Creating a VSI (Vertical Speed Indicator) Using EasyGauge

Getting Started

Before we begin we need to have a few things in place.

- Photo of a real gauge to use as a reference.
- BMP of VSI gauge face (Static element)
- BMP of VSI gauge needle (Needle element)
  Important: ALL needle elements must point **East** or to the **right**. This is a MSFS requirement.
- A basic understanding of what function the real gauge performs.
- An understanding of how to create transparent areas of BMPs using an RGB value of 0,0,0. (If not, see the **Tips** section at the end of this tutorial.)

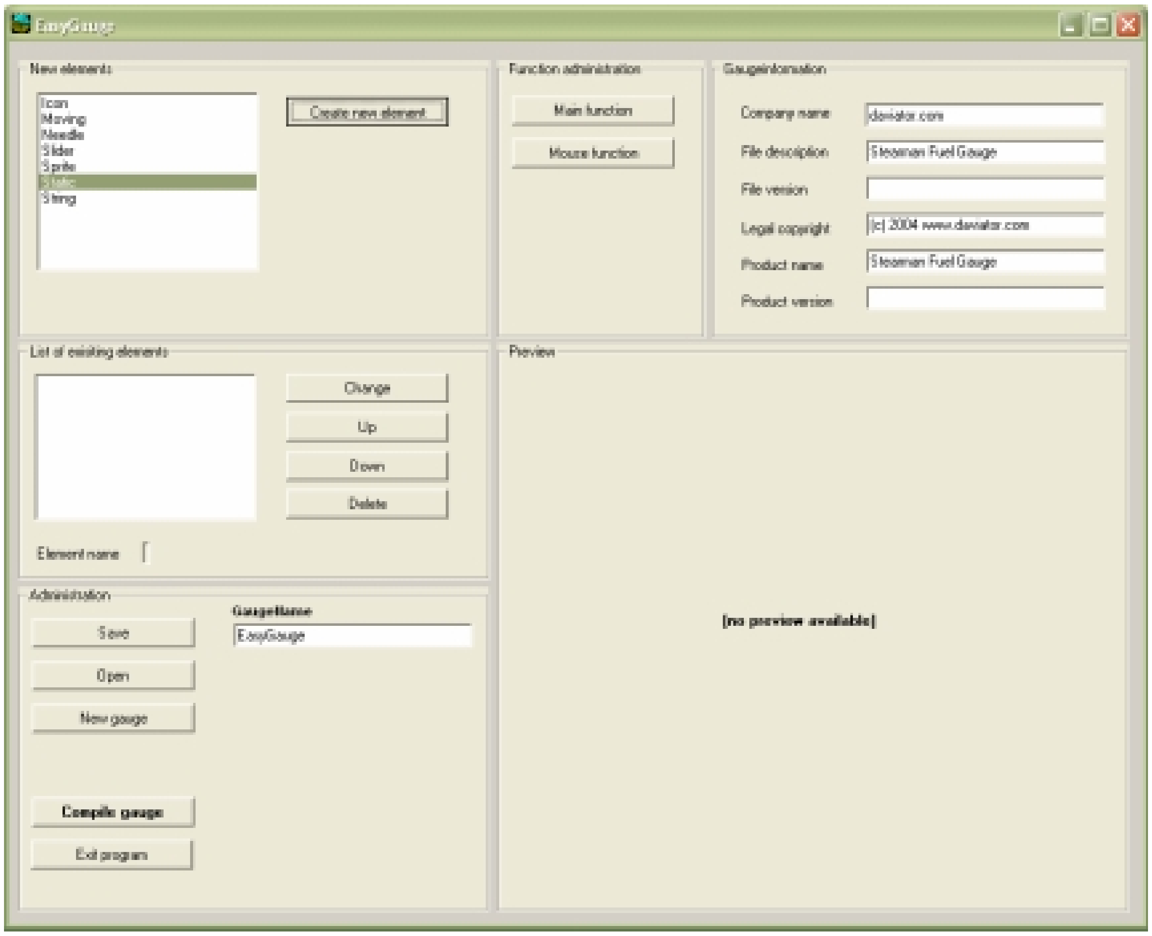
Creating the Gauge Elements

As with many functions in MSFS (Microsoft Flight Simulator) and EG (EasyGauge) there might be more than one option to choose from when creating a gauge. Trial and error might be necessary to get the desired results. I personally choose the path of least resistance. I want a working gauge not a long lesson in Algebra. Work smart, not hard.

Begin by opening EasyGauge.

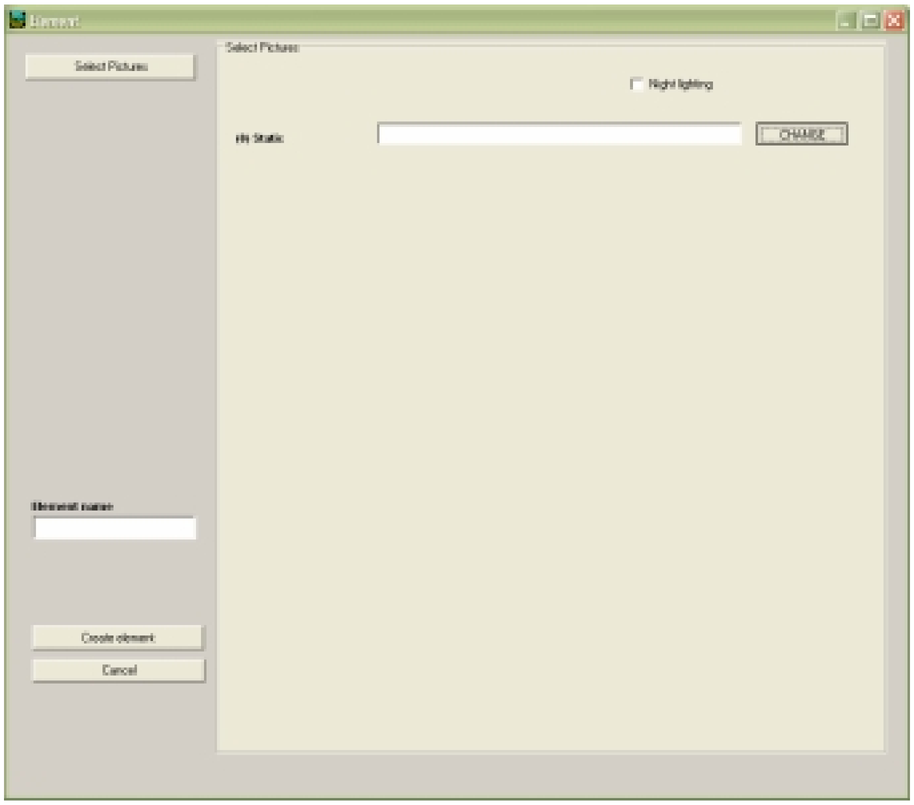
In this case we are creating a VSI or vertical Speed Indicator. This gauge measure how fast the aircraft is climbing or descending.

The first step is to select the background or ***Static*** element.
Click once on **Static** in the selection list in the **New element** area and then click on the **Create new element** button.



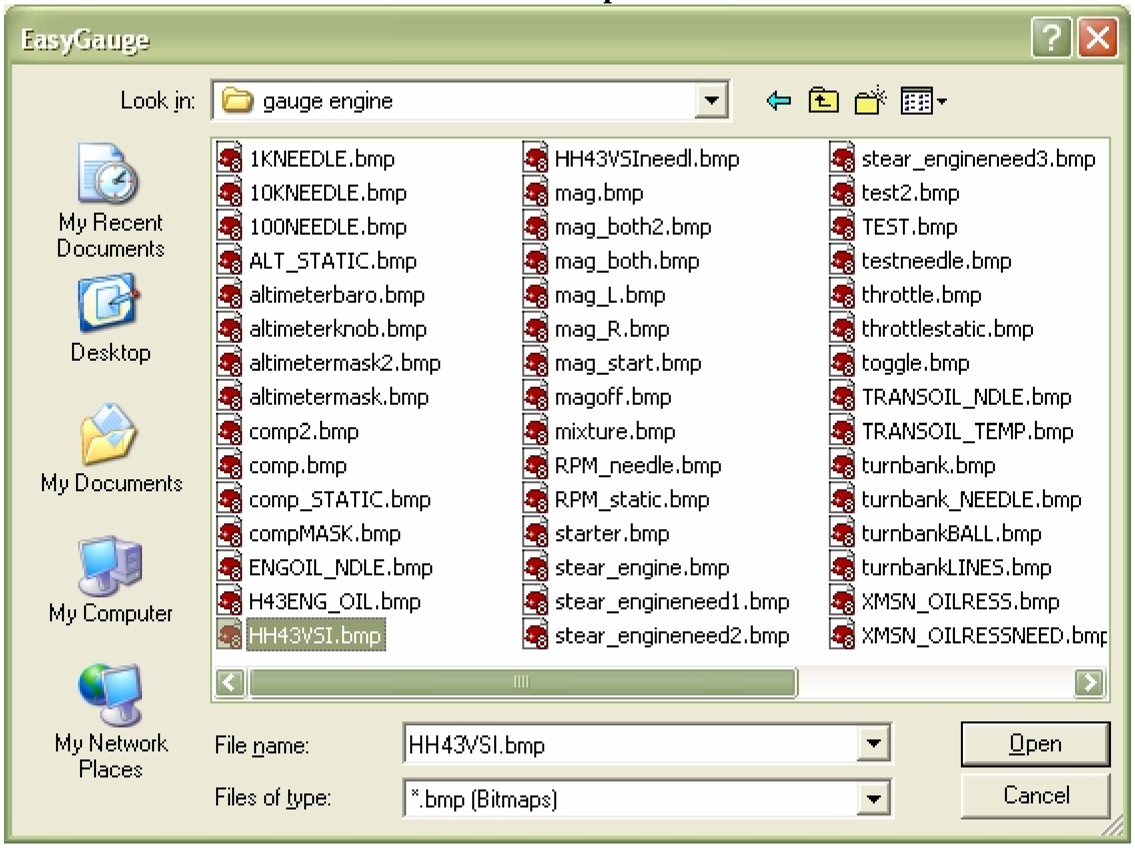
Company name: daviator.com
File description: Stearman Fuel Gauge
File version:
Legal copyright: (c) 2004 www.daviator.com
Product name: Stearman Fuel Gauge
Product version:
GaugeName: EasyGauge
Element name:
[no preview available]

Click on the **CHANGE** button on in the upper right of the **Element** window.



