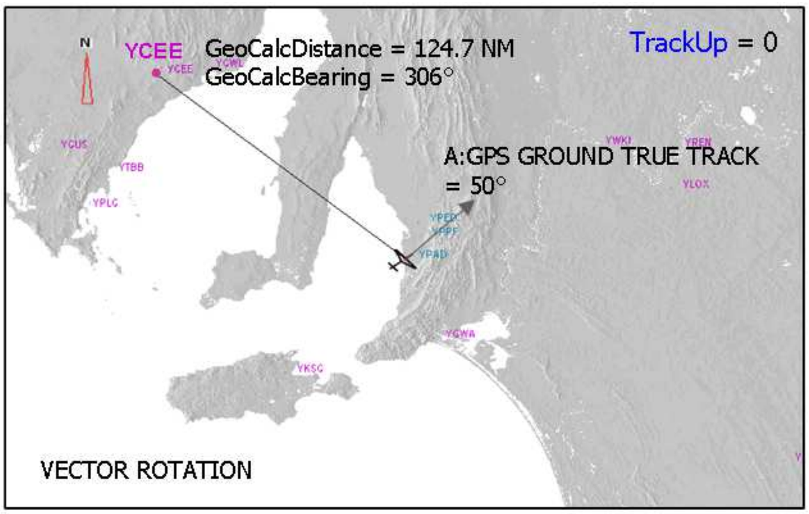
```
1  (L:ZFactor, number) 1852 * (>L:Zoom, number)
2  (L:Reference Latitude, radians) (C:ITrafficInfo:C:PLANE LATITUDE, radians) - (>L:TCAS_DeltaLat, radians)
3  (C:ITrafficInfo:C:PLANE LONGITUDE, radians) (L:Reference Longitude, radians) - (>L:TCAS_DeltaLon, radians)
4  (C:ITrafficInfo:C:PLANE LATITUDE, radians) cos (>L:TCAS_CosLat2, number)
5  (L:TCAS_DeltaLat, radians) (L:EarthRadius, meters) * (>L:TCAS_ArcLen_Y, meters)
6  (L:TCAS_DeltaLon, radians) (L:EarthRadius, meters) * (L:TCAS_CosLat2, number) * (>L:TCAS_ArcLen_X, meters)
7  (L:TrackUp, bool) 0 ==
8     if{
9        (L:TCAS_ArcLen_Y, meters) (L:Scale_Y, number) / (L:ZFactor, number) / (>L:TCAS_DeltaGU_Y, number)
10       (L:Zoom, number) 500000 &lt;
11          if{ (L:TCAS_ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) / (>L:TCAS_DeltaGU_X, number) }
12          els{ (L:TCAS_ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) / (L:TCAS_CosLat2, number) /
               (>L:TCAS_DeltaGU_X, number) }
13       (L:TCAS_DeltaGU_Y, number) (L:CenterY, number) + (>L:TCAS_Gauge_Y, number)
14       (L:TCAS_DeltaGU_X, number) (L:CenterX, number) + (>L:TCAS_Gauge_X, number)
15    }
16 (L:TrackUp, bool) 1 == (L:Zoom, number) 500000 &lt; and
17    if{
18       (L:TCAS_ArcLen_X, meters) (A:GPS GROUND TRUE TRACK, radians) /-/ cos * (L:TCAS_ArcLen_Y, meters)
            (A:GPS GROUND TRUE TRACK, radians) /-/ sin * - (>L:TCAS_ArcLen_X2, meters)
19       (L:TCAS_ArcLen_X, meters) (A:GPS GROUND TRUE TRACK, radians) /-/ sin * (L:TCAS_ArcLen_Y, meters)
            (A:GPS GROUND TRUE TRACK, radians) /-/ cos * + (>L:TCAS_ArcLen_Y2, meters)
20       (L:TCAS_ArcLen_X2, meters) (L:Scale_X, number) / (L:ZFactor, number) / (>L:TCAS_DeltaGU_X, number)
21       (L:TCAS_ArcLen_Y2, meters) (L:Scale_Y, number) / (L:ZFactor, number) / (>L:TCAS_DeltaGU_Y, number)
22       (L:TCAS_DeltaGU_Y, number) (L:CenterY, number) + (>L:TCAS_Gauge_Y, number)
23       (L:TCAS_DeltaGU_X, number) (L:CenterX, number) + (>L:TCAS_Gauge_X, number)
24    }
```

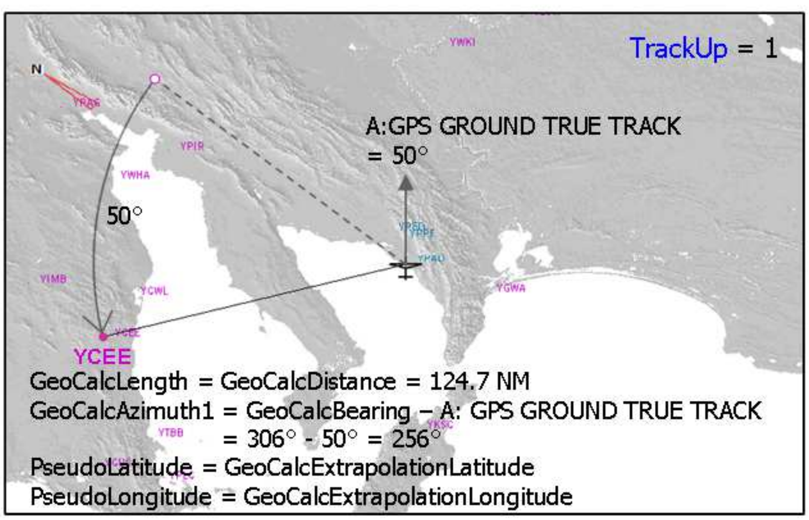
where the coordinate rotation script for TrackUp = 1 starts at line 18. ArcLen_X, ArcLen_Y are (x_1, y_1) and ArcLen_X2, ArcLen_Y2 are (x_2, y_2). The rotation angle, α, is (A:GPS GROUND TRUE TRACK, radians).

Vector Rotation given Distance and Bearing

The second approach creates rotated coordinates for the overlay object by subtracting the rotation angle from the true bearing to the overlay object, then computing a "pseudo" latitude and longitude given distance and the adjusted bearing. Subsequently, the pseudo latitude and longitude are converted to gauge units for display by XML. Refer to the diagram below.

Formulas for latitude and longitude given distance and bearing can be found in the excellent reference, "Aviation Formulary v.1.46" by Ed Williams, but the most convenient way to compute the pseudo latitude and longitude is to use the built in gps GeoCalc variables, GeoCalcLength, Azimuth1, ExtrapolationLatitude and ExtrapolationLongitude.





The FSX XML:

```
1  (L:ZFactor, number) 1852 * (>L:Zoom, number)
2  (A:PLANE LATITUDE, radians) (C:ITrafficInfo:C:PLANE LATITUDE, radians) - (>L:DeltaLat, radians)
3  (C:ITrafficInfo:C:PLANE LONGITUDE, radians) (A:PLANE LONGITUDE, radians) - (>L:DeltaLon, radians)
4  (C:ITrafficInfo:C:PLANE LATITUDE, radians) cos (>L:CosLat2, number)
5  (L:DeltaLat, radians) (L:EarthRadius, meters) * (>L:ArcLen_Y, meters)
6  (L:DeltaLon, radians) (L:EarthRadius, meters) * (L:CosLat2, number) * (>L:ArcLen_X, meters)
7  (L:TrackUp, bool) 0 ==
8    if{
9      (L:ArcLen_Y, meters) (L:Scale_Y, number) / (L:ZFactor, number) / (>L:DeltaGU_Y, number)
10     (L:Zoom, number) 500000 &lt;
11       if{ (L:ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) / (>L:DeltaGU_X, number) }
12       els{ (L:ArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) / (L:CosLat2, number) /
13         (>L:DeltaGU_X, number) }
14     (L:DeltaGU_Y, number) (L:CenterY, number) + (>L:Gauge_Y, number)
15     (L:DeltaGU_X, number) (L:CenterX, number) + (>L:Gauge_X, number)
16   }
17 (L:TrackUp, bool) 1 == (L:Zoom, number) 500000 &lt; and
18   if{
19     (A:PLANE LATITUDE, degrees) (>C:fs9gps:GeoCalcLatitude1, degrees)
20     (A:PLANE LONGITUDE, degrees) (>C:fs9gps:GeoCalcLongitude1, degrees)
21     (C:ITrafficInfo:C:PLANE LATITUDE, degrees) (>C:fs9gps:GeoCalcLatitude2, degrees)
22     (C:ITrafficInfo:C:PLANE LONGITUDE, degrees) (>C:fs9gps:GeoCalcLongitude2, degrees)
23     (C:fs9gps:GeoCalcBearing, degrees) (A:GPS GROUND TRUE TRACK, degrees) - dnor
24       (>C:fs9gps:GeoCalcAzimuth1, degrees)
25     (C:fs9gps:GeoCalcDistance, nmiles) (>C:fs9gps:GeoCalcLength, nmiles)
26     (A:PLANE LATITUDE, degrees) (C:fs9gps:GeoCalcExtrapolationLatitude, degrees) -
27       (>L:PseudoDeltaLat, degrees)
28     (C:fs9gps:GeoCalcExtrapolationLongitude, degrees) (A:PLANE LONGITUDE, degrees) -
29       (>L:PseudoDeltaLon, degrees)
30     (C:fs9gps:GeoCalcExtrapolationLatitude, radians) cos (>L:PseudoCosLat2, number)
31     (L:PseudoDeltaLat, radians)  (L:EarthRadius, meters) * (>L:PseudoArcLen_Y, meters)
32     (L:PseudoArcLen_Y, meters) (L:ZFactor, number) / (L:Scale_Y, number) / (>L:DeltaGU_Y, number)
33     (L:PseudoDeltaLon, radians)  (L:EarthRadius, meters) * (L:PseudoCosLat2, number) *
34       (>L:PseudoArcLen_X, meters)
35     (L:PseudoArcLen_X, meters) (L:Scale_X, number) / (L:ZFactor, number) / (>L:DeltaGU_X, number)
36     (L:DeltaGU_Y, number) (L:CenterY, number) + (>L:Gauge_Y, number)
37     (L:DeltaGU_X, number) (L:CenterX, number) + (>L:Gauge_X, number)
38   }
```

The vector rotation script starts at line 19. Explanation of the GeoCalc variables can be found in the FS9GPS Module Guidebook.

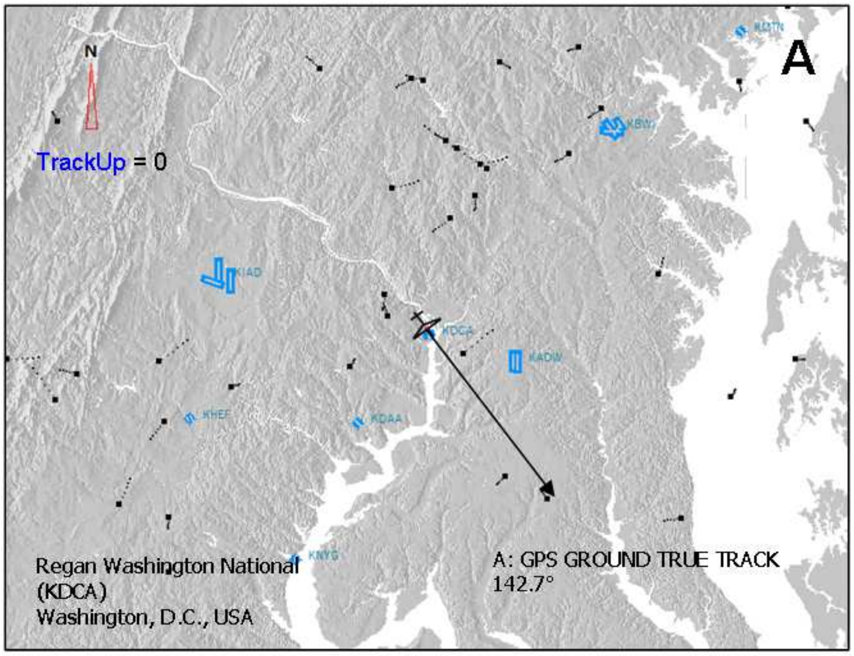
The TrackUp=1 rotations described above are valid in FS9 at all zoom levels and in FSX at zooms less than 500 km where sinusoidal projection is used. Unfortunately, FSX introduces an unexpected incremental rotation at zooms greater than or equal to 500 km with the equidistant cylindrical projection and consequently, the rotation methods I use do not work. Until I can determine how to predict this, I have no solution for TrackUp=1 for zooms >= 500 km.

