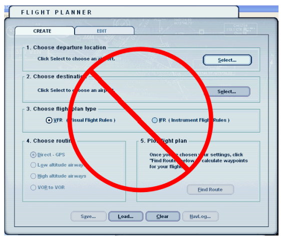
Using plotting software with Microsoft Flight Simulator 2004

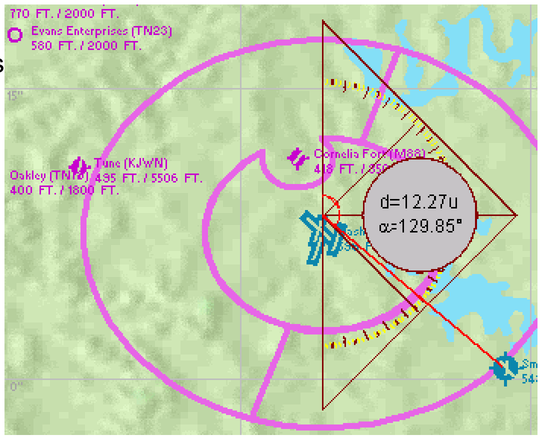
Acknowledgment

My thanks and recognition is extended to Markus Bader for developing the software that is demonstrated in the following pages and for allowing it to be distributed freely with this tutorial. Found within FSPlotTu.zip, the tutorial archive, is mbruler30.zip. That file contains the software, MBRuler, and associated instructions. Mr. Bader's website address is:
http://www.Markus-Bader.de/MB-Ruler

Overview

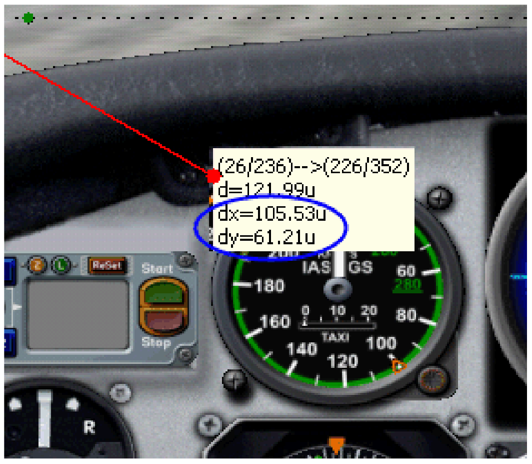


What is plotting software and what will it do within the MSFS environment? MBRuler is a fully-functioning, stand-alone, freeware application. Among its diverse talents is the ability to determine headings, bearings, and distances directly from the charts that reside within Microsoft Flight Simulator. It does this while FS2004 is running. This allows for accurate flight planning without getting involved with the default Flight Planner, at left, found in FS9. Flight Planner is limited to known waypoints because it must have latitude/longitude coordinates for its calculations. On the other hand, with MBRuler you can navigate to any point because it uses a different way to determine headings and distances.



At right, a typical chart from MSFS shows one example of how MBRuler can determine the heading and distance between two points. The brown triangle is one interface to MBRuler. It has another. The red line extending between the large airport to the smaller airport is the course line drawn by MBRuler. Inside the grey circle, the data shows this course has a distance of about 12-nautical miles and a heading of about 130-degrees. Bearings for a WP, VOR or NDB would be determined in a similar fashion.

The triangular interface has to ability to rotate to any value desired. Doing so would have the effect of canceling out magnetic variation when angles are measured. This function allows both true bearings and magnetic bearings to be read directly from charts.



```
(26/236)-->(226/352)
d=121.99u
dx=105.53u
dy=61.21u
```

MBRuler also has the capability to determine the X-Y pixel coordinates needed for the panel.cfg file when new gauges are being installed on panels. It can do this because the application can measure distances directly in pixels. Measuring coordinates can speed up the installation procedure by eliminating the trial and error technique often used to determine those coordinates.

At left, portions of a second interface are shown. The red dot is the end of a line extending from this interface. The coordinate value of the dot shows an X-value of 105-pixels and a Y-value of 61-pixels. From the panel.cfg below, you can see those are the correct coordinates.

```
gauge33=KingAir!Altimeter_Selector, 553, 43
//gauge34=KingAir!Airspeed, 105, 61
gauge34=TriSpeedGau220!TriSpeedGau220, 105, 61
gauge35=KingAir!Autopilot, 552, 106
```

The remainder of this tutorial will be devoted to using MBRuler with MSFS. I will not detail many of the graphic functions that MBRuler can perform outside of Flight Simulator. Some of those will be briefly mentioned but particulars will be left for the reader to discover.

Setting up

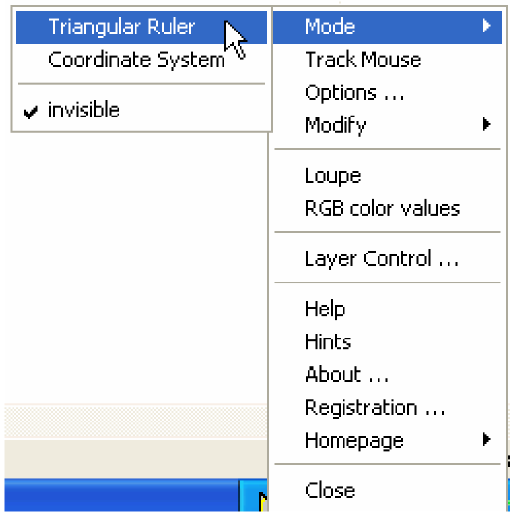
The tutorial file, FSPlotTu.zip, should be extracted into a working folder of your choice. Inside that archive is the ruler software, mbruler30.zip. That also should be extracted to a suitable folder. The ReadMe.txt inside mbruler30.zip instructs that the program can be installed on your computer in the usual manner by running the Setup.exe file. However, it also states that the application can be run directly by clicking on the MB-Ruler.exe file. The latter is my preferred method. It is handy to make a shortcut of the MB-Ruler.exe file and place it on your desktop. When launched, the program will automatically minimize itself to the taskbar so that it is available for use during a MSFS session. It can then be activated when desired using one of several methods discussed below.

Features of MBRuler

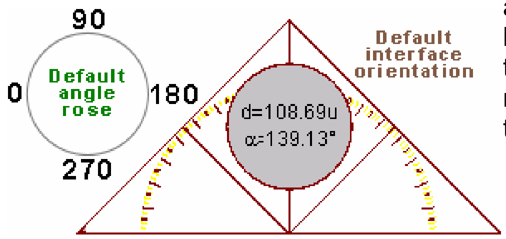
Note: To use all important features of MBRuler during a simulator session requires that MSFS be temporarily switched from full-screen mode to window mode. After gathering navigational or other data with MBRuler, FS can then be returned to full-screen. Not following this procedure will cause FS to be minimized (not a crash) to the taskbar when you try to use MBRuler.

MBRuler was not designed specifically for users of Microsoft FS. As such, some of its features are not applicable to our hobby. I will begin this section by describing how various parts of this software function and how options should be set so that MBRuler gives data suitable for FS2004.

As mentioned, when the application is launched from the desktop, it automatically minimizes to the taskbar. From there it can be activated in several ways; **(1)** hotkey **Ctrl+Alt+M**, **(2)** right-clicking on the taskbar icon (shown at right), **(3)** left-clicking on the icon.

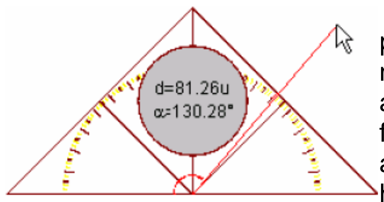


Right-clicking the icon will open the control box shown at right. Choosing **Mode/Triangular Ruler** will toggle the "invisible" status of the program to off and open the triangular-shaped interface shown below. A second type of interface can also be opened from this box. It is called **Coordinate System**. It will be used later in this tutorial to determine pixel coordinates for gauge installations. The box at right can also be opened from the interface by right clicking on the grey circle.