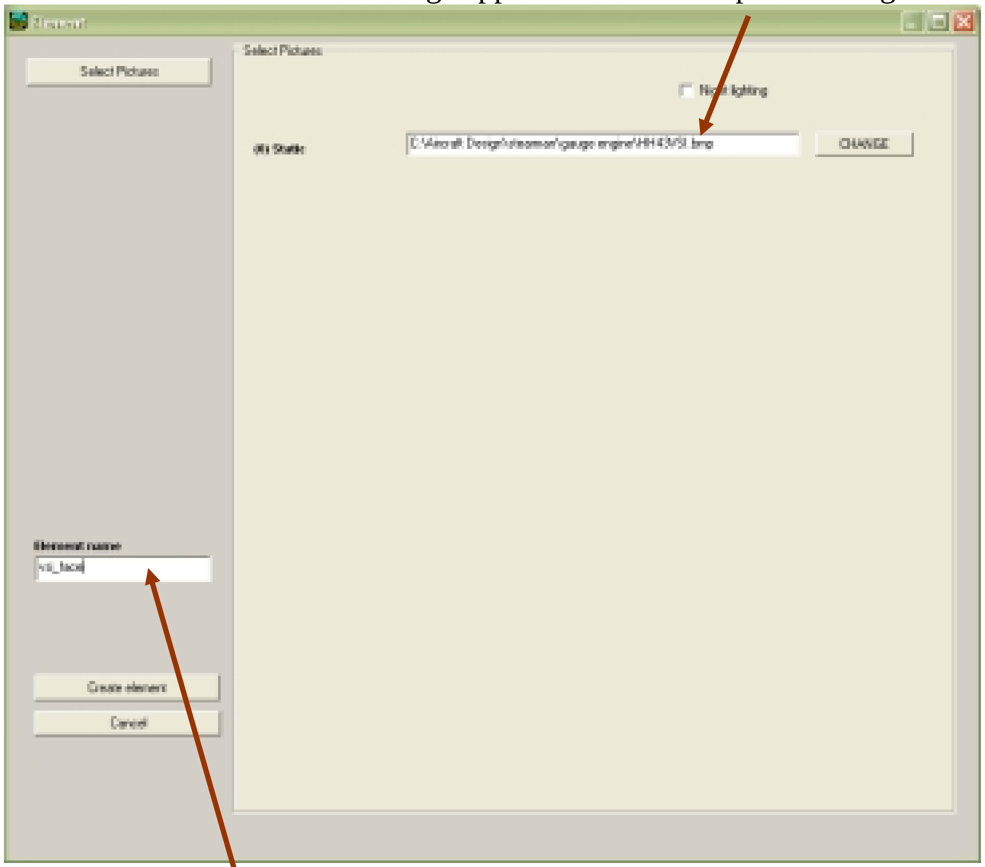
Select the BMP file you wish to use for the **Static** element of your VSI gauge.
Tip: Code always seems to work better when file names have no spaces. Use _ as a space.

Select the BMP for **Static** element and click **Open**.



Look in: gauge engine
File name: HH43VSI.bmp
Files of type: *.bmp (Bitmaps)

Note: You will NOT see the image appear but rather the path to image next to the **CHANGE** button.

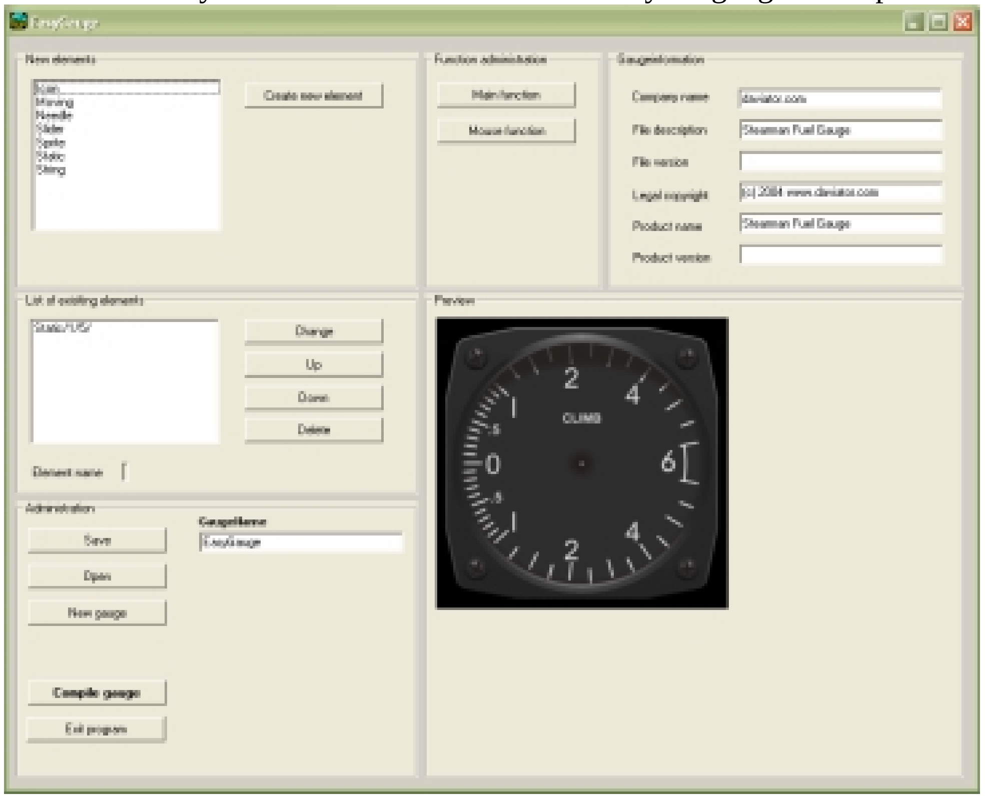


(4) Static: C:\Aircraft Design\stearman\gauge engine\HH43VSI.bmp

Element name: vsi_face

Name the **Element.** In this case I used **vsi_face**
(Naming Elements is not required but it makes working with Elements easier).

Click on the **Create element** button to create the **Static** element for your gauge**.** You are now back to the main EG screen and you can see the **Static** element for your gauge in the preview area.



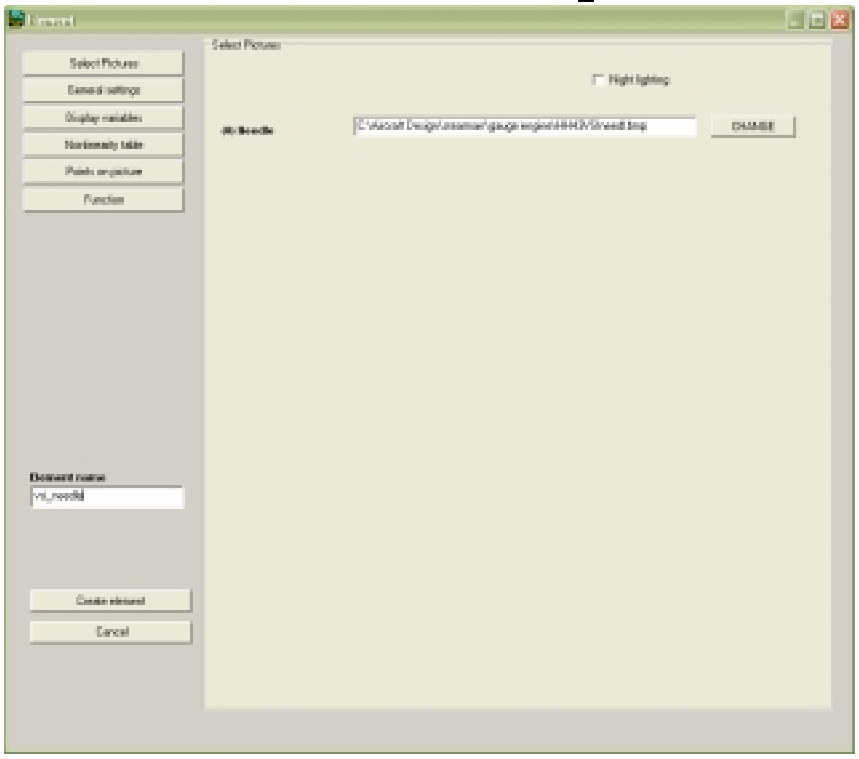
Company name	dasiator.com
File description	Stearman Fuel Gauge
File version	
Legal copyright	(c) 2004 www.dasiator.com
Product name	Stearman Fuel Gauge
Product version	

