«Easy Gauge» User Manual

Content

This “Easy Gauge” user manual provides the rules on how to create new and modify existing “Easy Gauge” instruments by editing the instrument specification file.

First check the ‘Requirements and Features’ section to find out if “Easy Gauges” are what you are looking for.

The section ‘How to Create a New “Easy Gauge” instrument’ is a Quick Start Guide for the newcomer.

A newcomer should also read the ‘“Easy Gauges” Concept’ section to familiarize himself with the terminology used throughout this document.

The remaining sections are just more detailed descriptions of “Easy Gauge” features.

There are some advanced options described in the “Advanced Features” section that require some LUA knowledge.

Document version 1.0.0-b2

Contact

GitHub Account: Cr4O13

SimInnovations Forum: Tetrachromat

Requirements and Features

To create or modify “Easy Gauges” instruments you need the following resources:

- Air Manager app and license
- A flight simulator app
- “Easy Gauges” release assets
- External editor

Air Manager App and License

If not already installed, download the latest version from the SimInnovations WIKI website. A “HOME USE” license must be purchased to be able to run and/or create personal instruments.

Flight Simulator

It is planned that “Easy Gauges” can be used with all flight simulators supported by Air Manager.

In the first release of “Easy Gauges” Microsoft Flight Simulator 2020 (FS2020) is the simulator of choice. X-Plane 12 will be the next candidate sim. FSX and P3D may follow.

“Easy Gauge” Release Assets

To create and run “Easy Gauge” instruments you need at least the “Easy Gauge” release code. The code is part of the release asset from my GitHub account.

Download Release Asset

My ”Easy Gauges” GitHub account is https://github.com/Cr4O13/easy-gauges. This is a public repository with the development code and additional resources. The code is maintained using a LUA development environment. As such the shown code cannot be executed directly in Air Manager.

Instead, I will provide stable builds in the “Releases” section on https://github.com/Cr4O13/easy-gauges/releases

From the ‘Assets’ drop down list choose the ”Easy Gauges” TAR file. It is named something like

```
easygauge-<release>-<build>.tar
```

Where <release> is the release number (e.g., 1.0.0) and build is the build type (currently ‘home’ only).

Sometimes I will release development releases named something like

```
easygauge-1.0.0-home-beta-1.tar
```

Development builds should be stable but may provide features that will change until the final stable build is released.

Always download the assets from the latest stable build if you start out as a new user.

The Asset File

The easygauge TAR file is the only asset needed. It contains all the required resources to create “Easy Gauge” instruments. The file is a zipped archive of the resources in “tar” format. I recommend 7-Zip as the archiving application to extract the content, but you can choose your preferred extractor if it supports the “tar” format.

The archive contains

- A sample instrument LUA script file ‘logic.lua’
- A ‘lib’ folder with
  - The built LUA code file ‘easygauge.lua’
- A ‘resources’ folder with
  - A specification template JSON file ‘easygauge.json’
- A ‘docs’ folder with
  - This user manual as a PDF file ‘Easy Gauges User Manual.pdf’
  - A reference manual as a PDF file ‘Easy Gauges Reference.pdf’
  - A Snippets file ‘Easy Gauges Snippets.sqlite’ for import into the ‘Notepad++’ snippets plugin.

The folder structure is the same as the folder structure of any Air Manager instrument. If you create a new personal instrument in Air Manager and extract the content of the tar file into the instrument root folder, you have all resources in place for creating a "Easy Gauge" instrument.

The 'docs' folder is not a required folder for "easy Gauge" instruments, you should extract or move the folder to somewhere else to save some space in the instrument.

"Easy Gauge" Specifications

Creating a "Easy Gauge" instrument does not need any knowledge in LUA programming.

To create a "Easy Gauge" instrument you just need to specify the instrument features in the 'easygauge.json' file.

The specification file uses the JSON object syntax to specify the instrument. See the section 'JSON Syntax' at the end of this document for a short introduction to the syntax.

There is a collection of ready-to-run "Easy Gauge" instrument specifications available in my GitHub repository folder "instruments". Just download them and try some out.

There is also a "Easy Gauge" reference manual available on GitHub. The reference document gives the complete rules for the JSON objects, fields, and values that are needed.

External editor

In the current Air Manager release you cannot directly edit the "Easy Gauge" instrument specification file. Therefore, an external editor is required to create and edit the "easygauge.json" files in the "resources" folder.

I requested that the next release of Air Manager (4.2) supports editing of text files (.json, .txt and .csv type files) in the "resources" folder.

In the meantime, I recommend "Notepad++" as the external editor. But you can use your preferred editor.

In "Notepad++" you can use my "Easy Gauges Snippets" file from the downloaded asset file. Just, make sure you have the "Snippets" plugin activated, import the 'Easy Gauges Snippets.sqlite' file in Notepad++.

"Easy Gauge" Instrument Features

Current Features

With "Easy Gauges" you can currently create instruments with some or all the following features:

Static features:

- A background image,
- A canvas to show
  - Any number of 'circular' scales with specified center position.
  - Each scale with any number of sections with different scaling.
  - Each scale section with specified divisions limited by major scale markings (optionally invisible).
  - Each section division with subdivisions separated by minor scale markings.
  - Scale values added to selected (visible) major scale marks in selected position and style.
  - Any number of scale arcs drawn in selected position and width.
  - Each scale arc with any number of arc segments
  - Each segment with a specified color (optionally invisible).
  - Any number of redline markings in selectable colors.
  - Any number of markup text in selected position and style.
- A cover image to hide some canvas areas or add glass covers
- A bezel image to add graphic features (logo, mounting rings, screws and the like)

Dynamic features

- Any number of indicating gauges on one instrument
- Each gauge with
  - a subscription to a simulator (MSFS2020) or Air Manager (SI) variable
  - a 'rotating' indicator image to indicate the values received on the assigned scale
  - some animation features

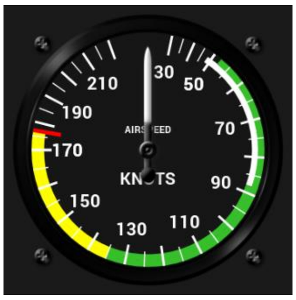
Planned features

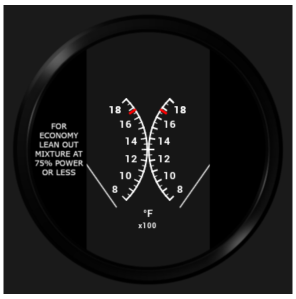
In coming releases of “Easy Gauges” the following features will follow:

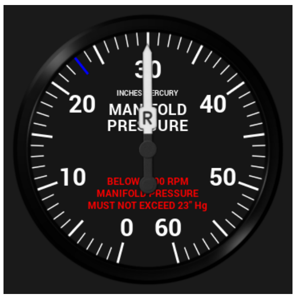
- Horizontal and Vertical Scales,
- Subscriptions to other simulators supported by Air Manager (X-Plane 12, FSX, Prepar3D),
- Some simple user input controls (dial, toggle buttons),
- Display digital values, and
- If feasible, some requested features by the Air Manager community.

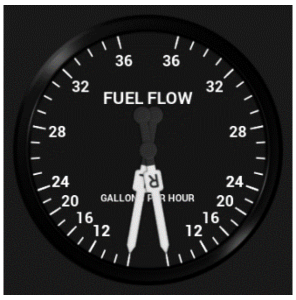
Sample Instruments

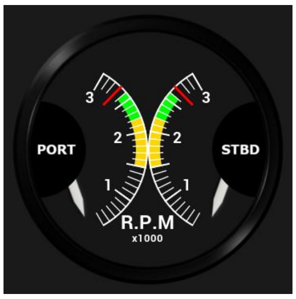
Below you can see a collection of “Easy Gauge” instruments (specification files available on my GitHub account).

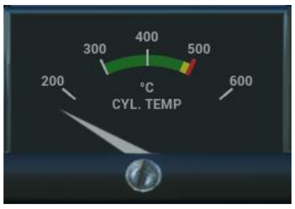


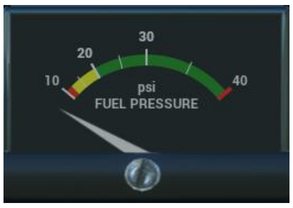


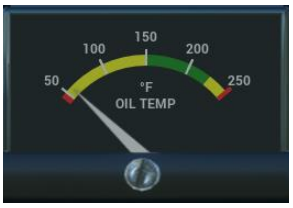


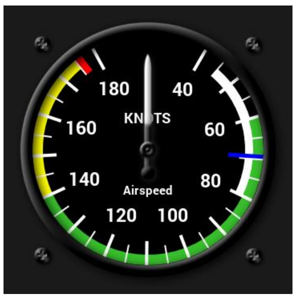


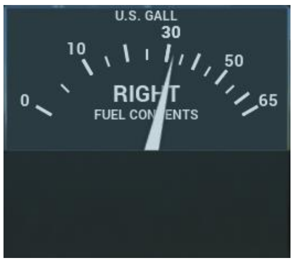


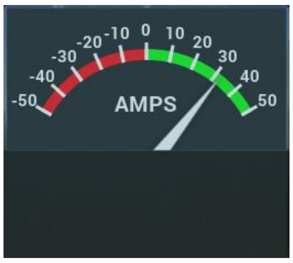


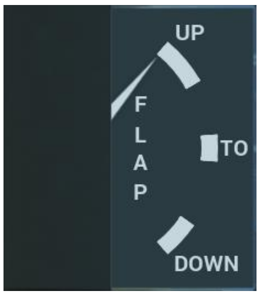








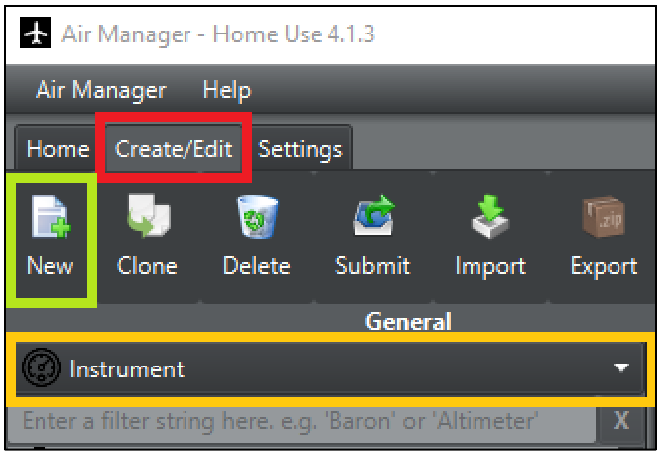




How to Create a New “Easy Gauge” Instrument

This is the “Quick Start Guide” to “Easy Gauges”.

