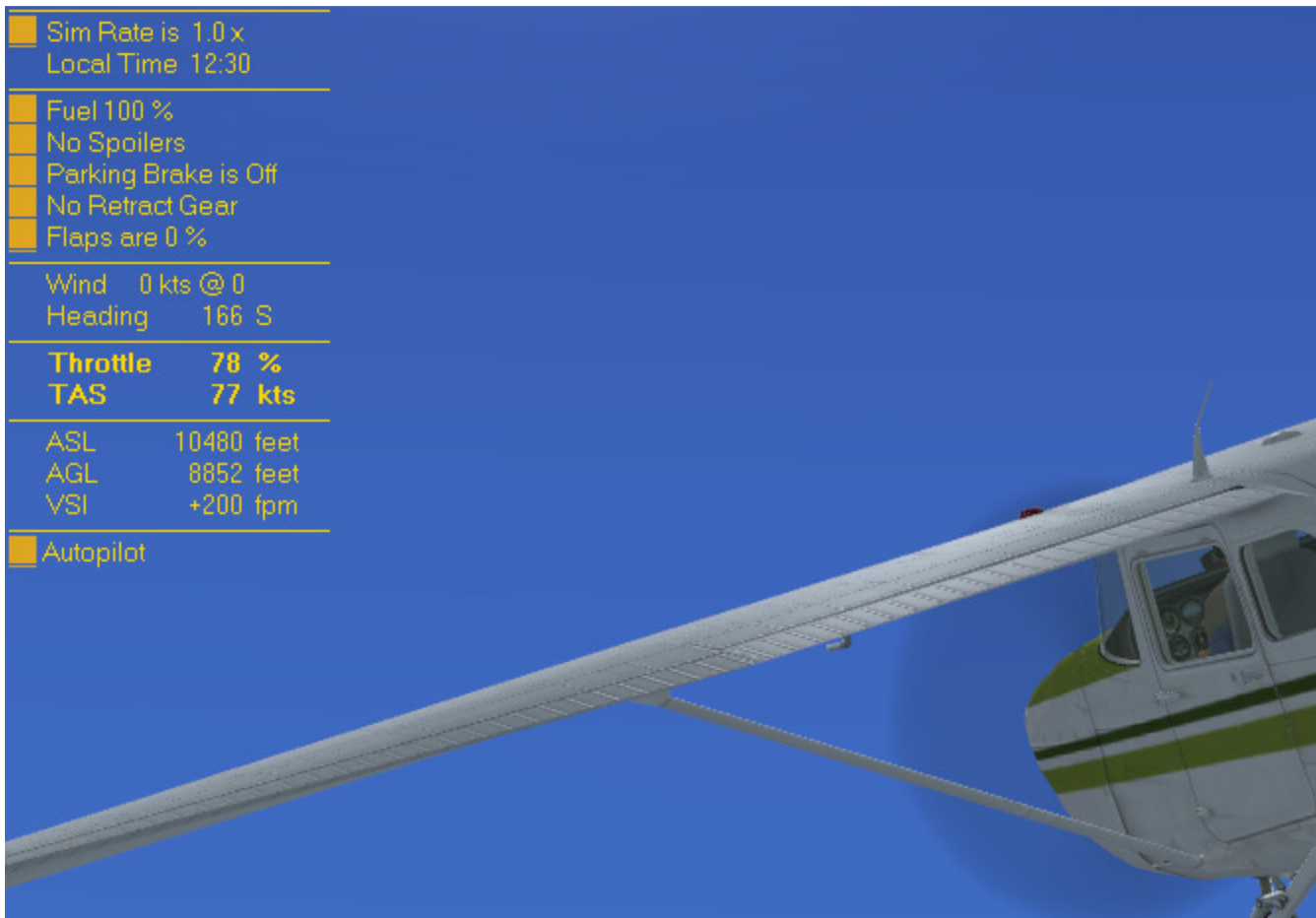


TextInfo Gauge for Flight Simulator X

Version 3.0



Rick Federmann
Pville211@gmail.com
April 2014

Overview	3
Features of TextInfo Gauge.....	4
Requirements.....	4
What it Displays.....	4
Using the simulation rate feature	8
Quick Installation.....	10
Step-by-Step Installation	11
Step 1 - Copy TextInfo Gauge into the FSX Gauges Folder	11
Step 2 - Edit the Panel.cfg in the Aircraft Panel Folder	11
Step 3 - Run TextInfo Gauge.....	13
Step 4 - Adjust the Size of the Display.....	13
Change the Colors	16
Turn Off FSX Screen Messages	16
TextInfo Gauge XML Script Explained	17
Structure of TextInfo Gauge.....	17
Formatting the display	18
Begin displaying the data	20
{if} statements.....	21
Display simple text and data	22
Display text and two data readouts in the same line	22
Text and data with a plus/minus sign in front of the data.....	23
Percentage calculation.....	23
Display an item only if it is engaged, and use an alternate color	23
Display an item only if the aircraft is equipped with the item.....	24
Change the color of the readout based on an either/or condition	24
Blinking warning indicator.....	24
Use multiplier for number rounding for less rapid changes in readout	25
Only display if aircraft has item AND item is engaged AND change readout color based on condition	25
Display heading in degrees and also abbreviated direction (N, W, S, E)	27
Use one line for various readouts, with color changes and flashing text, as needed.....	27
Display several indicators in the same line at the same time.....	29
Mouse click spots in a transparent text gauge that has no graphic	29
FSX Shift-Z Feature	32
Configuring Shift-Z: Parameters.....	32
Configuring Shift-Z: The Variables.....	35
Configuring Shift-Z: Font Color.....	35

Helpful Resources.....	37
Microsoft Resources.....	37
Other Resources.....	37

Overview

TextInfo Gauge displays real-time flight data as text on the Flight Simulator X screen. It also allows toggling equipment and changing flight settings. It is a single, small XML file that does not impact display frame rates or program operations.

The gauge is an alternative to the Shift-Z function in Flight Simulator X (FSX). Shift-Z provides a very simple display of flight data such as airspeed and altitude. It has a very limited list of data readouts and has almost no formatting options.

The data displayed by Shift-Z is defined in the [TextInfo.x] sections in the FSX.cfg file. This is where TextInfo Gauge gets its name. Like Shift-Z, TextInfo Gauge displays flight data as screen text. Unlike Shift-Z the TextInfo Gauge can display nearly any FSX variable, will accept mouse input to control aircraft functions and settings, and the display format can be modified in many ways.

Installing and using TextInfo Gauge is easy. Experienced users only need to follow four simple steps in the Quick Installation section to get TextInfo Gauge up and running.

Following the quick installation instructions are detailed installation information, reference information for modifying the gauge and tips for XML scripts in general.

info

TextInfo is a standalone gauge that does not replace Shift-Z. The Shift-Z function will be not be affected by TextInfo Gauge.

info

Shift-Z configuration is limited to selecting readouts from a very short list of flight data, on a few display lines. A common opinion is that the text color cannot be changed. That is incorrect. See the Shift-Z section near the end of this document for details about how to do it.

Features of TextInfo Gauge

- Completely text based. No impact to frame rates or performance.
- Can display a wide variety of flight data.
- Can be popped up from any camera view.
- Remains on the screen when cycling through camera views.
- Transparent; displays text without blocking a portion of the screen.
- Has mouse click spots for settings and equipment operations.
- Autopilot items are not displayed when the AP is disengaged to reduce the screen footprint.
- Small, uncomplicated file that is easily modified to change readouts, formatting, colors, etc.
- XML script is formatted to facilitate modification by nonprogrammers.

info

Conventional thinking is that a gauge must have a background graphic to enable mouse clicks. TextInfo displays text without a graphic, yet has mouse click spots for a variety of actions. An explanation of how to do this is provided later in this document.