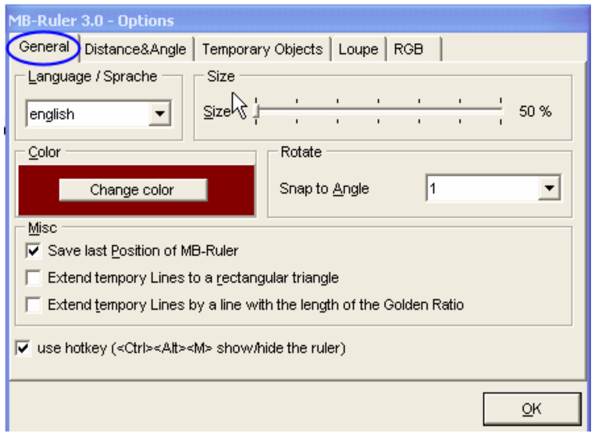


Consecutive left-clicks on the taskbar icon will toggle the program through the two interfaces, and then back to the "invisible" state. When invisible, it is not longer seen on the screen because it has been minimize to the taskbar. When visible, it is persistent and will stay above other windows.

The triangle above shows the default orientation of the interface. That graphic also illustrates how default angles are measured. This orientation is obviously unsatisfactory for Flight Simulator because zero-degree is at the West position. Fortunately, this is easily changed so that angles can be measured according to the compass rose. We will perform this task shortly. For now, let's continue exploring the features inside the above control box.



The next item in that box is **Track Mouse**. Activating this feature produces a line that is continuously drawn between the interface and the mouse cursor. That effect is shown at left. As the mouse cursor moves about, the red line sticks to it like gum to a shoe. It is this line that is used for the course between two points. MBRuler will calculate the same angle/distance values regardless of whether the line is present or not. But having it temporarily visible often supplies good visual feedback as course lines are plotted and distances and angles are measured.


MB-Ruler 3.0 - Options

General | Distance&Angle | Temporary Objects | Loupe | RGB

Language / Sprache: english

Size: Size 50 %

Color: Change color

Rotate: Snap to Angle 1

Misc

☑ Save last Position of MB-Ruler

☐ Extend tempory Lines to a rectangular triangle

☐ Extend tempory Lines by a line with the length of the Golden Ratio

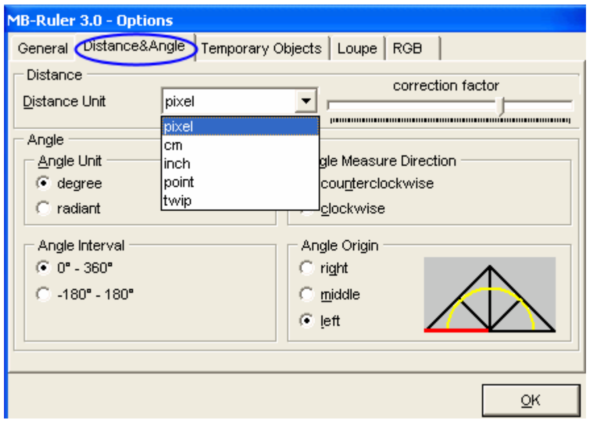
☑ use hotkey (<Ctrl><Alt><M> show/hide the ruler)

OK

Next, **Options** appears in the list inside the control box. This feature opens the box at left which is loaded with choices. Many of them are cosmetic such as changing the color of the interface inside the General tab. As seen, a brown color is used in this example. Green is the default color. Most of the choices under this tab are self-explanatory. The "Snap to Angle" works with the interface if you rotate it. With a value of 1, the interface will snap to every degree around the compass rose. We will put rotation to work when magnetic variation is discussed.

Under the Distance & Angle tab, more choices can be seen. In the various sections the radio buttons should be set as shown. Not seen clearly, the Angle Measure Direction is set to clockwise. This causes angle values to increase clockwise, just like the compass rose.

The Distance Unit drop-down box is shown opened. All the distance units are present except one that is called <user>. This means you, the user, can define units to meet a particular need. This function allows flight simmers to calibrate the software using the navigational charts inside MSFS so that distance can be read in nautical miles. This will be detailed later.


MB-Ruler 3.0 - Options

General | Distance&Angle | Temporary Objects | Loupe | RGB

Distance

Distance Unit: pixel (pixel, cm, inch, point, twip)

correction factor

Angle

Angle Unit: ● degree ○ radiant

gle Measure Direction: ○ counterclockwise ○ clockwise

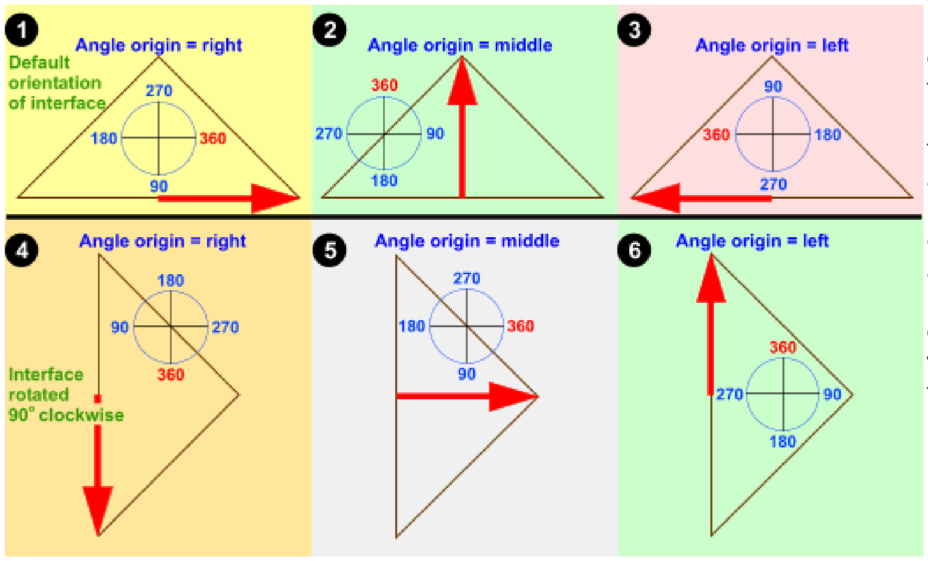
Angle Interval: ● 0° - 360° ○ -180° - 180°

Angle Origin: ○ right ○ middle ● left

OK

The Angle Origin section can be a bit confusing until you realize that the three origin choices (right, middle, left) identify a particular corner of the triangular interface. The corner containing the angle origin will always point in the direction of 360-degrees. Those corners retain that identity even if the interface is rotated.

Below, blocks 1, 2, and 3 identify the Angle Origin in each or the three corners of the interface. In each block, you can see the angle rose has its origin aligned toward that corner. Blocks 4, 5, and 6 confirm that the triangle still retains those corner designations even though the interface has been rotated 90-degrees. As the triangle was rotated, the angle rose rotated also.

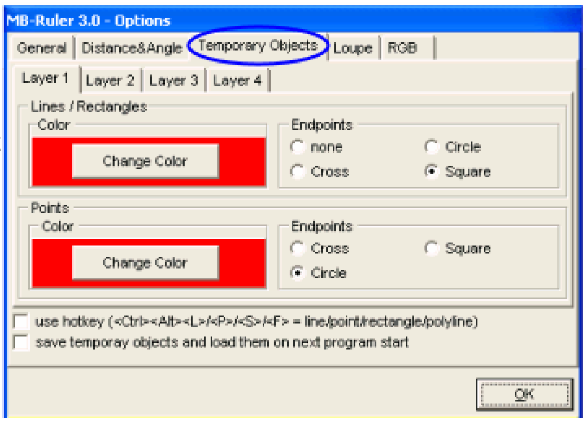


Only the blocks colored in green, 2 and 6, are suitable for MSFS because the angle rose is oriented to a compass rose with North (360-degrees) at the top.

The choices inside the Temporary Objects tab seen at right have no great impact when using MBRuler with FS2004. There, the user is given the ability to change colors of objects (points, lines, and rectangles). Sometimes this is helpful to produce colors that contrast against the graphics used by Flight Simulator.

Temporary Objects are often constructed and used for various tasks such as the calibration of distances. Techniques that use these object and are applicable to Flight Simulator will be covered later.