Open Air Manager:



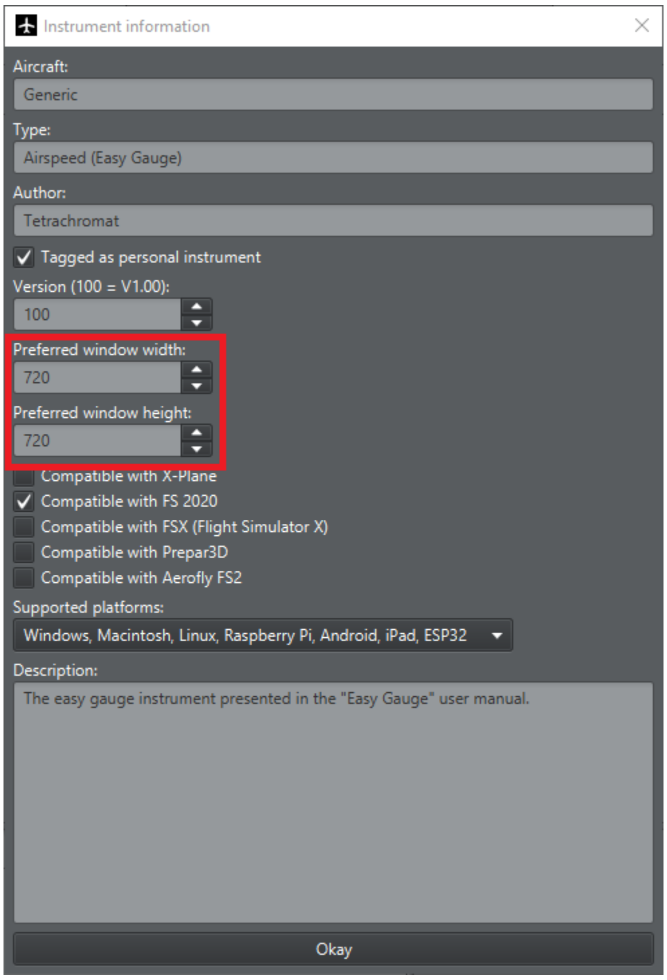
“Easy Gauge” instruments are basically regular personal Air Manager instruments. You create them with the following steps:

Select the “Create/Edit” tab

Make sure the “Instrument” list is selected

Select the “New” option.

The “Instrument information” dialog opens.



Instrument information

Aircraft:
Generic

Type:
Airspeed (Easy Gauge)

Author:
Tetrachromat

✓ Tagged as personal instrument

Version (100 = V1.00):
100

Preferred window width:
720

Preferred window height:
720

☐ Compatible with X-Plane
✓ Compatible with FS 2020
☐ Compatible with FSX (Flight Simulator X)
☐ Compatible with Prepar3D
☐ Compatible with Aerofly FS2

Supported platforms:
Windows, Macintosh, Linux, Raspberry Pi, Android, iPad, ESP32

Description:
The easy gauge instrument presented in the "Easy Gauge" user manual.

Okay

Set the attributes as you would do for a regular personal AM instrument.

‘Aircraft’ and ‘Type’ are used to identify the instrument in the “Instrument” list.

Leave it tagged as personal instrument.

The two settings to remember are the **Preferred window width and height**.

In this example, they are set to create a square instrument with a side length of 720 pixels.

But you can use any width or height values that fit your instrument.

Close the dialog using the ‘Okay’ button at the bottom.

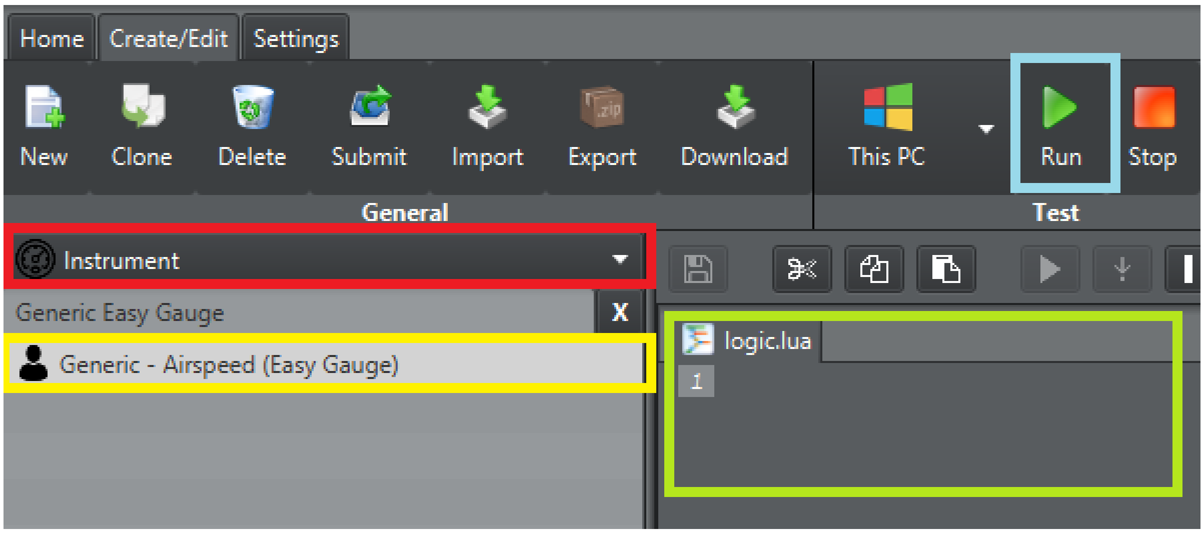
In the background, Air Manager creates a folder structure for the new instrument and some basic files

HINT: The instrument folder is created in

```
        C:\Users\<username>\Air Manager\instruments\OPEN_DIRECTORY\
```

Using a generated unique UUID as the folder name.

In Air Manager the new instrument is shown in the "Instrument" list. You can use the filter box to reduce the number of listed instruments.



When you select your new instrument, the editor pane on the right shows the LUA script file 'logic.lua' with an empty line (line number '1').

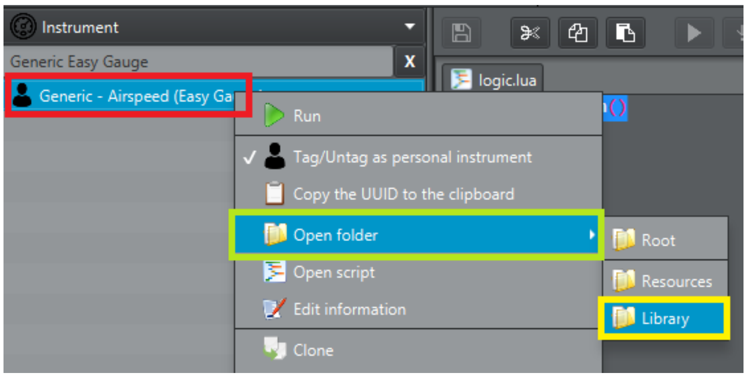
If you "Run" the instrument, Air Manager will display an empty white window in the preferred size.

Add The "Easy Gauge" library

For a regular instrument you would now start programming in LUA. Programming means writing some LUA code in the editor pane, which will be saved in the 'logic.lua' file.

But for a "Easy Gauge" instrument you don't need to write LUA code. All required code is already written for you. This code is contained in a file "easygauge.lua". The file can be downloaded from my GitHub account. See the 'Requirements' section at the top of this document.

Now, if not already done so, download the latest release assets, and place the file "easygauge.lua" into the Instrument "lib" folder.



You can open the "Library" folder from the context menu.

Drop the file into the folder and close the folder window.

If you intend to export instruments for import into another Air Manager installation or push them to an Air Player instance, the "easygauge.lua" library file in the 'lib' folder is included in the export / push.

Alternative

You can drop the file into the "lua_libs" folder in the Air Manager installation directory. This has the advantage that you only need to update the library file in one location when a new release of the "easygauge.lua" library is available. Otherwise, you would need to update every instance of the file in all instruments requiring the update.

But you need to copy it over to another Air Manager / Air Player installation directory manually, when exporting / pushing instruments.

If you “Run” the instrument now, Air Manager will again just display an empty white window in the preferred size.

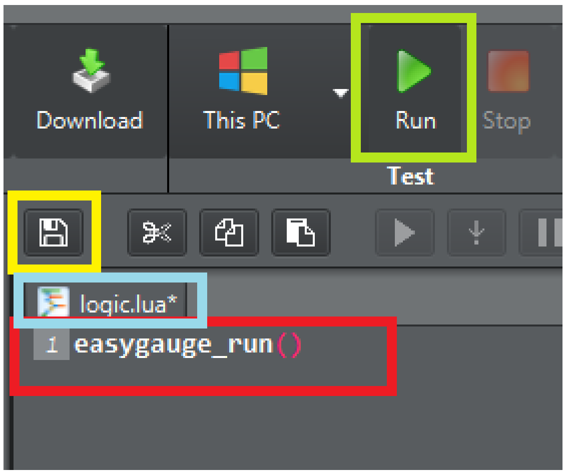
By intention, the library code is not executed automatically when the instrument is “Run”.

This behavior leaves the usage of the library code under the control of the user. This is important especially when the library code is placed in the “lua_libs” folder in the Air Manager installation directory.

Additionally, it allows to use some advanced features, described at the end of this document.

Add One Line of Code

To use the library code, you need to insert just one line of code to the ‘logic.lua’ file.



```
easygauge_run()
```

Enter the following code at line ‘1’

```
easygauge_run()
```

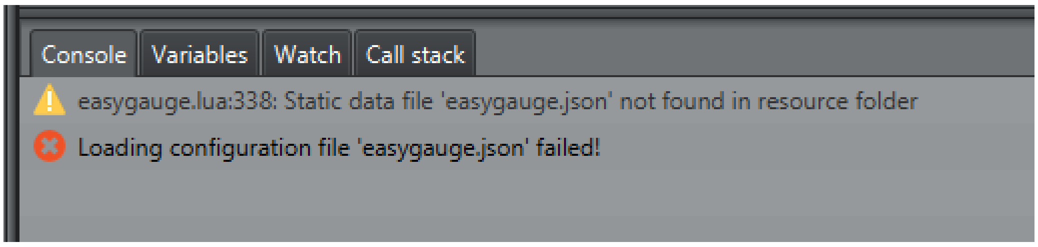
Save the changes to the instrument code.

Whether there are changes to the code or not is indicated with the “star” in the script tab (logic.lua*).

HINT: Pressing the “Run” button, also saves any changes in the code editor.

If you “Run” the instrument now, Air Manager will again just display an empty white window in the preferred size.

But in the “Console” pane below the editor pane, you will notice two messages, one warning and one error message:



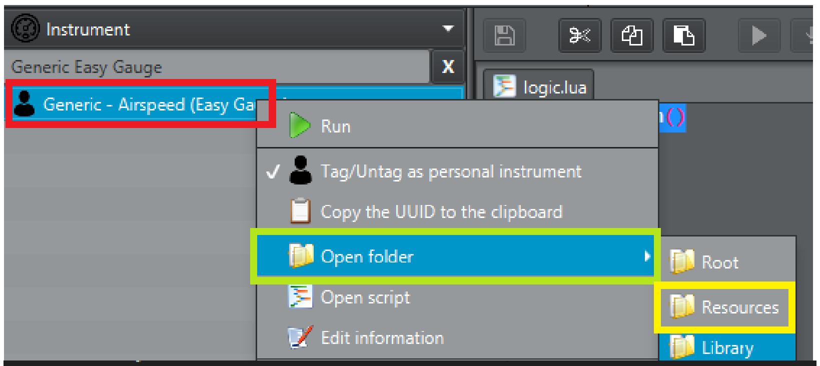
```
easygauge.lua:338: Static data file 'easygauge.json' not found in resource folder
Loading configuration file 'easygauge.json' failed!
```

The warning message occurs because we currently do not have an instrument specification file provided. Consequently, loading the instrument specification fails with an error message.