Requirements

- TextInfoGauge30.xml is the only file that is required.
- No bitmap graphic file.
- TextInfo Gauge was tested with Microsoft Flight Simulator Gold (FSX Accelerator).

What it Displays

In its default configuration TextInfo Gauge provides real-time readouts for:

- Simulation rate (speed of the simulation)
 - Can be changed by hovering over mouse spot and scrolling the mouse wheel.
 - Increase the sim rate for long, boring flights. Cross large bodies of water in minutes.
 - Decrease the sim rate to less than 1x to practice difficult approaches or maneuvers.
- Local time: 24-hour clock

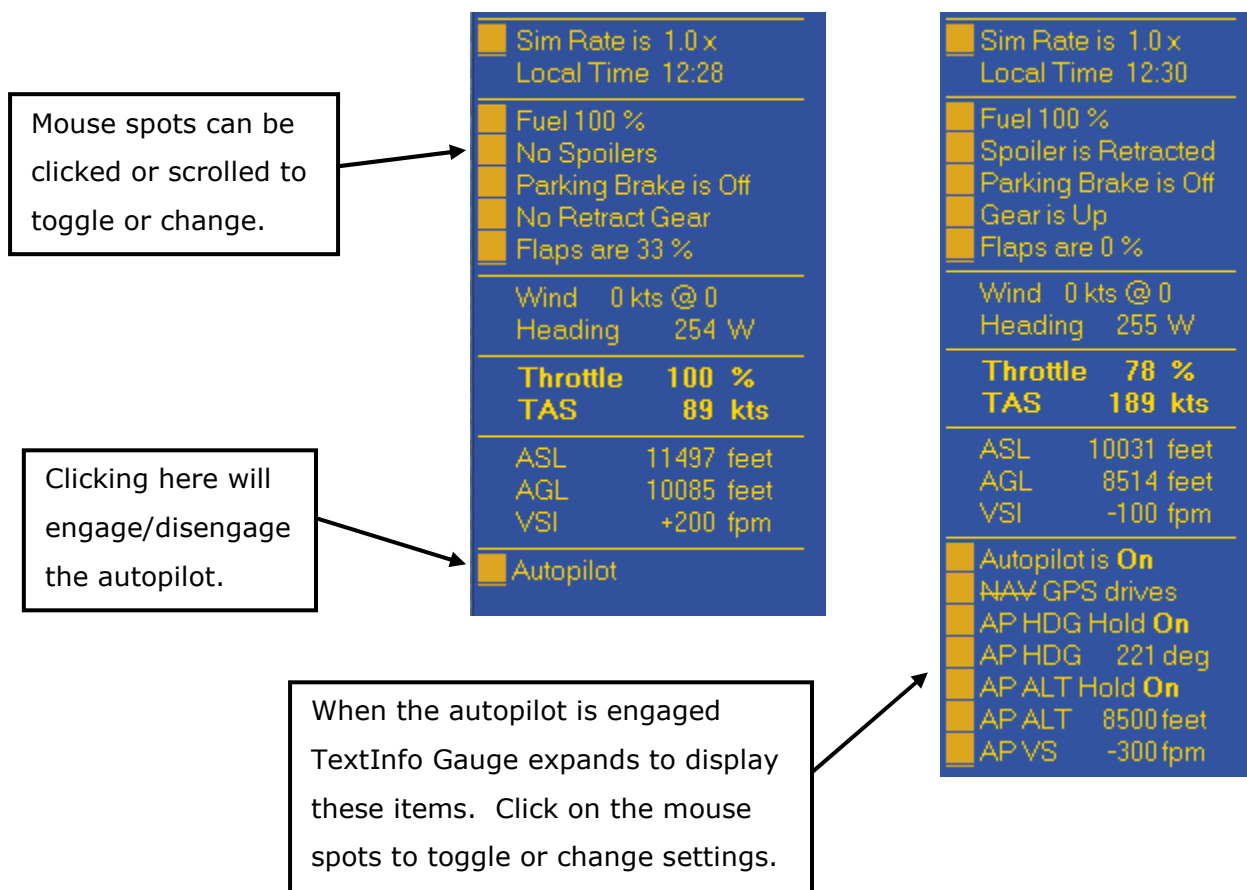
info

The ability to instantly change the simulation rate is convenient. However, there are some nuances that should be understood before using it. See the section below that describes simulation rate usage.

- Fuel status
 - Displays the remaining % of fuel.
 - When remaining fuel falls below 10% the line is highlighted.
 - When remaining fuel falls below 5% the line is highlighted and flashes.
 - Clicking on the mouse spot adds fuel in increments of 25% of capacity, even while in flight.
- Stall and overspeed warnings
 - Displayed as needed on same line as fuel readout to reduce the gauge's screen footprint.
 - Warnings are highlighted and flashing.
- Parking brake status
 - Indicates if parking brake is on or off.
 - Clicking on the mouse spot toggles the parking brake on and off.
- Spoilers status
 - Indicates if spoilers are not available on the aircraft.
 - Highlighted and flashing when deployed.
 - Clicking on the mouse spot extends and retracts the spoilers.
 - Arming auto-spoilers must be done via keyboard or controller but the armed status will be indicated in the gauge.
- Landing gear
 - Indicates if retractable landing gear is not available on the aircraft.
 - Indicates if the gear is fully up or fully down.
 - Indicates real-time progress percentage during extension and retraction.
 - Clicking on the mouse spot extends and retracts the landing gear.
- Flaps
 - Indicates if flaps are not available on the aircraft.
 - Indicates % position of the flaps (0%, 50%, etc.).
 - Clicking on the mouse spot extends or retracts the flaps in increments.
- Ambient wind: real-time velocity and direction
- Heading in degrees and direction (N, E, S, W)

- Throttle percentage
- True airspeed
- Altitude above sea level
- Altitude above the ground
- Vertical speed: rounded to 100 fpm to make it easier to follow real-time variations
- Autopilot master switch
 - Only displayed if the aircraft has an autopilot.
 - The remaining autopilot items below are displayed only when the autopilot is engaged, to reduce the gauge's screen footprint.
 - Clicking on the mouse spot engages and disengages the autopilot.
- GPS drives NAV
 - Displays if NAV or GPS is steering the aircraft.
 - Clicking on the mouse spot toggles the setting.
- Autopilot heading hold
 - Displays if autopilot is controlling the heading.
 - Clicking on the mouse spot toggles heading hold on and off.
- Autopilot heading bug
 - Displays the autopilot heading setting in degrees.
 - Can be changed by hovering over the mouse spot and scrolling the mouse wheel.
- Autopilot altitude hold
 - Displays if autopilot is controlling the altitude.
 - Clicking on the mouse spot toggles altitude hold on and off.
- Autopilot altitude setting
 - Displays the autopilot altitude setting in feet.
 - Can be changed by hovering over mouse spot and scrolling the mouse wheel.

- Autopilot vertical speed setting
 - Displays the autopilot vertical speed (climb / descend rate) setting in feet per minute.
 - Can be changed by hovering over mouse spot and scrolling the mouse wheel.