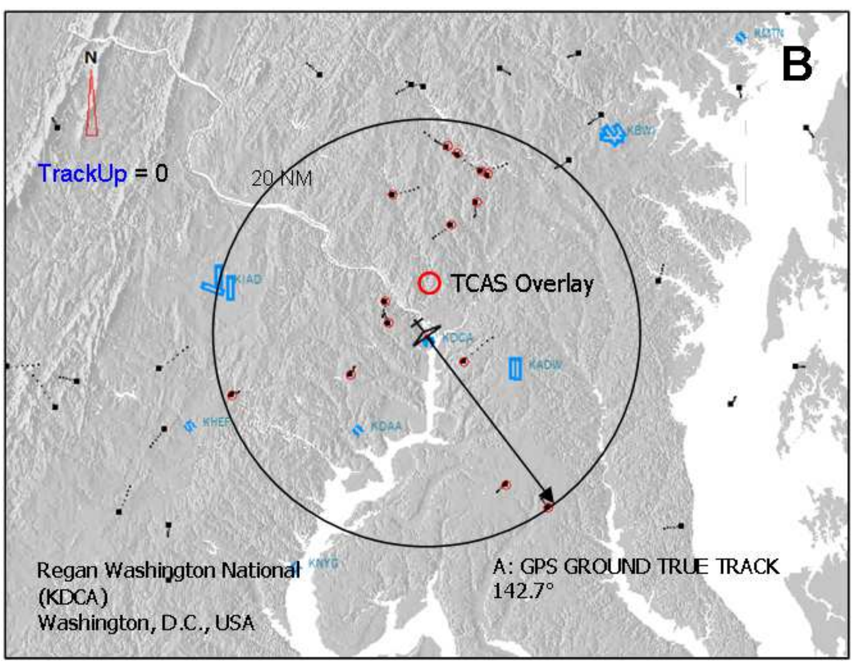
		Zoom	
		<500 km	>=500 km
FS9	TrackUp=0	✓	✓
FS9	TrackUp=1	✓	✓
FSX	TrackUp=0	✓	✓
FSX	TrackUp=1	✓	✗

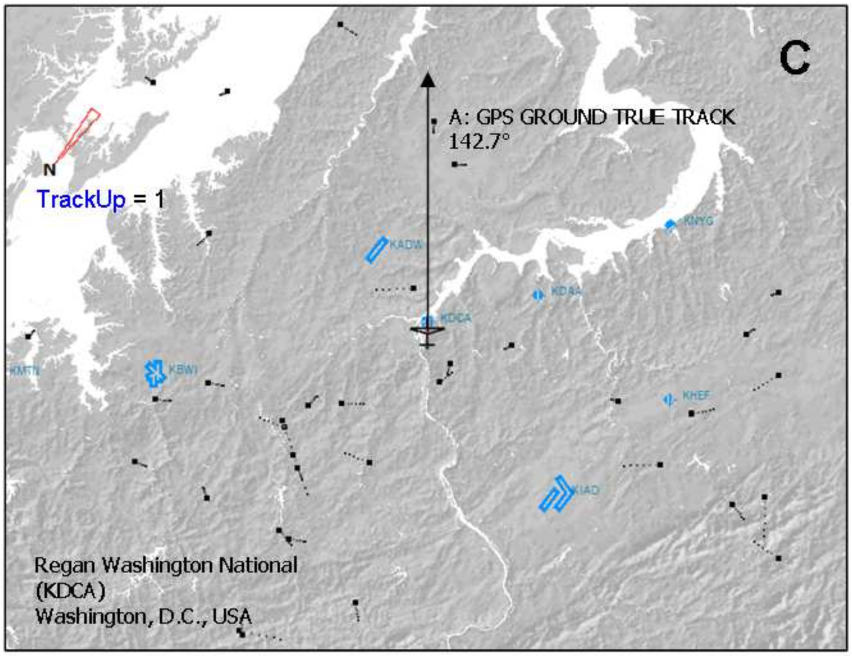
TCAS Overlay Example (FSX)

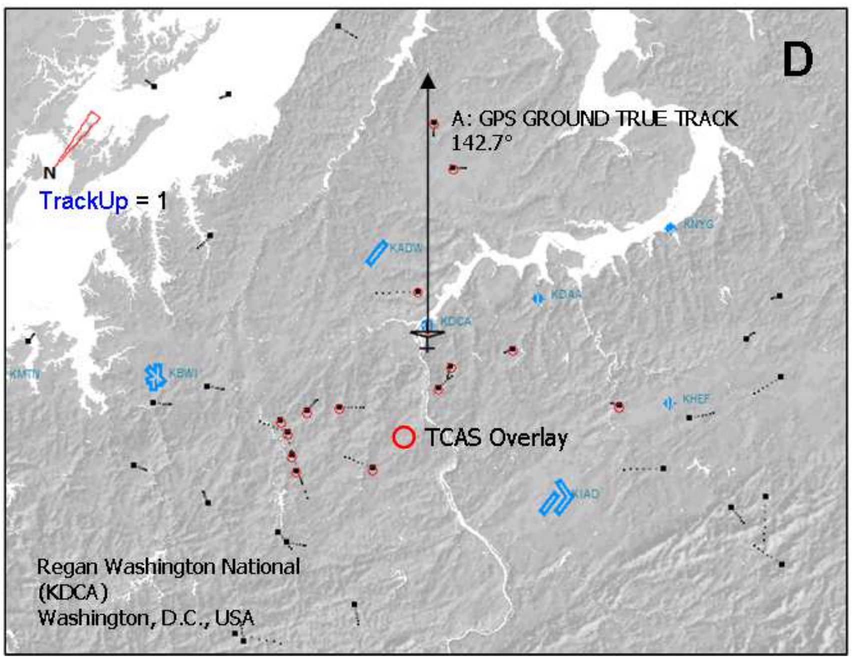
The maps below demonstrate a TCAS overlay that can be scripted using XML. Figure **A** is a TrackUp = 0 map showing all airborne traffic in the vicinity Regan Washington National Airport using the ATC small black square ■ symbol. The small red circles ○ in Figure **B** are the overlay points whose gauge display coordinates were generated with the XML previously described.

Figures **C** and **D** are the equivalent maps but with TrackUp = 1.









Advantages of a TCAS overlay include:

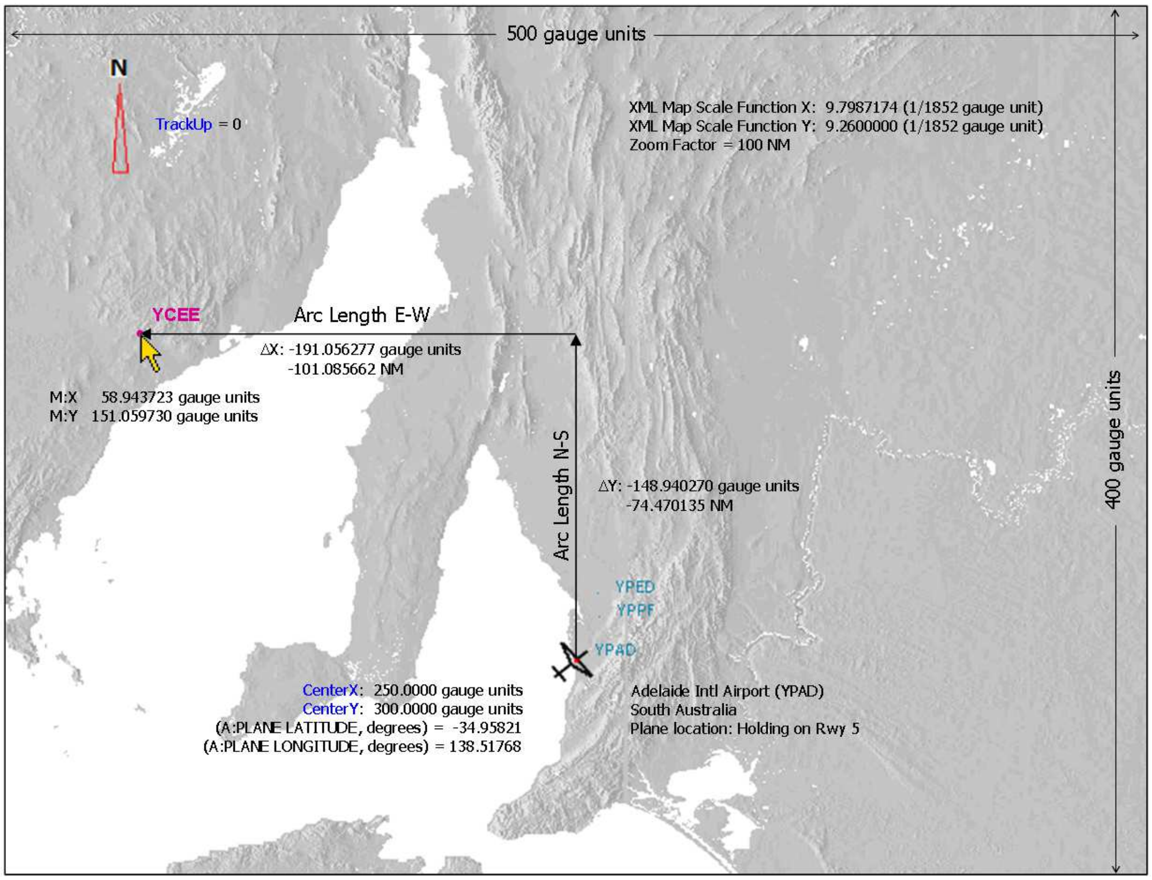
- The TCAS search radius can be limited to ~20 - 30 NM which is the design specification for real TCAS systems. LayerVehicles displays all aircraft traffic in map view, even traffic in excess of 30 NM that would never be seen by real TCAS.

- Accurate intruder alerts (Proximate Traffic, Traffic Advisory and Resolution Advisory) can be computed in real time from information returned by the ITrafficInfo search together with a Closest Point of Approach algorithm.
- An overlay allows utilization of the CustomDraw map terrain base, if desired.
- Realistic looking, alert status dependent, custom TCAS traffic symbols can be displayed, all positioned accurately with respect to the underlying CustomDraw moving map (in TCAS mode, LayerVehicles would not be displayed, only the TCAS overlay would be displayed).
- Some dis-advantages include 1) many <Element>, 2) interrogation geometry that is circular rather than ellipsoidal front-looking.

The TCAS chapter discusses this in more detail.

- **Transforming Gauge Units (Mouse Click) to Lat/Lon Coordinates: Determining Distance, Bearing, Latitude and Longitude from a Mouse Click**

This straight forward solution involves calculation of N-S and E-W arc lengths (distances) of the clicked point from the aircraft position using mouse coordinates, XML map scale functions and Zoom Factor and then computing latitude and longitude from the resulting spherical angles. Distance and bearing to the mouse click point are calculated using gps variables after that.



In the example above, the aircraft is holding on Rwy 5 at Adelaide International Airport, South Australia, and the airport symbol for Cleve Airport, Cleve South Australia (YCEE), is clicked.

Mouse functions M:X and M:Y return the gauge unit coordinates of the mouse click, which, in the example, are X: 58.943723 and Y: 151.059730 gauge units.

The coordinate calculations that follow are valid for TrackUp=0, or, top of the map is True North. When TrackUp=1, a rotation to reverse out the aircraft ground track is required for the correct latitude, longitude and bearing of the mouse click point. After click Latitude and Longitude are calculated, distance and bearing are easily determined using GeoCalc variables.

Mouse Y (M:Y):	**151.059730**	gauge units
Mouse X (M:X):	**58.943723**	gauge units
PLANE LATITUDE (Lat1):	**-34.958211**	degrees
PLANE LONGITUDE (Lon1):	**138.517680**	degrees
Plane Y (CenterY):	**300**	gauge units
Plane X (CenterX):	**250**	gauge units
XML Map Scale Y:	**9.260000**	1/(1852 gauge units)
XML Map Scale X:	**9.798717**	1/(1852 gauge units)
Z Factor:	**100**	NM
Average Earth Radius:	**3440.065**	NM
Flight Simulator X? :	**1**	1 = FSX; 0 = FS9
Mouse Delta Y:	-148.940270	gauge units
Mouse Delta X:	-191.056277	gauge units
Zoom:	185200	meters
Arc Length_Y (N-S):	-74.470135	NM
Delta Latitude Radians:	-0.021648	radians
Click Point Latitude:	**-33.717878**	degrees
Cosine Average Latitude:	0.825724	unitless
Sinusoidal Projection Adjust:	1	unitless
Arc Length_X (E-W):	-101.085658	NM
Delta Longitude Radians:	-0.035587	radians
Click Point Longitude:	**136.478711**	degrees

	A	B	C
1	Mouse Y (M:Y):	**151.059730**	gauge units
2	Mouse X (M:X):	**58.943723**	gauge units
3	PLANE LATITUDE (Lat1):	**-34.958211**	degrees
4	PLANE LONGITUDE (Lon1):	**138.517680**	degrees
5	Plane Y (CenterY):	**300**	gauge units
6	Plane X (CenterX):	**250**	gauge units
7	XML Map Scale Y:	**9.260000**	1/(1852 gauge units)
8	XML Map Scale X:	**9.798717**	1/(1852 gauge units)
9	Z Factor:	**100**	NM
10	Average Earth Radius:	**3440.065**	NM
11	Flight Simulator X? :	**1**	1 = FSX; 0 = FS9
12	Mouse Delta Y:	=B1-B5	gauge units
13	Mouse Delta X:	=B2-B6	gauge units
14	Zoom:	=B9*1852	meters
15	Arc Length_Y (N-S):	=B12*B7*B9/1852	NM
16	Delta Latitude Radians:	=B15/B10	radians
17	Click Point Latitude:	**=B3-DEGREES(B16)**	degrees
18	Cosine Average Latitude:	=COS(RADIANS((B3+B17)/2))	unitless
19	Sinusoidal Projection Adjust:	=IF(B14>=500000,B18,1)	unitless
20	Arc Length_X (E-W):	=B13*B8*B9*B19/1852	NM
21	Delta Longitude Radians:	=B20/(B10*B18)	radians
22	Click Point Longitude:	**=B4+DEGREES(B21)**	degrees

The yellow variables are the mouse click coordinates returned by the M:X and M:Y functions, the variables in red are knowns, or givens, and the rest are simple calculations. The green variables are the values initially being sought; latitude and longitude of the mouse click.

The equivalent XML is shown below. This should be placed within an <Update> section of the code.