Add A “Easy Gauge” Instrument Specification

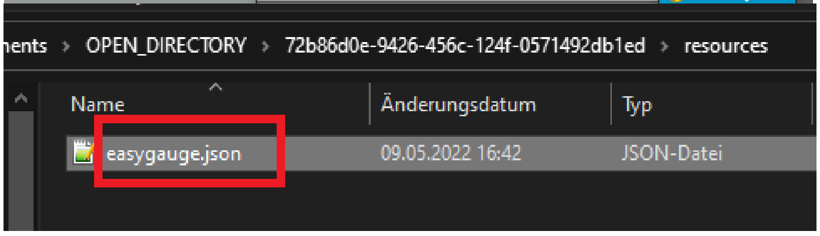
To display something meaningful we need to specify the instrument features. The instrument features must be specified in a file named “easygauge.json” located in the ‘resources’ folder.

I prefer the following procedure to create a new instrument specification file, as it is cumbersome to navigate the Air Manager instrument folder structures from external apps.



Select the new instrument in the "Instrument" list

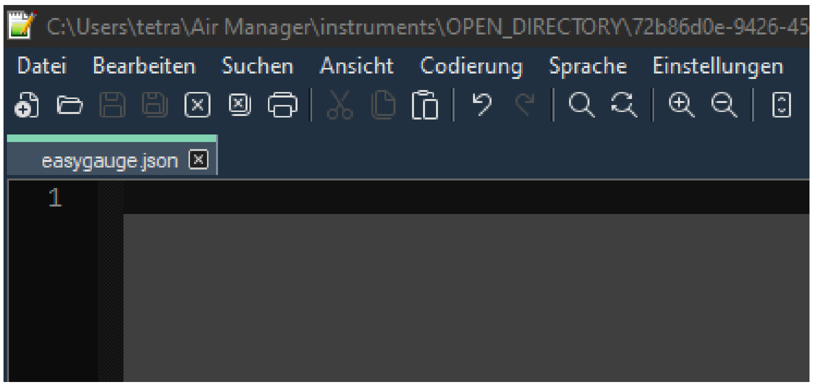
Open the instrument "resources" folder from the Air Manager context menu,


ents › OPEN_DIRECTORY › 72b86d0e-9426-456c-124f-0571492db1ed › resources

Name	Änderungsdatum	Typ
easygauge.json	09.05.2022 16:42	JSON-Datei

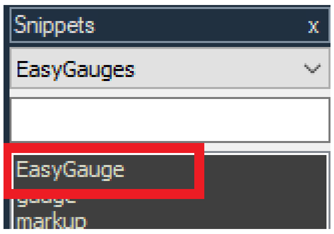
Create a new text document and name it "easygauge.json".

Confirm the message that appears because you changed the extension from .txt to .json



Open the file in your preferred editor.

HINT: Assign the file type .JSON to your preferred editor, to open the specification from the explorer window.



If you use "Notepad++" with the "Snippets" plugin and my "EasyGauges" Snippets (download from my GitHub account) you can insert the template with a DoubleClick on the "EasyGauge" snippet.

Otherwise add the following specification template code:

```
{ "name": "aircraft – instrument (EasyGauge)",
  "background": [ "background.png" ],
  "canvas"    : [ 0, 0, 720, 720 ],
  "cover"     : [ "cover.png" ],
  "bezel"     : [ "bezel.png" ],
  "markups" : [
  ],
  "scales": [
  ],
  "gauges" : [
  ]
}
```

If you save the file and "Run" the instrument, no warning and/or error messages will show up in the "Console" window as we now have a valid specification.

But Air Manager will again just display an empty white as we have not specified anything to show.

If you check the Air Manager log files you may see that there are some messages about image files not found. But you can ignore them for the moment.

Alternative

Instead of creating your own specification, you could just drop one of the instrument specification files from my GitHub account into the 'resources' folder.

You can use it as is or start to tweak its instrument features by changing the content.

“Easy Gauge” Concepts

Before you start specifying the instrument features, you should gain a basic knowledge of the six “Easy Gauge” concepts:

- Layers
- Images
- Canvas
- Scales,
- Markups and
- Gauges.

Layers

A final “Easy Gauge” instrument is drawn on your screen as a stack of 4 layers:

- A “background” layer,
- a “canvas” layer,
- a “cover” layer and
- a “bezel” layer.

Background Layer

The “background” layer is drawn first before any other layer. Currently the background layer just provides a background image for the instrument. As the name says the image stays in the background.

Canvas Layer

The canvas layer is above the background but below the cover and bezel layer.

The canvas layer by itself does not display anything on your screen. You need to specify scales, gauges, and markups. Whatever you specify is drawn on the canvas in a certain order.

Cover Layer

The cover layer is above the canvas but below the bezel layer.

The cover layer is used, as the name suggests, to cover / hide some element drawn on the canvas. But it can be used for other purposes as well.

Currently the cover layer uses an image file.

Note that not every instrument requires a cover layer.

Bezel layer

The bezel layer is above all other layers and the last layer drawn.

The bezel layer function is, as the name suggests, to display some bezel elements like bezel rings or mounting screws but is not limited to that.

Currently the cover layer uses an image file.

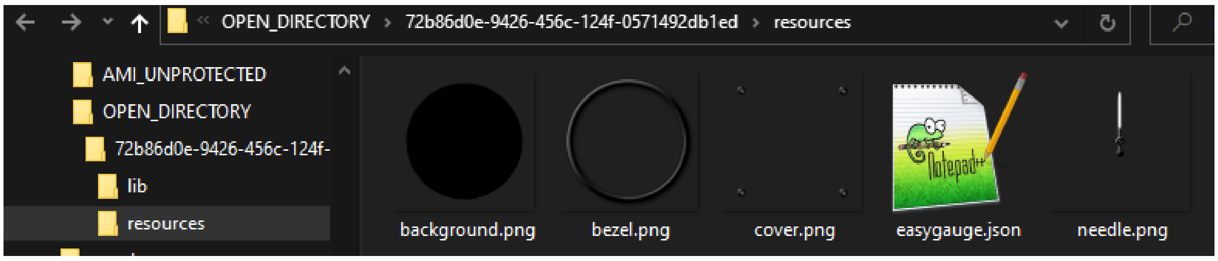
Not every instrument requires a cover layer.

Images

As already explained three of the four layers use images to display instrument features.

Additionally, “Gauges” also use images for their indicators (see the ‘Gauge Indicator’ section for more details).

All images are contained in image files (PNG files are the preferred image types). The image files must reside in the instrument “resources” folder.



For our sample airspeed instrument, I have dropped some simple image files into the “resources” folder:

They are named to reference the layer name. This is not required but helps to simplify specification.

To create new “Easy Gauge” instruments, you can create your own images in your preferred toolset, or reuse images from the community instruments, just give some credits to the original artist.

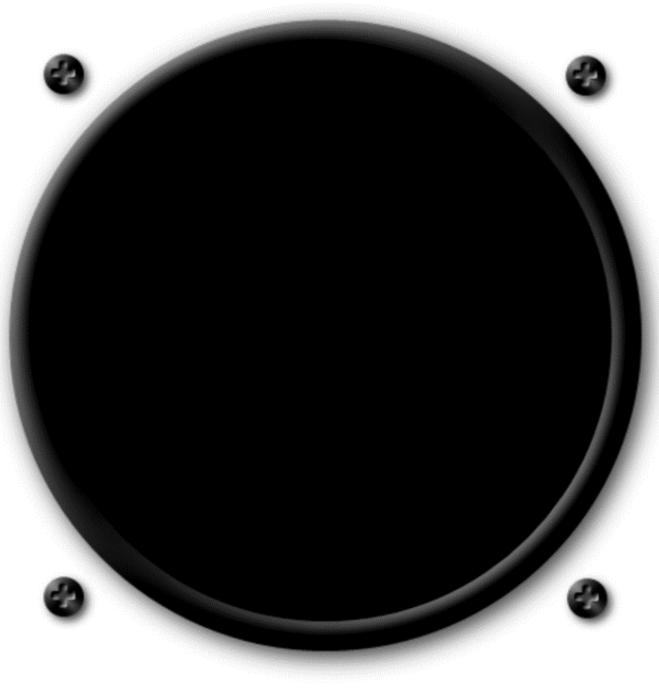
As I’m not a great artist when it comes to image creation, I prefer to reuse images from community instruments.

For the sample instrument, credits go to Russ Barlow for the bezel, cover (screws) and needle images.

The background image I created myself in “Inkscape”.

By the way the final sample instrument with all required resources can be downloaded from my GitHub account as Air manager export (SIFF) file for importing into your installation.

If the image files exist in the folder, you can “Run” the instrument again.



The screen will show the background, the screws from the cover image and the bezel ring from the bezel image.

The canvas is empty as we have not specified any instrument features beside the images so far.

See also:

- ‘Instrument Background’
- ‘Instrument Cover’
- ‘Instrument Bezel’

Canvas

As mentioned in the 'Canvas Layer' section, the canvas is the layer that displays the specified instrument features. Instrument features are:

- Scales,
- Markups and
- Gauges.

Scales are drawn onto the canvas first, followed by markups. Gauges are placed last but before the cover and bezel layers.

See also:

- 'Instrument Canvas'

Scales

A "Easy Gauge" instrument may show one or more scales.

A scale is used by a gauge to indicate the current value. But a scale can have additional features.

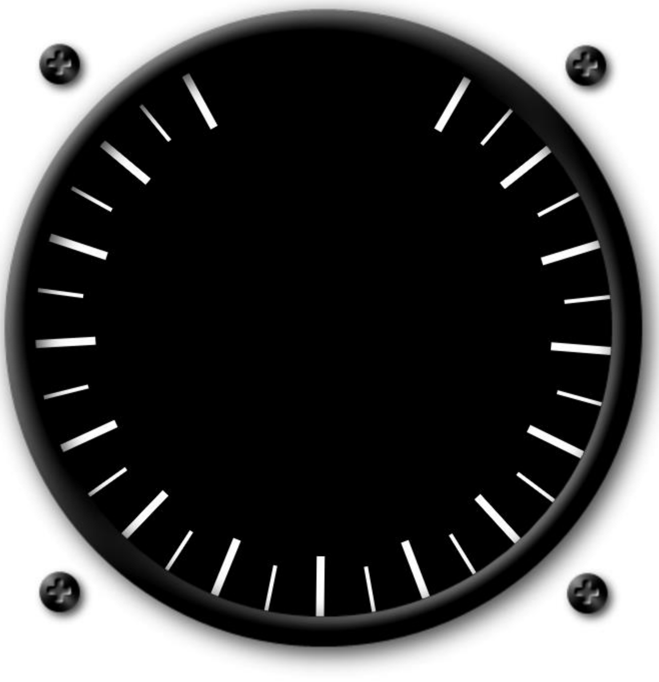
Usually, a scale has

- scale arcs,
- scale marks,
- scale values and sometimes
- redline marks.