Using the simulation rate feature

TextInfo Gauge makes it easy to quickly change the simulation rate. Speed up the sim rate to quickly fly a long distance. Reduce the simulation rate to move in slow motion without affecting the flight dynamics while practicing difficult maneuvers and approaches.

Simulation Rate vs Camera Zoom

Using TextInfo Gauge to change the simulation rate will appear to affect your ability to zoom in and out in the camera views. You increase the sim rate to 2x, then you attempt to zoom in on the aircraft and instead of zooming in the sim rate increases to 4x. This is normal FSX behavior and is easy to resolve.

Some items in FSX use "increase selection" and "decrease selection" commands. These are implemented via the -/+ keys. In a camera view you can press the minus or plus keys to change the zoom level. Changes in simulation rate are accomplished using these same keys.

How does FSX avoid a conflict with using the same commands for simulation rate and camera views? By assigning the -/+ keys to the *current* selection.

In default FSX operations pressing the "R" key makes simulation rate the current selection and you can decrease/increase the sim rate by pressing -/+. When done, click anywhere on the screen to make the screen the current selection, thus restoring normal -/+ zoom operations.

Using TextInfo Gauge to change the simulation rate involves the same behavior. [After changing the sim rate simply click anywhere on the screen to resume normal camera zoom operations.](#)

An alternate method is to press the Backspace key after changing the sim rate. In addition to restoring normal zoom operations, however, it also resets the zoom level. This may not be a good choice for many users.

Sim Rate Limit on Ground

While the aircraft is on the ground the simulation rate is limited to 4x.

Sim Rate Limit of Autopilot

The autopilot is limited to a simulation rate of 16x.

If the autopilot is already engaged before you change the simulation rate you will find that you cannot increase the simulation rate beyond 16x.

If the simulation rate is already more than 16x then the rate will drop to 16x when the autopilot is engaged.

Quick Installation

1. Create a TextInfoGauge folder in the FSX Gauges folder and copy TextInfoGauge30.xml into it.
2. Edit the aircraft's panel.cfg file.

```
[Window Titles]
```

```
Windowxx=TextInfo Gauge
```

```
[Windowxx]
```

```
size_mm=60,300           // must match dimensions described in the XML file
```

```
background_color=0,0,0   // make panel background transparent
```

```
window_size_ratio=0.5    // edit to adjust size of gauge in each aircraft
```

```
position=0               // positions gauge at top left corner of screen
```

```
visible=1                // makes gauge available when aircraft is loaded
```

```
ident=MISC_POPUP_10      // use any unused ident name
```

```
window_pos= 0.000, 0.025 // edit to position display's top left corner
```

```
type=special              // keeps gauge on screen when cycling camera views
```

```
render_3d_window=1       // forces gauge to render as an element of the 3D
```

```
//      window, not as a separate window; prevents
```

```
//      accidentally grabbing and dragging the gauge
```

```
//      when clicking on items in the gauge
```

```
gauge00=TextInfoGauge!TextInfoGauge30, 0,0
```

3. Load the aircraft and pop up TextInfo Gauge to see how it looks.
4. Adjust the size of the data display in panel.cfg by changing the `window_size_ratio` setting.



`window_pos = 0.000, 0.025` positions the panel all the way to the left and slightly below the top of the screen. This allows room for the FSX menu bar at the top of the screen (i.e. prevents TextInfo Gauge from overlaying the FSX menu bar).

Step-by-Step Installation

These instructions assume the FSX folders are in their default locations in Windows 7. Your environment may be different.

Step 1 - Copy TextInfo Gauge into the FSX Gauges Folder

This step is a one-time FSX action. It will not be repeated for each aircraft.

1. Go to C:\Program Files (x86)\Microsoft Games\Microsoft Flight Simulator X\Gauges\
2. Create a subfolder named TextInfoGauge.
3. Copy TextInfoGauge30.xml into the TextInfoGauge folder.

Step 2 - Edit the Panel.cfg in the Aircraft Panel Folder

This step must be done for each aircraft that will use TextInfoGauge.

1. Go to C:\Program Files (x86)\Microsoft Games\Microsoft Flight Simulator X\SimObjects\Airplanes\AircraftName\panel\
X\SimObjects\Airplanes\AircraftName\panel\
2. **IMPORTANT:** Save a backup copy of panel.cfg. If things go wrong you can simply copy the backup copy back into this folder to recover.
3. Open panel.cfg in a plain-text editor (e.g. Notepad).

info

If an aircraft has more than one panel folder then look in the aircraft.cfg file to determine which folder to edit. Each [fltsim.x] section in aircraft.cfg defines an individual aircraft and has a "panel=" line indicating which panel folder to use. Example: if [fltsim.2] has "panel=G1000" then the panel folder is "panel.G1000." If no folder is defined then it uses the generic "panel" folder.



Always use a plain-text editor (e.g. Notepad) to edit FSX files to avoid introducing hidden control codes.

4. Add TextInfo gauge as a panel in the aircraft:

- a. At the top of the file there is a section called [Window Titles]

In the list of windows add a line:

```
Windowxx=TextInfo Gauge
```

*For purposes of these instructions we will use **Window08=TextInfo Gauge**.*

```
[Window Titles]
Window00=Main Panel
Window01=Radio Stack
Window02=GPS
Window03=IFR Panel
Window04=Landing View
Window05=Mini Panel
Window08=TextInfo Gauge
```

- b. Add a [Window08] settings section:

```
[Window08]
size_mm=60,300
window_size_ratio=0.5
position=0
background_color=0,0,0
visible=1
ident=MISC_POPUP_10
window_pos= 0.000, 0.025
type=special
render_3d_window=1

gauge00=TextInfoGauge!TextInfoGauge30, 0,0
```

info

Windowxx can be any unused number between 0 and 63. The numbers do not have to be sequential. Only Window00 through Window08 use shortcut keys to pop them up in the simulator. Window09 and above can be used, but can only be popped up via the FSX menu (Views > Instrument Panel). If you use Window08 it is likely that most aircraft will have that slot available. This will allow setting up various aircraft with the same shortcut key and programming it on a controller button.

info

The name in the [Window Titles] section does not have to be "TextInfo Gauge." This is simply the name you will see in the FSX menu (Views > Instrument Panel).

info

The [Windowxx] sections are not required to be in number order. In other words, the [Window01] section does not have to appear in the file above [Window02]. In this example, if you add [Window08] above the other [Windowxx] sections it will be easier to locate it when you return to the panel.cfg file to adjust the gauge display size on your screen.



The "ident" name must be unique within each aircraft panel, and can only be specific numbers or names. MISC_POPUP_10 is typically a safe choice. See the last page in this document for a link to the panel.cfg reference information for details regarding the "ident" setting.

5. Save the file.

Step 3 - Run TextInfo Gauge

1. Start FSX and load the aircraft that you edited in the preceding steps.
2. Once the simulation begins, pop up TextInfo Gauge.
 - a. In the example above we added TextInfo Gauge as Window08, so the shortcut key is Shift-9.
 - b. You can also pop up the display using the FSX menu: Views > Instrument Panel > TextInfo Gauge.

info

To pop up a panel the shortcut key is Shift + [panel+1]. For example, to pop up the panel defined as Window08 the shortcut is Shift + 9.

It may be confusing that we added TextInfo Gauge as a panel in panel.cfg. What we actually did was simply use the gauge *name* (TextInfo Gauge) as the panel name. Each panel can have multiple gauges. In this case, we added a panel named "TextInfo Gauge" that has a single gauge named TextInfoGauge30.xml.