- Lines 3 - 17: The click latitude and longitude calculations.
- Lines 18 - 24: The distance and bearing calculations using GeoCalc variables.
- Lines 28 – 35: Latitude, longitude and bearing calculation for the TrackUp = 1 case.

```
1  (L:CalculateClikDist, bool)
2    if{
3      (L:ZFactor, number) 1852 * (>L:Zoom, number)
4      (L:Mouse_X, number) (L:CenterX, number) - (>L:MouseDelta_X, number)
5      (L:Mouse_Y, number) (L:CenterY, number) - (>L:MouseDelta_Y, number)
6      (L:MouseDelta_Y, number) (L:Scale_Y, number) * (L:ZFactor, number) * 1852 /
7        (>L:ClikArcLength_Y, nmiles)
8      (L:ClikArcLength_Y, nmiles) (L:EarthRadius, nmiles) / (>L:DeltaLatitude, radians)
9      (A:PLANE LATITUDE, degrees) (L:DeltaLatitude, degrees) - (>L:ClikLatitude, degrees)
10     (A:PLANE LATITUDE, radians) (L:ClikLatitude, radians) + 2 / cos (>L:CosAvgLat, number)
11     (L:Sim_FSX, bool) 1 == (L:Zoom, number) 500000 &lt; and
12        if{ (L:MouseDelta_X, number) (L:Scale_X, number) * (L:ZFactor, number) *  1852 /
13          (>L:ClikArcLength_X, nmiles) }
14       els{ (L:MouseDelta_X, number) (L:Scale_X, number) * (L:ZFactor, number) *
15          (L:CosAvgLat, number) * 1852 / (>L:ClikArcLength_X, nmiles) }
16     (L:ClikArcLength_X, nmiles) (L:EarthRadius, nmiles) (L:CosAvgLat, number) * / rddg
17       (A:PLANE LONGITUDE, degrees) + (>L:ClikLongitude, degrees)
18     (A:PLANE LATITUDE, degrees) (>C:fs9gps:GeoCalcLatitude1, degrees)
19     (A:PLANE LONGITUDE, degrees) (>C:fs9gps:GeoCalcLongitude1, degrees)
20     (L:ClikLatitude, degrees) (>C:fs9gps:GeoCalcLatitude2, degrees)
21     (L:ClikLongitude, degrees) (>C:fs9gps:GeoCalcLongitude2, degrees)
22     (C:fs9gps:GeoCalcDistance, nmiles) (>L:ClikDistance, nmiles)
23     (C:fs9gps:GeoCalcBearing, degrees) d (>L:ClikBearingTrue, degrees) (A:MAGVAR, degrees) -
24       (>L:ClikBearingMag, degrees)
25
26     (L:TrackUp, bool) 1 == (L:Zoom, number) 500000 &lt; and
27         if{
28           (A:PLANE LATITUDE, degrees) (>C:fs9gps:GeoCalcLatitude1, degrees)
29           (A:PLANE LONGITUDE, degrees) (>C:fs9gps:GeoCalcLongitude1, degrees)
30           (L:ClikDistance, nmiles) (>C:fs9gps:GeoCalcLength, nmiles)
31           (L:ClikBearingTrue, degrees) (A:GPS GROUND TRUE TRACK, degrees) + dnor
32             (>C:fs9gps:GeoCalcAzimuth1, degrees)
33           (C:fs9gps:GeoCalcAzimuth1, degrees) (>L:ClikBearingTrue, degrees)
34           (C:fs9gps:GeoCalcExtrapolationLatitude, degrees) (>L:ClikLatitude, degrees)
35           (C:fs9gps:GeoCalcExtrapolationLongitude, degrees) (>L:ClikLongitude, degrees)
36         }
37     0 (>L:CalculateClikDist, bool)
38   }
```

The mouse code:

```
1  <Area Name="CLICK LAT LON" Left="150" Top="10" Width="500" Height="400">
2    <Cursor Type="Normal"/>
3    <Click Kind="LeftSingle+RightSingle">
4      (M:Event) 'LeftSingle' scmp 0 ==
5       if{
6         1 (>L:DisplayClikDistanceOnMap, bool)
7         1 (>L:CalculateClikDist, bool)
8         (M:X) (>L:Mouse_X, number) (M:Y) (>L:Mouse_Y, number)
9       }
10     (M:Event) 'RightSingle' scmp 0 ==
11      if{
12        0 (>L:DisplayClikDistanceOnMap, bool)
13      }
14   </Click>
15 </Area>
```

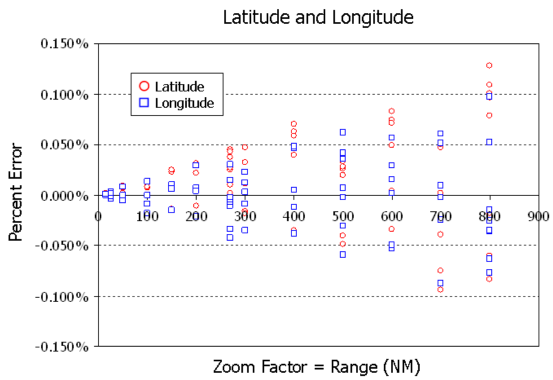
- Line 1: The upper left corner of the CustomDraw map display is located at gauge unit X=150, Y=10. The map display is 500 gauge units wide by 400 gauge units high. Consequently, this line establishes the entire map display as a clickable area
- Line 6: A left mouse click enables display of Latitude and Longitude information on the screen – a readout of the lat, lon, dist, brg calculations
- Line 7: A toggle that allows the calculation code to be executed
- Line 12: A right click disables display of the lat, lon, dist, brg readout

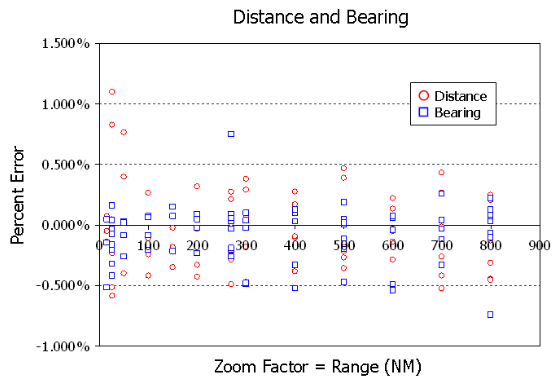
These calculations fail to yield accurate latitude, longitude and bearing information in the FSX case where zoom exceeds 500 km.

		Zoom	
		<500 km	>=500 km
FS9	TrackUp=0	✓	✓
FS9	TrackUp=1	✓	✓
FSX	TrackUp=0	✓	✓
FSX	TrackUp=1	✓	✗

Accuracy

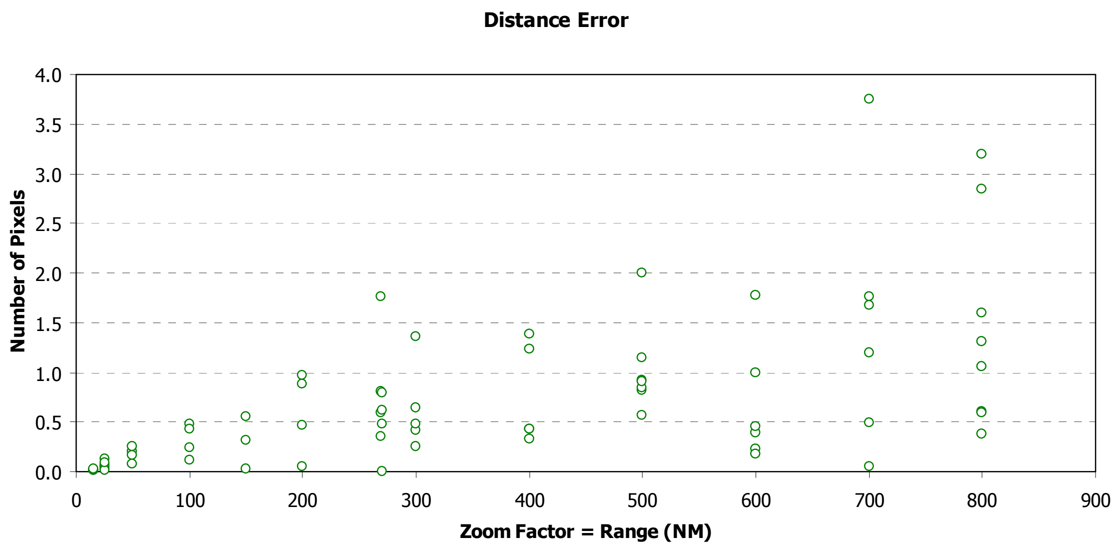
Assuming the XML Map has been calibrated carefully, accuracy of coordinates calculated from a mouse click is quite acceptable I believe.





The cross-plots above show the percent error for coordinates and distances measured using mouse clicks. Latitude and Longitude errors relative to gps variables GeoCalcLatitude and Longitude rise with increasing Zoom Factor, but are less than about 0.1 percent error through 800 NM range. Distance and Bearing errors calculated using the Click Latitude and Longitude are also very low, averaging less than 0.5 percent.

To put those errors in perspective, the plot below translates the errors into physical screen pixels. At Zoom Factors less than 200 NM, accuracy is within one physical pixel, or, in practical terms, essentially no error. Errors reach up to about 3 physical pixels at the 800 NM Range level.



Key Equations

Gauge Units = Meters / (Scale x Zoom Factor)

Scale = Meters / (Gauge Units x Zoom Factor)

Offset Lat Lon given N-S and E-W arc lengths:

Lat_2 = Lat_1 + [ArcLenY / EarthRadius]

Lon_2 = Lon_1 + [ArcLenX / (EarthRadius x cos(Lat_2))]

Lat_2 - Lat_1 = [ArcLenY / EarthRadius]

Lon_2 - Lon_1 = [ArcLenX / (EarthRadius x cos(Lat_2))]

ArcLenY = (Lat_2 - Lat_1) x EarthRadius

ArcLenX = (Lon_2 - Lon_1) x (EarthRadius x cos(Lat_2))

Coordinate rotation:

$X_1 = x_0\cos(a) - y_0\sin(a)$

$Y_1 = x_0\sin(a) + y_0\cos(a)$

TCAS

Traffic Alert and Collision Avoidance System in FSX

Acknowledgements

A discussion of Flight Simulator TCAS must begin with acknowledgement of the original FS9 TCAS developed by Arne Bartels in 2006 and Doug Dawson's re-packaging of it for FSX use in 2012. Arne's FS9 TrafficRadar module can be downloaded free of charge at:

http://library.avsim.net/index.php?CatID=fs2004gau (log-in, then search for "DLL for XML traffic radar / TCAS")

Doug's work is also in the public domain and can be downloaded from the AVSIM Library above, or from the FSDeveloper Downloads site:

http://fsdeveloper.com/forum/downloads.php?do=file&id=104

or here (Doug's web site):

http://www.douglassdawson.ca/

Lastly, acknowledgement is mostly owed to Microsoft for making ITrafficInfo variables available in FSX.

XML TCAS in FSX

The discussion of TCAS that follows ties together previous topics such as scale calibration, lat/lon transforms, vector rotations and ITrafficInfo variables. It shows the math behind simple threat identification and demonstrates one approach for development of a XML-based TCAS overlay that can be superimposed on CustomDraw Map or used in a stand-alone gauge.

The ITrafficInfo group variables in FSX enable interrogation and tracking of AI and multiplayer aircraft traffic. This is the platform of an XML-based TCAS system for FSX.

All real-world TCAS systems contain a traffic display and some level of traffic threat alert capability: Proximate Traffic, Traffic Advisory (TA) and Resolution Advisory (RA). TCAS systems in use today are divided into two groups, TCAS I and TCAS II. For Flight Simulator purposes, TCAS I and TCAS II differ in the sophistication of the alerting capability and collision avoidance maneuver instructions.

TCAS SYSTEM COMPONENTS	TCAS I	TCAS II	Can Be Modeled By Flight Simulator?		Key Flight Simulator Variables
Traffic Display	Required	Required	Yes	FSX	ITrafficInfo variables and A:PLANE variables
Proximate Traffic Threshold	Required	Required	Yes	FSX	ITrafficInfo variables and A:PLANE variables
Traffic Advisory Threshold	Required	Required	Yes	FSX	ITrafficInfo and A:PLANE variables and Closest Point of Approach algorithm
Resolution Advisory Threshold	N/A	Required	Yes	FSX	ITrafficInfo and A:PLANE variables and Closest Point of Approach algorithm
Resolution Advisory Maneuvers and Display	N/A	Required	Yes	FSX	ITrafficInfo variables and A:PLANE variables

In Flight Simulator X, the TCAS Traffic display overlay, TCAS I and TCAS II Proximate Traffic and Traffic Advisory Status, and many features of TCAS II Resolution Advisories can be modeled with XML. Resolution Advisory Maneuvers and a Resolution Advisory Display are beyond the scope of this guidebook; however, references listed at the end of this chapter will be useful guidance to those that want to replicate this capability. Complementary RAs (mutual avoidance maneuvers or TCAS/TCAS coordination) with AI traffic are not possible in Flight Simulator because AI traffic have no collision avoidance capability.

An approach to XML TCAS in FSX

One approach for a FSX XML-based TCAS II system is to construct it from three parts:

- Nearest traffic search using ITrafficInfo variables
- Search interrogation loop. Real-time, continuous assessment of Proximate aircraft, Traffic Advisory and Resolution Advisory status based on US FAA or ICAO TCAS II protocol, and computation of Gauge_X, Gauge_Y map position for each intruder aircraft
- Traffic display overlay for the CustomDraw map which enables the use of custom traffic bitmaps or polygons (i.e., realistic looking symbols that change according to alert status) instead of the stock FSX CustomDraw traffic symbol.

A general logic flow is shown on the next page. In my application, the ITraffic search instructions and the interrogation loop are contained within the <Update> section and the traffic display overlay in <Element>.