Important to note is that the features are drawn in the order given above. That means a redline mark may cover a scale value, a scale value may cover a scale mark and scale marks may cover parts of scale arcs.

Marks

Scale Marks represent positions along the scale where specific values can be indicated.



There are major and minor marks dividing the indication range into divisions and subdivisions.

The scale marks can be specified in different length and sizes.

Marks are usually drawn in white onto the canvas but can be specified in any color available.

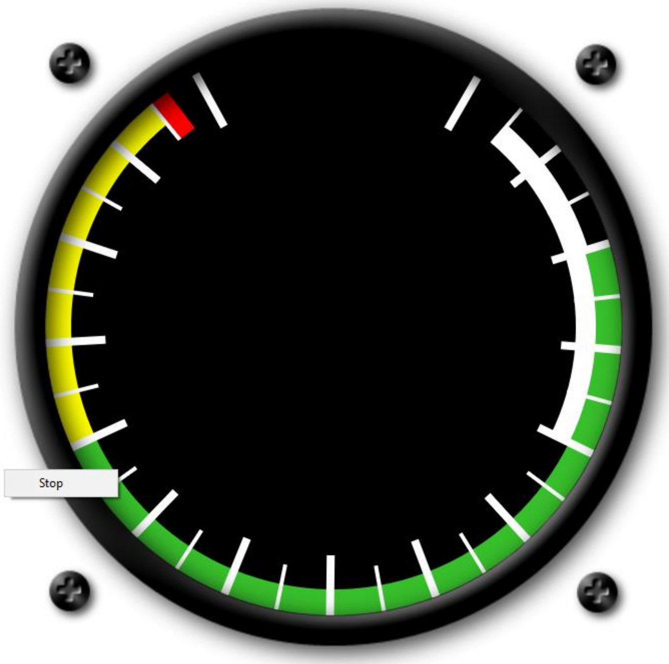
Optionally, a scale can be divided into multiple sections with independent scaling factors. Each section with its own divisions and subdivisions.

See also:

- 'Scale Marks'

Arcs

Scale Arcs indicate operational ranges of scale values. A scale can show multiple arcs. A single arc can then be divided into segments with different colors.



In our example airspeed instrument, there are two arcs.

In the first arc, the green arc segment indicates the normal operating values. The yellow arc segment indicates acceptable operating values for flight without excessive maneuvers. A small red arc segment is used to indicate the never exceed speed.

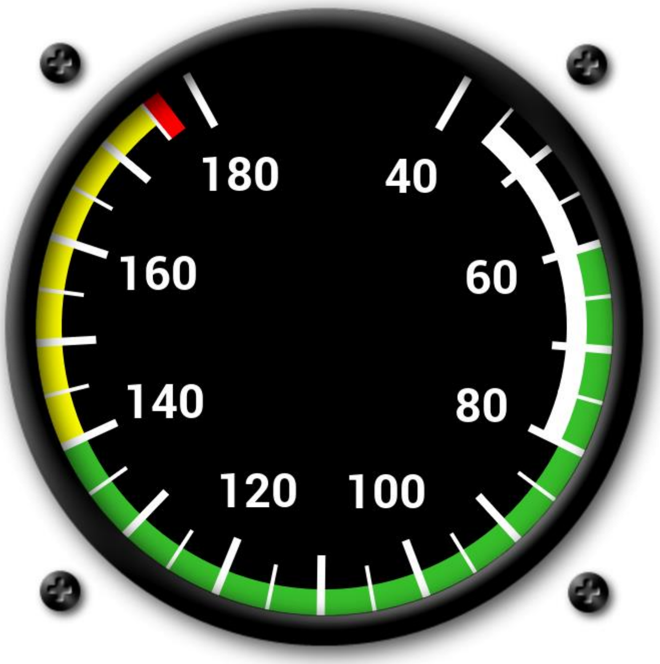
In the second arc, the single white arc segment indicates the operating values with fully extended flaps.

See also:

- ‘Scale Arcs’

Values

Scale values are used to interpret the indicated values. The style and placement of values can be specified.



Values can be shown for every major mark or for every 2nd, 3rd, 4th … mark

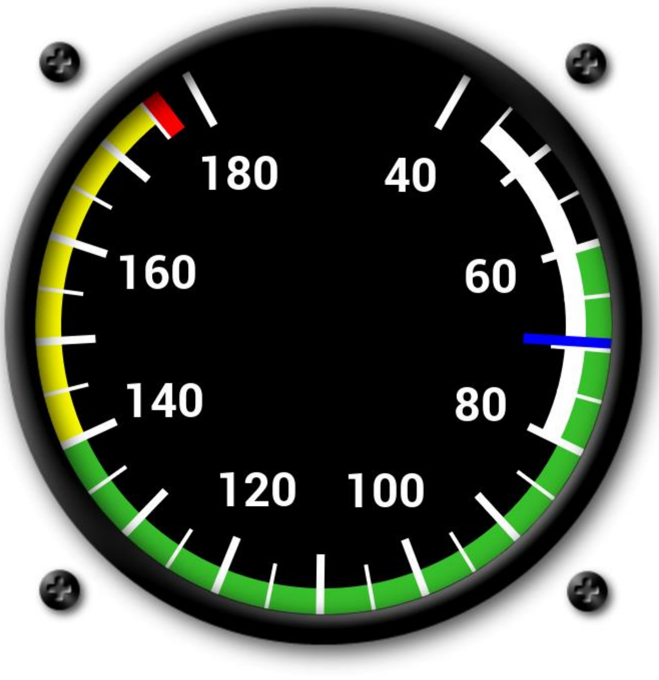
Additionally, the value displayed can be divided by a power of 10 to save space for large numbers.

See also:

- ‘Scale Values’

Redline Marks

Redline marks are used to indicate values that have a special meaning to the pilot. The style, size, and placement of redline markings can be specified.



Redlines do not necessarily need to be displayed in “RED”.

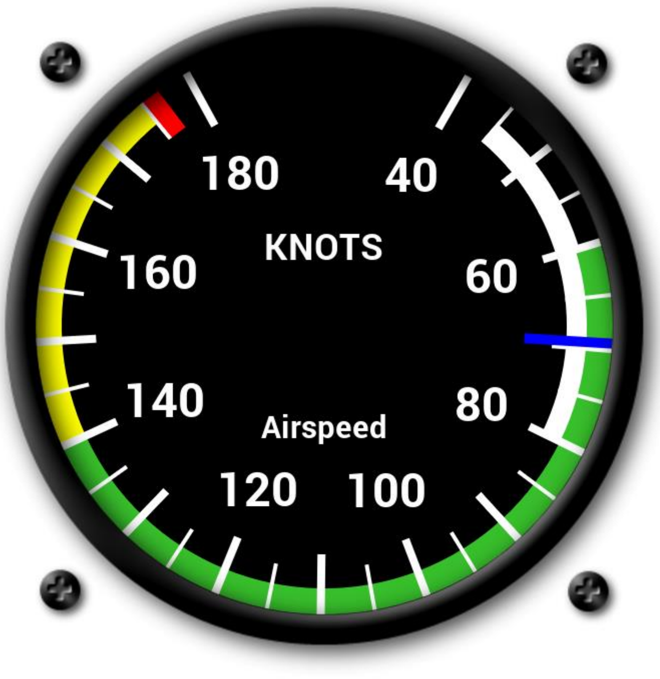
In our sample airspeed instrument a blue marking has been placed at the value for the recommended final approach speed.

See also:

- ‘Scale Redlines’

Markups

Markups are used to place additional information on the canvas. Currently only text markups are supported. There are instrument markups as well as scale specific markups. Both types are treated the same by “Easy Gauges”. The style and placement of markups can be specified.



In our example, markup text elements have been placed on the canvas:

“Airspeed” to display what the gauge is indicating.

And “KNOTS” as the unit of measurements.

NOTE: Markups are placed on the canvas after all scale features are drawn. That means a markup element may cover scale features if not placed in an empty area.

See also:

- ‘Scale Markups’
- ‘Instrument Markups’

Gauges

All the previous described instrument features are static. Meaning, they cannot be changed after the instrument is drawn on startup.

Gauge objects are the dynamic part of instruments. A "Easy Gauge" instrument may have one or more gauge objects.

The responsibilities for a single gauge are

- Subscribing to a simulation variable
- Receiving value updates
- Indicating the new values

Subscribing to a Simulation Variable

For each gauge the source, a variable name and a unit of measurement must be specified. With this you can subscribe to either your flight simulator or the Air Manager internal broker (SI bus).

Currently Microsoft Flight Simulator (FS2020) is supported only. It is planned to support X-Plane 12 and other flight simulators supported by Air Manager.

The Air Manager internal broker (SI) can be used for communicating with other instruments. Currently "Easy Gauges" only supports subscribing to SI. You can't write to SI.

Receiving value updates

Currently only numeric values can be received. It is planned to allow Boolean values as well but converting them to the integers 0 (false) and 1 (true).

The update process is currently hidden from the user. It just saves the new value to a local 'value' variable.

An initial value for the gauge can be specified.

Indicating the new values

For each gauge, one of the specified scales can be assigned. But one scale can be referenced by multiple gauges.

Initially the gauge will indicate the specified initial value. Receiving a value update will trigger a new indication.

180
40
KNOTS
160
60
140
80
Airspeed
120
100

For a visible indication, an indicator image file (needle) is required.

The indication process is currently also hidden from the user.

For each new value received, it first calculates the corresponding position/angle in the scale, then moves/rotates the indicator to that position.

Some animation for the movement may be specified.

Concept Remarks

The above is just the basic concepts and terminology we use for "Easy Gauges".

If you could follow the explanations without hiccup, you should be ready to specify new "Easy Gauge" instruments on your own. The sections on the following pages describe how to specify the instrument features in detail.

Instrument Specification

An instrument specification file must contain the following top level JSON object:

```
{ "name": "aircraft - instrument (EasyGauge)",
  "background": [ "background.png" ],
  "canvas"    : [ 0, 0, 720, 720 ],
  "cover"     : [ "cover.png" ],
  "bezel"     : [ "bezel.png" ],
  "markups" : [
  ],
  "scales": [
  ],
  "gauges" : [
  ]
}
```

If you use "Notepad++" with the "Snippets" plugin and my "EasyGauges" Snippets (from my GitHub account) you can insert the template with a DoubleClick on the "EasyGauge" snippet.

Instrument Name

The instrument "name" field is optional. If missing the name is built from the instrument properties AIRCARFT and TYPE defined in the "Instrument information" dialog.

```
   "name": "Generic - Airspeed (EasyGauge)"
```

The instrument name is used for logging and documentation only.