Step 4 - Adjust the Size of the Display

1. Go C:\Program Files (x86)\Microsoft Games\Microsoft Flight Simulator X \SimObjects\Airplanes \AircraftName\panel\
2. Open panel.cfg in Notepad.
3. In the [Window08] section change the **window_size_ratio** setting.
4. Save the file. You do not have to close the file; just save it so that the changes are written to the file on your hard drive.



If the gauge does not pop up in the simulation then verify that it is running. Use the FSX menu (Views > Instrument Panel).

If TextInfo Gauge is listed and there is a check mark in front of it then the most likely cause is that the gauge is too small to be seen. You may need to significantly increase the window_size_ratio setting.

info

Window_size_ratio is not a linear value, it is a ratio. Changes may have a lesser or greater affect than you anticipate. Start with small changes (e.g. 1.0 > 1.2), then make larger changes once you have a feel for how a particular aircraft display is reacting to the changes.

5. Reload the aircraft in FSX to force it to read the updated panel.cfg file.
 - a. FSX menu: Aircraft > Select Aircraft > OK
 - b. Or you can program a key on your keyboard for faster reloading: Options > Settings > Controls > Buttons/Keys tab > Aircraft (reload)
6. Pop up TextInfo Gauge to see the updated display size.
7. When you are satisfied with the results save and close the panel.cfg file.

info

"Aspect ratio" is the proportional relationship between the width and height of an object. When an object is enlarged or shrunk it may retain its aspect ratio. This is common when changing the dimensions of graphic objects.

For example, let's look at a 4 x 6 graphic. If the width is increased to 8 and the aspect ratio is preserved then the enlarged object will be 8 x 12. The dimensions changed, but not the proportions.

TextInfo Gauge is a text-based gauge, not a graphics-based gauge. **The display size is controlled by window_size_ratio**, which is different than aspect ratio. When the gauge size is changed it does not automatically preserve proportions.

For example, you may find that a window_size_ratio of 1.00 is too small. A setting of 2.50 may be large enough but you discover that the text overlaps or it disappears past the right edge of the gauge. Yet, a window_size_ratio of 2.45 may be a good size and everything looks good.

Experiment with the window_size_ratio in each aircraft to acquire the size and appearance that fits your needs.



When aircraft are reloaded over and over again in the same session then FSX may exhibit these behaviors:

- FSX may freeze (hang).
- Unexpected changes in how TextInfo Gauge is displayed.
- Edits to TextInfo Gauge may not be displayed at all. You make a change, but that change doesn't show up when running the simulation.

This is a known FSX behavior and the fix is to simply restart FSX. It is a good practice to proactively restart FSX after making a lot of changes and after reloading an aircraft multiple times.

Examples of window_size_ratio settings for the author's aircraft at a resolution of 1366 x 768:

Aircraft	Source	window_size_ratio
Air Creation 582SL	FSX	1.80
Aviatika MAI-890	Tim Conrad	2.10
B747-400	FSX	1.25
BD-5	Jez G	0.40
Beech Baron 58	FSX	1.10
Cessna C172	FSX	0.35
Douglas DC-3	FSX	1.50
Ercoupe 415C	Classic Wings	0.75
FA-18	FSX	0.75
Mooney Bravo	FSX	1.75
Piper J3 Cub	FSX	2.00
PT-19	Tim Conrad	0.40
SportCruiser CZAW	WSSimulation	0.40
Stingray Pro	Simon Smeiman	0.40
Swamp Wallaby	Ant's Airplanes	0.40

Change the Colors

To change the color settings look for these two lines in the TextInfoGauge30.xml file:

```
Color="Gold"                Text color for the overall gauge
<Color Value="Goldenrod"/> Color of the mouse spots (small squares in left margin)
```

You can use any color that is supported by HTML. You can use the color name or the color hexadecimal value. Color charts are easily found with an Internet search.

There are many available colors. The fundamental colors supported by the widest variety of platforms are:

black (#000000)	silver (#C0C0C0)	gray (#808080)	white (#FFFFFF)
maroon (#800000)	red (#FF0000)	purple (#800080)	fuchsia (#FF00FF)
green (#008000)	lime (#00FF00)	olive (#808000)	yellow (#FFFF00)
navy (#000080)	blue (#0000FF)	teal (#008080)	aqua (#00FFFF)

Turn Off FSX Screen Messages

FSX has screen messages (e.g. red warnings for brakes, overspeed, etc.) that may appear in the same location as TextInfo Gauge. To turn off the FSX screen messages:

1. Navigate to C:\Users\UserName\AppData\Roaming\Microsoft\FSX
2. Open FSX.cfg in Windows Notepad
3. Locate the section titled [MAIN]
4. Add a line: HideInfoText=1 (if this line already exists then set it to 1).



Always use a plain-text editor (e.g. Notepad) to edit FSX files to avoid introducing hidden control codes.

TextInfo Gauge XML Script Explained

This section provides information for users desiring to modify the script. Some of the examples below are not used in the default configuration of TextInfo Gauge or do not exactly match the default XML script; they are intended to facilitate changes. The script format (indents, spacing, etc.) does not strictly follow XML programming standards; it is formatted to facilitate understanding by nonprogrammers.

Structure of TextInfo Gauge

TextInfoGauge30.xml has a very simple structure. When you look at the file it will appear to be a stack of many small sections. In reality, it is just one long string of commands, followed by mouse click mappings. XML ignores extra spaces and line breaks between commands, which allows formatting the file into a more readable layout. But there is only one `<String>` section, and that is followed by a `<Mouse>` section.

At the top of the data readout script is a `<Visible>1</Visible>` command. This means `Visible = True`. Some aircraft will not display the gauge if this is not declared at a level that wraps around the readout script. This necessitates an extra `<Element>` layer.



Before editing the script make a backup copy of the file. If things go awry you can simply replace the non-working file with the backup copy.

```
<?xml version="1.0" encoding="utf-8"?>
<Gauge Name="TextInfo Gauge" Version="3.0">

  <Element>
    <FloatPosition X="0" Y="0" />
    <Visible>1</Visible>

    <Element>
      <FloatPosition X="0" Y="0" />

      <FormattedText X="60" Y="300"
        <String>
          Sim Rate readout
          Local time readout
          Fuel readout
          Etc.
        </String>
      </FormattedText>
    </Element>
  </Element>

  <Mouse>
    Map mouse click for sim rate setting
    Map mouse click for adding fuel
    Etc.
  </Mouse>
</Gauge>
```

Formatting the display

The script positions the display area then uses the FormattedText command to define the appearance of the readouts.

<code><FloatPosition X="0" Y="0" /></code>	Positions gauge element at top left corner of the panel.
<code><FormattedText X="60" Y="300"</code>	<p>Must match the size_mm setting in the aircraft's panel.cfg file. Because this is a text-based gauge (no graphic file) the text dimensions define the relative size of the gauge.</p> <p>This is NOT the size that will actually be displayed. The display size is controlled with by the <code>window_size_ratio</code> in the aircraft's panel.cfg file.</p> <p>The default TextInfo Gauge script only requires FormattedText dimensions of 60 x 210. Extra height has been included (60 x 300) to allow space for users to add more display elements.</p>
<code>Font="MS Sans Serif"</code>	Must be a font that is available on your computer.
<code>FontSize="8"</code>	Do not change. This is a text-only gauge so the gauge size is defined by the text dimensions. The <code><FormattedText></code> X,Y settings and <code>FontSize</code> in the script, combined with the panel.cfg settings determine the size of the display on your simulation screen, which in turn determines the displayed font size.
<code>LineSpacing="0"</code>	Do not change. This is a text-only gauge that relies on font settings and line breaks to format the gauge.
<code>Adjust="Left"</code>	Left-justify the text.
<code>Color="Gold"</code>	This line defines the text color. It does not define the color of the mouse click spots (those are defined below as <code>Color Value</code>). The text can be any color in standard HTML color charts.
<code>Bright="Yes"</code>	When flying at night FSX uses cockpit lighting to illuminate the gauges. However, this is a text-based gauge so FSX dims the gauge text but the cockpit lighting does not illuminate it. This setting prevents FSX from dimming the text at night.