List of existing elements: Static/"VSI"

Element name:

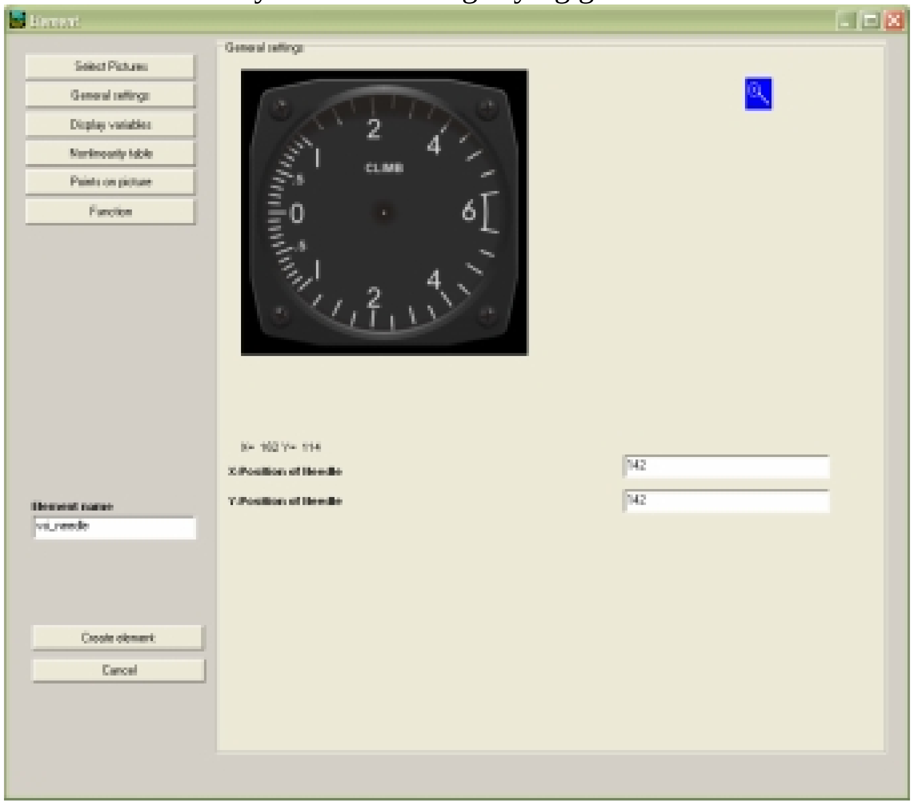
GaugeName: EasyGauge

Creating the Needle
Click on **Needle** in the **New element** area and click **Create new element**
Select the BMP for the needle the same way you selected the Static element BMP by clicking on the **CHANGE** button. As with the Static element NO image appears at this time.
Name the Element. In the case I named it **vsi_needle**



(#) Needle: [illegible]
Element name: vsi_needle

Click on the **General settings** button. This is where you tell EG at which point the needle with rotate around. Click on the point of rotation. In this case I created a small white dot in the center of the gauge. It will not be seen when the gauge is completed as it will be covered by the needle. You will notice that the X and Y values will be filled in for you. Use the magnifying glass button to zoom in one time for greater accuracy.



X= 102 Y= 114
X Position of Needle: 142
Y Position of Needle: 142
Element name: vsi_needle

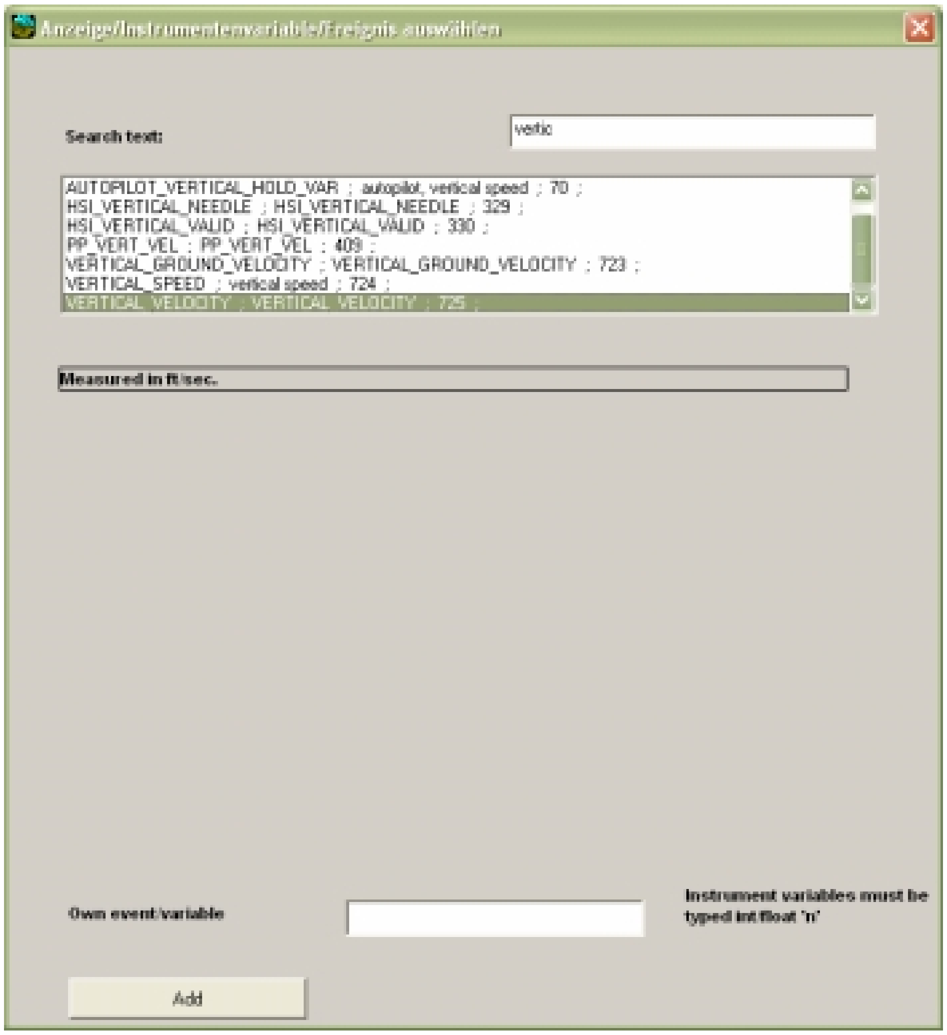
Display Variable
The Display Variable is the function EG uses to animate the gauge.
Note: As mentioned before sometimes there is more than one function that can be used to achieve the desired results.
Click on the **CHANGE** button to display the **Anzeige/Instrumentenvariable/Ereignis auswählen** window. (From here on out known as the **AIE** window).
You will notice that you cannot directly select a variable form the list but must either:

Search for one and then select it.

- Enter part or all of a search word in the **Search text** box. In the case I tried **vertic**. Notice that we have several variable with the term vertical in them. If you click on the variable it will display the unit of measure or the value returned by the function.
  I chose **VERTICAL_SPEED** first but it returned a measurement of 1/256 of a meter per second. (huh?)
  Next I tried **VERTICAL_VELOCITY**. This measures feet per second. Now that makes more sense to me.

- **Add your own.**
  You can also add your own variable.
  Note: EG is notorious for have many missing variables. Some can be located by downloading the **Panel SDK** from MS here: http://www.microsoft.com/games/flightsimulator/fs2004_downloads_sdk.asp

Click **Add** to add the variable.


Anzeige/Instrumentenvariable/Ereignis auswählen

Search text: vertic

```
AUTOPILOT_VERTICAL_HOLD_VAR ; autopilot, vertical speed ; 70 ;
HSI_VERTICAL_NEEDLE ; HSI_VERTICAL_NEEDLE ; 329 ;
HSI_VERTICAL_VALID ; HSI_VERTICAL_VALID ; 330 ;
PP_VERT_VEL ; PP_VERT_VEL ; 409 ;
VERTICAL_GROUND_VELOCITY ; VERTICAL_GROUND_VELOCITY ; 723 ;
VERTICAL_SPEED ; vertical speed ; 724 ;
VERTICAL_VELOCITY ; VERTICAL_VELOCITY ; 725 ;
```

Measured in ft/sec.

Own event/variable

Instrument variables must be typed int/float 'n'

Now that we have a **Static** image, the **Needle** and the **Variable** for the gauge it's time to make it work. But at this point we have to stop and make some decisions.