Instrument Background

For the 'Instrument Background', you can specify the image file name, where to place the image and in what size.

Default Instrument Background Specification

If the 'Instrument Background' is not specified, "Easy Gauges" will

- load an image file named "background.png",
- resize it, if necessary, to the instrument size (preferred width and height) and
- place it at the root (top-left) position of the instrument.

It is good practice to name the background file "background.png" and remove the background layer specification. This should help to shorten the instrument specification.

No Background Image Required

If the no background image is required, it is best practice to specify an empty list:

```
   "background": [],
```

This will make it obvious to the reader that it is intentionally left out.

Background Image File with Custom Name

When using a background image file with another name than "background.png" you must specify at least the image file name:

```
   "background": [ "my_background.png" ],
```

The given file name must include the extension (.png recommended)

"Easy Gauges" will

- load the image file specified,
- resize it, if necessary, to the instrument size (preferred width and height) and
- place it at the root (top-left) position of the instrument.

The image file must reside in the instrument "resources" folder. If a file with the given name does not exist, you will notice an error message in the Air Manager log file. And no image will be visible.

Placing and/or Sizing Background Images

If you require the background image to be placed and/or sized differently, you must specify position or position and size:

```
"background": [ "my_background.png", 100, 50 ],
"background": [ "my_background.png", 100, 50, 520, 620 ],
```

The image position must specify the offset from

- left instrument border (e.g., 100)
- from top instrument border (e.g., 50)

The image size must specify the

- required width (e.g., 520)
- required height (e.g., 620)

If the image size is specified, the position must be specified as well, possibly as 0, 0,

Instrument Cover

For the 'Instrument Cover', basically the same rules apply as for the 'Instrument Background', with the following changes: The default cover image file name is "cover.png".

Instrument Bezel

For the Instrument Bezel, basically the same rules apply as for the 'Instrument Background', with the following changes: The default bezel image file name is "bezel.png".

Instrument Canvas

For the 'Instrument Canvas', you can specify the canvas size and where to place the canvas in relation to the instrument borders.

It is important to remember that all scale features and markups are positioned in relation to the canvas or canvas center, not in relation to the instrument borders.

Default Instrument Canvas Specification

The 'Instrument Canvas' is required to draw the scale features.

If the 'Instrument Canvas' is not specified, "Easy Gauges" will

- create a canvas with the size of the instrument (preferred width and height) and
- place it at the root (top-left) position of the instrument.

It is good practice to remove the canvas layer specification if you want the canvas to cover the whole instrument. This should help to shorten the instrument specification.

In most cases no custom canvas is required. That means you can skip the canvas specification and use the default canvas.

Placing and Sizing a Custom Canvas

If you want to restrict the area, where scale features are drawn, you must specify position and size for a custom canvas:

```
"canvas": [ 60, 0, 600, 720 ],
```

The position must specify the offset from

- left instrument border (e.g., 60)
- from top instrument border (e.g., 0)

The canvas size must specify the

- required width (e.g., 600)
- required height (e.g., 720)

If a custom canvas is required, position AND size must be specified.

Instrument Scales

As explained in the concept sections, a “Easy Gauge” instrument may have one or more scales. This is specified with a list of scale objects in the following form:

```
"scales": [
   <scale-spec>,
   <scale-spec>,
   ...
   <scale-spec>,
],
```

The instrument might have no scale at all, but this is an advanced feature, not described in this document. In such a case the list would be empty:

```
"scales": [],
```

In most cases an instrument has just one scale (like our example instrument from the concept sections). In those cases, the list contains just one scale object

```
"scales": [
   <scale-spec>
],
```

For some instruments (like engine gauges) there might be two scales. More than two scales are a very rare use case. Anyway “Easy Gauges” supports an unlimited number of scales if needed.

But a single scale might be used by more than one gauge. How this is handled is described in the ‘Instrument Gauges’ section.

Scale Object

A scale object specifies all the scale features. Some have been already explained in the concept sections.

A template for a JSON scale object looks like this (hint: insert the ‘scale’ snippet):

```
{ "kind": "circular",
  "center": [ 360, 360 ],
  "marks": {
    "offset": [300, 280, 260 ],
    "width": 4,
    "sections": [
    ]
  },
  "values": {
    "every"  : 2,
    "offset" : 260,
    "style" : { "size": 48 },
    "power"  : 0
  },
  "arcs": [
    { "offset"  : 290,
      "width"   : 10,
      "segments": [
      ]
    }
  ],
  "redlines": { "offset" : [ 300, 250 ], "values" : [ 0.0, 100.0], "width"  : 6 }
},
```

Scale Kind

The scale kind defines how the scale appears in the instrument. Currently only ‘circular’ scales are supported.

```
"kind": "circular",
```

This is the kind of scale from our example instrument in the concept sections.

It is planned to also support ‘horizontal’ and ‘vertical’ scales in later releases.

“Circular” is the default kind because this is the most common kind of gauge scales in real life.

If you want to replicate such a circular scale, you can omit the “kind” specification. “Easy Gauges” will assume “circular” as the scale kind.

Scale Center

The scale center is the reference location for scale features. All scale features will be placed in relation to the specified scale center. Specify the center position as

- distance from the left canvas border
- distance from the top canvas border

Both distances are need if a center position is specified.

```
"center": [ 360, 360 ],
```

If the scale “center” is not specified, “Easy Gauges” will assume the scale center at the center of the canvas. That is at

- half the canvas width for distance from the left canvas border, and
- half the canvas height for distance from the top canvas border.

If the canvas covers the whole instrument area (if for example no canvas has been specified explicitly), this position is congruent with the center of the instrument.

Scale Marks

Scale marks are the main features of a scale.

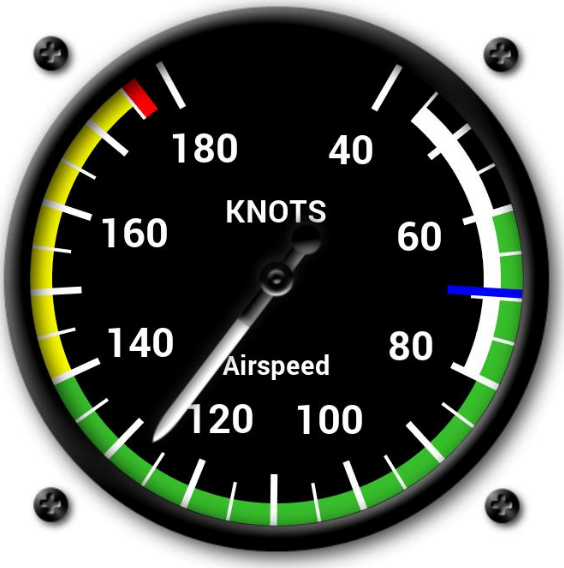
They are usually drawn in white as shown in the picture here.

Scale marks allow to read an indicated value from the gauge.

In the picture the indicator needle points to a scale value of about 127 knots.

To draw scale marks on the canvas, you need to specify the placement, size and how the scale is divided into sections.

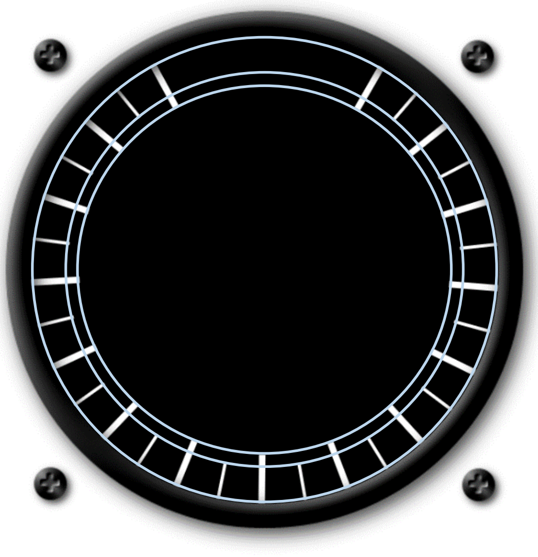
```
"marks": {
  "offset": [ 288, 242, 228 ],
  "width": 8,
  "sections": [
  ]
},
```



Offset

The placement is specified using a list of three **offset** values.

```
"offset": [ 288, 242, 228 ],
```



The 3 offset values specify the distances from the scale center to

- the start of the major and minor marks
- the end of the minor marks
- the end of the major marks

For 'circular' scales, the values represent the radii of the circles around the minor resp. major marks.

The (absolute) difference between the first and second values defines the length of the minor marks.

The (absolute) difference between the first and third values defines the length of the major marks.

For each specified scale, the offset list is required. There are no default values for the offsets.

Width

The size of the markings is specified using a **width** value.

```
"width": 8,
```

The value represents the width of the mark lines in pixels.

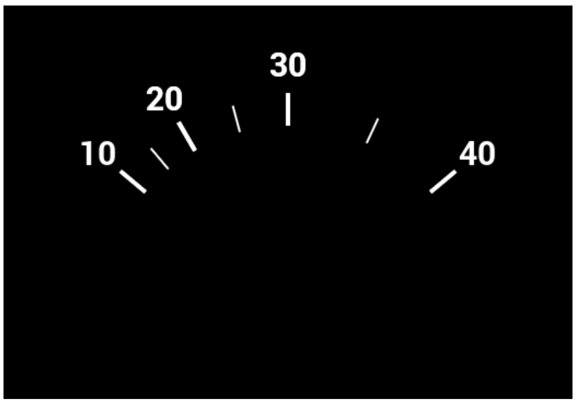
If not specified, the width defaults to a value of 4.

Scale Sections

A scale may be divided into sections with different properties. The sections are specified through a sequence of section specifications.

```
"sections": [
   [    0, -60.0,   0          ],
   [   10, -50.0,  10, 5, 0.5 ],
   [   20, -30.0,  10, 5, 0.5 ],
   [   30,   0.0,  10, 5, 0.5 ],
   [   40,  50.0,   0          ],
   [   50,  60.0               ]
 ]
```

The given section list above results in a scale with 5 different sections as shown in the picture here:



- A hidden (unmarked) section from 0 to 10
- A marked section ranging from 10 to 20
- A larger marked section ranging from 20 to 30
- An even larger section from 30 to 40 and
- Another hidden section from 40 to 50

For each specified scale, a sections list is required. There are no default values for the sections.

At least the start of the scale and the end of the scale must be specified.

- the start of the scale, is specified in the first section.
- the end of the scale is specified in the last section.

You can insert any number of additional sections between the first and the last section spec.

Section Specification

A section defines

- the start value of the section,
- the position of the first major mark in the section,
- optionally, how the section is divided into (major) divisions,
- optionally, how the divisions are divided into (minor) subdivisions and
- optionally, how much “thinner” minor marks should be drawn

Example:

```
[   10, -50.0,  10, 5, 0.5 ],
```

All sections must at least specify the start value and the position value.

All sections, except the last section spec, may contain the optional values.

Section start value

The first value defines the gauge value at which the section begins.

It is important that the start value for the first section spec represents the lowest value the gauge must indicate. If the gauge can indicate negative values, that start value for the first section spec must be the largest negative value.