Nearest Traffic Search

ITrafficInfo Nearest Traffic Search Parameters

C:ITrafficInfo:Filter	80 (Awake and In Air)
C:ITrafficInfo:Radius	20 Nautical Miles (14 to 30, user preference)
C:ITrafficInfo:MaxVehicles	30
C:ITrafficInfo:Latitude	A:PLANE LONGITUDE
C:ITrafficInfo:Longitude	A:PLANE LATITUDE

ITrafficInfo Search Results Interrogation Loop

Loop through Intruder Aircraft returned by the ITraffic Search

Intruder Aircraft 1 through C:ITrafficInfo:ListSize

→ Interrogate:

- C:TrafficInfo:C:PLANE LATITUDE
- C:TrafficInfo:C:PLANE LONGITUDE
- C:TrafficInfo:C:PLANE ALTITUDE
- C:TrafficInfo:C:VERTICAL SPEED
- C:TrafficInfo:C:GROUND VELOCITY

Calculate:

- Time of Closest Point of Approach (Range Tau)
- Time of Co-Altitude (Vertical Tau)

Assess:

- Proximate, TA or RA Status, Relative Altitude, VSI Arrow

Compute:

- Gauge_X and Gauge_Y map position

Store:

- Proximate, TA or RA Status, Relative Altitude, VSI Arrow
- Gauge_X, Gauge_Y into L:VARs or XMLVARS

Next

Traffic Display Overlay

TCAS Map Display

Display Element:

- Proximate, TA and RA Aircraft Symbol as appropriate
- Display using:

```
<Shift><Value>'Intruder_1_GaugeX</Value><Scale X="1"/></Shift>
<Shift><Value>'Intruder_1_GaugeY</Value><Scale Y="1"/></Shift>
```

FAA TCAS II Protocol

Example code written for this guidebook incorporates TCAS II protocol described in the following US F.A.A. reference, "Introduction to TCAS II Version 7.1" (February, 2011):

http://www.faa.gov/documentLibrary/media/Advisory_Circular/

TCAS%20II%20V7.1%20Intro%20booklet.pdf

XML design elements include (page number refers to the F.A.A. reference):

- Page 13: Proximate Traffic definition 6 NM and +/- 1200 feet
- Page 13-14: Traffic Display Symbology
- Page 17: Simultaneously track up to 30 transponder equipped aircraft within a nominal range of 30 nmi.
- Page 22: TCAS Control panel switch StandBy, TA-Only, and TA-RA
- Page 22-23: TCAS Sensitivity Levels (SL) based on TCAS control panel switch position and altitude
- Page 23: Tau. Time-to-go to Closest Point of Approach and Co-Altitude. Range Tau and Vertical Tau alarm thresholds as indicated in Table 2

Table 2. Sensitivity Level Definition and Alarm Thresholds

Own Altitude (feet)	SL	Tau (Seconds)		DMOD (nmi)		ZTHR (feet) Altitude Threshold		ALIM (feet)
		TA	RA	TA	RA	TA	RA	RA
< 1000 (AGL)	2	20	N/A	0.30	N/A	850	N/A	N/A
1000 - 2350 (AGL)	3	25	15	0.33	0.20	850	600	300
2350 - 5000	4	30	20	0.48	0.35	850	600	300
5000 - 10000	5	40	25	0.75	0.55	850	600	350
10000 - 20000	6	45	30	1.00	0.80	850	600	400
20000 - 42000	7	48	35	1.30	1.10	850	700	600
> 42000	7	48	35	1.30	1.10	1200	800	700

Introduction to TCAS II Version 7.1, US Department of Transportation, Federal Aviation Administration, Feb 28, 2011

- Page 23-25: Distance Modification (DMOD) and Altitude Threshold (ZTHR) alarm threshold modifications
- Page 28: Target on ground determination. Advisory Inhibit if intruder Radio Height (simplifying assumption) is less that 360 feet
- Page 29: Inhibit threat declaration against intruder aircraft with vertical rates in excess of 10,000 fpm

Range Tau and Vertical Tau

TCAS computers primarily incorporate time separation calculations rather than distance separation to determine traffic alerts. Alert criteria, or thresholds, are divided into vertical and slant time components. In the vertical dimension, the time to co-altitude is called Vertical Tau and in the slant (range) dimension, the time to closest point of approach is called Range Tau. A TA or an RA is issued only when both the range tau and vertical tau are less than certain threshold values that are a function of altitude (Sensitivity Levels, see Table 2).

Range tau is equal to the slant range divided by the relative closing speed between own aircraft and the intruder. It can be calculated by comparing changes in slant distance from one ITraffic interrogation cycle to the next as follows:

$Slant\_Distance_1$ (NM) = Slant Distance previous interrogation cycle

$Slant\_Distance_0$ (NM) = Slant Distance current interrogation cycle

$Time_1$ = Time of previous interrogation cycle

$Time_0$ = Current time

Delta_Distance (NM) = $Slant\_Distance_1 - Slant\_Distance_0$

Delta_Time (seconds) = $Time_0 - Time_1$

Closing_Speed (NM per sec) = Delta_Distance / Delta_Time

Range Tau_0 (seconds) = $Slant\_Distance_0$ / Closing_Speed

ITrafficInfo:CurrentDistance is a slant range distance that would appear well suited for this calculation. However, Flight Simulator updates this variable every two seconds only, so it is not ideal for a TCAS application. Consequently, I prefer to derive slant distance using GeoCalc and system variables as follows:

$$Slant\_Distance_0 = \sqrt{(GeoCalcDistance_0^2 + Relative\_Altitude_0^2)}$$

This calculation involves the following variables, all of which are updated each gauge update cycle:

(C:ITrafficInfo:C:PLANE ALTITUDE, feet)

(A:PLANE ALTITUDE, feet)

(A:PLANE LATITUDE, degrees)

(A:PLANE LONGITUDE, degrees)

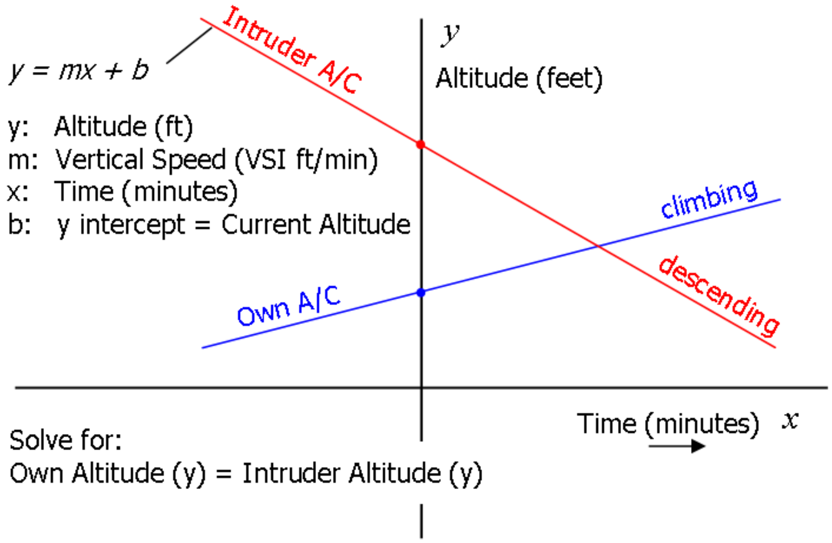
(C:ITrafficInfo:C:PLANE LATITUDE, degrees)

(C:ITrafficInfo:C:PLANE LONGITUDE, degrees)

(C:fs9gps:GeoCalcDistance, nmiles)

Vertical tau can be solved by an intersection of two lines method:

$$\text{Vertical Tau (seconds)} = \frac{Intruder\_Altitude_0 - Own\_Altitude_0}{Own\_VSI_0 - Intruder\_VSI_0} \times 60$$



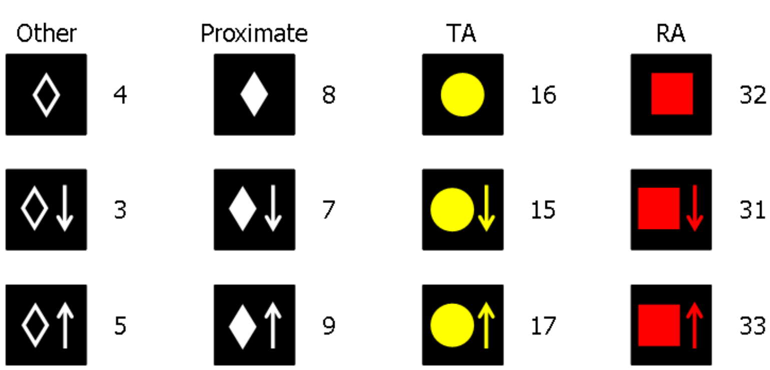
DMOD and ZTHR

TA and RA thresholds are further modified for low closure rate situations where an intruder can come very close while range and vertical tau remain above standard thresholds. These modifications are discussed in the FAA TCAS II v7.1 reference.

Display Variables and Arrays

At the conclusion of every ITrafficInfo interrogation cycle, five values must be calculated and stored for each intruder aircraft to provide information for a TCAS overlay.

1. TCAS display symbol code (L:TCAS_Symbol, enum), a function of alert status (Other, Proximate, TA, or RA). An example:



The display symbol code determines which TCAS symbol is used for the display. Note that most real TCAS systems display intruder aircraft on a black background, not superimposed on a color terrain base. If this is the users' preference, then in TCAS mode, set LayerTerrain = 0 and BackgroundColor = 0x010101, or create a stand-alone TCAS gauge that is not an overlay for the CustomDraw map.

2. Relative Altitude measured in hundreds of feet
3. Relative altitude label position shift. When the intruder is above user's aircraft, relative altitude is displayed above the TCAS symbol; when the intruder is below the user's aircraft, relative altitude is displayed below the TCAS symbol
4. Gauge_X position of each intruder aircraft
5. Gauge_Y position of each intruder aircraft

This establishes the need for variable arrays, for example

(L:TCAS_Symbol_0, enum) thru (L:TCAS_Symbol_*n*, enum)

(L:RelativeAltitudeHundreds_0, enum) thru (L:RelativeAltitudeHundreds_*n*, enum)

(L:RelAltPositionShift_0, enum) thru (L:RelAltPositionShift_*n*, enum)

(L:IntruderGaugeX_0, enum) thru (L:IntruderGaugeX_*n*, enum)

(L:IntruderGaugeY_0, enum) thru (L:IntruderGaugeY_*n*, enum)

where "_0" is the index number of the first (nearest) intruder aircraft and "_*n* " is the index of the last intruder aircraft returned in the ITrafficInfo search.

XMLVars for Dynamic Variable Arrays

Traditional L:Vars can be used to create the arrays but the code is lengthy. The easiest solution utilizes Tom Aguilo's **XMLVars** module to create a dynamic variable array each time the ITrafficInfo search results are interrogated. In the example TCAS gauge provided in the BlackBox website, I use XMLVars to store the interrogation results.

The XMLVars module can be downloaded free of charge from:

http://fsdeveloper.com/forum/downloads.php?do=file&id=105

Follow the installation and operation instructions contained in the ReadMe file.

TCAS Overlay Display Example

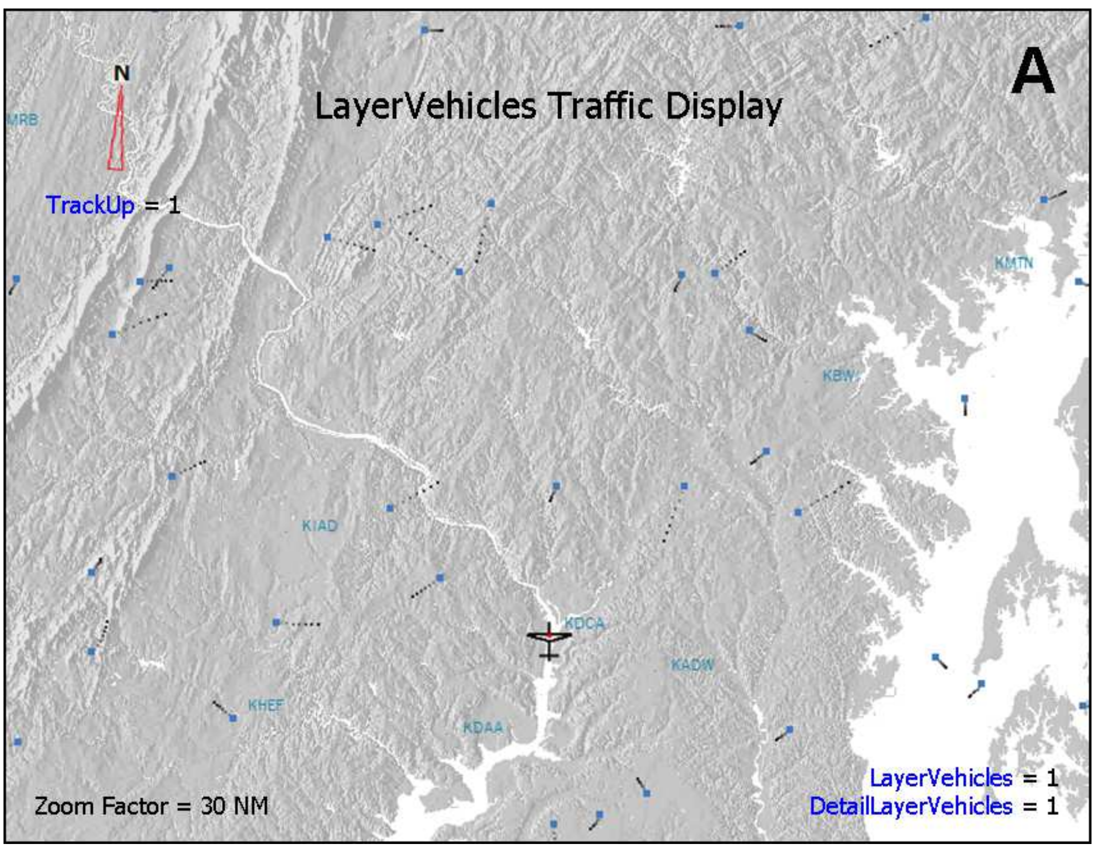
Figure **A** shows airborne aircraft vehicles displayed by LayerVehicles.