A Few Things to Note:

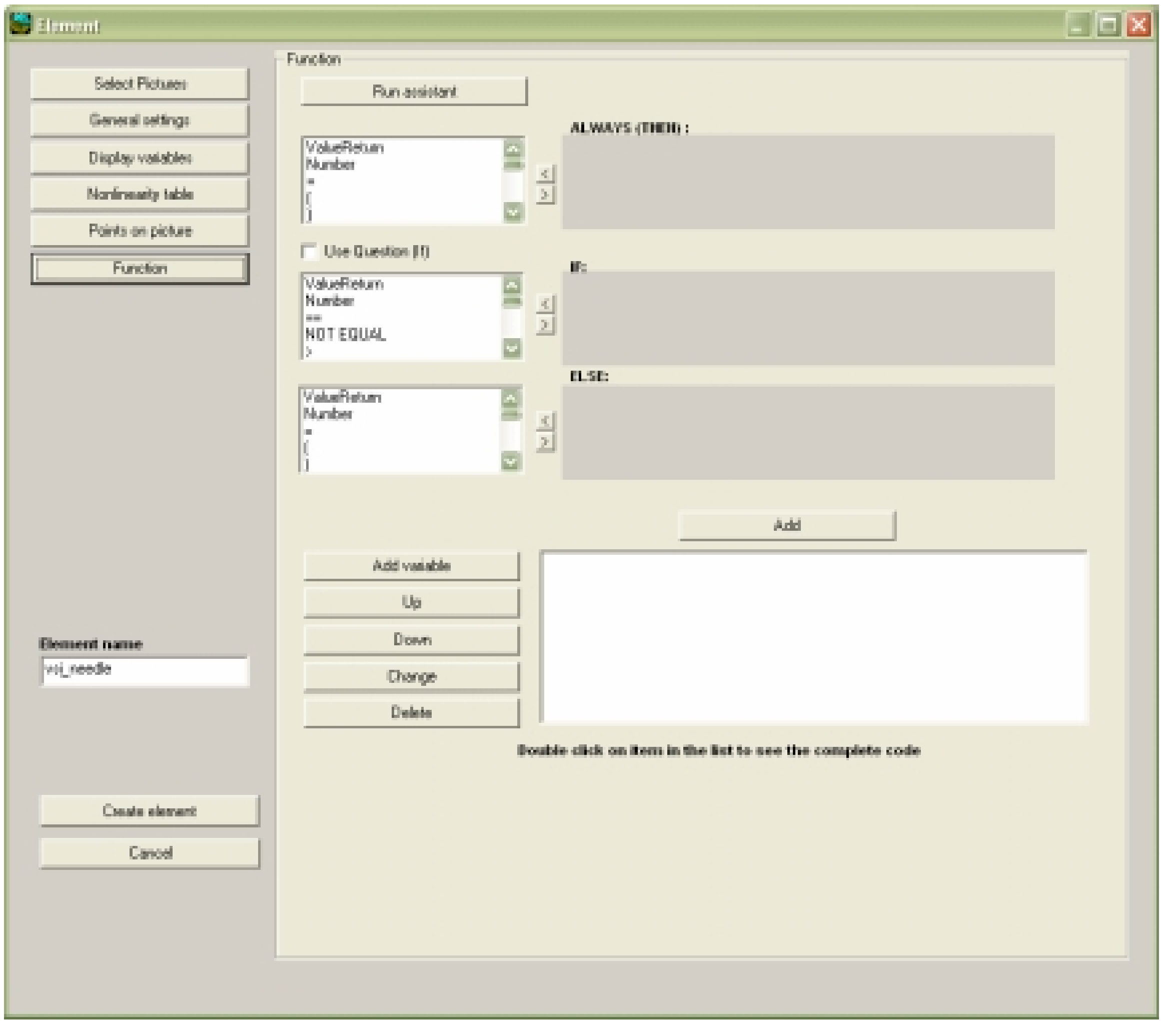
- The real gauge measures **feet per minute (fpm)** but remember our variable measures **feet per second (fps).**
- The needle on the real gauge stops at **+6000 fpm** and **-6000 fpm** *even if the velocity of the aircraft exceeds these speeds.*
- Negative numbers are involved. (Apparently EG doesn't always work correctly with negative numbers.)

To proceed we need to do the following:

- Convert fps to fpm. *Simple*: fpm=fps * 60 (seconds)
- Figure out a way to trick EG into thinking all numbers are positive. Let's see, our lowest value is -6000 fpm so if we add 6000 the lowest value it becomes 0. (This will make more sense later on.)
- Find a way to stop the needle at +6000 fpm and -6000 fpm

Adding a Function

The best way to perform the steps above to create a **Function**.
Skip the **Nonlinearity table** and **Points on picture** buttons for now and click on the **Function** button.
The **Function** window now appears. It looks daunting but don't worry it is.



Element name: vsi_needle

This is where we can add custom items to our gauge to make it function correctly.

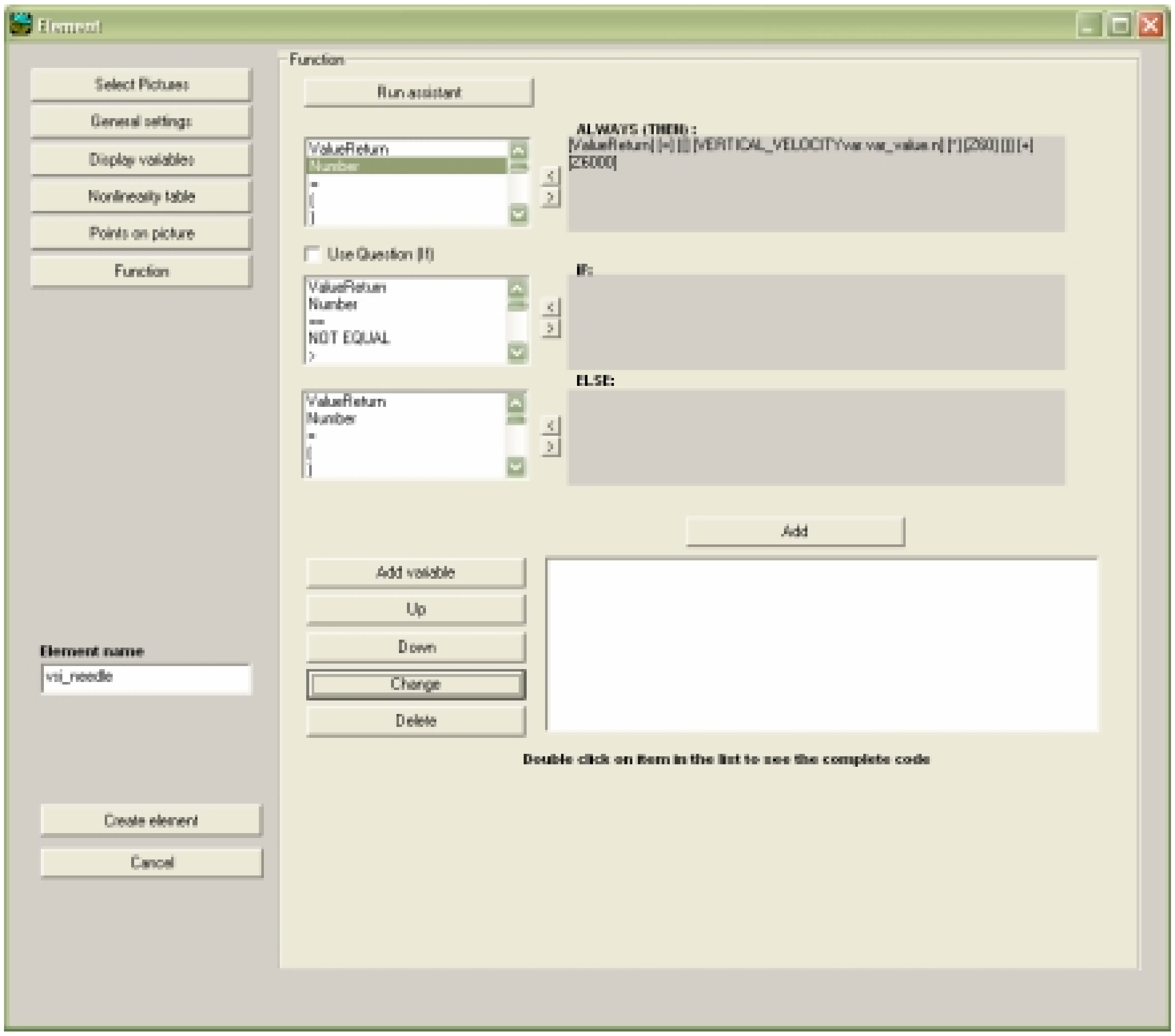
Let's start by converting fps to fps and fooling EG into thinking that the starting point of our gauge is 0.

The number returned or displayed by the gauge is called the **ValueReturn**.
So what we want to say is: **ValueReturn = fps * 60 (seconds) + 6000 feet (to get a starting point of 0)**

To add the custom code select the item you wish to in the list next to the **AWAYS (THEN) :** window.
Click on the right arrow button > to add the item.
Note: brackets [ ] will be inserted automatically by EG. Also numbers will have a leading Z added as well.

So for this process we added:
ValueReturn
=
(
Add flightsimulator variable (we chose our variable of VERTICAL_VELOCITY)
*
Number (key in 60 for 60 seconds)
)
+
Number (key in 6000 to make starting point 0 and eliminate negative numbers)

Click **Add** to add the line of code.