The start value for each succeeding section must have a value that is greater than all the previous start values and smaller than all following start values, giving a sequence of ascending start values.

This implies also that no two sections have the same start values.

When drawing scale values, additional restrictions may apply (see ‘Scale Values’ below).

Position of first major scale mark

The second value defines the position where the section begins.

For ‘circular’ scales this is the angle of a clockwise rotation from a 12’o clock position. This value must be given in degrees.

- 0 degrees is the 12’o clock position
- 90 degrees is the 3’o clock position
- 180 degrees is the 6’o clock position
- 270 degrees is the 9’o clock position
- 360 degrees is the 12’o clock position again

Note that this scheme requires that the indicator image (needle) is pointing to the 12’o clock position (see the ‘Gauge Indicator’ section).

Major Divisions

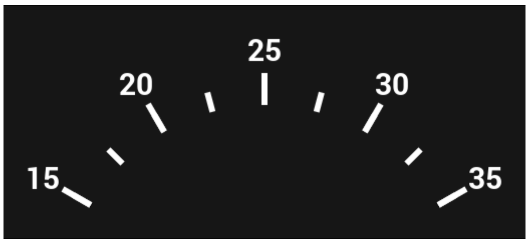
A section may be divided into divisions for placing additional major marks.

It is recommended to divide a section into equal divisions. If this is not done properly the last division of a section might be noticeably smaller than all the other divisions.

To do it properly a divider value must be specified that divides the difference between the start value and the next section start value without remainder.

A simple example with just one section:

```
"sections": [
  [  15.0, -60.0, 5, 2.5 ],
  [  35.0,  60.0         ]
]
```



In the resulting picture shown here, the difference of the two consecutive start values is 20 (35 minus 15).

Any divider value that divides 20 into equal division would be fine.

Here a value of 5 was chosen that divides the range into 4 equal divisions limited with major marks.

It is recommended that the values at the division limits represent integer values.

If subdivisions are required, the divisions themselves should be divisible using to the same rules (see below).

If the divider is not specified or given as zero (0), no major marks would be drawn in that section. Also, no scale values are drawn, as scale values are bound to major marks (see also ‘Scale Values’ below).

This can be used to have hidden sections without markings and values drawn.

Minor Divisions

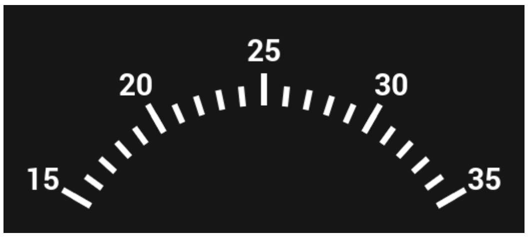
A section division may be divided into subdivisions for placing minor marks.

The same rules as given for major divisions apply for minor subdivisions. Except that the values at the minor marks do not need to be integers.

In our example above the value range for a division is 5 which is further divided into 2 subdivisions using a subdivider value of 2.5

A value of 1 for the subdivider would create 5 subdivisions separated with minor marks, as shown in the picture here.

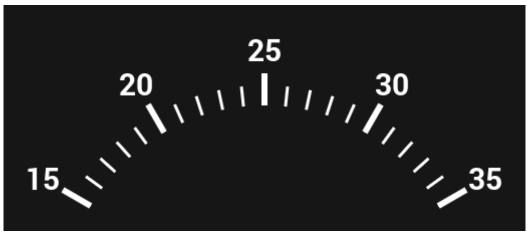
If not specified, no minor marks are drawn on the canvas.



Minor Mark Sizing

If minor subdivisions are specified, the width of the minor marks may be reduced to allow better distinction between major and minor marks. This must be specified as a number value between 0.0 and 1.0

The value of 1.0 gives the same width as specified for the major marks.



A value of 0.5 will use half the width of the major mark, as shown in the picture here.

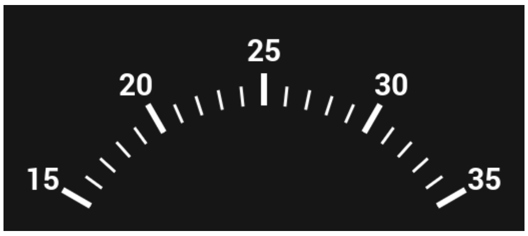
If not specified, a value of 1.0 is assumed.

Do not use the value 0.0, as it has the same effect as specifying no subdivisions. It would just make the specification unclear about the intentions.

Scale Values

Scale values are a scale feature to further help identify the indicated value. Without scale values it would be difficult to determine the indicated value out of a full range scale.

```
"values": {
  "offset" : 255,
  "style"  : { "size": 40 },
  "every"  : 1,
  "power"  : 0,
},
```



Scale values are always bound to major marks.

Every major mark may have its own value (text) drawn on the canvas.

For scale values you can specify

- Where the value (text) should be placed
- Optionally In what text style the value is drawn
- For which major marks values are drawn
- What value is drawn

NOTE:

All values drawn are formatted without decimal parts before drawing. Therefore, it is recommended to use integer values as the start values for the scale sections. If using section divisions, make sure that the dividers are integers and the section start values are multiples of the dividers.

Placing of Scale Value

Scale values should always be placed close to their major marks. The position is defined by

- The position of the major mark, as specified in the scale marks specification
- A specified distance (offset) from the scale center

```
"offset" : 255,
```

For 'circular' scales, the value can is placed along the radial from the center to the major mark, either inside or outside the mark. The offset value gives the distance to the scale center.

Style of Scale Value

Scale values can be drawn in a variety of styles. Font, size, color, and alignment can be specified.

See the 'Text Styles' section for a description of text style specifications.

Reducing the Number of Scale Values

If there are many major marks in a scale, it is recommended to reduce the number of values drawn. Otherwise, the canvas will be cluttered with too many values, making it difficult to see what value stands for which mark.

With the optional "every" value, you can specify that only every n-th value should be drawn. This rule does not account for the first value in a scale section, as this value is always drawn.

For example, with a "every" value of 3 and a sequence of values like

'1', '2', '3', '4', '5', '6', '7'

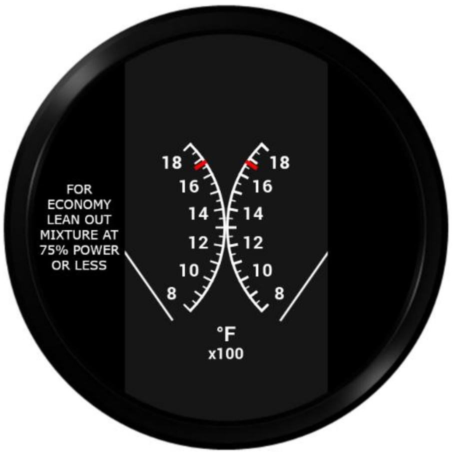
only every 3rd value after the first one will be drawn. This would be the values

'1'   '4'   '7'

Reducing the Number of digits per Values

If the values to be drawn have more than 2 or 3 digits (e.g., a value like ‘1600’) again cluttering of the canvas my occur.

In such cases it is common practice to draw only the relevant digits (e.g., ‘16’) and give a corresponding multiplier as markup text (“ x100) to indicate the reduction.



As in the picture of this gauge indicating exhaust gas temperature (EGT) in the range of 800 to 1800 °F.

When reading the gauge the pilot must multiply the value indicated (‘16’) with the multiplier (‘x100) to get the real value of ‘1600’.

The reduction is always in power of tens. To specify the reduction, the “power” value is the exponent of the power function with base 10. A “power” value of

- 3 defines a divider of 1000
- 2 defines a divider of 100
- 1 defines a divider of 10
- 0 defines a divider of 1 (no reduction)

“Easy Gauges” even allows negative “power” values to produce integer values from small decimal values. For example, a “power” value of

- -1 would define a divider of 0.10

This would produce an integer value of 12 from a real value of 1.20

Scale Arcs

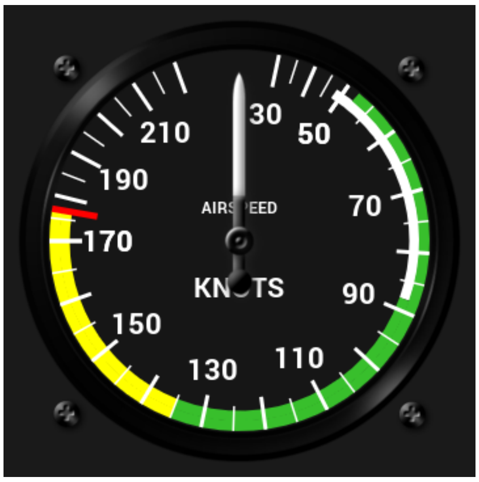
As explained in the concept sections, a “Easy Gauge” scale may have one or more arcs. This is specified with a list of arc objects in the following form:

```
"arcs": [
   <arc-object>,
   <arc-object>,
   ...
   <arc-object>,
],
```

The “arcs” field is optional. If the field is missing or the specified list does not contain any arc objects, no arcs are drawn.

A list may contain any number of arc objects. In the picture shown here two arcs are specified:

- The first arc has 2 segments, one segment in green and one segment in yellow.
- The second arc with just one segment in white.



The arc objects are drawn in the order as they are specified. Here the white arc is on top of the first arc.

Each arc object specifies the following arc features:

- The Position of the arc
- The width of the arc
- The segments of the arc

The arc object for the first arc in the picture looks like this:

```
{  "offset"   : 275,
   "width"    : 30,
   "segments" : [
      [  52, "lime"  ],
      [ 135, "gold"  ],
      [ 177          ]
   ]
}
```

Position of Scale Arcs

The arc is positioned with the “offset” field. The field is required and must give a numeric value. The value specifies the distance of the arc center line from the scale center.

For ‘circular’ scales this is the radius of the arc.

For ‘horizontal’ scales this is the vertical distance. For ‘vertical’ scales this is the horizontal distance.

Width of Scale Arc

The arc “width” field specifies the stroke size for drawing the arc. The field is optional. If missing a default width is assumed. If specified the value must be a number, giving the stroke width in pixels.

Note that half of the width is inside the center line, half of the width is outside the center line. This must be accounted for when positioning the arc (see above).

Arc Segments

Each arc may consist of any number of arc segments. Each arc segment with its own color. The arc segments are specified with the “segments” field. If the field is missing or the list is empty, no arc segments are drawn.

The “segments” field must specify a list of segments.

A segment defines

- the start value of the segment, and
- the color of the segment.

```
"segments" : [
   [  50, "lime"  ],
   [ 134, "gold"  ],
   [ 177          ]
]
```

All segments must at least specify the start value. The start value must be a numeric value out of the scale range. Do not specify rotation angles, as the rotation angle is interpolated for the specified value from the scale sections.

If a color is specified an arc segment is drawn from the specified start value to the start value of the next segment.

The last segment ends at the scale end. If its color is not specified, no segment is drawn from that start value till the scale end.

Scale Redlines

Redlines are used in real life instruments to indicate operational limits. The “redlines” field is used to specify a list of redline objects.

```
"redlines" : [
   <redline-object>,
   ...
   <redline-object>
]
```

The “redlines” field is optional. If missing no redlines are drawn.