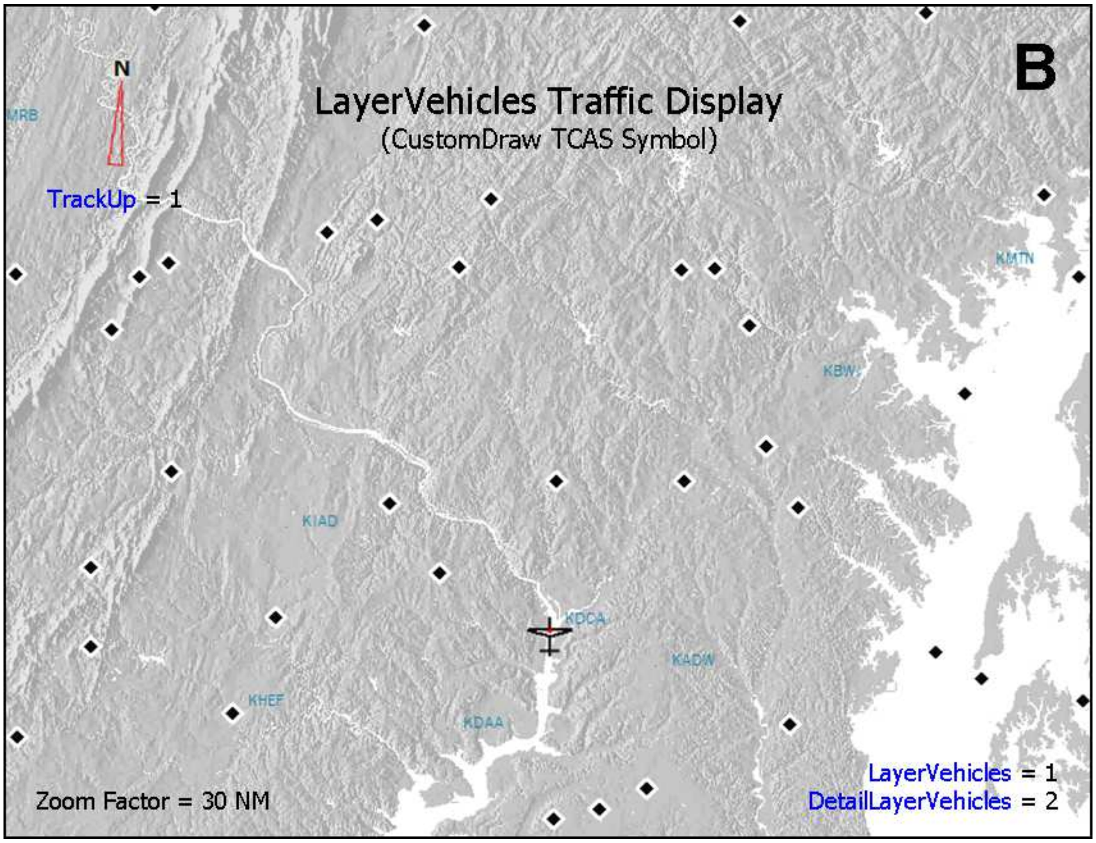
Figure **B** is the same display but with DetailLayerVehicles = 2, CustomDraw's TCAS symbol. Some drawbacks of using FSX CustomDraw's LayerVehicles TCAS symbols:

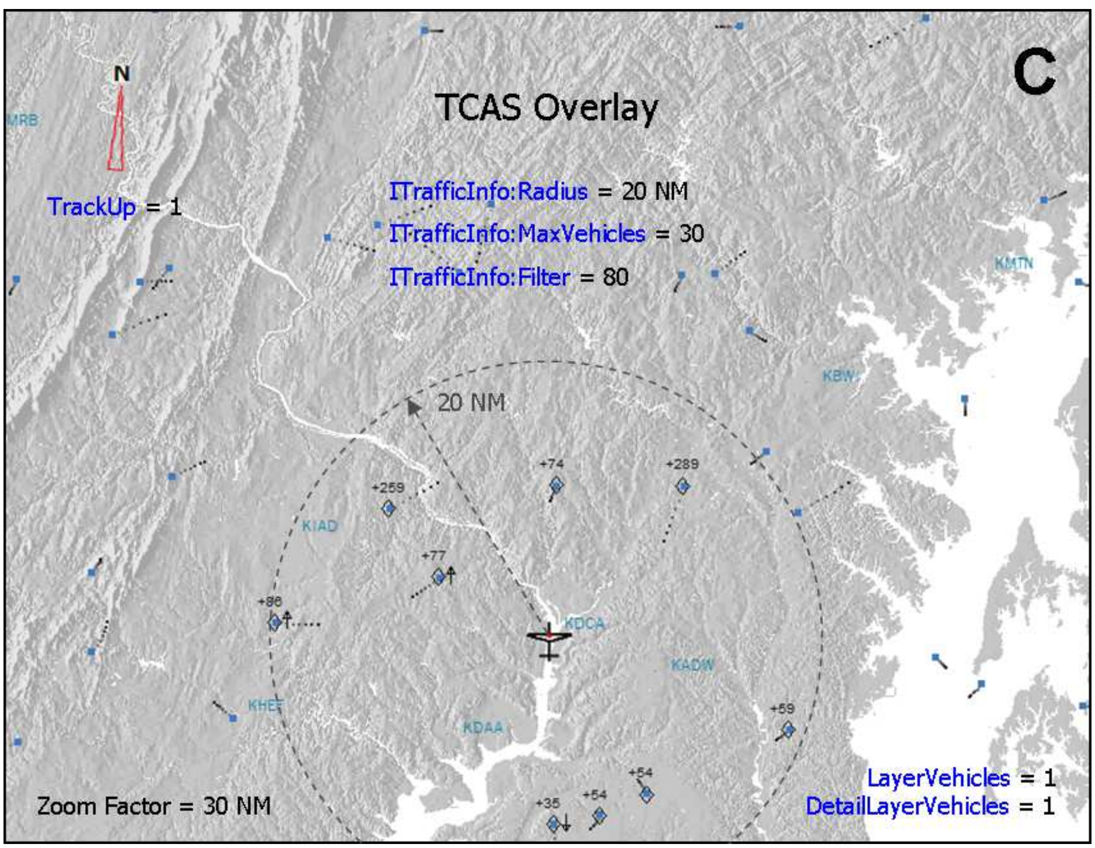
- Symbol does not change as alarm status changes. FSX does not provide TCAS II alarm capability
- Relative altitude and climb/descent arrows are not available
- All AI aircraft in map view are displayed, even those outside real TCAS interrogation limits

Figure **C** shows a TCAS XML overlay on the CustomDraw Map base. ITrafficInfo search parameters consistent with real TCAS units are used.

Finally, Figure **D** shows the TCAS overlay on the CustomDraw map base, but with LayerVehicles = 0.




B
LayerVehicles Traffic Display
(CustomDraw TCAS Symbol)
N
TrackUp = 1
MRB
KMTN
KBWI
KIAD
KDCA
KADW
KHEF
KDAA
Zoom Factor = 30 NM
LayerVehicles = 1
DetailLayerVehicles = 2


C
TCAS Overlay
N
TrackUp = 1
ITrafficInfo:Radius = 20 NM
ITrafficInfo:MaxVehicles = 30
ITrafficInfo:Filter = 80
20 NM
MRB
KMTN
KBWI
KIAD
KDCA
KADW
KHEF
KDAA
Zoom Factor = 30 NM
LayerVehicles = 1
DetailLayerVehicles = 1

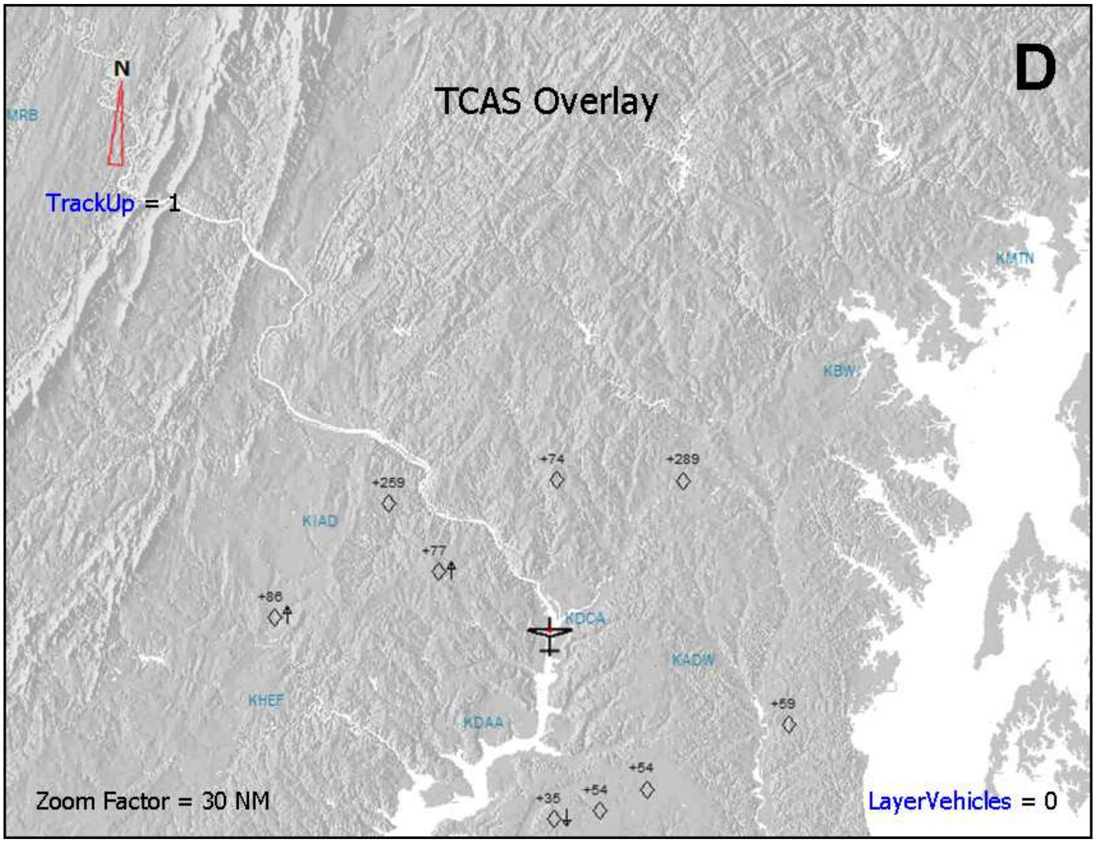
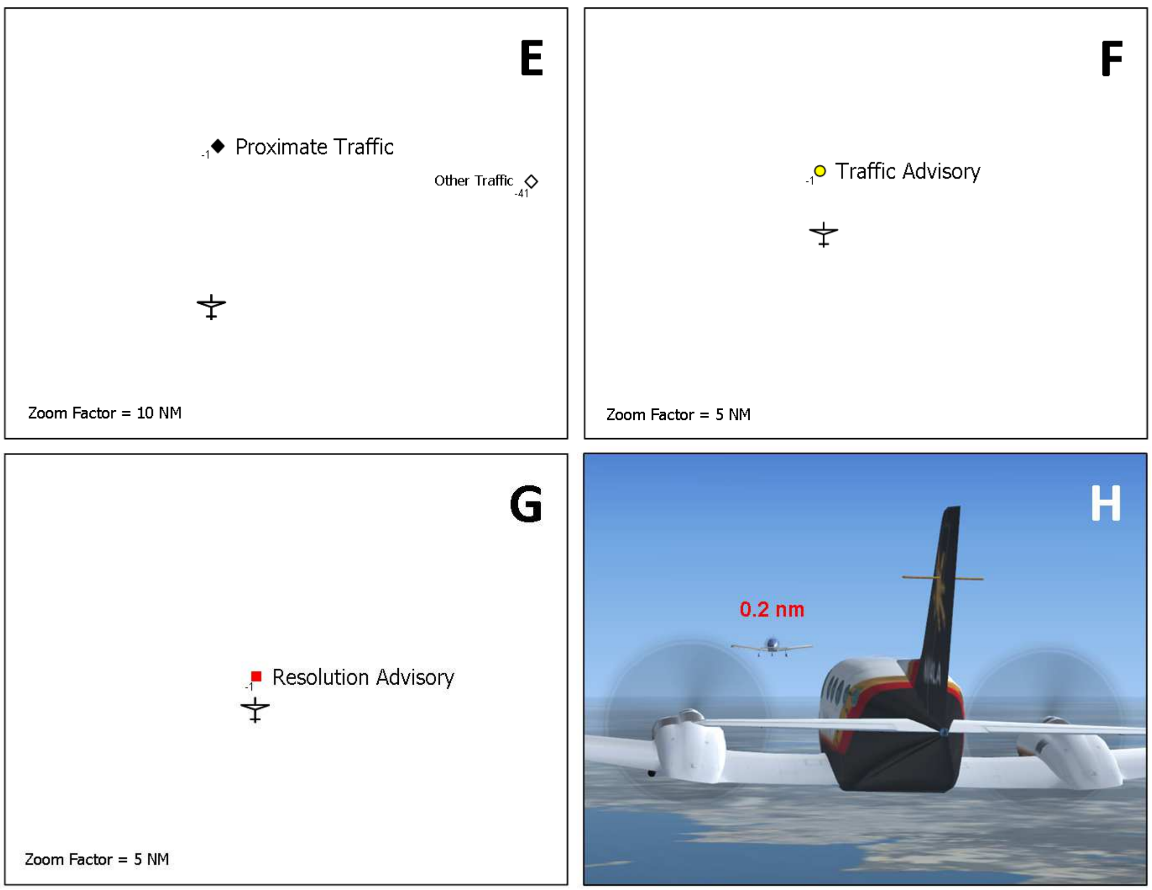

D
TCAS Overlay
N
MRB
TrackUp = 1
KMTN
KBWI
+74
+289
+259
KIAD
+77
+86
KDCA
KADW
+59
KHEF
KDAA
+54
+54
+35
Zoom Factor = 30 NM
LayerVehicles = 0

Figure **E** demonstrates the traffic symbol change as the intruder becomes Proximate Traffic (within 6 NM distance and +/- 1200 feet altitude).

Figure **F** shows the intruder aircraft when alarm status is Traffic Advisory.

Figure **G** is alarm status Resolution Advisory.

Figure **H** is a spot plane view taken at the same time.



Example TCAS XML gauge available from BlackBox website

A fully functional XML gauge is available for download from the BlackBox website. It demonstrates several concepts discussed in the guidebook including scale calibration, “click distance” application, TAWS and TCAS overlay.

References

1. *"Introduction to TCAS II Version 7.1"*, US Department of Transportation, Federal Aviation Administration, February 2011

http://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf

2. *"ACAS II Guide Airborne Collision Avoidance System II (incorporating version 7.1)"*, The European Organisation for the Safety of Air Navigation (EUROCONTROL), January 2012

http://www.eurocontrol.int/msa/gallery/content/public/documents/ACAS_guide71.pdf

3. *"Overview of ACAS II (incorporating version 7.1)"*, The European Organisation for the Safety of Air Navigation (EUROCONTROL), January 2012

http://www.eurocontrol.int/msa/gallery/content/public/documents/Training_ACAS_overview.pdf

4. Kochenderfer, M.J., Chrysanthacopoulos, J.P., Kaelbing, L.P. and Lozano-Perez, T., *"Model-Based Optimization of Airborne Collision Avoidance Logic"*, Lincoln Laboratory, Massachusetts Institute of technology, January 2010

http://www.ll.mit.edu/mission/aviation/publications/publication-files/atc-reports/Kochenderfer_2010_ATC-360_WW-18658.pdf

5. Bartels, Arne, *"XML Traffic Radar 2.0.1"*, July 2006

http://library.avsim.net/index.php?CatID=fs2004gau

LayerRacePoints

FSX Only

As of the release date of this Guidebook (February, 2013), I have not taken the time to study this layer.

A future update to the guidebook may contain a discussion of this layer.

CustomDraw: Rose

CustomDraw Rose is a separate class that renders a compass rose overlay for the CustomDraw Map. Its XML must be placed *below* the map code in order to display on top of the map.