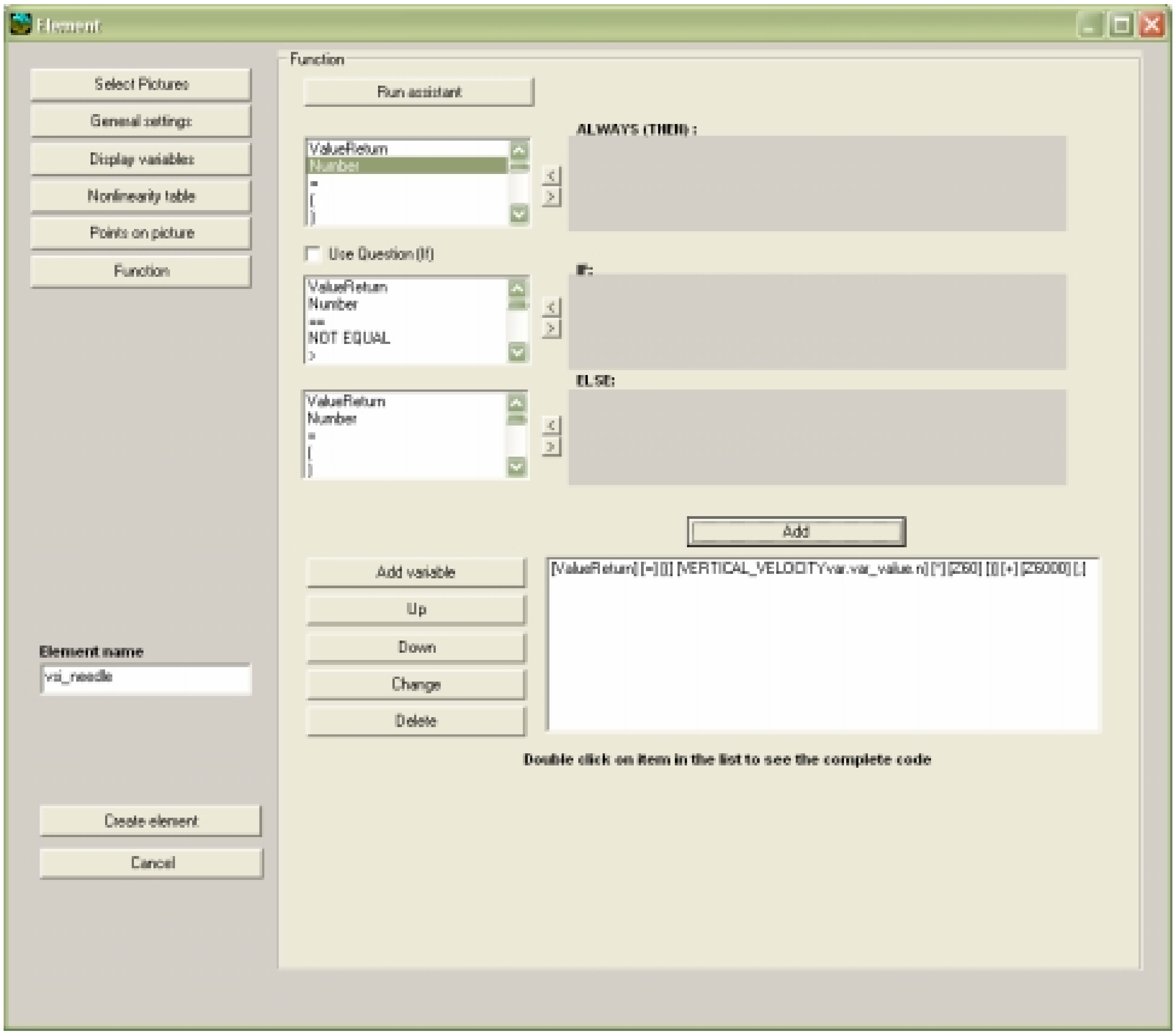
```
[ValueReturn] [=] [(] [VERTICAL_VELOCITY\var:var_value.n] [*] [Z60] [)] [+]
[Z6000]
```

Notice that your function has now been added below the **Add** button.
The completed code looks like this:

```
[ValueReturn] [=] [(] [VERTICAL_VELOCITYvar. var_value .n] [*] [Z60] [)] [+] [Z6000] [ ; ]
```

To recap: What the needle shows = variable * 60 seconds + 6000 feet

To change the code, click one on it and then click on the **Change** button.



```
[ValueReturn] [=] [(] [VERTICAL_VELOCITYvar.var_value.n] [*] [Z60] [)] [+] [Z6000] [;]
```

Element name: vsi_needle

Double click on item in the list to see the complete code

Limiting the Needle Travel
Next we need to add the functions that limit the range the needle will travel.

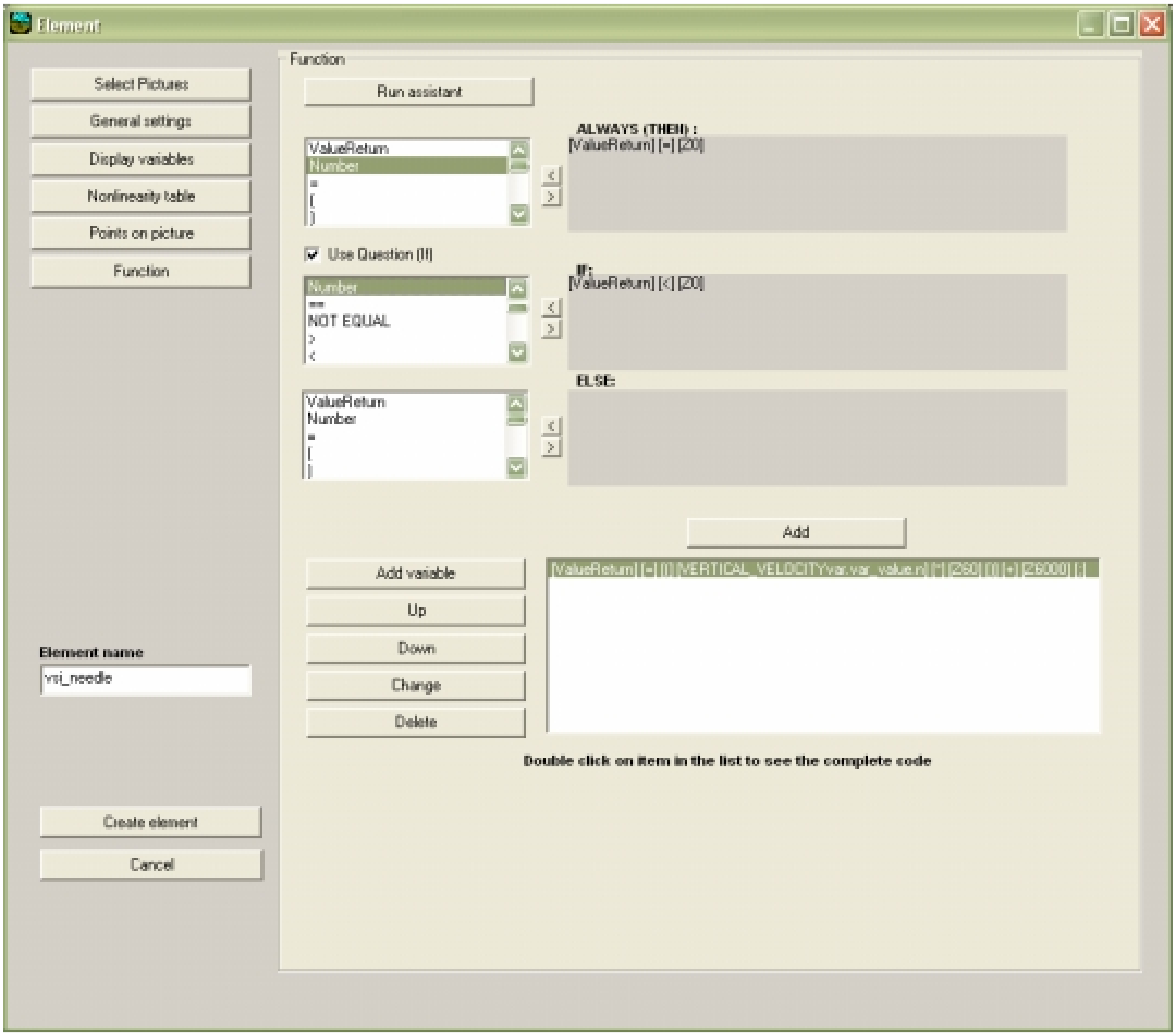
Remember we want the needle to stop at +6000 fpm and -6000 fpm so we need to add another function.

Since we told EG that the starting point is 0 (remember we added 6000 to the variable) we also want tell RG that the needle should never go below 0.

So we would say: If the ValueReturned in less than then 0 then ValueReturn = 0

We need to create an “IF” statement. With EG you must enter the FIRST part of the IF statement in the SECOND window. It seems backwards to what might seem intuitive.
Note: Be sure to check ☑ the **Use Question [If]** box when creating an IF statement.