The final two tabs, Loupe and RGB, contain the expected. You can discover these features for yourself. They are not particularly helpful for this tutorial but can certainly find use outside MSFS.


MB-Ruler 3.0 - Options

General | Distance&Angle | Temporary Objects | Loupe | RGB

Layer 1 | Layer 2 | Layer 3 | Layer 4

Lines / Rectangles
- Color: 🟥 Change Color
- Endpoints: none, Circle, Cross, [✓ Square]

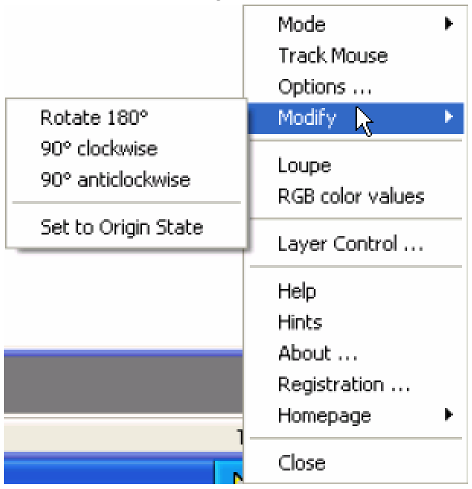
Points
- Color: 🟥 Change Color
- Endpoints: Cross, Square, [✓ Circle]

use hotkey (<Ctrl><Alt><L>/<P>/<S>/<F> = line/point/rectangle/polyline)

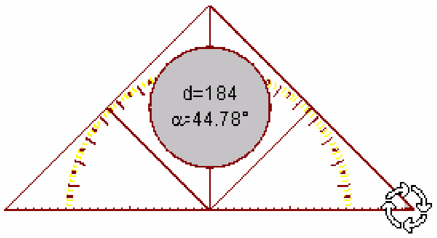
save temporay objects and load them on next program start

OK

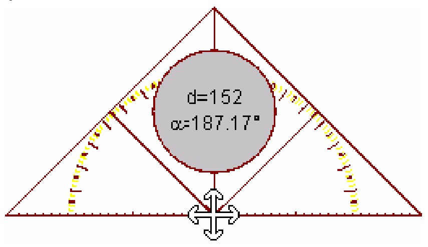
Continuing with the control box at left, **Modify** contains rotation controls. They provide a way to quickly change the orientation of the rectangular interface in either 90 or 180-degree steps. Or, the interface can be set back to its Origin State (the default orientation).

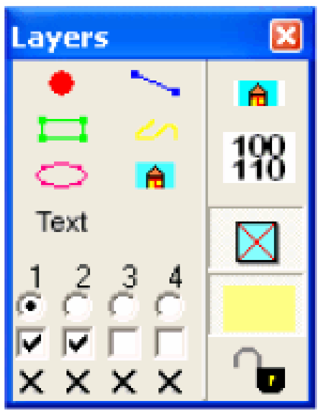


But this control is not the only way to rotate the interface. The mouse can also be used for the same purpose. At right, the mouse cursor has changed into rotational arrows as it passed over the lower, right corner of the triangle. The left corner works the same way. Using either position, the triangle can be turned to any angle required.



Speaking of the mouse; the cursor will change into directional arrows when placed at the base of the triangle as seen at right. From that position the interface can be moved about on the FS screen.



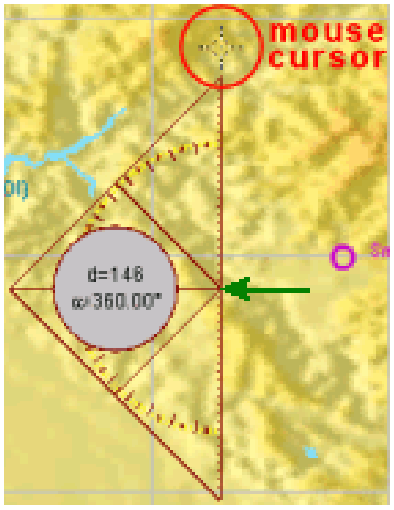


The **Layer Control** is the next stopping point on the control box. Clicking that control will open the small Layers box shown at left. Inside that box are the Temporary Objects discussed above. The solid, red circle is a point. These can be used to plot way points on FS charts. The blue line is used to draw course and bearing lines. This is also the object that will be used to calibrate distances in MBRuler. The green rectangle produces four-sided polygons of any desired size. One task for this temporary object is to outline a square bitmap that contains a round gauge. That is helpful when determining the coordinates of the upper, left corner of these graphics because that point defines the location of the entire gauge.

Many of the other features of the Layers box will be of no concern in this tutorial. However, give them a try. You may discover uses as yet unknown.

I will end this section with the last item on the control box, **Close**. That, of course, is a fast way to terminate the MBRuler session.

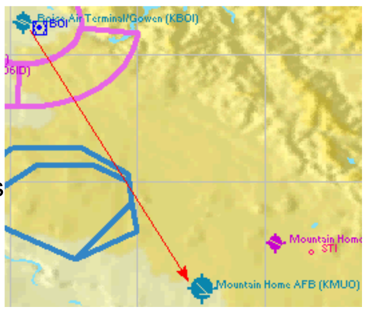
A word from pilots Tail Wheel and Sweet Pea
DANGER
Pull Up
Holy Mackerel
Glenn Copeland 2005
When your admiring cabin guests request photographs, smile like it's your last one.



Headings and bearings

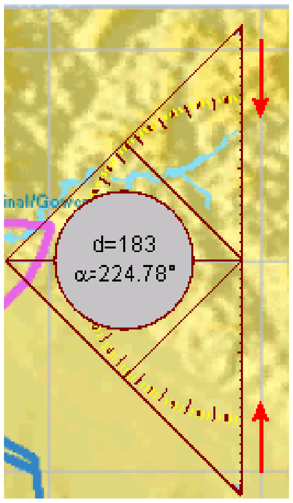
The first step in determining headings and bearings on the FS charts is to set up the triangular interface so that it reads angles with 0/360-degrees in the North position. All information necessary to do this has been covered on previous pages. If needed, refer back to the diagram at the bottom of page 3 to find two suitable examples.

At left, one correct orientation is shown. If you look closely inside the red circle you can see the mouse cursor. The grey data circle confirms that the cursor sits at the 360-degree mark. Keep in mind that angles are measured and course lines are drawn from the base of the triangular interface. That point is identified by the green arrow. That location remains constant even if the triangle is rotated.

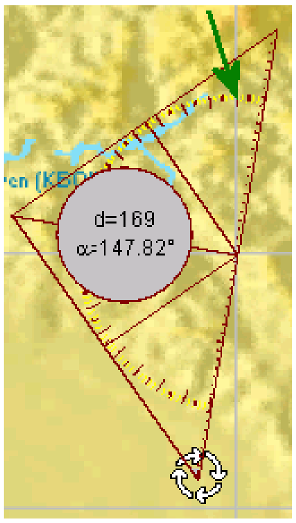


At right, the first task using MBRuler will be to determine the heading between two airports. The red arrow defines the track that will be flown. What is the compass heading for that course? Compass headings require that the magnetic variation for that area of the world be known. For this example let's make it an even 10-degrees east. East is least so the compass heading will be 10-degrees less than the true heading. Let's set up the interface so it will measure the compass heading by automatically taking into account the variation.

Adjusting for magnetic variation

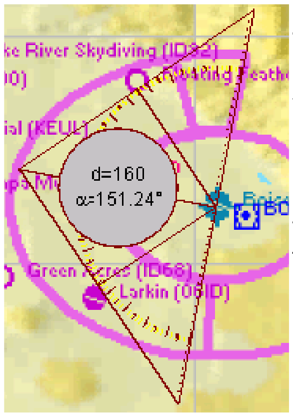


In the diagram at left, the interface has been positioned next to a longitude line (between the red arrows). The baseline of the triangle and the longitude line are parallel. This means the compass rose is aligned to true North. Any angles measured in this position would result in a true heading. To compensate for the 10-degrees East variation, the interface must be rotated 10-degrees. But, in which direction? East variation requires a clockwise rotation; west variation, a counterclockwise rotation.