The start tag is similar to fs9gps:map, as follows:

```
<Element Name="Compass Rose">
<Position X="150" Y="10"/>
<CustomDraw Name="fs9gps:rose" X="500" Y="400" Bright="Yes">
```

Position, X and Y are normally the same as used for fs9gps:map.

- ❑ **Heading (radians)**

Heading is the direction to which the top of the rose points. I prefer

```
<Heading>
   (L:TrackUp, bool) 0 ==
      if{ 0 }
      els{ (A:GPS GROUND MAGNETIC TRACK, radians) }
</Heading>
```

- ❑ **CenterX**
- ❑ **CenterY (gauge units)**

Center of the compass, normally the user's aircraft position.

- ❑ **Radius (gauge units)**

Radius of the compass measured in gauge units along the short axis.

- ❑ **Color (BGR hexadecimal)**

Line color of the rose, tick marks, and degrees labels.

- ❑ **BackgroundColor (BGR hexadecimal)**

Background color of the degrees markings.

❑ LineWidth (screen pixels)

Line width of the rose, in screen pixels.

❑ Font (string)

Font used for the degrees markings, for example, Arial.

❑ FontSize (enum)

Font size of the degrees markings.

❑ BigFontSize (enum)

Size of the "N", "S", "E", "W" labels in the LabelAllTicks=0 case.

❑ FullCircle (bool) FSX Only

FullCircle=1 for a complete circle rose. FullCircle=0 for a half rose.

❑ LabelAllTicks (bool) FSX Only

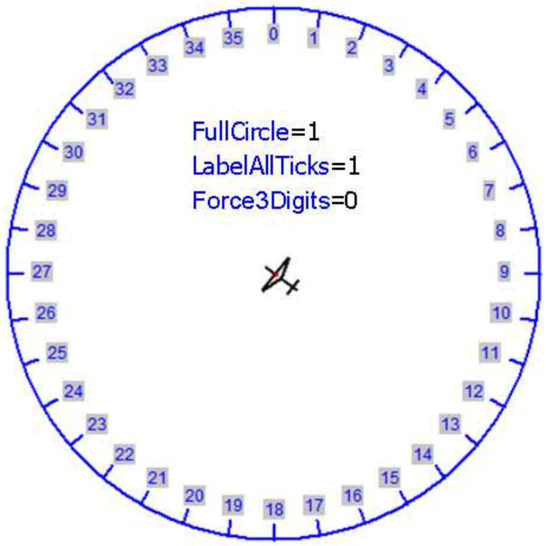
LabelAllTicks = 1: Ticks are drawn and annotated every 10 degrees.

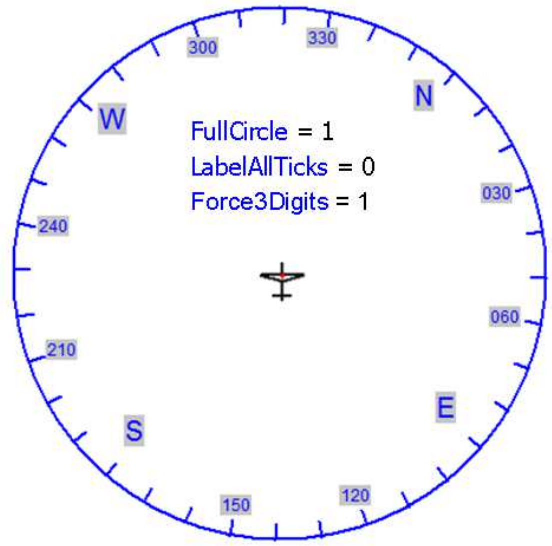
LabelAllTicks = 0: Ticks are drawn every 10 degrees but annotated every 30 degrees.

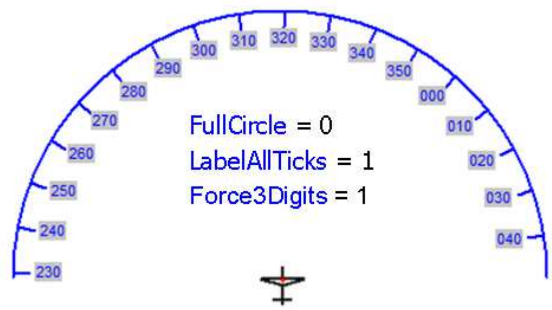
Additionally, the cardinal directions, "N", "S", "E", and "W", are displayed using BigFontSize.

❑ Force3Digits (bool) FSX Only

Three digits are used to annotate degrees. For example, 60 degrees is displayed as 060. See diagram below.


FullCircle=1
LabelAllTicks=1
Force3Digits=0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35


FullCircle = 1
LabelAllTicks = 0
Force3Digits = 1
N
030
060
E
120
150
S
210
240
W
300
330


FullCircle = 0
LabelAllTicks = 1
Force3Digits = 1
230 240 250 260 270 280 290 300 310 320 330 340 350 000 010 020 030 040

Example XML Map Gauges

Included in the BlackBox/CustomDraw Map website are two XML gauge examples available for download. They demonstrate several of the topics discussed in this guidebook, including:

- ITrafficInfo variables
- FSX map projection schemes
- Calibration of CustomDraw Map and XML overlay map scales
- Creation of XML map overlays and coordinate rotation transforms

Some interesting map applications can be written using XML overlays to the CustomDraw map base. The example gauges include:

- TCAS Map with functional Proximate, Traffic Advisory, and Resolution Advisory alarm status and appropriate TCAS symbols
- TAWS map display. As close as you can get with what FSX offers
- Click Distance, Bearing, Lat and Lon. Click anywhere on the map to return Distance and Bearing from users aircraft, and Latitude and Longitude of the click point. This opens the door to interesting applications such as "touch screen" MFD displays (well, the mouse is your finger)
- Nearest search centered on mouse click point rather than users aircraft. Click anywhere on the map to see details of the 10 nearest airports to the mouse click. A variation on this is to click on or simply near any airport shown on the map to see any or all details about that airport that are available from the gps database – you don't need to enter an Ident or ICAO to identify the airport or other facility you are interested in, just point to it by clicking on the map
- Click to add Waypoint to Flight Plan. Click anywhere on the map and add a new waypoint at that location
- Stationary Map rather than normal Moving Map. Click the **M_M** icon and the map stops moving but the airplane symbol starts moving – like the flight map that passengers can view on an airliner. Toggle **M_M** "On" and "Off" to see map reset feature

These are fully functional gauges written using FS9 XML syntax, but should be used in FSX as they demonstrate some features available only in FSX.

- ❑ **Gauge Setup**

The gauges are large, **520 x 700 gauge units**, and are intended to be set up as a separate window in your panel.cfg file, for example:

```
[Window19]  or whatever window number is appropriate in your panel
position=5
size_mm=520,700
visible=1

gauge00=FSMAP!ExampleMovingMap1, 0,    0,    520, 700
```

Use whatever path information is consistent with your panel. My installation has a folder named "FSMAP" in which I keep the XML gauge file. The FSMAP folder is located within the Panel folder of my aircraft.

XMLVars

The TCAS application uses variable arrays to store necessary information about intruder aircraft. An easy way to create such arrays is through the use of the XMLVars class in Tom Aguilo's XMLTools module that can be freely downloaded from:

http://fsdeveloper.com/forum/resources/xmltools-2-0-xml-expansion-module-for-fsx.148/

Follow the installation and operation instructions contained in the ReadMe file.

My example gauges will not function without XMLTools first being installed.

❑ Download Gauge Examples

The **ExampleMovingMap1.xml** gauge contains all of the applications listed above except Stationary Map.

The **ExampleStationaryMap1.xml** gauge adds the stationary map feature. Because so many reference points are changed when switching to a stationary map, I decided to save this as a separate file. It is easier to understand my approach to making an overlay by inspecting the script in the **ExampleMovingMap1.xml** gauge.

❑ Description of Features

- **Figure 1** identifies the click spots of the gauges

- **Figure 2** shows the sequence to retrieve Click Distance information and to compare that with the gps module GeoCalc distances. The GeoCalc reference is setup up to function only with airport facilities, not with VORs or NDBs, for example. Enter the three to four character airport Ident, not the full ICAO identifier

- **Figures 3 through 6** demonstrate the map calibration sequence. Note that map calibration must always be done at zooms under 500 km (i.e., zoom factors = ranges = of 269 NM or less). As well, TrackUp must be set to 0, and the simulation should not be in Pause mode. Calibration needs to be done each time the size of the map changes

  I recommend that the calibration sequence be repeated to double check consistency of M:X and M:Y returns. If the scales are different the second time, it is because the initial mouse clicks returned slightly different X and Ys than the second attempt. I have not figured out why, … yet.

- **Figures 7 and 8** show the process to add a new waypoint to a loaded flight plan by using a mouse click

- **Figures 9 through 11** describe the Nearest search from a click point. In this example, I use the click latitude and longitude to perform a nearest airport search relative to the click point.

  Note the code within the "NEAREST AIRPORT SEARCH TABLE". I use GeoCalcDistance and GeoCalcBearing to return the distance and bearing relative to the user aircraft rather than displaying the normal distance and bearing to the nearest search origin point which in this case is not the user aircraft.

  Additionally, I utilize XMLVars to store the gauge unit X and Y and Ident of the 10 nearest airports. The following code,

```
%(
(@c:NearestAirportSelectedLatitude, radians) (>L:OverlayObject_LAT, radians)
(@c:NearestAirportSelectedLongitude, radians) (>L:OverlayObject_LON, radians)
@GaugeXY
'ClikNrstY_' l31 scat @FindIndex (L:Gauge_Y, number) @WriteNumber
'ClikNrstX_' l31 scat @FindIndex (L:Gauge_X, number) @WriteNumber
'ClikNrstIdent_' l31 scat @FindIndex (@c:NearestAirportCurrentIdent) @WriteString
)
```

  does not display information as the rest of the <String> does, it is used within the <String> loop to assign values to ClikNrstY, X, and Ident arrays.

  The array capability of XMLVars is very useful indeed.

- **Figure 12** shows the TAWS map operation. When TAWS mode is active, the TAWS click button displays the radar altimeter which is useful reference to Q/C the TAWS display

- **Figures 13 and 14** show the TCAS operation.

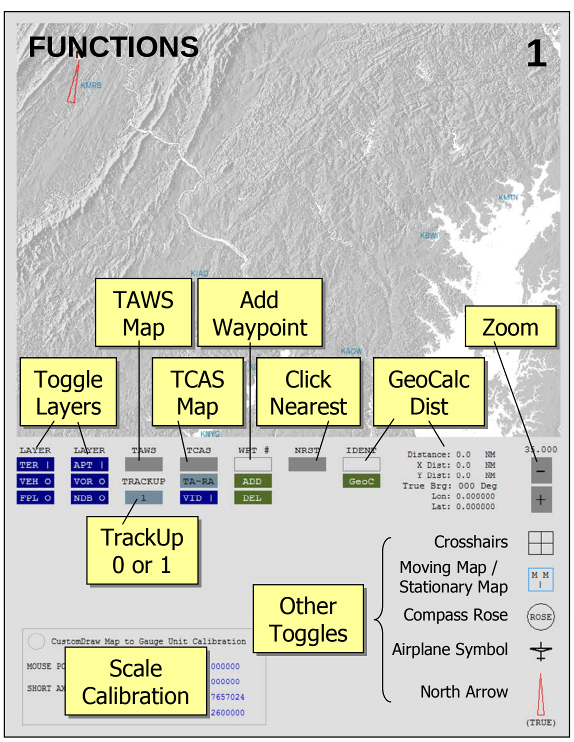
  When TCAS mode is active, the number of intruder aircraft within the search radius (30 NM search radius and 30 aircraft maximum, in my example) is displayed in the TCAS click button. The ITrafficInfo search will return the user's aircraft as Index 0 with a VID=1. You might want to add 1 to the MaxVehicles to account for this.

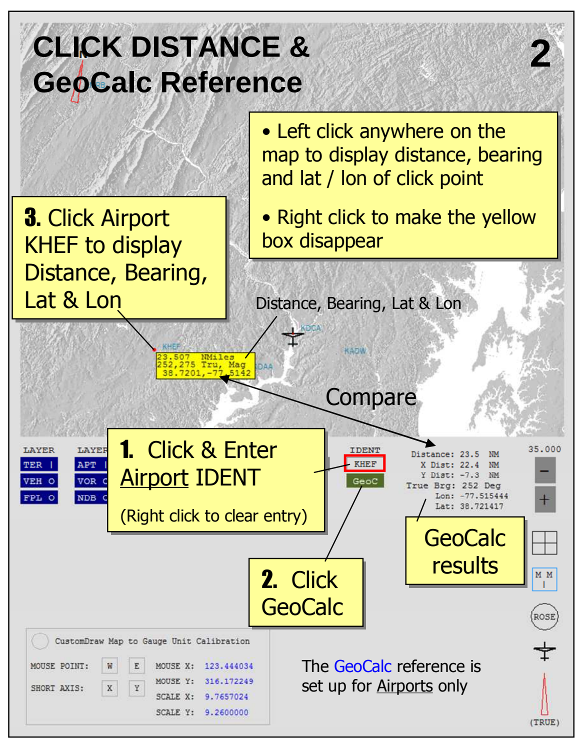
  The Vehicle ID is not a real-world TCAS display element but is included here for Q/C purposes. It can be toggled "On-Off" by clicking the VID button.