Let’s set the stop at the -6000 point.
Enter the values as shown below. Click **Add** when completed.
Remember that we told EG that the starting point is 0 and not -6000



ALWAYS (THEN) :
[ValueReturn] [=] [Z0]

IF:
[ValueReturn] [<] [Z0]

ELSE:

[ValueReturn] [=] [(] [VERTICAL_VELOCITYvar.var_value.n] [*] [Z60] [)] [+] [Z6000] [;]

Double click on item in the list to see the complete code

Element name
vsi_needle

The new line should look like this. [IF] [ValueReturn] [<] [Z0] [then] [ValueReturn] [=] [Z0] [;] [}]

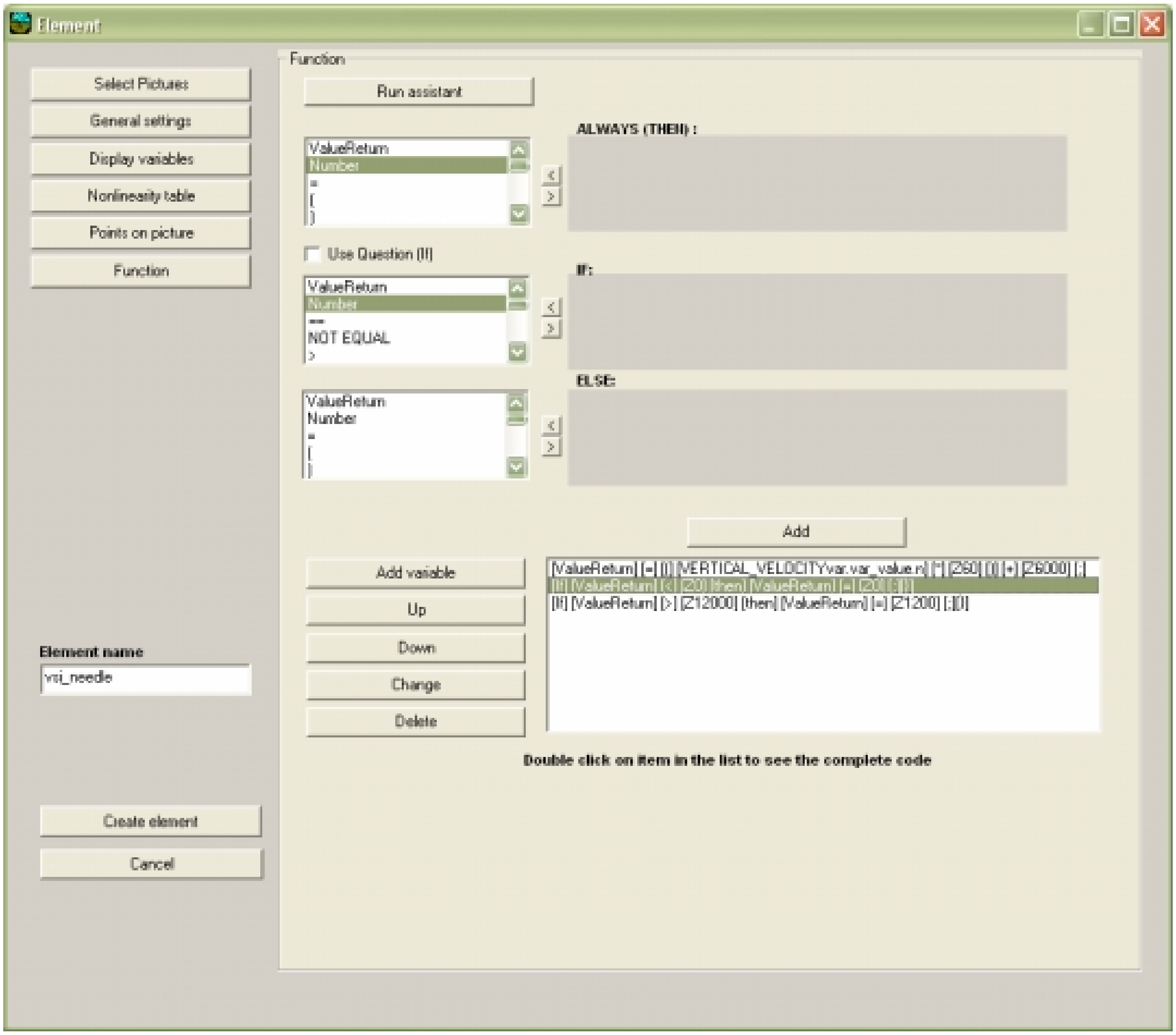
Finally we need to limit the swing in the positive direction.
On the real gauge the difference in needle direction is from -6000 fpm to +6000 fpm or a total distance of 12000 fpm. Since we have fooled EG into thinking that we're starting at 0 we must tell it that needle must go no further than 12000 fpm. As with the last statement we need to create an IF statement.

So we would say that IF the ValueRetun is greater than 12000 then ValueRetun = 12000

In EG the statement should look like the following:
[IF] [ValueReturn] [>] [Z12000] [then] [ValueReturn] [=] [12000] [;] [}]

Below we see the **Function** window with three statements that do the following:

- Convert FPS to fpm + 6000 to set start point at 0
- Limit needle travel in negative direction
- Limit needle travel in positive direction



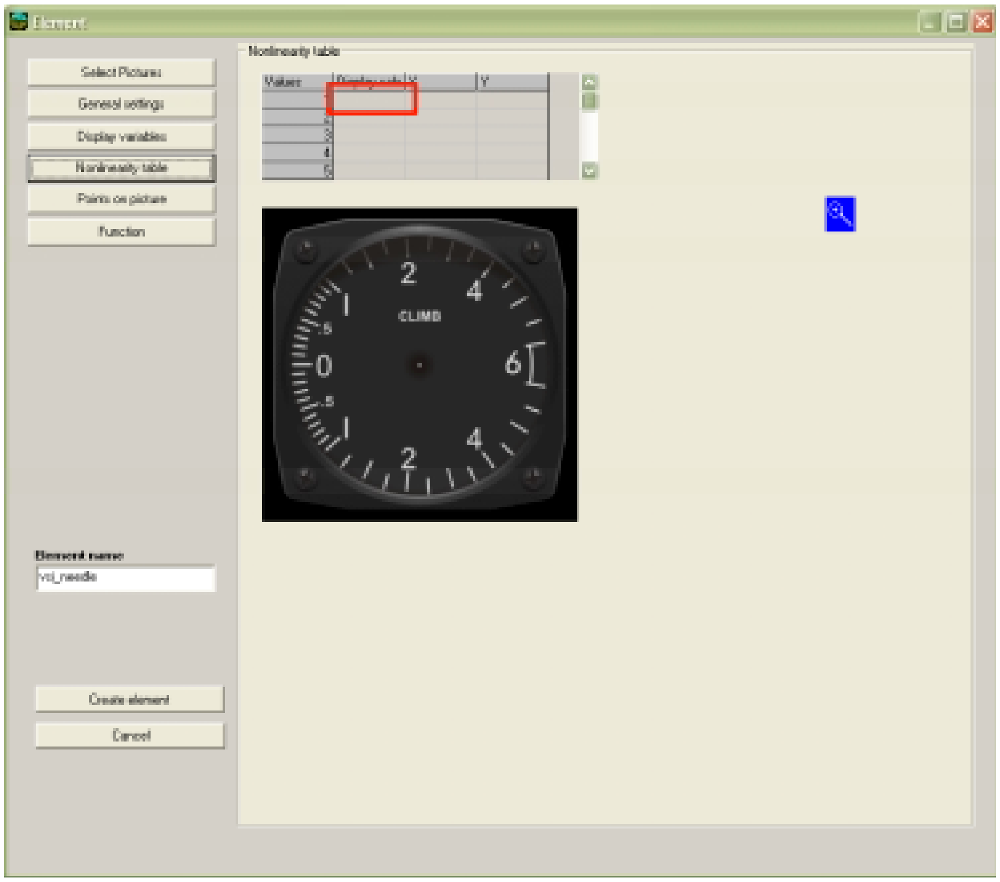
```
[ValueReturn] [=] [(] [VERTICAL_VELOCITYvar.var_value.n] [*] [Z60] [)] [+] [Z6000] [;]
[If] [ValueReturn] [<] [Z0] [then] [ValueReturn] [=] [Z0] [;] [}]
[If] [ValueReturn] [>] [Z12000] [then] [ValueReturn] [=] [Z1200] [;] [}]
```

Element name: vsi_needle

Double click on item in the list to see the complete code

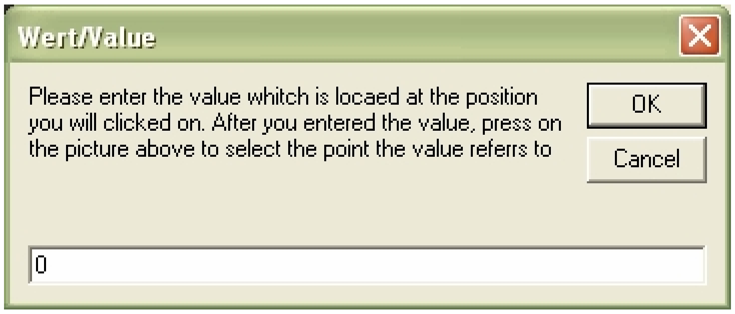
Nonlinearity table
We now need to tell EG where on the Static the needle should point for the different vertical speeds.