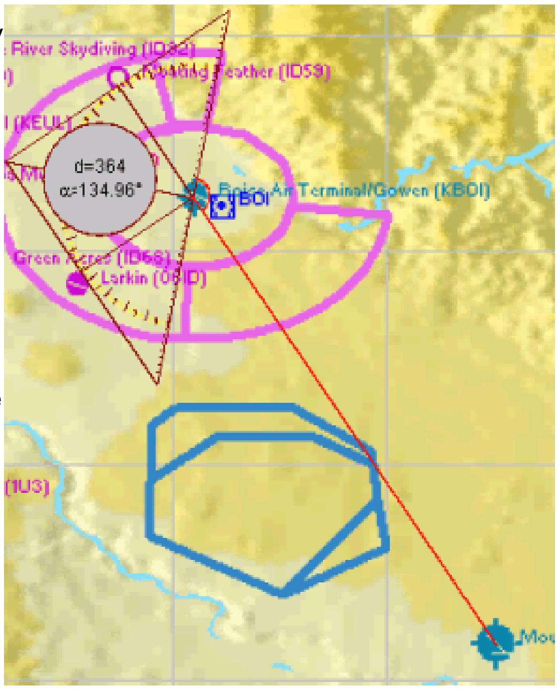


A fast and easy way to rotate the interface accurately is shown at right. There, the base of the triangle is positioned at a latitude/longitude intersection. The lower corner of the triangle is then rotated with the mouse until the 10-degree mark in the visible protractor is centered over the longitude line (green arrow). That action results in a 10-degree clockwise rotation and will effectively cancel out the East variation. Compass heading can now be measured directly by MBRuler.

The interface is then moved until the base of the triangle is positioned over the departure airport. You can see that setup at left. The numbers seen inside the grey data circle are meaningless at this point. They are simply reflecting values gathered as the software tracks the cursor position as this illustration was prepared.



For the heading demo, I will activate **Track Mouse** by right-clicking inside the grey data circle and choosing that option from the control box. I have the software set so that this function draws a red line but any color that gives a contrast to the chart will work.



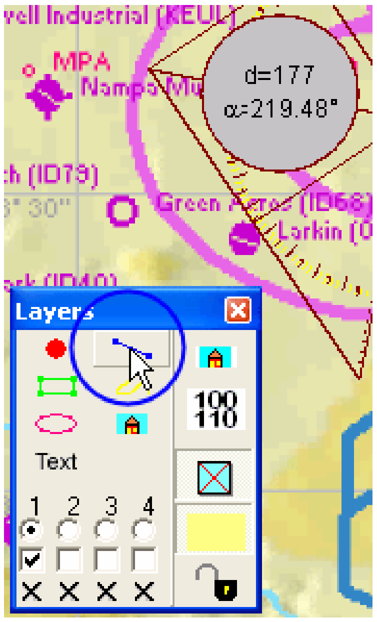
Moving the mouse cursor to the arrival airport will cause the red course line to be extended between the two airports. This can be seen at right. You can now read the magnetic heading inside the grey data window. In this example, a compass heading of about 135-degrees is shown.

The d-value (distance) shows 364. What does that number mean? At this point, it has nothing to do with navigation. Why? Because we have not yet calibrated the software to measure distances in nautical miles. It is still reading in the default unit of pixels.

In practice, after orienting the interface to account for magnetic variation, the distance would then be calibrated. Once these two items are set, you can measure angles and distances to your heart's content.

Calibrating the distance

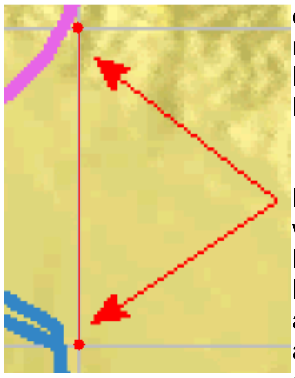
Depending on the zoom of a chart, Flight Simulator can display latitude and longitude lines at 15-minute, 30-minute, 1, 5, 10, or 15-degree intervals. The distance between these lines equates to 1-nautical mile (nm) per minute of latitude (60 nm per degree). It is these lines that will be used as the guide to calibrating MBRuler.



Distance calibration begins at the interface. A right-click on the grey circle opens the control box. Inside that box, choose **Layer Control**. That will open the Layers box seen in the illustration at right. The time has arrived to construct our first Temporary Object.

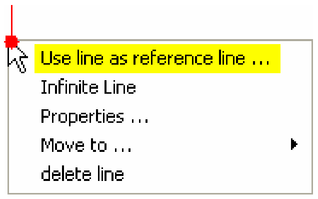
We will use the blue line (inside the blue circle) to draw a calibration object. This is simply a line drawn between two of the latitude lines on the FS chart. Those are spaced every 15-minutes in the charts seen here. That will produce a line of known length, in this case, 15-nautical miles. It is important to construct this line as accurately as possible keeping it parallel to one of the longitude lines.

At left, the red temporary line has been positioned between two lat/long intersections. After the line is drawn, the round end points

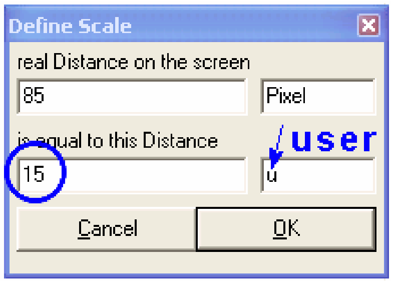


can be adjusted, if needed, by using the mouse. In this example you can see why the line is kept parallel to a longitude. If the red line had any slant, it would be longer than the 15-nm needed for this calibration.

We need to inform MBRuler that this line is to be used as the calibration factor when measuring distances. This is done by left-clicking on either end of the temporary line. That opens the box shown at right. You can see all the choices available, but for calibration, “Use line as reference line” is our choice. When through with calibration, use the “delete line” to zap the line from the screen. That is why it is called a temporary object.



Clicking “Use line as reference line” will activate the box at left. There, the calibration unit of 15 is entered. OK this value and the software becomes



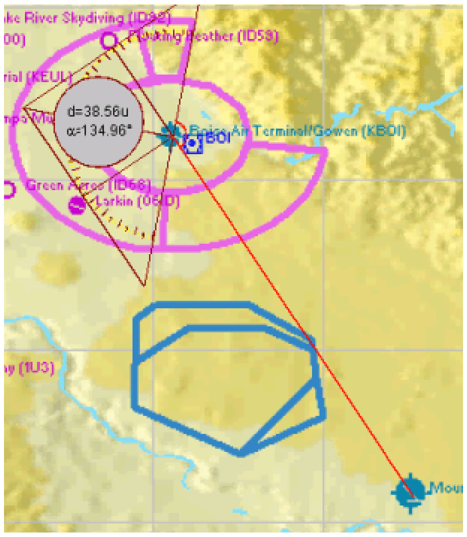
Define Scale

real Distance on the screen: 85 Pixel

is equal to this Distance: 15 u

distance-calibrated. It now knows that 85-pixels are equal to 15 user-defined units. In this example, that means 15-nm. MBRuler now has the capability to convert any pixel value into the correct nautical mile value. Distances other than nautical miles could be calibrated in the same manner.

Returning to the FS chart, we can now read the distance for the course between the two airports. In this case we see it is about 39-nautical miles at a compass heading of 135-degrees. The Flight Planner inside FS calculated this route at a distance of 37-nm and a heading of 137-degrees. The difference between the two methods lies in the way each application makes heading and distance calculation.

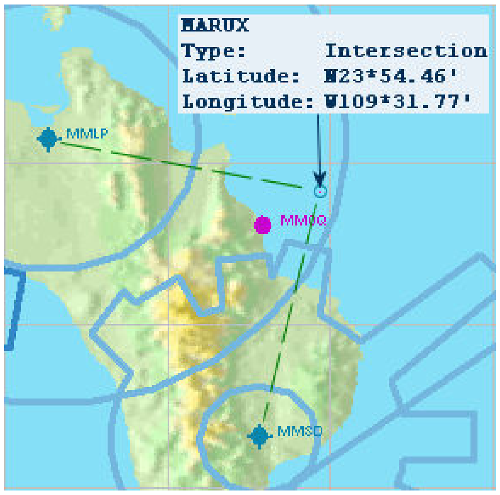


A gain in calibration accuracy can be achieved by using longer distances. For example, instead of using 15-min of latitude, use a span of 30-min when possible.