<pre>Tabs="0L, 7L, 25L, 44R, 45L"</pre>	<p>Sets tabs that are used to align the data display.</p> <p>L = justify left C = justify center R = justify right</p> <p>The <code>%\t</code> command is used to implement the tabs later in the script.</p>
<pre><Color Value="Goldenrod"/> <Color Value="DarkOrange"/> <Color Value="DarkViolet"/></pre>	<p>Alternate colors. The default color is defined in a previous line (<code>Color="Gold"</code>).</p> <p>Specific colors can be called later in the script for specific actions:</p> <p><code>\{clr1}</code> = default color (Gold) <code>\{clr2}</code> = first alternate color (Goldenrod) <code>\{clr3}</code> = second alternate color (DarkOrange) <code>\{clr4}</code> = third alternate color (Dark Violet)</p> <p>Although several alternate colors are shown in this example, TextInfo Gauge only has one alternate color that is used for the mouse click areas.</p>
<pre> </pre>	<p>Alternate fonts. Various parameters can be used, such as Font, FontWeight, FontSize, etc.</p> <p>Note that alternate font numbers differ from alternate colors. The first alternate font is <code>\{fnt1}</code>, whereas the first alternate color is <code>\{clr2}</code>. The default font is invoked with <code>\{fnt}</code> (no number indicated).</p> <p>TextInfo Gauge does not use alternate fonts. This info is provided for users who wish to modify the script using alternate fonts.</p>
<pre>%\n</pre>	<p>Line break command (carriage return). Has the same effect as pressing the Enter key in a word processor.</p>



info

At the end of this document are links to resources that provide all of the commands and variables used in this script.

Draws a horizontal line as a separator between display items.

`%\n\{lsp=4} \{line=59}\{lsp}%\n`

- `%\n` is a carriage return to start a new line.
 - `\{lsp=4}` sets the line spacing to a small number to snug it up against the preceding line. **Note:** The blank space following this command is required or else the new line spacing setting will not work. If the line has no content (in this case, a space character) then it has no work to do and it will use the default line spacing.
 - `\{line=59}` draws a line that is 59 characters long.
 - `\{lsp}` resets the line spacing to the default value.
 - `%\n` finishes the line with a carriage return.
 - Additional escape sequences can be used. For example, `%\n\{lsp=4} \{clr2}\{line=47}\{clr1}\{lsp}%\n` will produce a horizontal line in the first alternate color, then will restore the default color for the items that follow after the horizontal line.
-

Begin displaying the data

Once the display formatting is defined the script begins the data display (the flight data readouts) with the `<String>` command.

Do not include any comments in the script between the `<String>` and `</String>` commands. This will cause the script to fail.

The items within the `<String>` can be moved around. For example, airspeed can be moved so that it is displayed above wind speed.



If you pop up the gauge in the simulation and you see only a momentary flash, as if the gauge attempted to display but then did not, then the most likely reason is a syntax error in the script.

Examples of errors:

- Missing closure tag (e.g. `<Element>` without a final `</Element>`)
- Missing quotation marks (e.g. `Color = "Blue)`)
- Unsupported elements (e.g. including a comment within a `<String>` element)

{if} statements

TextInfo Gauge uses a lot of {if} statements. To nonprogrammers they can be daunting, but the concept is actually very simple:

```
If (something) is True then perform an action,  
Otherwise, do something else,  
Then end this part of the script.
```

For example,

```
If the aircraft is in a stall condition then display a warning message,  
Otherwise, display "OK",  
Then we are done.
```

The script for this example is:

```
%((A:Stall Warning,bool) 1 == ){if}  
%( 'STALL' )%!s!  
{else}  
%( 'OK' )%!s!  
{end}
```

The initial question in an XML {if} statement is read from right to left. The remainder of the script reads in a normal left-to-right manner.

In this example, if [%{if}] the aircraft variable for a stall warning [A:Stall Warning] is True [1 ==], then display a string that says "STALL" [%('STALL')%!s!]. Otherwise [%{else}], display a string that says "OK" [%('OK')%!s!]. After we display something then we are done [%{end}].

Nested `{if}` statements are `{if}` statements that are layered, one within another.

```
If (something) is True then perform an action,  
    Otherwise, if something else is True then perform another action,  
        Otherwise, if yet another thing is True then do another action,  
        Then end this part of the script.  
    Then end this part of the script.  
Then end this part of the script.
```

The easiest way successfully interpret a nested `{if}` statement is to separate the layers into understandable terms. Look at each layer separately rather than trying to understand it all at once by simply reading from top to bottom.

If you get weird results with nested `{if}` statements the first thing to check is that the number of `{end}` statements match the number of `{if}` statements. Then check how they are layered to determine that the `{if}` and `{end}` statements wrap around one another (good) rather than overlapping each other (bad).

Display simple text and data

This example displays "TAS" then tabs to the next tab setting and displays the True Airspeed readout. Technically, the double parentheses around the variable are not required for a simple readout that has no additional arguments. However, testing yielded mixed results when some variables were used with single parentheses. It is recommended that you use double parentheses.

```
%( 'TAS' )%!s!\t%((A:Airspeed True, knots))%!\-d!%
```

Example: TAS 120 kts

Display text and two data readouts in the same line

In this example "Wind" is followed by the wind speed readout, then "@", then by wind direction followed by "mag". This example is on two separate lines but the code can all be on a single line.

```
%( 'Wind ' )%!s!%((A:Ambient Wind Velocity, knots))%!\-d!  
%( ' @ ' )%!s!%((A:Ambient Wind Direction, degrees))%!\-d!%( ' mag' )%!s!
```

Example: Wind 14 @ 245 mag

Text and data with a plus/minus sign in front of the data

This example adds a +/- sign in front of the data by adding a plus sign to the formatting code. Instead of using %!-d! it uses %!+d!

```
%( 'VSI ' )%!s!%(A:Vertical Speed, feet per minute))%!+d!
```

Example: VSI +850

Percentage calculation

This example calculates the % of remaining fuel.

```
%( 'Fuel ' )%!s!%(A:Fuel Total Quantity, gallons) (A:Fuel Total Capacity, gallons)
/ 100 *)%!-d!%( ' %' )%!s!
```

Example: Fuel 78 %

Display an item only if it is engaged, and use an alternate color

In this example the parking brake only appears if it is engaged, displayed in an alternate color.

If the first color (default color) is defined at the beginning of the script as Blue in <FormattedText> then it is invoked as \{clr1}. If an alternate color is defined as Red in <Color Value="Red"/> then it is invoked as \{clr2}.