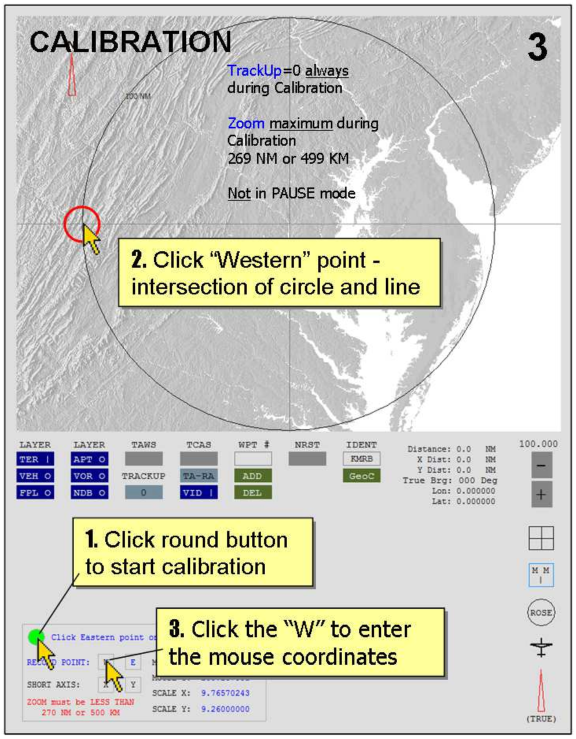
Additional features – the icons in the lower right toggle "On-Off" the following:

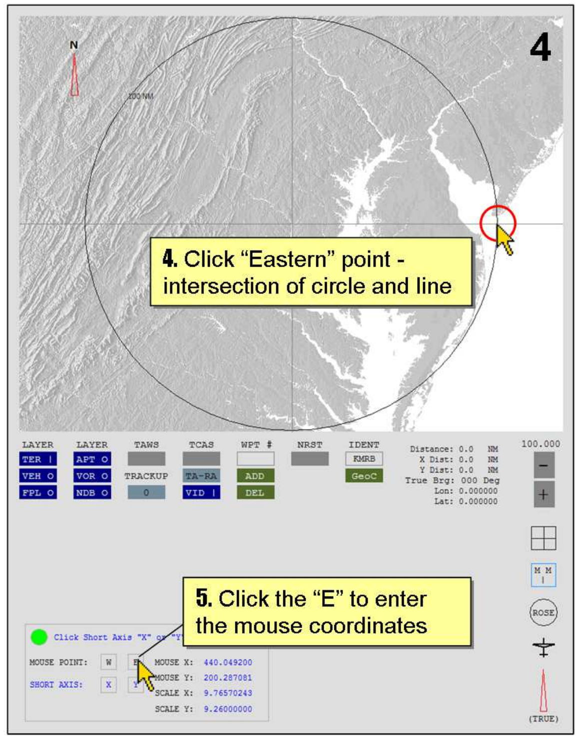
- North arrow
- User aircraft symbol
- Compass rose
- Moving Map – Stationary Map toggle
- Cross hairs

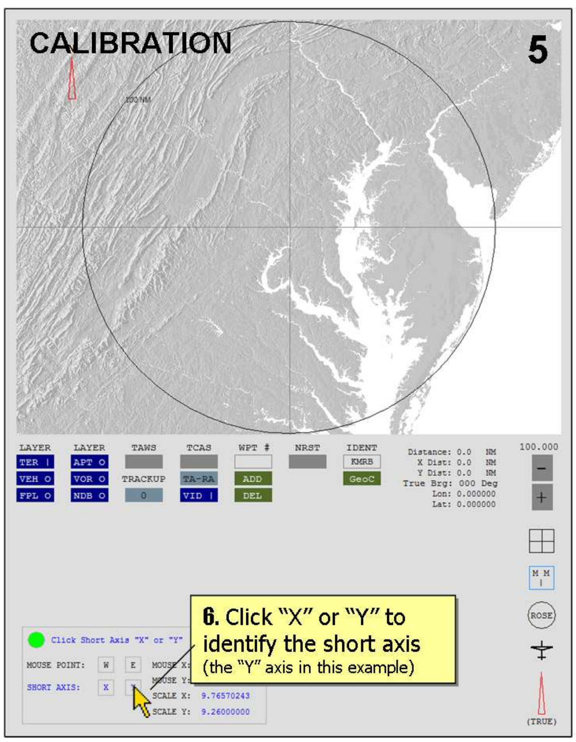
Zoom: Map zoom is achieved through use of the Zoom "**+**" and " **–** " toggle. The ZFactor, or Range, is displayed above the toggles. The height of the map is 2 times the ZFactor.

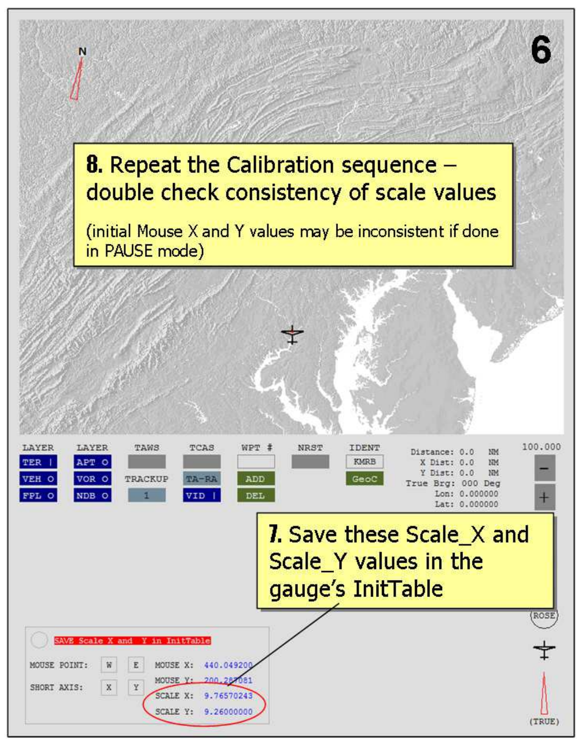


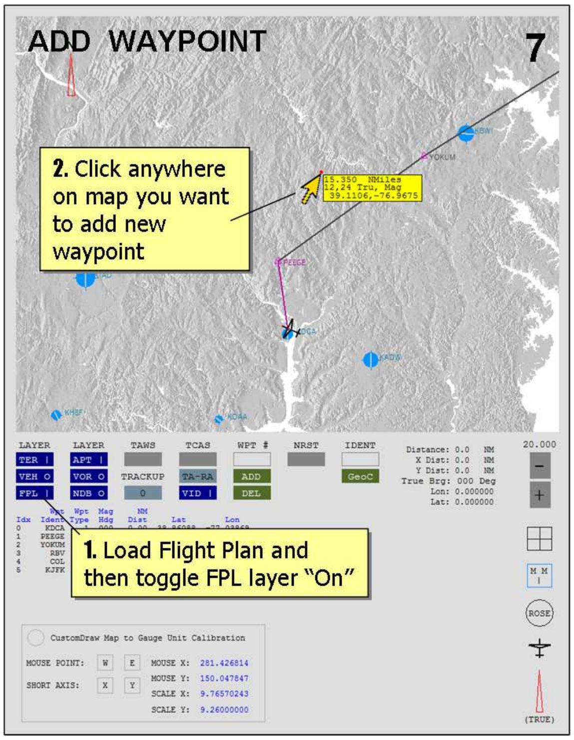


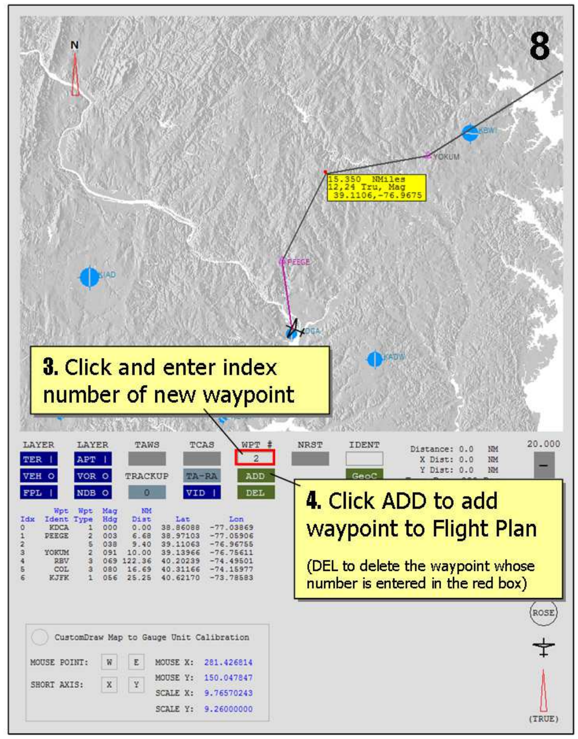

CALIBRATION
3
TrackUp=0 always
during Calibration
Zoom maximum during
Calibration
269 NM or 499 KM
Not in PAUSE mode
2. Click "Western" point -
intersection of circle and line
1. Click round button
to start calibration
3. Click the "W" to enter
the mouse coordinates
ZOOM must be LESS THAN
270 NM or 500 KM


4
4. Click "Eastern" point -
intersection of circle and line
5. Click the "E" to enter
the mouse coordinates

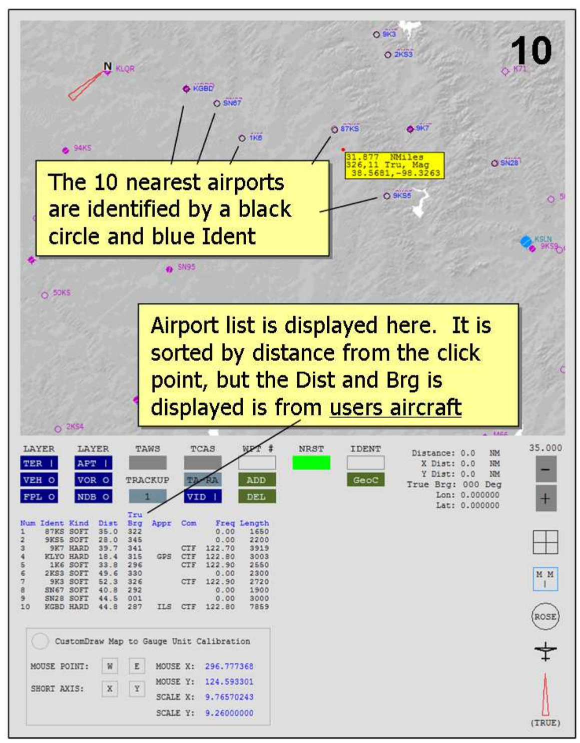

CALIBRATION
5
6. Click "X" or "Y" to
identify the short axis
(the "Y" axis in this example)

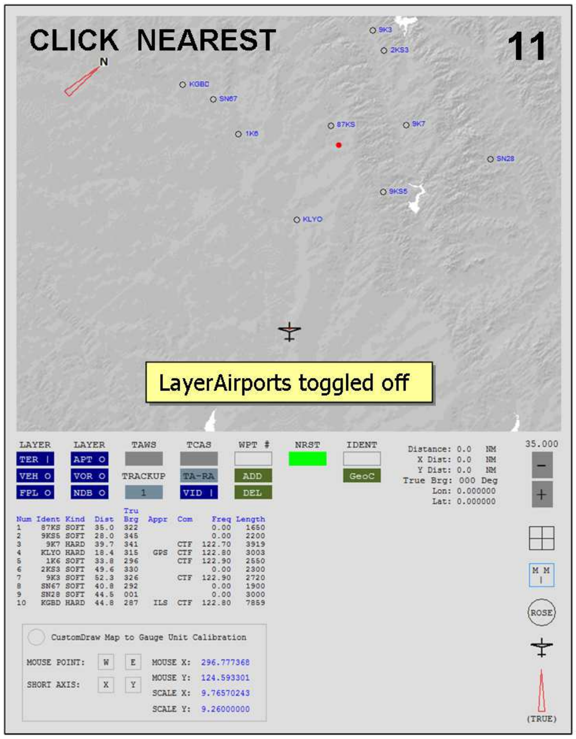

6
8. Repeat the Calibration sequence –
double check consistency of scale values
(initial Mouse X and Y values may be inconsistent if done
in PAUSE mode)
7. Save these Scale_X and
Scale_Y values in the
gauge's InitTable

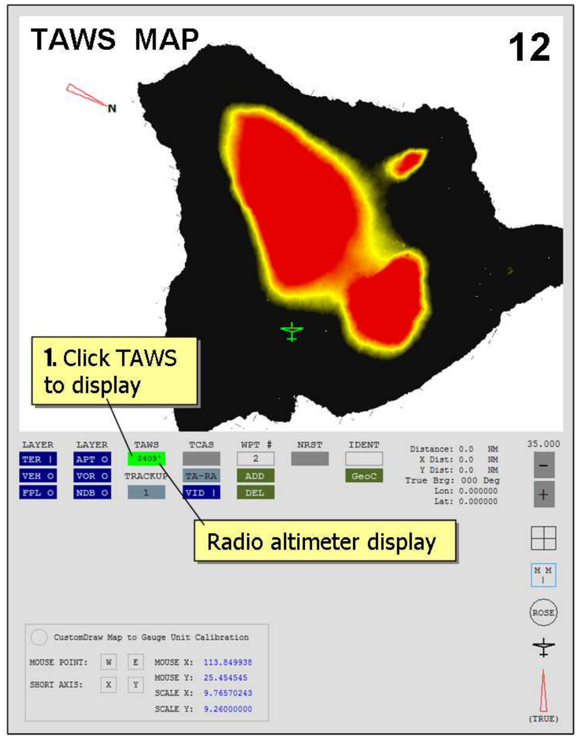

ADD WAYPOINT
7
2. Click anywhere on map you want to add new waypoint
1. Load Flight Plan and then toggle FPL layer "On"

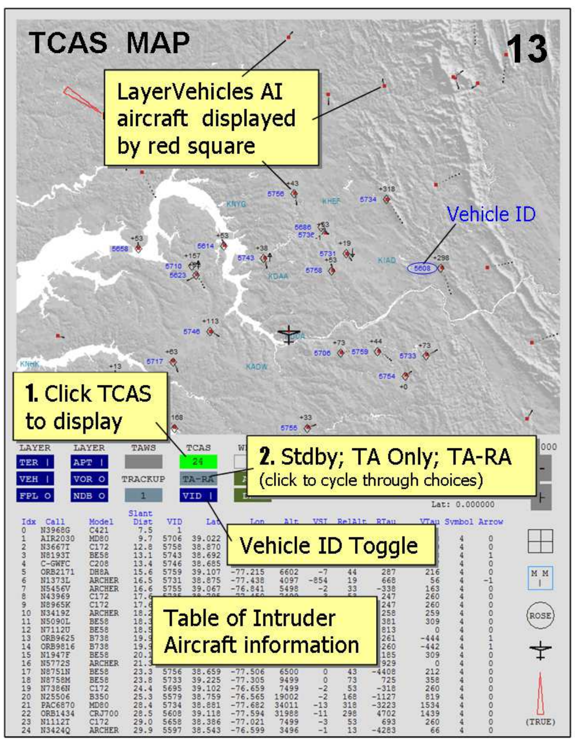

8
3. Click and enter index number of new waypoint
4. Click ADD to add waypoint to Flight Plan
(DEL to delete the waypoint whose number is entered in the red box)