Click on the **Nonlinearity table** button
You will now see the Static along with a table to enter the values for needle speeds.
Notice that on this gauge that first 1000 feet in either direction have more lines than then rest of the gauge.
To define a point click in the table's first column and row. (outlined in **RED**)



Element name: vsi_needle

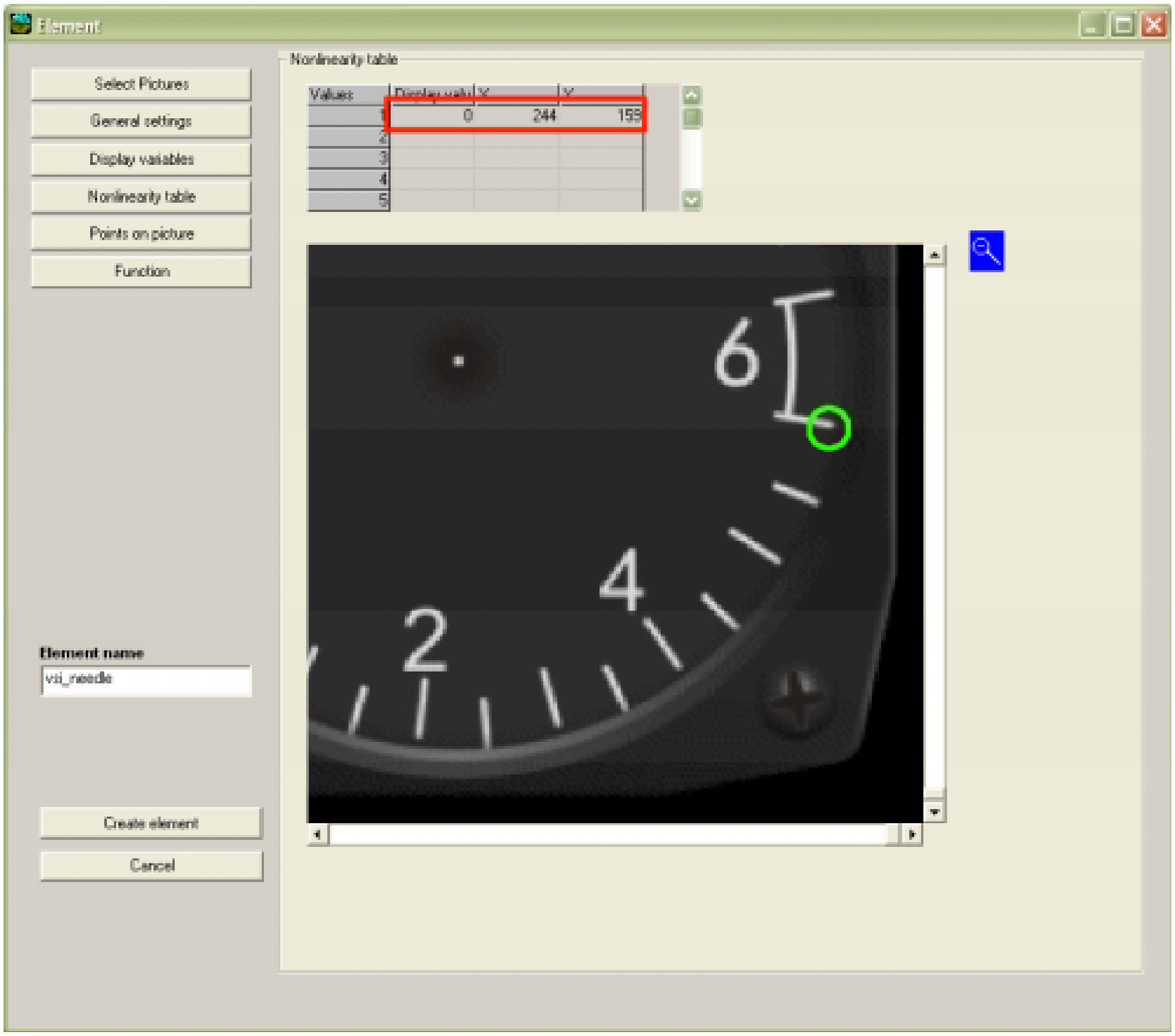
Enter a value for the needle position.
Let's start at the low end of the scale and work our way up. Enter 0
Click **OK**.



Please enter the value whitch is locaed at the position you will clicked on. After you entered the value, press on the picture above to select the point the value refers to

0

Click on the point on graphic where 0 would be displayed. (circled in **Green**) Remember, we set -6000 to be 0. Use the magnifying glass to zoom in if necessary.
Notice that once you click on the graphic the X and Y values are automatically entered by EG. (Outlined in **Red**)



Nonlinearity table

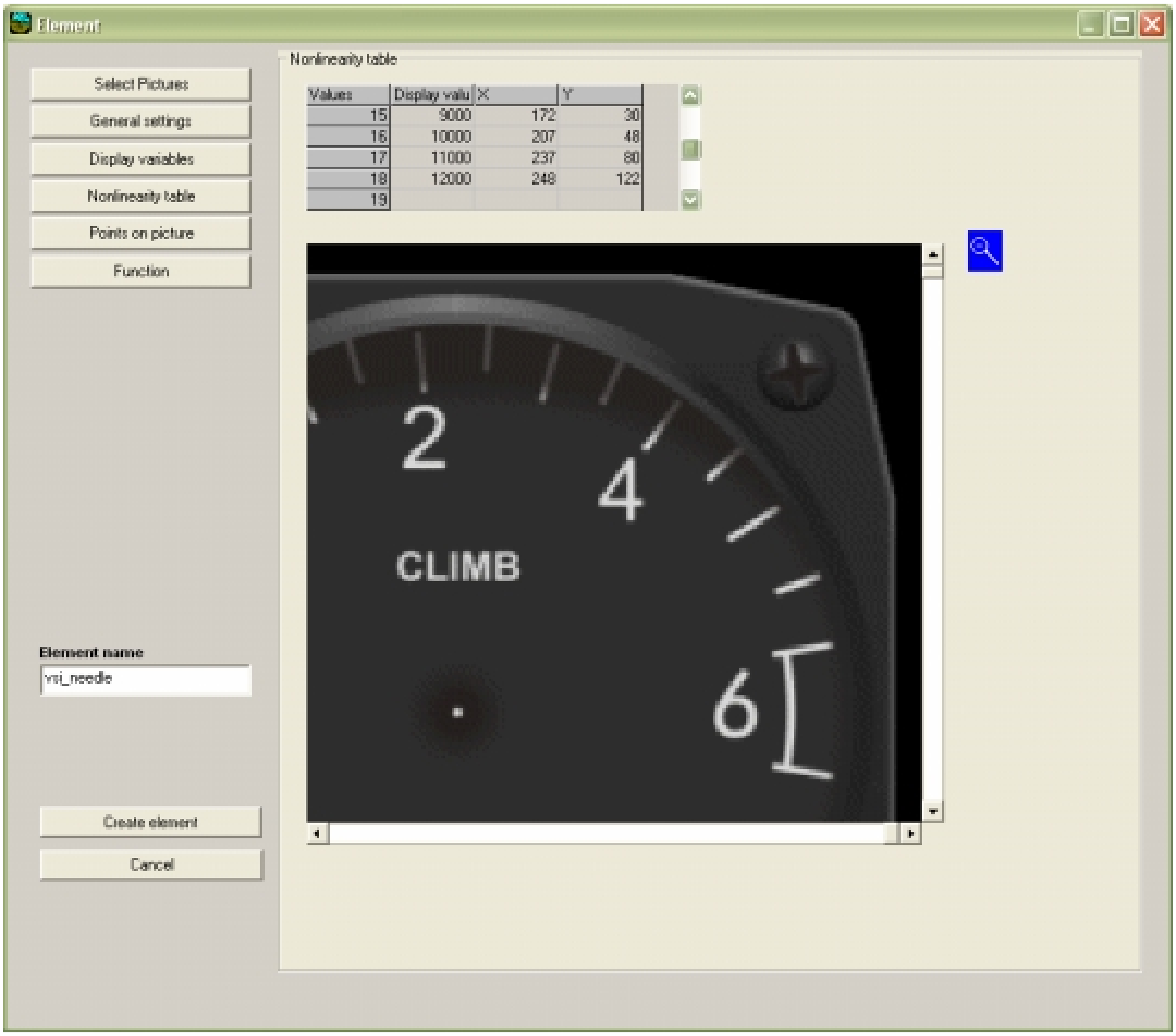
Values	Display valu	X	Y
1	0	244	159
2			
3			
4			
5			

Element name: vsi_needle

Continue the process to complete the gauge.
Important: Remember that you started at 0 so -4000 fpm on the gauge face should be entered in the table as 2000, -2000 fpm should be entered as 4000, 0 fpm is entered as 6000 and so on.

Note: The more points you define the more accurate the gauge. For the larger numbers 1000-6000) I defined points every 1000 fpm. For 0-1000 I define points every 200 fpm.

When completed, your Nonlinearity table should look like the following.
Your X and Y numbers may not be the same as below but they should be close.