Another example

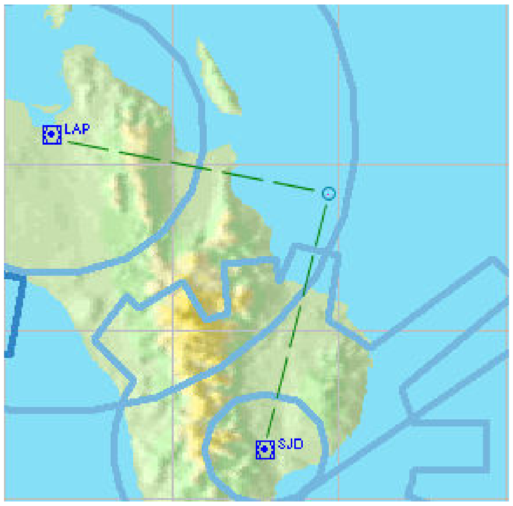
For the second example, a more complex dogleg flight will be plotted. A couple of bearings from VORs will be thrown in for navigation.

The area chosen is the lower tip of the Baja peninsula in Mexico. This whale-watcher’s adventure will depart from MMLP; fly out to a reported pod of humpbacks with albino calves; then return to the coast for a landing at MMSD. And if we can’t find whales, at least we will not get lost.



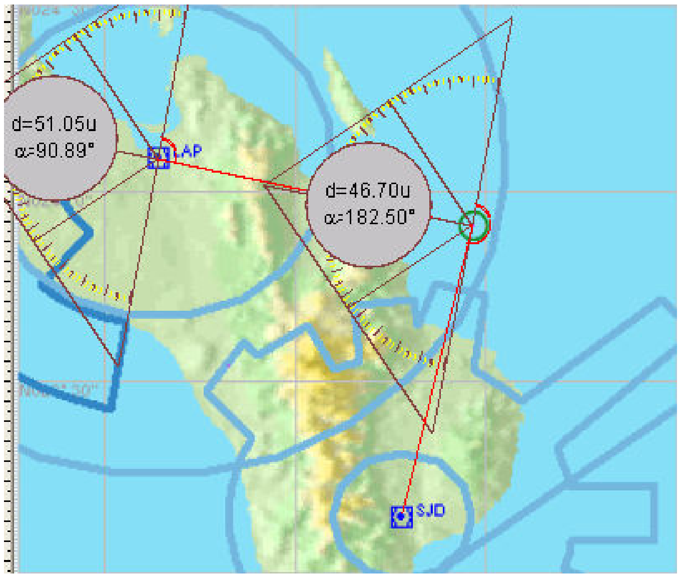
With MBRuler you are not limited to only those waypoints required by the FS Flight Planner such as airports, navigational aids, or intersections. Now you can plot courses between any points desired and fly where you wish.

However, for this demonstration an intersection called MARUX was chosen as the dogleg points so that a flight plan prepared by MBRuler could be compared to the same plan as computed by Flight Planner. The chart at left shows the two legs of this adventure



At right, the two navigational VORs near each airport are shown. The game plan is to track outbound on a radial of LAP located at MMLP. That course will be maintained until we intersect a radial from SJD located at MMSD. That will position us at the intersection inside the green circle. If no whales are found, we will track into MMSD using the SJD radial to guide us to our destination. For this flight, we need to measure the VOR radials and distance of each leg. For compass headings the magnetic variation of 10 1/2-degrees East will be used.

The first step in flight planning is the orientation of the triangular interface to the compass rose. The interface is then rotated clockwise to offset the East magnetic variation. Then, distance units are calibrated to nautical miles using two latitude lines on the chart. The lat/long spacing on these demo charts are in 30-minute intervals, that is, 30-nautical miles.

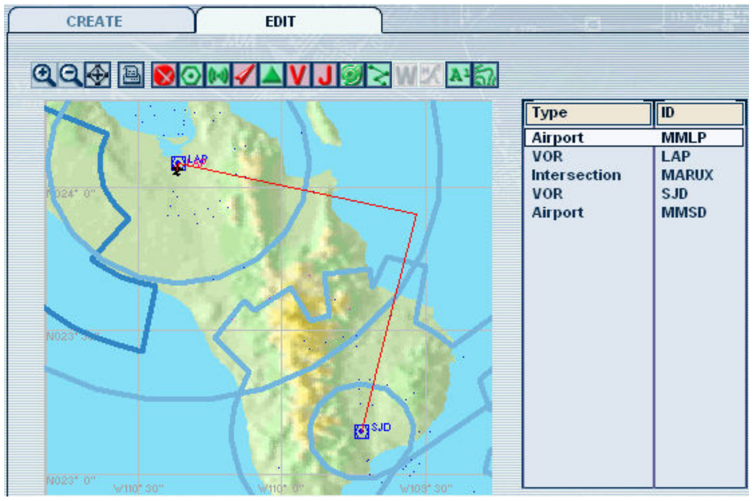


At left, the interface was initially positioned at LAP. The red course line was extended out to the dogleg point. The distance between those two points is about 51nm. The aircraft will track outbound on the 91-degree radial from the VOR located at MMLP.

After obtaining the values for the first leg, the interface was repositioned at the dogleg. A new course line was extended downward to SJD. The second leg has a distance of 47nm. The bearing to the VOR is 183-degrees. For this flight, compass headings will equal the radials of the VORs.

Following the example just shown, you can plot and measure as many points as will fit on a Flight Simulator chart. Bear in mind, if you zoom the chart, distance calibration will no longer be correct. If you shift the chart in any direction, the interface will not shift with it because it is a separate program running independently of Microsoft Flight Simulator.

Let's use Flight Planner to set up the same two legs, then compare the data collected by both flight planning methods.



Type	ID
Airport	MMLP
VOR	LAP
Intersection	MARUX
VOR	SJD
Airport	MMSD

The illustration at left shows Flight Planner's layout for the whale-watcher's adventure. It insisted on including the legs between the airports and their associated VOR when the routing was VOR to VOR even though these legs were not wanted. When I tried to use Direct-GPS routing, then the courses were drawn using the airports but without any VORs.

Below, the printout of the flight plan is shown. The comparison figures for the first leg are circled in blue. The second leg is circled in green.

MBRuler measured the distance of the first leg at 51nm; the heading was 91-degrees. For the second leg, MBRuler calculated the distance at 47nm with a heading of 183-degrees.

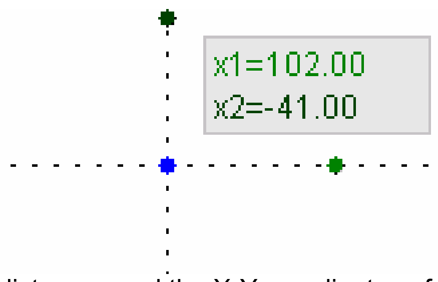
MSFS calculates a distance mathematically using latitude/longitude coordinates. This is why Flight Planner is limited to known waypoints when being used for flight planning.

On the other hand, MBRuler calculates distances based on pixel values measured on the computer screen. This allows the distance between any two points to be converted into nautical miles. But pixels are whole units. The software cannot measure half a pixel.