Redline Object

A redline object specifies

- the values where redlines should be drawn
- the length of a redline,
- the width of a redline,
- the “color” of a redlines

The order of the fields is irrelevant.

Example :

```
{  "values"   : <value-list>
   "offset"   : <offset-list>,
   "width"    : <width>,
   "color"    : <color>,
}
```

Redline Values

The redline value list specifies where redline marks are placed along the scale. The list may contain any number of values. If the “values” field is missing or the value list is empty, no redlines are drawn.

The value must be a NUMBER in the range of the scale values. Otherwise, a redline at the closer scale end is drawn (CAPPING).

Example:

```
"values": [ 15, 125 ]
```

Line Position

The redline offset list specifies the length and at what distance from the scale center redlines are drawn.

The “offset” field is required. If the field is missing or the offset list is empty, no redlines are drawn.

Example:

```
"offset": [ 218, 265 ]
```

For circular scales, the offsets constitute the circle radii from the scale center to the line start and end.

For horizontal scales, the offsets specify the vertical distance from scale center to line start and end.

For vertical scales, the offsets specify the horizontal distances from scale center to line start and end.

Line Width

The “width” field specifies the width (thickness) of the redline marking. The value must be the number of pixels used for drawing the stroke.

Example:

```
"width": 5
```

The width field is optional. If missing a default redline width value of 4 is used.

Line Color

The “color” field specifies the color of the redlines. Color values are either color names (like “red”) or RBG-strings(like “FF0000FF”).

Example:

```
"color": "F1F1F1FF"
```

The color field is optional. If missing the default redline color value “red” is used.

Scale Markups

Scale Markups are created like Instrument Markups.

See the ‘Instrument Markups’ section.

Instrument Markups

Markups can be specified for the instrument as well as for each scale.

In both cases the “markups” field is used to specify a list of markup objects. Any number of markup objects can be specified. The “markups” field is optional. If missing or if the list is empty, no markups are drawn.

Example:

```
"markups": [
   <markup-object>,
   ...
   <markup-object>,
]
```

NOTE: Currently only text markup objects are supported.

Text Markup Objects

A text markup object specifies attributes for text elements to be placed onto the canvas.

```
   {  "position" : <position>,
      "style" : <style-spec>,
      "text" : <string>
   },
```

If a markup object is specified, the “text” and the “position” fields are REQUIRED.

Markup Position

The “position” field specifies the position of the text markup. The field is required. If missing not markup is drawn.

The value must be the list of two numbers.

Example:

```
"position": [ 112, 345 ],
```

The order of the values is relevant:

1. the first value specifies the distance from the left **canvas** border
2. the second value specifies the distance from the top **canvas** border

Markup Text

The “text” field defines the text string to place on the canvas. The field is required. If missing no markup is drawn.

The value must be a string value.

Example:

```
"text": "x10 °C",
```

Optionally the placeholder ‘%s’ can be placed anywhere in the string. The placeholder will be replaced by the value for the user property “Index” for the instrument. See a description for the Index property in the ‘Advanced Features’ section.

Markup Style

The “style” field specifies some attributes to control the appearance of text elements placed onto the canvas.

The “style” field is optional. If the “style” field is missing a default style is used.

See the ‘Text Styles’ section for a description of text style specifications.

Instrument Gauges

You can specify any number of gauges in a “Easy Gauge” specification. Most real life gauges have just one gauge, but some have more than one gauge.

Each gauge is specified in a separate JSON gauge object. All gauge objects are listed in the gauges list.

```
"gauges" : [
   <gauge-object>,
   ...
   <gauge-object>
]
```

Gauge Object

The gauge object specifies a single gauge.

The template for a gauge specification looks like this:

```
{ "value"    : 0.0,
  "scale"    : 1,
  "indicator": [ "needle.png", 720, 720, 0, 0 ],
  "variable" : [ "SIMVAR", "UNIT", "FS2020" ],
  "animation": [ "LOG", 0.05  ]
},
```

NOTE: Use the ‘gauge’ snippet from my “EasyGauges” snippet list in Notepad++ to insert a gauge template.

Gauge Value

Every gauge object maintains its current value. You can specify the initial value with the “value” field:

```
"value" : 29.92,
```

The specified value must be a LUA number, a floating point or integer literal value, in decimal notation, like

```
4
0.4
4.57e-3
0.3e4
5E+20
```

or in hexadecimal notation, like

```
0xff
0x1A3
0x0.2
0x1p-1
0xa.bp2
```

If not specified, the initial value is assumed to be zero (0.0).

Gauge Scale

Every gauge uses a scale to indicate its current value on. You can specify the scale with the “scale” field:

```
"scale" : 2,
```

The specified value must be an index number of a scale object specified in the instrument “scales” field. The first scale object specified is assigned the index number ‘1’. For each additional scale specified the assigned index number increases by 1.

If not specified, the first scale (1) is assumed.

NOTE: A scale can be referenced by more than one gauge.

Gauge Indicator

To indicate the current value a gauge needs a moveable indicator (aka needle, hand, pointer).

“Easy Gauge” uses images to display indicators.

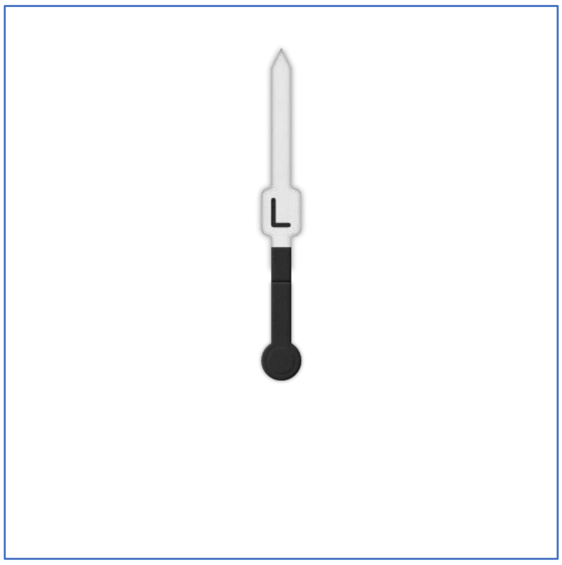


Image File

You can specify the name of the file containing the indicator image with the “indicator” field.

```
"indicator": [ "needle_left.png" ],
```

If not specified, the file name “needle.png” will be used.

The image file, whether specified or default, must reside in the instrument “resources” folder.

Indicator images for rotating indicators must have

- the pivot point at the center of the image, and
- the indicator (needle) pointing to the 12’o clock position.

Image Size

By default, “Easy Gauges” uses the image in its original size (resource width/height).

To adjust the size to fit the scale, you can specify an alternate (width and height) size in the “indicator” field.

```
"indicator": [ "alt_1000.png", 670, 670 ],
```

If not specified, the size is the original width and height of the image.

Image Placement

“Easy Gauges” positions the image such that the pivot point of the image (the image center) is at the scale center. This is independent from any sizing of the image.

If it is necessary to adjust the position to fit certain requirements, you can, in addition to the size attributes, specify a (horizontal and vertical) offset in the “indicator” field.

```
"indicator": [ "needle.png", 670, 670, 0, -10 ],
```

The offsets will be added to the calculated position.

If not specified, the offsets are 0.

Gauge Variable

The main purpose of a gauge is to indicate changing values. As the maintained value does not change by itself, we need to select a source variable that provides the changing values.

Variable Name and Unit

For the selected variable, you can specify the variable name and the unit of measurement for the values.

```
"variable" : [ "AIRSPEED INDICATED", "KNOTS" ],
```

The available variable names and units of measurements are documented in the SDK documentation that are available online.

By default, "Easy Gauges" will subscribe for the specified variable and unit through Air Manager. Air Manager handles the subscriptions in its plugin and forwards the updated values whenever the variable value changes.

The new value is maintained by the gauge until a new value is received

Optionally the placeholder '%s' can be placed anywhere in the variable name string. The placeholder will be replaced by the value for the user property "Index" for the instrument. See a description for the Index property in the 'Advanced Features' section.

Variable Source

The default source for all variable subscriptions is Microsoft Flight Simulator 2020 (FS2020).

If you need the variable from another source, you can specify the source in the "variable" field.

```
"variable" : [ "AIRSPEED INDICATED", "KNOTS", "SI" ],
```

Currently, two sources are supported

- "FS2020" for Microsoft Flight Simulator 2020 (the default), and
- "SI" for the SimInnovations internal broker.

The Air Manager internal broker (SI) is used for inter-instrument communication. That is, a value published by one instrument, can be subscribed to by any number of other instruments.

This is the only way to share information between instruments in Air Manager.

NOTE: Currently a "Easy Gauge" instrument can subscribe to a "SI" variable, but not publish values.

Gauge Animation

Every time a new value is received, the indicator image is moved to point to the corresponding value in the scale.

For 'circular' scales this means the image is rotated around the pivot point (image center). Where to point the indicator is interpolated from the scale's sections table, using the start values and position values.

Using the same sections table for drawing the scale and calculating the indicator movement guarantees that the indicator will always point to the correct scale value for a received value. No conversion must be applied.

Just make sure you subscribe with the unit of measurements that was used to specify the section start values.

Smoothing the Movements

The rotation described above is immediate. When the received values change in large steps, the movements of the indicator can be quite “jerky”.

To smooth the movements, you can specify a list of animation attributes with the “animation” field.

```
  "animation" : [ "LOG", 0.05 ]
```

The first value specifies the type of animation

- “LINEAR” for equal sized steps between intermediate values,
- “LOG” for a logarithmic stepping (a larger step first, then continuously smaller steps), and
- “OFF” for no smoothing.

“LOG” gives the most natural smoothing.

The second value specifies the “speed” of the animation, with a value between 1.0 and 0.0, where

- 1.0 is the fastest animation (few intermediate values)
- 0.0 is the slowest animation (many intermediate values)

From experience, values between 0.06 and 0.03 give nice and smooth animations.

The third value specifies the direction of the rotation

- “DIRECT” will rotate clockwise for positive and counterclockwise for negative rotation angles,
- “CW” will always rotate clockwise (negative angles are added to 360°)
- “CCW” will always rotate counterclockwise (positive angles are added to -360°)
- “FASTEST” will rotate in the direction with the smaller amount of rotation
- “SLOWEST” will rotate in the direction with the larger amount of rotation

For “Easy Gauges” the direction of rotation is only relevant for scales that use the full circle (0-360°).

On such scales the indicator would rotate counterclockwise a full circle when the value goes from close to 360° to above 0° as the difference is negative, and vice versa from close to 0° to below 360°.

If you observe such a behavior, you should specify animation smoothing and use “FASTEST” as the direction value.

NOTE:

Smoothing is done by Air Manager. I do not know if this is implemented using software or hardware timers. Anyway, using a lot of timers, could impact the behavior of Air Manager.

I only use smoothing for the gauges, that are under continuous observation during a flight.

Text Styles

Text style specifications are used for drawing scale values and markup text.