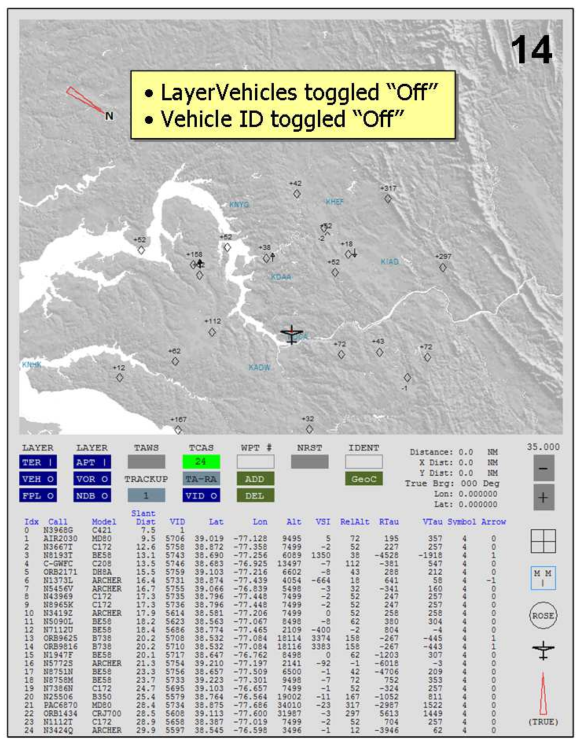

CLICK NEAREST
9
2. Click anywhere on the map to find the 10 nearest airports to the click point
1. Click NRST


10
The 10 nearest airports are identified by a black circle and blue Ident
Airport list is displayed here. It is sorted by distance from the click point, but the Dist and Brg is displayed is from users aircraft


CLICK NEAREST
11
LayerAirports toggled off


TAWS MAP
12
1. Click TAWS to display
Radio altimeter display


TCAS MAP
13
LayerVehicles AI aircraft displayed by red square
Vehicle ID
1. Click TCAS to display
2. Stdby; TA Only; TA-RA
(click to cycle through choices)
Vehicle ID Toggle
Table of Intruder Aircraft information

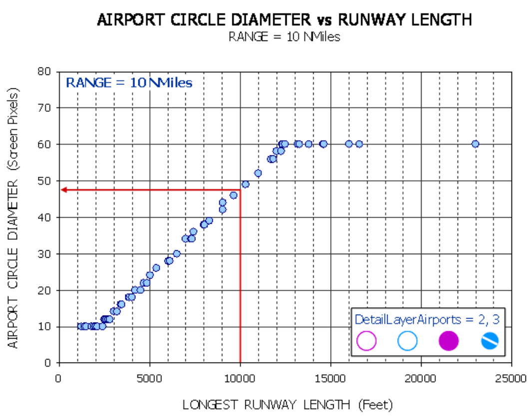

14
• LayerVehicles toggled "Off"
• Vehicle ID toggled "Off"

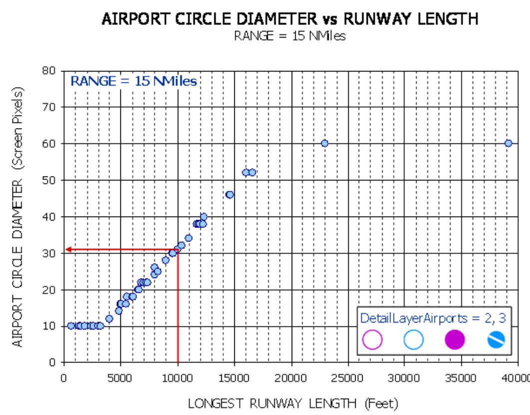
LayerAirports

Additional Information

Additional detail for LayerAirports relating to airport symbol and text generated by Flight Simulator:

Airport Symbol Size – A Function of Runway Length and Zoom





The size (diameter) of the symbol is proportional to length of the longest runway and the zoom setting. The relationship for index 2 and 3 symbols for Range = 10 and 15 NMiles is shown in the graphs above. The minimum size rendered is always 10 screen pixels, and the maximum size, regardless of runway length or zoom, is 60 screen pixels. Airport symbols become smaller as the map is zoomed out. Note that a 10000 ft. runway has a 48 pixel symbol at Range = 10 NMiles, but a 31 pixel symbol at Range = 15 NMiles. Index 4 and 5 have different size relationships but are similarly rendered proportionate to runway length and zoom. Index 1 (dot, which is always 1 screen pixel), Heliports, and Seaplane Base Index 2, 3, and 4 are not drawn according to runway length.

TextDetailLayerAirports – A Function of Zoom

DEFAULT TEXT DISPLAY ZOOM RANGES

FSX: 1600 x 1200

	Zoom range (m)			**ZoomFactor range (NM)**		
Runway Numbers	80	*to*	4,447	0.043	*to*	2.401
Frequencies	80	*to*	10,970	0.043	*to*	5.923
Elevation & Length	80	*to*	14,825	0.043	*to*	8.005
Name	80	*to*	22,237	0.043	*to*	12.007
Ident	80	*to*	148,250	0.043	*to*	80.049
Nothing	148,251	*to*	5,000,000	80.049	*to*	2699.784

FSX: 1600 x 900

	Zoom range (m)			**ZoomFactor range (NM)**		
Runway Numbers	80	*to*	3,316	0.043	*to*	1.790
Frequencies	80	*to*	8,177	0.043	*to*	4.415
Elevation & Length	80	*to*	11,050	0.043	*to*	5.967
Name	80	*to*	16,575	0.043	*to*	8.950
Ident	80	*to*	110,500	0.043	*to*	59.665
Nothing	110,501	*to*	5,000,000	59.666	*to*	2699.784

FSX: Permissible Zoom range for fs9gps:Map is 80 to 5,000,000 meters

FS9: 1600 x 1200

	Zoom range (m)			**ZoomFactor range (NM)**		
Runway Numbers	100	*to*	8,745	0.054	*to*	4.722
Frequencies	100	*to*	21,862	0.054	*to*	11.805
Elevation & Length	100	*to*	43,725	0.054	*to*	23.610
Name	100	*to*	145,750	0.054	*to*	78.699
Ident	100	*to*	291,500	0.054	*to*	157.397
Nothing	291,501	*to*	5,000,000	157.398	*to*	2699.784

FS9: Permissible Zoom range for fs9gps:Map is 100 to 5,000,000 meters

Airport Symbol Type Overrides Text Index Selection

FSX: Despite user selection of a TextDetailLayerAirports Index, the text actually displayed will be limited by the choice of airport symbol – the lower the DetailLayerAirports Index, the less label information that is displayed as summarized in the tables below. It is a little complicated, but it's all part of the default de-cluttering scheme.

TextDetailLayerAirports Index = 1

Airport Symbol	Ident	Name	Elevation & Rwy Length	Control and Advisory Freq	Runway Numbers
Dot - **1**	✓				
Circle - **2**	✓				
Circle runways - **3**	✓				
Block runways - **4**	✓				
Runways - **5**	✓				

TextDetailLayerAirports Index = 2

Airport Symbol	Ident	Name	Elevation & Rwy Length	Control and Advisory Freq	Runway Numbers
Dot - **1**	✓				
Circle - **2**	✓				
Circle runways - **3**	✓				
Block runways - **4**	✓	✓			
Runways - **5**	✓	✓			

TextDetailLayerAirports Index = 3

Airport Symbol	Ident	Name	Elevation & Rwy Length	Control and Advisory Freq	Runway Numbers
Dot - **1**	✓		✓		
Circle - **2**	✓		✓		
Circle runways - **3**	✓		✓		
Block runways - **4**	✓	✓	✓		
Runways - **5**	✓	✓	✓		

TextDetailLayerAirports Index = 4

Airport Symbol	Ident	Name	Elevation & Rwy Length	Control and Advisory Freq	Runway Numbers
Dot - **1**	✓		✓		
Circle - **2**	✓		✓		
Circle runways - **3**	✓		✓		
Block runways - **4**	✓	✓	✓	✓	
Runways - **5**	✓	✓	✓	✓	

TextDetailLayerAirports Index = 5

Airport Symbol	Ident	Name	Elevation & Rwy Length	Control and Advisory Freq	Runway Numbers
Dot - **1**	✓		✓		
Circle - **2**	✓		✓		
Circle runways - **3**	✓		✓		
Block runways - **4**	✓	✓	✓	✓	
Runways - **5**	✓	✓	✓	✓	✓

Example 1: Even if TextDetailLayer index is **5**, only Ident and Elevation & Rwy Length will be displayed if the airport symbol index is 1, 2, or 3.

Example 2: Runway Numbers are displayed only when airport symbol and text index are both 5.

Font Type, Font Size and Label Offset

FSX: fs9gps:Map uses an Arial font, rasterized without anti-aliasing.

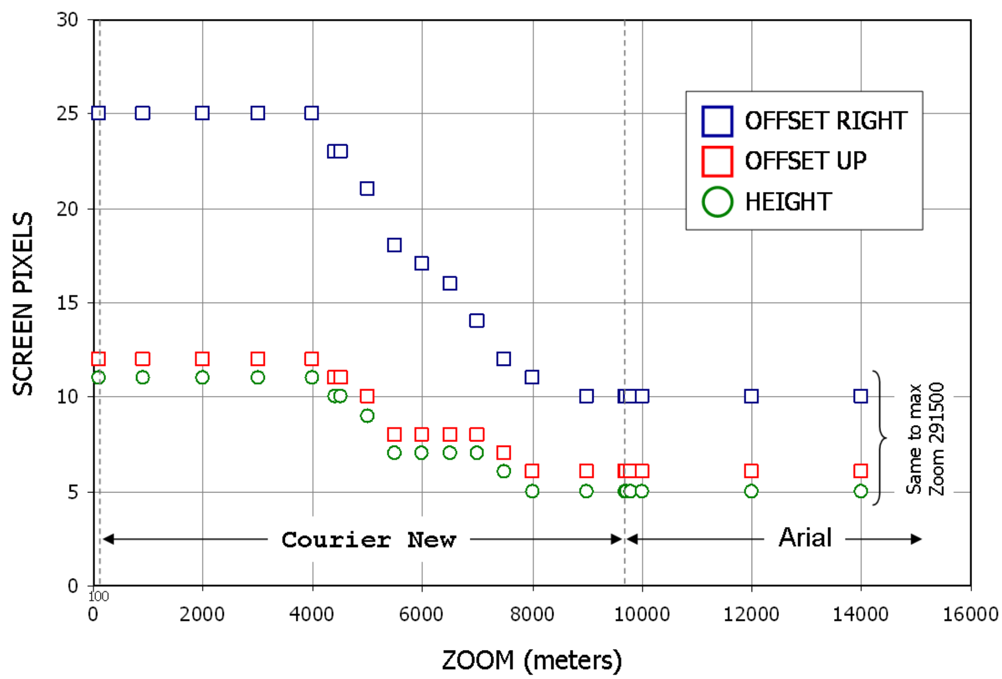
At zoom levels of 7412 meters and below (a very zoomed-in view), font height is 8 screen pixels, and placement of the text (offset of the upper left corner of the Ident or Name) is 16 screen pixels right and 9 screen pixels up from the airport location. Any additional lines of text are listed below this.

At zoom levels of 7413 meters and greater (zoomed-out view), font height is 7 screen pixels, and placement of the text is 13 screen pixels right and 8 screen pixels up from the airport location. Any additional lines of text are listed below this.

These are the only two size and offset variations. They are automatic and cannot be changed.

FS9: fs9gps:Map uses Courier New font, rasterized without anti-aliasing. It demotes to Arial font as Zoom increases. The font height and offsets are shown in the chart below. They are automatic and cannot be changed.

FS9: FONT TYPE, FONT SIZE and OFFSET



De-cluttering

Map symbols need to be reduced in size or removed from display as Zoom increases (as you zoom out). This is known as de-cluttering. The stock gps_500 gauge decluttering settings are shown below. Additionally, CustomDraw has default de-cluttering settings as described throughout the guidebook.

Stock gps_500 De-cluttering scheme

	Case Step:	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
										Nautical Miles Zoom, Range (NM)															
	Range, ZoomFactor (NM):	**2000**	**1500**	**1000**	**500**	**350**	**200**	**150**	**100**	**50**	**35**	**20**	**15**	**10**	**5.0**	**3.5**	**2.0**	**1.5**	**1.0**	**0.576**	**0.329**	**0.247**	**0.165**	**0.082**	**0.000**
ObjectDetailLayerAirports	Hex	0	5	5	5	5	15	15	15	15	15	15	15	1F	1F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F
ObjectDetailLayerAirports	Decimal	0	5	5	5	5	21	21	21	21	21	21	21	31	31	95	95	95	95	95	95	95	95	95	95
TextDetailLayerAirports	Integer	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TextDetailLayerVORs	Integer	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TextDetailLayerILSs	Integer	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TextDetailLayerNDBs	Integer	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TextDetailLayerIntersections	Integer	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

			Private	**Heliport**	**Water**	**Soft**	**Hard**	**Non-Twr**	**Tower**
ObjectDetailLayerAirports	Hex	**0**	0	0	0	0	0	0	0
ObjectDetailLayerAirports	Hex	**5**	0	0	0	0	1	0	1
ObjectDetailLayerAirports	Hex	**15**	0	0	0	1	1	1	1
ObjectDetailLayerAirports	Hex	**1F**	0	0	1	1	1	1	1
ObjectDetailLayerAirports	Hex	**5F**	1	0	1	1	1	1	1

fs9gps:Map Guidebook Updates

v.2.0

Page	**Edit**
19	Added bullet point on Map Object Color syntax
26	Added section on Number Formats
36	Added example of Terrain Shadow = 1 effect on color schemes
53	Revised VOR graphic
83	Corrected LayerApproachLeg description error
86	Added note that TrackUp is True North
94	Corrected definition of ITrafficInfo:Filter Ground_Vehicles

v.2.0.1

Page	**Edit**
19	Corrected text color **RGB** syntax statement