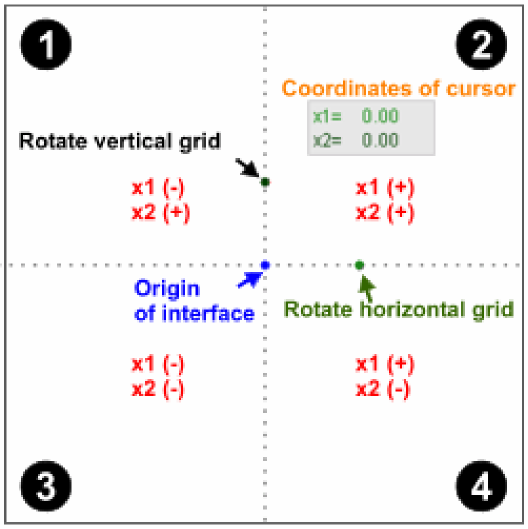
Microsoft Flight Simulator Flight Plan
Gen Manuel Marquez De Leon -> Los Cabos Intl
Distance: 95.3 nm
Estimated fuel burn: 34.8 gal / 233.0 pounds
Estimated time en route: 0:18

Waypoints	Route	Alt (ft)	Hdg	Distance	GS (kts)	Fuel	Time off
				Leg		340.3	0:00
MMLP				Rem	Est	Est	ETE
				95.3	Act	Act	ATE
LAP (112.30)	-D->	650	004	1.4	297	0.5	0:00
				93.9			
MARUX	-D->	11500	092	46.7	302	17.1	0:09
				47.3			
SJD (114.00)	-D->	490	182	46.9	302	17.1	0:09
				0.3			
MMSD	-D->	358	154	0.3	303	0.1	0:00
				0.0			

Coordinate System interface



The remainder of the tutorial will concern itself with demonstrating the second interface available in MBRuler. Coordinate System is basically designed to do two things; first, it tracks and displays the X-Y pixel location of the mouse cursor as it moves about on the computer screen. However, it does this a bit differently than you may be used to. This will be demonstrated shortly. Second, the Coordinate System interface can be calibrated to a user defined unit in a way that is similar to the triangular interface. After calibration, MBRuler will determine distances and the X-Y coordinates of points that have useful application in Microsoft Flight Simulator. It is this second application that will receive the major focus.



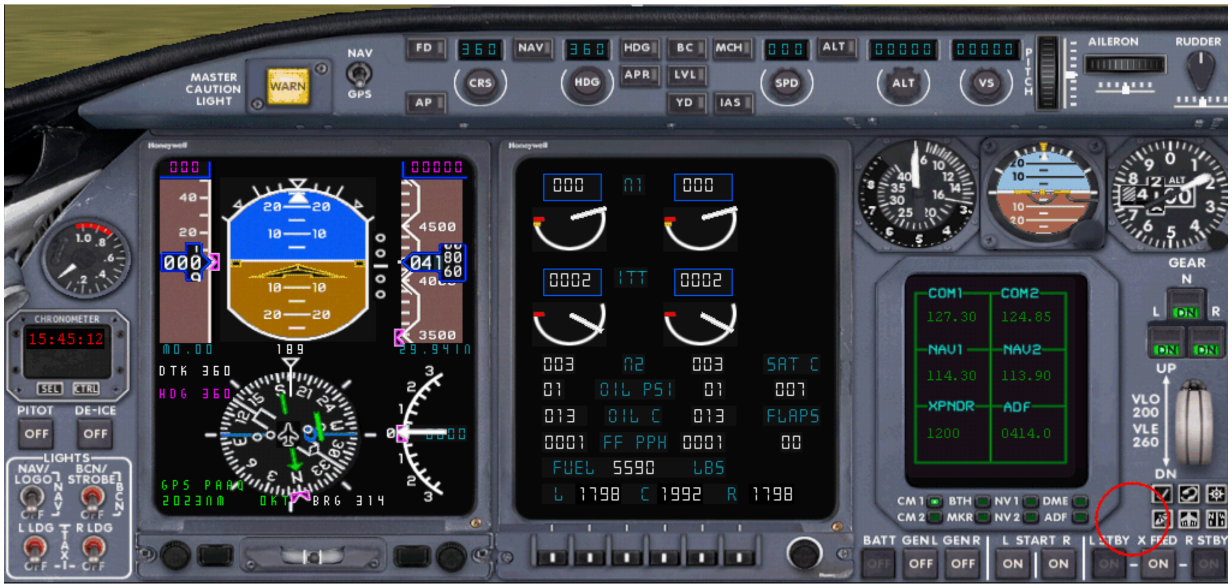
Many of the basic parts of the Coordinate System grid is shown here. The grey data rectangle operates in some ways like the grey data circle on the rectangular interface. For example, a right-click will open the same control box giving access to the same options described in the first part of this tutorial. The grey window can be moved with the mouse should its position interfere with any procedure that is underway.

The data inside the grey area are pixel coordinates of the mouse cursor. It tracks the mouse position; however, data is displayed with an x1-x2 designation instead of X-Y values. Think of x2 as the Y-value.

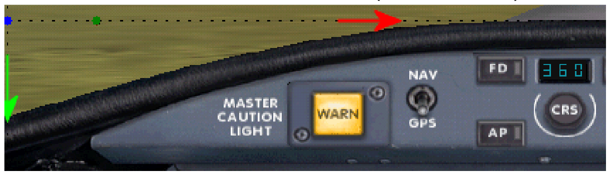
This system uses four quadrants (the numbered black circles) that surround the origin of the interface (blue dot). The origin has a value of x1= 0, x2= 0. Coordinate values of the mouse position can be either positive or negative depending on which value is being read in a particular quadrant. Values of x1 that are to the right of the vertical grid are positive. They are negative left of the vertical grid. Values of x2 that are above the horizontal grid are positive. They become negative below the grid.

Both horizontal and vertical grids can be rotated independently of each other. This is done by dragging either the black dot on the vertical grid or the green dot on the horizontal grid.

Not all of the highlighted features are helpful for this tutorial but the above description will serve as an introduction to many of the unfamiliar items.



The Lear 45 panel above will serve as the platform for the next demonstration. This project will use MBRuler to determine coordinates for the installation of a new gauge. In this example, a toggle switch that operates the parking brake will be located inside the red circle at the lower, right of the panel. The panel above looks a tad squashed because, as noted previously, FS can not be run in full screen mode when using any MBRuler interface. Attempting to do so will cause FS to be minimized (not a crash) to the taskbar when MBRuler is used.



For orientation, this interface is placed so the horizontal grid (red arrow) touches the top of the panel bitmap. The vertical grid (green arrow) touches the left side of the bitmap.

When oriented in this position, the Coordinate System will have its origin (blue dot) directly over the upper, left corner of the panel bitmap. That is the X= 0, Y= 0 corner for the panel graphic; therefore, we want the origin of the interface to coincide with that location.

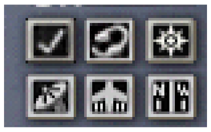
The manner in which panel bitmaps are displayed in Flight Simulator can be rather confusing. For example, the Lear has two such bitmaps. One measures 640 x 333; the other measures 1024 x 533. Do you know which bitmap you are seeing when flying the Lear? While this may not affect flight characteristics or pilot performance when jockeying that aircraft, it does play a tremendous role when gathering gauge coordinates for use in the panel.cfg file.

But other factors affect these coordinates as well. These can be found in various panel.cfg files within MSFS. For example, the "`size_mm=640`" statement from the configuration file of the Lear will alter bitmap coordinates. So will another type of statement seen in the Bell Jet Ranger configuration file, "`windowsize_ratio=0.632`".

How can one obtain accurate X-Y coordinates for gauges directly from a panel bitmap if the effects of the above factors are uncertain? Plus, add the fact that the bitmap is squashed while running in window mode, thus altering its width/height ratio by an unknown amount?

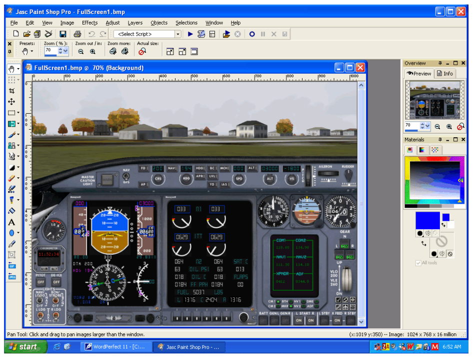
Calibrating the Coordinate System interface

All of the above uncertainly can be put to rest. The Coordinate System interface can be accurately and quickly calibrated by using gauges that are already installed. The X-Y coordinates for those are obtained from the appropriate panel.cfg file. Easy instruments to assist with this calibration are the small, square icons (at right) because the top, left corner (0, 0) of these gauges is easily identified. Shy away from using gauges that are round if the zero corner of their bitmap is not for certain.