Note that there is an {if} statement but no {else} statement. If the result is true then it displays the word "Brake." If the result is false there is nothing to do, so the logic simply ends, resulting in nothing being displayed.

```
%(A:Brake Parking Indicator,bool) 1 == )%{if}
\{clr2}Brake\{clr1}%{end}
```

Example: TAS 120 kts
 Fuel 78 %
 Brake
 Flaps 50 %

Display an item only if the aircraft is equipped with the item

This example tests the "Flaps Available" variable. If it is true, then the remainder of the script line is displayed ("Flaps" and the % extended). Note there is a space after "Flaps" and before "%." This results in a readable appearance in the display (e.g. "Flaps 50 %" instead of "Flaps50%").

```
%(A:Flaps Available,bool) 1 == )%{if}
%( 'Flaps ' )%!s!%(A:Flaps Handle Percent, percent))%!d!%( ' %' )%!s!
%{end}
```

Example: Flaps 33 %

Change the color of the readout based on an either/or condition

In this example, if the pitch is negative it is displayed in the default color (`\{clr1\}`). If the pitch is positive it is displayed in another color (`\{clr2\}`). Let's assume that we defined the default color as red and defined an alternate color as green.

```
%( 'Pitch ' )%!s!%(A:Plane Pitch Degrees, degrees) 0 > )%{if}
%(A:Plane Pitch Degrees, degrees) -1 * )%!-+d!%{else}
\{clr2\}%(A:Plane Pitch Degrees, degrees) -1 * )%!-+d!\{clr1\}%{end}
```

Example: Pitch -15
 Pitch +23

Blinking warning indicator

In this example a blinking stall warning is displayed during a stall. If the aircraft is not stalled then nothing is displayed because there is no `{else}` statement following the `{if}` statement.

`\{blnk\}` and `\{nr\}` are "escape sequences" that are listed in the FSX SDK reference content (see the links to the SDK content provided at the end of this document).

```
%(A:Stall Warning,bool) 1 == )%{if}
\{blnk\}STALL\{nr\}
%{end}
```


Use multiplier for number rounding for less rapid changes in readout

If a digital vertical speed indicator (VSI) displays info in feet per minute (e.g. 823 FPM) then it will have a constantly-changing data readout, even on autopilot. The data may be changing at a frantic pace. It is much easier to read when the information is provided as hundreds of feet per minute (e.g. 800 FPM).

In this example the script first checks to see if the aircraft is on the ground. If so, then it simply displays, "VSI 0 fpm."

Otherwise, it reads the data, multiplies it by 0.01, then appends "00." Using a number format of %!-+d! provides a number that indicates -/+ for descending or ascending data.

```
%( 'VSI ' )%!s!  
%((A:SIM ON GROUND,bool) 1 == )%{if}  
( '0' )%!s!%( ' fpm' )%!s!%{else}  
%((A:VERTICAL SPEED, feet per minute) 0.01 * )%!-+d!%( '00' )%!s!%( ' fpm' )%!s!  
%{end}
```

Only display if aircraft has item AND item is engaged AND change readout color based on condition

In this example the readout will be blank if the aircraft is not equipped with retractable landing gear. It will also be blank if the gear is fully retracted. When the gear is in the process of extending or retracting there will be a real-time display of the % that the gear is extended in an alternate color (\{clr3}). When the gear is fully extended it will display "Gear Down" in another alternate color (\{clr2}). Whether or not an aircraft has certain equipment is defined in each aircraft's aircraft.cfg file.

```
%(A:Is Gear Retractable,bool) 1 == )%{if}  
%(A:Gear Position:2,enum) 1 == )%{if}  
\{clr2}Gear Down\{clr1}%{else}  
%((A:Gear Total Pct Extended, percentage) 0 > )%{if}  
\{clr3}Gear %(A:Gear Total Pct Extended, percentage) 100 * )%!-d!  
%( ' %' )%!s!\{clr1}  
%{end}%{end}%{end}
```

Here's how this script works:

<pre>%(A:Is Gear Retractable,bool) 1 ==)%{if}</pre>	<p>Does the aircraft have retractable landing gear? If so, then go to the next line.</p> <p>If not, then do nothing and go to the end (% {end}).</p>
<pre>%(A:Gear Position:2,enum) 1 ==)%{if}</pre>	<p>We are on this line because the previous line determined that the aircraft has retractable gear. Next question: "Is the gear down?" If so, then go to the next line.</p>
<pre>\{clr2}Gear Down\{clr1}%{else}</pre>	<p>If the condition in line 1 is true (aircraft has retractable gear) and line 2 is true (the gear is down), then this line is displayed.</p> <p>The line invokes the first alternate color, then displays "Gear Down", then invokes the default color (otherwise everything that follows will also be in \{clr2}).</p> <p>If the gear is not down then the script skips this line and goes to the next line.</p>
<pre>%(A:Gear Total Pct Extended, percentage) 0 >)%{if}</pre>	<p>We reached this line because it was determined that the aircraft has retractable gear, but the gear is not down. That leaves us with two possibilities: either the gear is all the way up (retracted) or it is in the process of going up or down.</p> <p>So the script asks another question, "Is the gear extended more than 0%?" If so, then it is not all the way up (0% extended) and thus must be in motion because we already determined that it is not down.</p>
<pre>\{clr3}Gear %(A:Gear Total Pct Extended, percentage) 100 *)%!-d! %(' %')%!s!\{clr1}</pre>	<p>If the statement above is true then the gear is in motion. This line shows the % the gear is extended, using alternate color \{clr3}.</p> <p>If \{clr3} is blue then it will give a real-time display of the % progress of the gear in motion, in a blue font. It displays a percentage symbol after the data. The final piece is \{clr1} to ensure that everything following this line uses the default color.</p>
<pre>%{end}%{end}%{end}</pre>	<p>The number of %{end} statements must equal the number of IF statements.</p>

Display heading in degrees and also abbreviated direction (N, W, S, E)

This example begins with a simple data display of "Heading" followed by the heading readout. The subsequent lines are nested `{if}` statements that determine which direction to display based on the ranges of compass degrees.

Note that the abbreviated directions (N, W, S, E) do not use the formal string display script of `%('xxx')%!s!`. In some instances formatting the text is not required.

```
%( 'Heading ' )%!s!%(A:PLANE HEADING DEGREES TRUE, degrees))%!-d!
%( 0 22 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} N%{else}
%( 23 67 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} NE%{else}
%( 68 112 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} E%{else}
%( 113 157 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} SE%{else}
%( 158 202 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} S%{else}
%( 203 247 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} SW%{else}
%( 248 292 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} W%{else}
%( 293 337 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} NW%{else}
%( 337 360 (A:PLANE HEADING DEGREES TRUE, degrees) rng )%{if} N
%{end}%{end}%{end}%{end}%{end}%{end}%{end}%{end}%{end}%{end}
```

Example: Heading 125 SE

Use one line for various readouts, with color changes and flashing text, as needed

This example provides multiple functions in the same line, based on conditions. At any specific point in time only one of the five conditions listed below is displayed. Selecting which data will be displayed on the line is prioritized from most-immediate hazard to least-immediate hazard.

The reason for prioritizing from worst-to-best is because of the nature of `{if}` statements. Once an `{if}` statement is satisfied it takes the specified action and skips all of the subsequent `{else}` statements. Thus, if the top `{if}` statement determines that the aircraft is in a stall condition then it will ignore all of the remaining `{else}` conditions and you will see a stall warning. If you aren't in a stall, and you aren't over-speeding, but you have less than 5% fuel then it will warn you about the fuel and will skip the remaining `{else}` statements that follow the 5% fuel warning.