A template for the JSON style object looks like this (hint: insert the ‘style’ snippet):

```
"style" : {
  "font"   : "roboto_bold.ttf",
  "size"   : 48,
  "color"  : "white",
  "valign" : "center",
  "halign" : "center"
},
```

NOTE: Use the ‘style’ snippet from my “EasyGauges” snippets list in Notepad++ to insert a style template.

“Easy Gauge” defines some default values for each style attribute (see ‘Default Values’). You only need to specify the attributes that differ from the default value.

```
"style" : { "size": 32 },
```

Text Fonts

You can specify any TrueType font that Air Manager has access to.

NOTE:

Air Manager does not use the True Type fonts installed in the Windows Operating System. This is to provide a consistent interface across all supported platforms.

Air Manager provides some preinstalled fonts in the ‘fonts’ folder in the installation directory:

arimo_regular.ttf

arimo_bold.ttf

DIGITAL-7-MONO.TTF

inconsolata_regular.ttf

inconsolata_bold.ttf

roboto_regular.ttf

roboto_bold.ttf

If you want to use another font, you must copy the corresponding .ttf font file(s) into the ‘resources’ folder of your instrument.

If you intend to export instruments for import into another Air Manager installation or push them to an Air Player instance, font files in the ‘resources’ folder are included in the export / push.

Alternative

You can drop the font files into the “fonts” folder in the Air Manager installation directory. This has the advantage that the font files are accessible to all instruments.

But you need to copy it over to another Air Manager / Air Player installation directory manually, when exporting / pushing instruments.

Font Size

You can specify any font size that the specified TrueType font supports.

NOTE:

Not all creators of TrueType fonts adhere to common type size rules. If you change from one font to another, you might have to adjust the font size.

Font Color

You can specify a font color in two formats. Either

- As color name or
- As RBG(A) value.

Color names are the names defined for HTML/CSS colors, the basic colors as well as the extended colors.

```
"color"  : "aqua",
"color"  : "aquamarine",
```

RBG(A) values are specified as strings representing the hexadecimal color value.

```
"color"  : "#00FFFF",
"color"  : "#7fffd4",
```

Optionally a (A) value for the alpha channel (transparency) can be appended:

```
"color"  : "#00FFFFFF",
"color"  : "#7fffd480",
```

Text Alignment

You can specify vertical and horizontal text alignment.

```
"style" : {
   "valign" : "top",
   "halign" : "left"
}
```

If not specified, “center” alignment is used. For scale values, this helps to properly place the values on the radial from scale center to the corresponding scale mark. Also, placement of text markups is easier if “center” alignment is used.

Default Values

These are current defaults for “Easy Gauges”

```
EASYGAUGE = {
  IMAGE = {
    BACKGROUND = "background.png",
    COVER      = "cover.png",
    BEZEL      = "bezel.png",
    INDICATOR  = "needle.png"
  },
  MARKUP = {
    FONT   = "roboto_bold.ttf",
    SIZE   = 48,
    COLOR  = "white",
    VALIGN = "center",
    HALIGN = "center"
  },
  MARKS = {
    WIDTH   = 4,
    COLOR   = "white",
    SECTIONS = {}
  },
  ARC = {
    ZERO  = -90,
    WIDTH = 4,
    COLOR = "white"
  },
  REDLINE = {
    WIDTH = 4,
    COLOR = "red"
  }
}
```

Advanced Features

Change Default Values

To change the default values (see the ‘Default Values’ section above), you must add some code in the ‘logic.lua’ script file before the runner is started.

The necessary code is a LUA assignment statement, assigning the new default value to a predefined default variable.

Example:

```
DEFAULT.MARKUP.FONT = "arimo_bold.ttf"
```

The example will change the default font for markups (see the section ‘Default Settings’).

Any number of assignments can be inserted **before** the function `easygauge()` is called.

Index Property

Every “Easy Gauge” instrument provides a user property named “Index”. The index value can be set in the instrument user properties section, after a “Easy Gauge” instrument is inserted into a panel.

The index can be set to any value. In most cases the index is a numeric value like ‘1’ or ‘2’ (e.g., for an engine index) . But it could also be some alphanumeric value like ‘LEFT’ or ‘RIGHT’?

This is helpful to indicate values for multiple system components, for example engines, fuel tanks, oil pumps and many more, using identical instruments.

Index Value in Variables

A ‘Index’ value can be used in the “variable” field.

Two steps are required. You must

1. insert the placeholder “**%s**” into the variable name specification and
2. set the ‘Index’ property value.

On subscribing, the placeholder will be replaced by the set index property value.

Yo must insert the placeholder “**%s**” in the variable name, where it should finally appear. For most of the indexable variables the placeholder is positioned at the end (after a colon “:”).

Example:

```
"variable": [ "GENERAL ENG FUEL PRESSURE:%s", "PSI" ]
"variable": [ "APU VOLTS:%s", "VOLTS" ]
```

With the index property set to “1” the variable subscribed to is

```
"GENERAL ENG FUEL PRESSURE:1" resp.
"APU VOLTS:1"
```

There are a few indexable variables where the placeholder must be inserted at other positions (without a colon “:”).

Example:

```
"variable": [ "GEAR %s POSITION", “PERCENT OVER 100” ],
```

With the index property set to “LEFT” the variable subscribed to is

```
"GEAR LEFT POSITION"
```

Index Value in Markup Text

Indexing also works for markup text. When inserting the placeholder in a markup text string, the index value replaces the placeholder.

Example:

```
   { "style": {"size": 56}, "position": [260, 136], "text": "%s TANK" }
```

With an index value of “LEFT” the markup will appear as “LEFT TANK”.

Change Gauge Appearance

Normally a gauge uses the “center” and “sections” values from the referenced scale (scale index) to position the indicator image and move the indicator to indicate correct values.

- The center of the indicator image is co-located with the scale center
- The rotation angle is interpolated from the scale section data.

This requires that the pivot point of the indicator image is at the image center, and the indicator should point to “North”. This is the case for most of the community indicator imagery.

If the image does not comply with these rules, or other requirements dictate, the gauge object can specify its own center and sections fields:

Example:

```
"gauges"    : [
   {  ...
      "center"     : <center>,
      "sections"   : <section-list>
   }
```

Gauge Center

Specifying a gauge “center”, positions the center of the indicator image at the specified center position in relation to the canvas left and top borders.

The value must be the list of two numbers.

Example:

```
"center": [ 112, 345 ],
```

The order of the values is relevant:

1. Distance from the left **canvas** border (NUMBER)
2. Distance from the top **canvas** border (NUMBER)

The gauge field “center” is optional. If missing the center of the referenced scale (see ‘Gauge Scale’) is used.

Gauge Sections

Specifying a gauge “section-list”, interpolates the rotation angle from the specified section data.

In most cases a gauge should use the scale sections defined in the indexed scale for correct positioning of the indicator.

For special purposes a gauge may have its own section list.

```
   "sections" : [
      <gauge-section>,
      ...
      <gauge-section>
   ],
```

The gauge-section requires the start and position values only.

```
      [ <start>, <position> ],
```

Example:

```
   "sections" : [
      [  10, 120.0 ],
      [  22, 310.0 ]
   ],
```

Change Instrument Behavior (not implemented yet)

The easygauge behavior can be modified in the following aspects

- How gauge values are updated from received sim values
- How gauge values are indicated

Updating gauge values and indication of gauge values has been separated for easier modification.

Modify Updating of Gauge Values

Updating of gauge values is performed by a generic gauge “update” function, which is the callback function for the subscription process. Every time a value is received the gauge “value” is set to the received value and the generic gauge indicate function (see below) is called.

Modify Indicating of Gauge Values

Indicating of gauge values is performed by a generic gauge “indicate” function, which is called by the initial display of the instrument and on every update of the gauge value. Every time a value is received the gauge “value” is set to the received value and the generic gauge indicate function (see below) is called.

JSON Syntax

The instrument specification is written using the JSON syntax. This short description just explains what is needed to know to specify “Easy Gauge” instruments.

With JSON you specify the “Easy Gauge” instrument features. Instrument features are specified as JSON items.

JSON Items

A JSON iten is either

- a JSON value or
- a JSON list or
- a JSON object.

JSON Values

JSON supports the following value types

- Numeric (Integer or Float)
- String
- Boolean
- NULL

Numeric Values

Numeric values are written as numeric literals. Either in decimal notation or hexadecimal notation

In decimal notation, Integers are just plain integers ass you commonly write them without any interpunction

```
102, 0, 35645
```

Floating point numbers can be written in plain form (with decimal point) or using the exponential form with the exponent delimieter (‘e’ or ‘E’)

```
102.0, 0.031, 356.45
1.02E2, 3.1e-2
```

In hexadecimal notation, a # sign is placed in front of the HEX digits (0..9, a..f, A..F)

```
#ad00ef
```

String Values

String values are any alphanumeric characters enclosed in double quotation marks (“).

```
"variable", "GEAR %s POSITION", "PERCENT OVER 100"
```

Boolean Values

A Boolean value is one of the values

```
true
false
```

Do not enclose them in quotation marks.

NULL Values

The NULL value can be used to indicate an unspecified JSON object field. The value is

```
null
```

Do not use NULL values in lists as LUA handles them as if an item is left out.

JSON Lists

A JSON list is specified using square brackets "[" and "]" and contains any number of JSON items, separated with a comma (","):

```
[ item1, item2, ... itemN ]
```

or written on multiple lines

```
[  item1,
   item2,
   ...
   itemN
]
```

Do not place a comma after the last item. This is a common mistake. Such a JSON file will fail on loading in Air Manager.

Each item can be any JSON item: a JSON value, a JSON list or a JSON object (see below).

```
[ "LOG", 0.05 ]
[ [ 15.0, -60.0, 5, 2.5 ], [ 35.0,  60.0 ] ]
[ { "size": 32 }, 2 ]
```

The order of the items is relevant. Item1 must be placed first and itemN must follow itemN-1.

JSON Objects

A JSON object is specified using curly brackets "{" and "}" and contains any number of JSON fields, separated with commas (",").

```
{ field, field, … field }
```

or written on multiple lines

```
{  field,
   field,
   ...
   field
}
```

Do not place a comma after the last field. This is a common mistake. Such a JSON file will fail on loading in Air Manager.

JSON Fields

A JSON object field is a key-value pair. The key is a keyword placed between quotation marks. The value can be any JSON item: a JSON value, a JSON list or another JSON object.

Key and value are separated by a colon ":" (you can add some whitespace for formatting).

```
"color": "#7fffd4"

"animation" : [ "LOG", 0.05 ]

"marks": {
   "offset": [ 288, 242, 228 ],
   "width" : 8
}
```

The order of the fields is NOT relevant. "Easy Gauges" defines a lot of default settings that allow to omit many fields.

Whitespace and Line Breaks

Any whitespace, tabs and line breaks between items and separators are ignored by the loader (parser). Use this to make the specification more readable by formatting judiciously. Indenting items/fields in multiline lists/objects

Document Revisions

Document version 1.0.0-b2

+ allow multiple redline objects (-> change redlines object to redlines list) : page 26