Calibrating the Coordinate System interface using installed icon gauges can be done two ways. The first method requires another graphics program such as Paint Shop Pro. MS Paint has few zoom adjustments and is not a good choice for this job.

The reason for extra software is this: a screen shot of the Lear panel is made while FS is running in full-screen mode. This shot is pasted into the graphics program. An example can be seen at left. The pasted image is then used as if it were the actual panel viewed while FS was running. The image will allow MBRuler to be calibrated and X-Y coordinates to be obtained for gauge installation. The X-value and the Y-value can be calibrated together since the pasted image of the panel bitmap is full-screen and not under the distorting effect from running FS in window mode.


Jasc Paint Shop Pro - FullScreen1.bmp
FullScreen1.bmp @ 70% (Background)

In the illustration at left, the full-screen, pasted shot of the Lear panel is on display. It is viewed at 70-percent zoom allowing the entire panel to be visible without scrolling.

MS Paint does not display panel bitmaps in a manner where they can be seen in their entirety which is critical for this first method. Instead, in that program you must scroll about to see the entire panel. Scrolling causes the origin of the MBRuler interface to shift its location since it does not scroll too. That causes all coordinates to be measured incorrectly.

The second method of calibration requires no additional software since calibration and determining X-Y coordinates can be done entirely within Flight Simulator. The downside to this procedure is the requirement that FS be running in windows mode. The resulting distortion will necessitate that the X-value be calibrated separately from the Y-value. Instead of reading both X and Y coordinates in one step, two steps will be required. This is not a big deal because the procedure goes quickly once you get accustomed to using the MBRuler software.

Both methods of calibration will be demonstrated in the pages that follow.

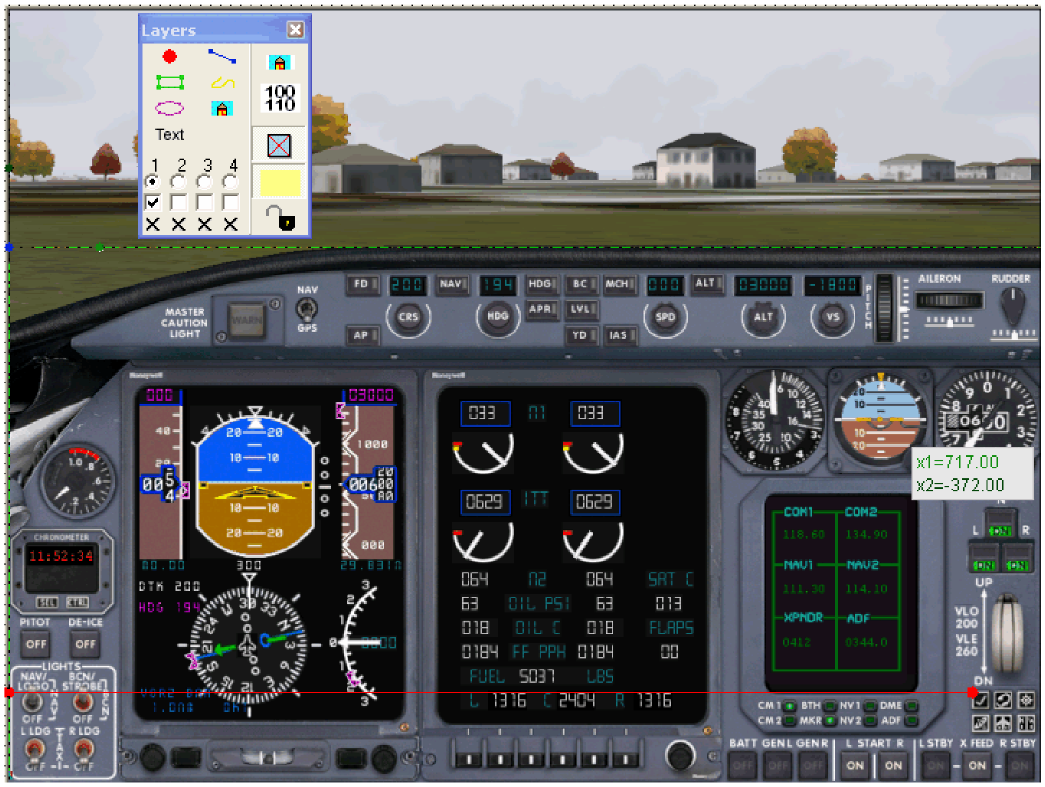
Calibration using a screenshot image

The first step in this method is to produce a screenshot of the panel while FS is running in full-screen mode. This can be done by pressing the **Print Screen** key on your keyboard. That places the screen image onto the clipboard. At this point, press the **Alt** key to bring up the menu in Flight Simulator. That will allow you to remove the checkmark at **View/Full Screen**. After FS changes to window mode, you can minimize it to the taskbar.

```
gauge30=Lear_45!Nav GPS Switch, 178, 24
gauge36=SimIcons!Kneeboard Icon, 597, 276
gauge37=SimIcons!ATC Icon, 611, 276
```

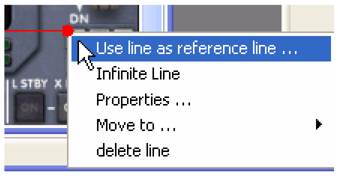
The clipboard data can then be pasted into your graphics program. Adjust the panel image so that you can see it in its entirety at the largest possible zoom.

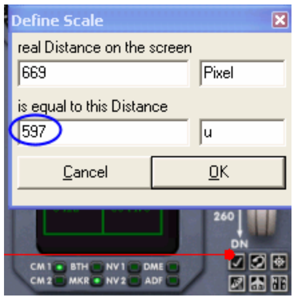
Once you have a suitable image, MBRuler can be calibrated. For this example we will use the kneeboard icon in the Lear panel. A check of the panel.cfg file (at right) shows the installation coordinates for that gauge as X= 597, Y=276. Either of these numbers can be used to calibrate the MBRuler interface. Here, we will use the longest distance, the 597-value.



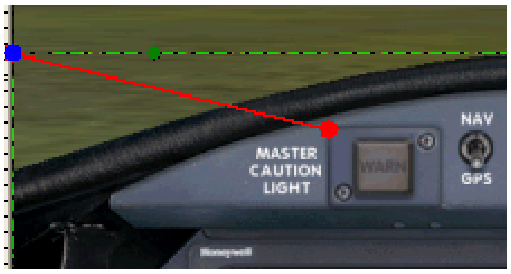
In the illustration above, several steps have been completed. First, the origin of the interface (blue dot) was placed over the upper, left corner of the panel bitmap. I highlighted the horizontal and vertical grids of the interface in bright green so they can be seen well. I have found no way to change the color of this interface as could be done with the triangular interface.

Next, the control box was opened by right-clicking on the origin of the interface and **Layer Control** was chosen. That opened the Layers box seen above the panel. Choosing the temporary line tool, a red calibration line was drawn from the corner of the kneeboard icon to the left side of the panel bitmap (the vertical grid). A right-click on either end of this line will cause the box at right to appear. From that box, the red temporary line can be calibrated to the kneeboard X-value of 597. At left, the Define Scale has received this value. An OK will enter the X-coordinate and will calibrate both the X and Y directions at the same time.

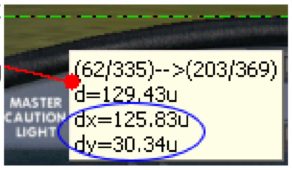




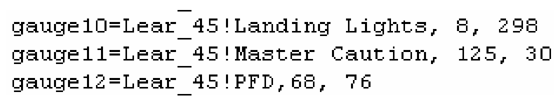
real Distance on the screen
669 Pixel
is equal to this Distance
597 u



To measure X-Y coordinates together requires that the left end of the temporary line be moved atop the origin of the interface. In the picture at left this has been done. Then, by moving the dot at the right end of the line, calibrated coordinates can be read. Let me demonstrate by measuring the top, left corner of the Master Caution gauge. You can see the end of the temporary lines sits above the corner of the Master Caution bitmap. By allowing the mouse cursor to linger above the red dot, a popup data box will appear with the coordinates inside. That data can be seen at right. The dx shows a value of about 126-pixels; the dy contains a value of about 30-pixels. Let's see how that compares with the installation coordinates for this gauge as found in the panel.cfg file. A portion of that file is shown below. You can see the two sets of figures are equal.