1. If in a **stall** condition then provide a flashing stall warning in the default color,
else
2. If in an **overspeed** condition then provide a flashing overspeed warning in the default color,
else
3. If remaining **fuel is less than 5%** then display remaining fuel % in a flashing alternate color,
else
4. If remaining **fuel is less than 10%** then display fuel % in an alternate color (not flashing),
else
5. If remaining **fuel is more than 10%** then simply display remaining fuel % in the default color.

The fuel warning limits are set at 5.5 and 10.5, because FSX rounds the numbers. If the readout is 11% it will change to 10% when the figure drops to 10.5%. Thus, using 10.5 (instead of 10) results in font color changes as soon as the displayed number changes, not delayed until FSX reaches the rounding point.

Note that conditional math in the fuel script is *very* sensitive to spacing and formatting. Make only one change at a time and test it, or insanity may result when you try to sort out what went wrong. For this reason it is better to copy this code from the script file than to copy it from this document.

```
%((A:Stall Warning,bool) 1 == )%{if}\{blnk}      STALL \{nr}%{else}
%(A:Overspeed Warning,bool) 1 == )%{if}\{blnk} OVERSPEED \{nr}%{else}
%( ' Fuel' )%!s!\t
%((A:Fuel Total Quantity, gallons) (A:Fuel Total Capacity, gallons) / 100 * 5.5
&lt;= )%{if}
\{clr5}\{blnk}
%((A:Fuel Total Quantity, gallons) (A:Fuel Total Capacity, gallons) / 100 *)%!-d!
\{nr}\{clr1}%{else}
%((A:Fuel Total Quantity, gallons) (A:Fuel Total Capacity, gallons) / 100 * 10.5
&lt;= )%{if}
\{clr5}
%((A:Fuel Total Quantity, gallons) (A:Fuel Total Capacity, gallons) / 100 *)%!-d!
\{nr}\{clr1}%{else}
%((A:Fuel Total Quantity, gallons) (A:Fuel Total Capacity, gallons) / 100 *)%!-d!
%{end}%{end}%{end}%{end}
%\t%( ' %' )%!s!%
```

Display several indicators in the same line at the same time

This example displays three indicators (landing light, parking brake and autopilot engaged) in a single line. It uses an `{if}` statement for each item to determine if the item will be displayed. It uses `%\t` tabs rather than `%\n` line breaks to place the items in the display.

```
%(A:Light Landing,bool) 1 == )%{if}
\{clr2} Light\{clr1}
%{end}
%\t
%(A:Brake Parking Indicator,bool) 1 == )%{if}
\{clr3} Brake\{clr1}
%{end}
%\t
%(A:Autopilot Available,bool) 1 == )%{if}
%(A:Autopilot Master,bool) 1 == )%{if}
\{clr2}Auto\{clr1}
%{end}%{end}
```

Example: Light Brake Autopilot

Mouse click spots in a transparent text gauge that has no graphic

Providing mouse click spots in a transparent text gauge is a challenge. If a word is mapped as a clickable item then it is possible to click on the text but you will need to click on the text. Specifically, directly on a text character. If you don't click exactly on a letter then nothing happens because the background behind the letter is transparent.

The solution is to provide something non-transparent for the mouse to click on. It does not have to be a graphic. Anything that covers the transparency will work.



One solution is the `\{rev}` escape code. This code reverses the text so that the text character is the color of the background, and the font background is the color of the text:

Normal text: Example

Reversed text: Example

In the example above, if the background is transparent then the letters in the reversed text will be transparent. If you land directly on a letter your click will be ignored because the letter is now a transparent to match the simulation background. But clicking anywhere on the red background will be successful.

Create a solid landing place by using the underscore character:

Normal text: 
Reversed text: 

The example above has underscore characters that will be transparent, but the odds are much better that you will click on a solid spot because most of the background is solid.

Here's an example of a script that uses this approach:

In the `<String>` section of the script:

<code>\{rev}%('___')%!s!\{nr}</code>	<code><- Create click box</code>
<code>%(' Parking Brake is ')%!s!</code>	<code><- Start text line</code>
<code>%(A:BRAKE PARKING INDICATOR, bool) 1 ==)%{if}</code>	<code><- Is the brake on?</code>
<code>%('On')%!s!%{else}</code>	<code><- If on, display "On"</code>
<code>%('Off')%!s!</code>	<code><- Else, display "Off"</code>
<code>%{end}</code>	

How it will appear on the screen:  Parking Brake is Off

In the `<Mouse>` section of the script:

```
<Area Left="0" Top="45" Width="3" Height="8">
<Cursor Type="Hand"/>
<Click Event="PARKING_BRAKES"/>
</Area>
```

If the `<Mouse>` settings for "Left" and "Top" match the location of the `<String>` red square in the gauge then clicking on the red square will result in toggling the parking brake on and off.

This approach also works for settable items. For example:

In the `<String>` section of the script:

```
\{rev}% ( '____' )%!s!\{nr}  
%( ' Autopilot Altitude Setting ' )%!s!  
%(A:AUTOPILOT ALTITUDE LOCK VAR, feet))%!d!  
%\t%( ' feet' )%!s!
```

How it will appear on the screen:  Autopilot Altitude Setting 1500 feet

In the `<Mouse>` section of the script:

```
<Area Left="0" Top="200" Width="3" Height="8">  
<Cursor Type="DownArrow"/>  
<Click Event="AP_ALT_VAR_DEC"/>  
</Area>  
  
<Area Left="3" Top="200" Width="3" Height="8">  
<Cursor Type="UpArrow"/>  
<Click Event="AP_ALT_VAR_INC"/>  
</Area>
```

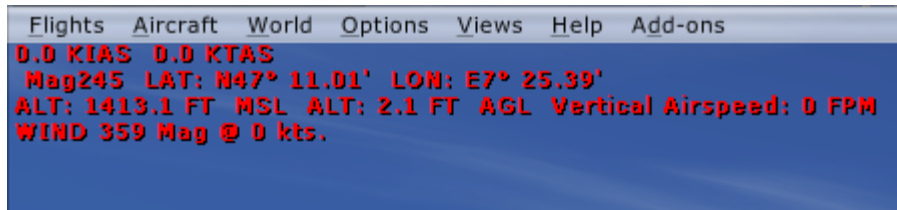
The two mouse scripts are at the same vertical level (Top=200). Thus, they will both be in the same display line.

They are also placed side by side to create the appearance of a single box. They both have the same width ("3") but they have different offset settings from the left margin. The first one is Left=0 and the second one is Left=3. Thus, the first box occupies spaces 0-2 and the second box occupies spaces 3-5, for a total horizontal length of 6 (0 through 5).

The result of this example is that when you mouse over the red block you can scroll the mouse wheel to change the altitude setting.

FSX Shift-Z Feature

Pressing Shift-Z in Flight Simulator X toggles a simple text display of flight data at the top of the screen. Although it has a limited catalog of display variables and almost no formatting capability it is nonetheless a very useful feature.



Shift-Z is a program-level function. It can be invoked for any aircraft at any time. This is handy when you download aircraft and want a quick on-screen display of airspeed and altitude while testing the aircraft.

Shift-Z can display some data that is not available as simulation variables. In other words, there are a few things that Shift-Z can display that cannot be displayed in an aircraft gauge. The most popular Shift-Z readouts are the `FrameRate` readouts.

Configuring Shift-Z: Parameters

Shift-Z parameters are in the `fsx.cfg` file. The default `fsx.cfg` file is located at `C:\Program Files (x86)\Microsoft Games\Microsoft Flight Simulator X\fsx.cfg`.