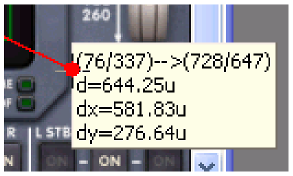
```
(62/335)-->(203/369)
d=129.43u
dx=125.83u
dy=30.34u
```



```
gauge10=Lear_45!Landing Lights, 8, 298
gauge11=Lear_45!Master Caution, 125, 30
gauge12=Lear_45!PFD,68, 76
```

All other desired location on the panel bitmap can be measured quickly by leaving the left end of the line at the origin of the interface. The right end of the line becomes the movable measuring point recording coordinates as needed.

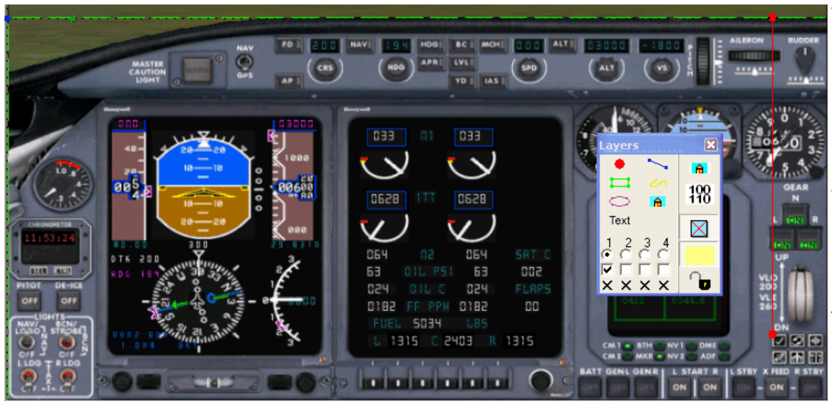
Let's take one more measurement. This time we will obtain a location for our parking brake gauge. That task is shown at right. The red dot gives an X-value of 582-pixels and a Y-value of 277-pixels. I will not install the gauge at this time but will wait until the next demonstration (using window mode) is completed. During that procedure I will again measure the location for the parking brake and compare those coordinates with the ones just measured. The gauge will then be installed to check the location.



```
(76/337)-->(728/647)
d=644.25u
dx=581.83u
dy=276.64u
```

Calibration using window mode

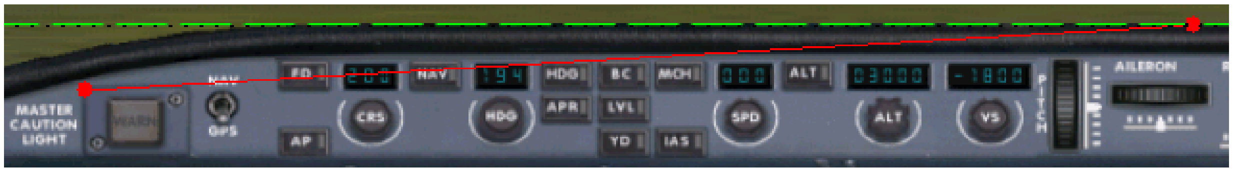
I will cheat a bit during this demonstration. Instead of making measurements in Flight Simulator while it runs in window mode, I will take a screenshot of that mode and take measurements from the pasted image. This will not change any procedures or results. Instead, it allows graphics for this tutorial to be prepared a lot faster without having to continually minimize FS to work in other applications. As this tutorial is written, I have several CPU hogs running simultaneous. If FS is kept minimized, the process is speeded up greatly.



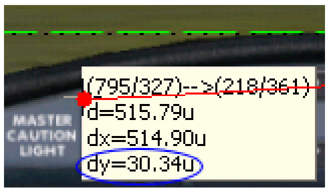
At left, the origin of the interface has been positioned at the origin of the panel bitmap. Calibrating the X-direction would be done exactly as previously shown in the first demonstration using the full-screen image and will not be repeated here. Instead, the Y-calibration will be shown. Two separate calibrations are necessary since this panel is squashed out of its original proportion.

A red temporary line has been extended from the corner of the kneeboard icon up to the horizontal grid of the interface. This line will be calibrated to a value of 276. If you remember, that is the value of the Y-coordinate for the kneeboard icon as shown in the panel.cfg.

When reading X and Y values separately, the end of the calibration line does not have to be moved to the origin of the interface. That step is required only when reading both X and Y coordinates at the same time. Here, the end of the line can be left in its original position atop the horizontal grid.



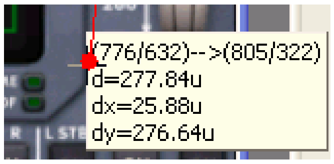
Above, the measuring end of the line has again been shifted to the origin of the Master Caution bitmap. The popup coordinate box contains the Y-calibrated data. The X-value is meaningless because it is not calibrated. The data is shown at right. You can see the Y-coordinate is 30-pixels, the same value shown in the panel.cfg file.



```
(795/327)-->(218/361)
d=515.79u
dx=514.90u
dy=30.34u
```

The last measurement in this demonstration is to obtain a Y-value for installing the parking brake gauge. The X-value will be exactly the same as was measured during the first demonstration because both procedures would calibrate the X-distance using identical values.

At left, the Y-coordinate (dy) was measured at about 277-pixels. That is the same value measured in the first demonstration. We now have installation coordinates for the parking brake gauge of X=582, Y= 277. Let's install the gauge using these values and see what the results are.

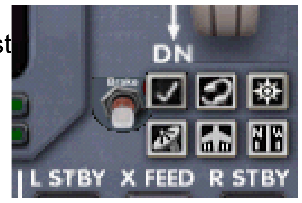


```
(776/632)-->(805/322)
d=277.84u
dx=25.88u
dy=276.64u
```

The panel.cfg was edited as shown next. The size-x and size-y values for the brake switch were determined when this gauge was under testing. They were repeated here to keep the gauge small because installation space in the targeted area is at a premium.

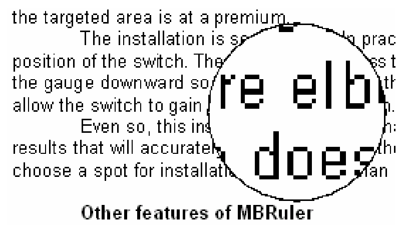
```
gauge32=SimIcons!ECU Icon, 611, 290
gauge33=SimIcons!Compass Icon, 625, 290
gauge34=PkBrk1_G!BrakeSw1, 582, 277, 18, 18
```

The installation is seen at right. In practice, this would not be the final position of the switch. The appearance is less than satisfactory and I would adjust the gauge downward so that it was even with the lowest row of icons. That would allow the switch to gain a bit more elbow room.



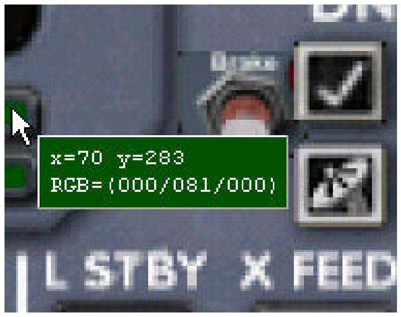
Even so, this installation does show that Coordinate System produces results that will accurately locate a gauge in the targeted position. You just have to choose a spot for installation a tad better than did I.

Other features of MBRuler



Available from the control box is the magnifier show at left. This tool has many adjustment options such as increasing/decreasing the magnification. Those are found inside the Options/Loupe tab.

A rather interesting tool is the RGB Color Tool also activated in the control box. It has a small window that follows the mouse cursor. The window not only shows color data but the window changes its background to reflect the color of the pixel under the mouse. This tool also has many adjustments inside the Options/RGB tab.



```
x=70 y=283
RGB=(000/081/000)
```

This ends the tutorial.