However, [the working file](#) is at `C:\Users\Username\AppData\Roaming\Microsoft\FSX\fsx.cfg`. FSX uses the working file. It reverts to the default file if it can't find the working file.

In the `fsx.cfg` the Shift-Z parameters are in the `[TextInfo.1]`, `[TextInfo.2]` and `[TextInfo.3]` sections. Each time you press Shift-Z it will cycle through these sections. Pressing Shift-Z while the last section is displayed results in turning off the Shift-Z display. Pressing Shift-Z once again displays the first `[TextInfo]` section.

Listed within each section are the readout parameters. Each parameter is followed by the line number and the position number of the readout. Here are the settings in the default fsx.cfg file:

```
[TextInfo.1]           // Section to display the first time Shift-Z is pressed
Latitude=1,1           // Line 1, item 1
Longitude=1,2          // Line 1, item 2
Altitude=1,3           // Line 1, item 3
Heading=1,4
AirSpeed=1,5
WindDirectionAndSpeed=1,6
[TextInfo.2]           // Section to display the second time Shift-Z is pressed
FrameRate=1,1
LockedFrameRate=1,2
GForce=1,3
FuelPercentage=1,4
[TextInfo.3]           // Section to display the third time Shift-Z is pressed
Latitude=1,1
Longitude=1,2
Altitude=1,3
Heading=1,4
AirSpeed=1,5
WindDirectionAndSpeed=1,6
FrameRate=2,1          // Begin line 2 of the display
LockedFrameRate=2,2
GForce=2,3
FuelPercentage=2,4
```

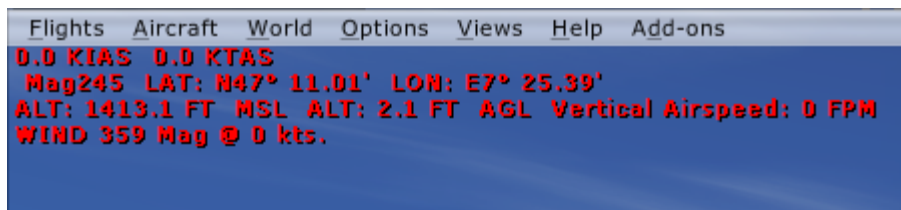
You can have as many [TextInfo] sections as you wish. Reportedly, Shift-Z can display up to 6 lines of information. The author has never been able to display more than 5 lines.

When running the simulation in window mode Line 1 is displayed on the same line as the FSX menu (Flights Aircraft World). If you display the FSX menu and Shift-Z line 1 at the same time then the first line of Shift-Z will be displayed behind the FSX menu line (i.e. unreadable).

Thus, if you are in window mode you may be limited to 4 lines of information because line 1 is behind the FSX menu and because line 6 may not be displayed at all.

To avoid the issue of line 1 displayed behind the FSX menu in window mode begin the Shift-Z configuration with line 2:

```
[TextInfo.1]
AirSpeed=2,1
TrueAirspeed=2,2
Heading=3,1
Latitude=3,2
Longitude=3,3
Altitude=4,1
AltitudeAgl=4,2
VerticalSpeed=4,3
WindDirectionAndSpeed=5,1
```



In each [TextInfo.x] section the variables can only be used once. If a variable is included a second time within the same [TextInfo.x] section then it will be ignored, but the line itself will still be displayed. If a particular line includes no unique variables (i.e. only uses variables from other lines) then you will see a blank line on the screen. The line space is there, but no data is displayed.

If you do not want to display a [TextInfo] section or a specific line within a [TextInfo] section, but you do not want to delete the information, then comment it out by adding "//" at the beginning of the line:

// [TextInfo.3]		[TextInfo.3]
// Altitude=1,1		Altitude=1,1
// AltitudeAgl=1,2	or	AltitudeAgl=1,2
// Heading=3,1		// Heading=3,1
// AirSpeed=4,1		AirSpeed=4,1

Configuring Shift-Z: The Variables

Altitude	GForce	WindSpeed	Latitude
AltitudeAgl	FuelPercentage	WindDirection	LatitudeDec
Heading	FuelRemainingGallons	VideoDevice	LatitudeHex
HeadingHex	FuelRemainingPounds	FrameRate	Longitude
HeadingTrue	VerticalSpeed	AverageFrameRate	LongitudeDec
Airspeed	AngleOfAttack	LockedFrameRate	LongitudeHex
TrueAirspeed	WindDirectionAndSpeed		

Configuring Shift-Z: Font Color

You can specify the colors for the text and the background of the Shift-Z display by including these parameters within a `[TextInfo]` section:

```
TextColor.1=255,255,255
BackColor.1=0,0,0,128
```

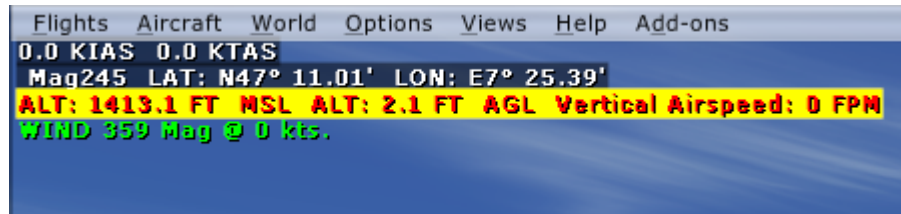
Here is how they are parsed:

Item	TextInfo Section	RGB Color Numbers	Alpha Channel Color Number
TextColor	.1	255,255,255	n/a
BackColor	.1	0,0,0	128

The alpha channel color number defines the transparency of the background. Any number between 0-255 can be used.

0	Transparent background
128	Semi-transparent background
255	Opaque (solid) background

Here is an example of coloring Shift-Z:



And here are the settings that defined the colors:

```
[TextInfo.1]
```

```
AirSpeed=2,1
```

```
TrueAirspeed=2,2
```

```
TextColor.2=255,255,255          // white text
```

```
BackgroundColor.2=0,0,0,128      // black semi-transparent background
```

```
Heading=3,1
```

```
Latitude=3,2                    // no color settings; uses the same color
```

```
Longitude=3,3                   // settings as the previous line
```

```
Altitude=4,1
```

```
AltitudeAgl=4,2
```

```
VerticalSpeed=4,3
```

```
TextColor.4=255,0,0            // red text
```

```
BackgroundColor.4=255,255,0,255 // yellow opaque background
```

```
WindDirectionAndSpeed=5,1
```

```
TextColor.5=0,255,0            // green text
```

```
BackgroundColor.5=128,128,128,0 // gray transparent background; can use any color
```

```
// for the background because it is transparent
```

```
// so the background color will not be seen
```

Helpful Resources

Microsoft Resources

Creating XML Gauges: <http://msdn.microsoft.com/en-us/library/cc526953.aspx>

FSX data variables (status): <http://msdn.microsoft.com/en-us/library/cc526981.aspx>

FSX GPS variables (status): <http://msdn.microsoft.com/en-us/library/cc526954.aspx>

FSX key events (do actions): <http://msdn.microsoft.com/en-us/library/cc526980.aspx>

FSX panel.cfg configuration info: <http://msdn.microsoft.com/en-us/library/cc526956.aspx>

FSX aircraft.cfg configuration info: <http://msdn.microsoft.com/en-us/library/cc526949.aspx>

FSX cameras.cfg configuration info: <http://msdn.microsoft.com/en-us/library/cc526984.aspx>

Other Resources

Color chart: http://www.w3schools.com/html/html_colornames.asp

The definitive forum for FSX programming: <http://www.fsdeveloper.com/>