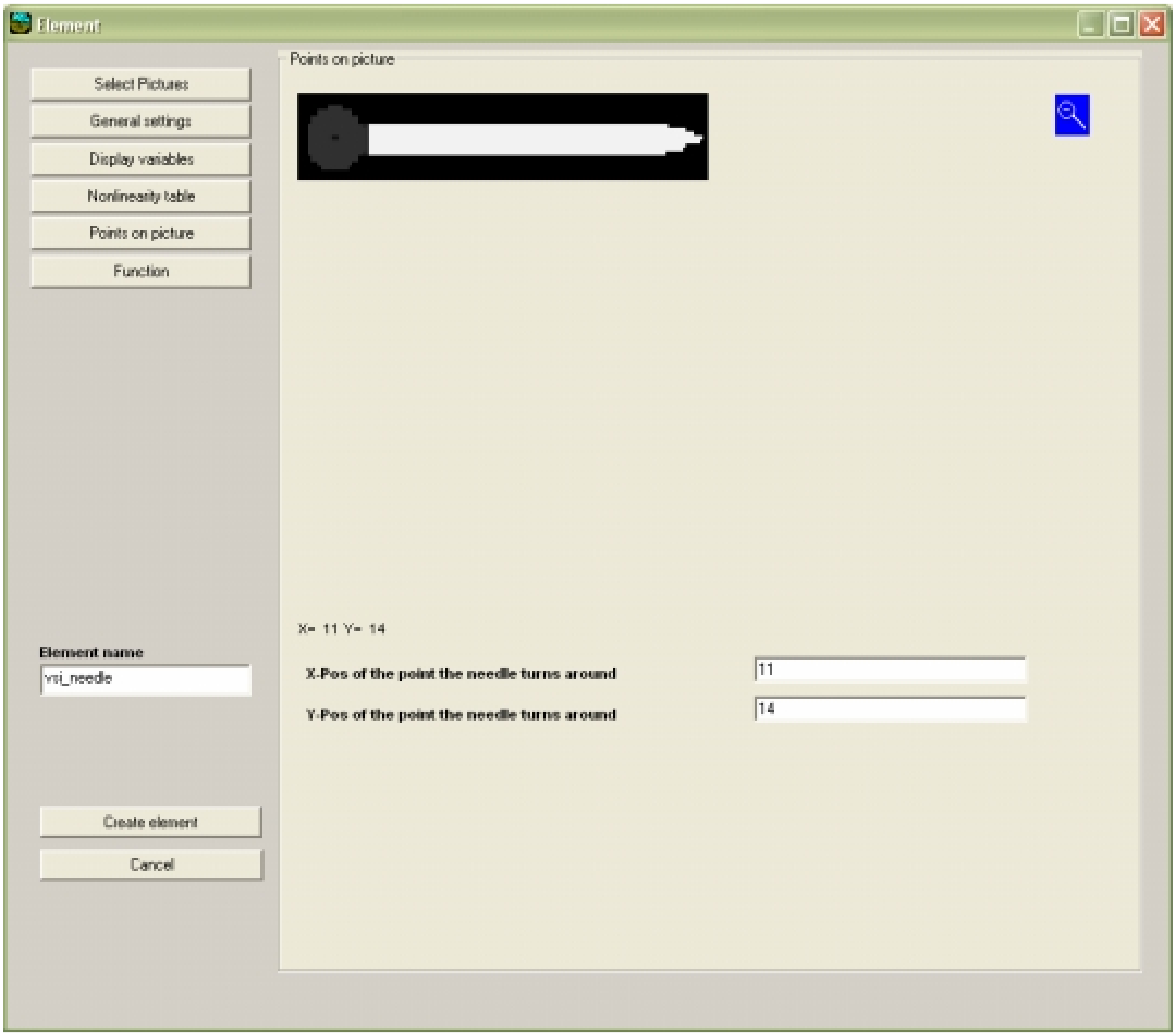
Values	Display valu	X	Y
15	9000	172	30
16	10000	207	48
17	11000	237	80
18	12000	248	122
19			

You can also change a value by clicking in the box in the nonlinearity table and reentering the needle position value.

Note: Once you enter a value and click on the Static, the X and Y values will continue to change if you continue to click on the Static. To select the next value you must click on the next line in Nonlinearity table and enter the next value.

Points on Picture
We have one more simple but crucial step before we can compile and test our gauge. We must tell EG where the pivot point is on the needle. Remember we did that for the Static earlier.

Click on the **Points on picture** button.
Click on the point where the needle will pivot. (Just like we did for the center of the Static)
Notice that the X and Y positions will be entered for you.
Use the magnifying glass to zoom in.
Once completed click **Create element**



Element name: vsi_needle

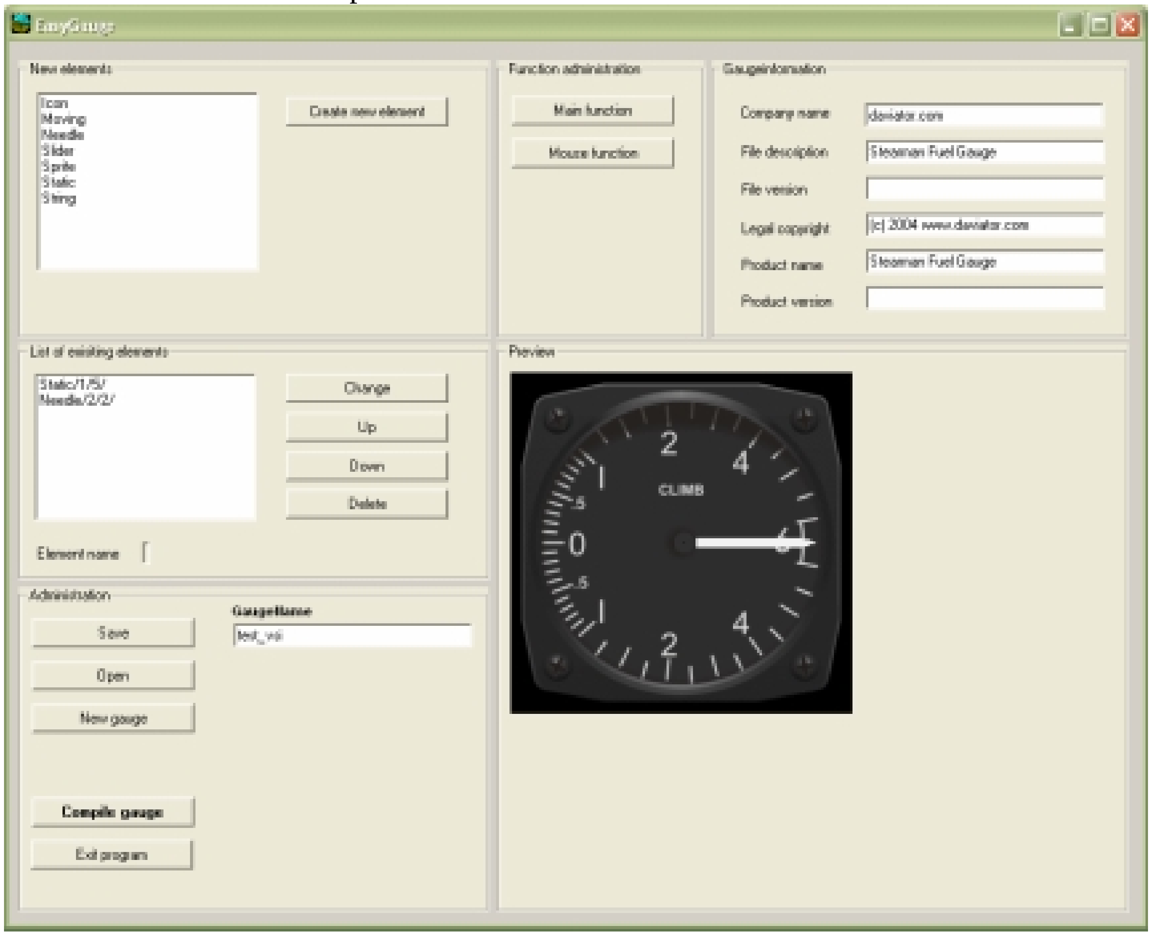
X= 11 Y= 14

X-Pos of the point the needle turns around: 11

Y-Pos of the point the needle turns around: 14

You should now be retuned back to the main EG screen.
You should see your needle on your gauge and it should be pointing due East.

If you need to adjust the needle location or pivot points you can select the element (Static/1/5/ or Needle/2/2) that you wish to change and click on the **Change** button. There you can redefine the General settings for the Static and/or the Points on picture for the needle.

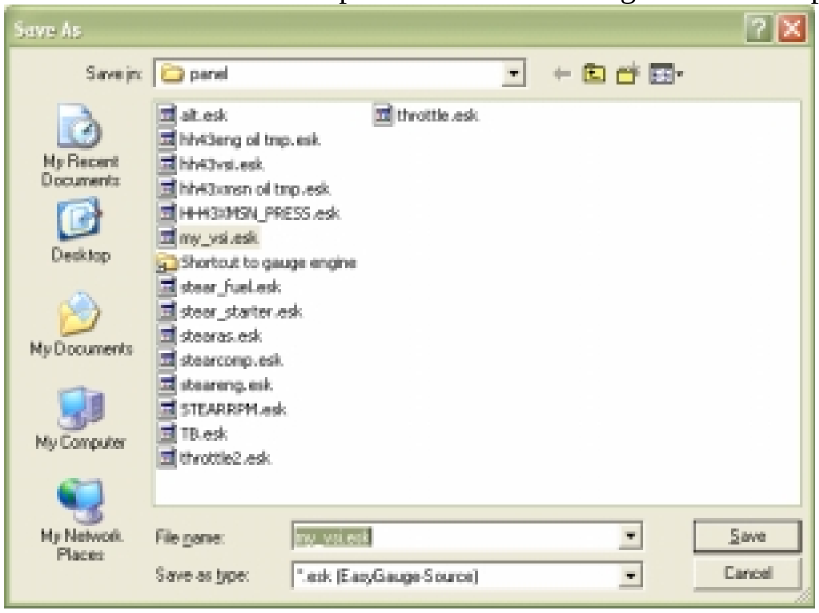


Company name: daviator.com
File description: Stearman Fuel Gauge
File version:
Legal copyright: (c) 2004 www.daviator.com
Product name: Stearman Fuel Gauge
Product version:

Static/1/5/
Needle/2/2/

GaugeName: test_vsi

At this point let's name and save the gauge before we compile it. All we need is for the compile to error out and lose all of our work I named the gauge **test_vs**
Click the **Save** button to open the **Save As** dialog. I named the project **my_vsi**



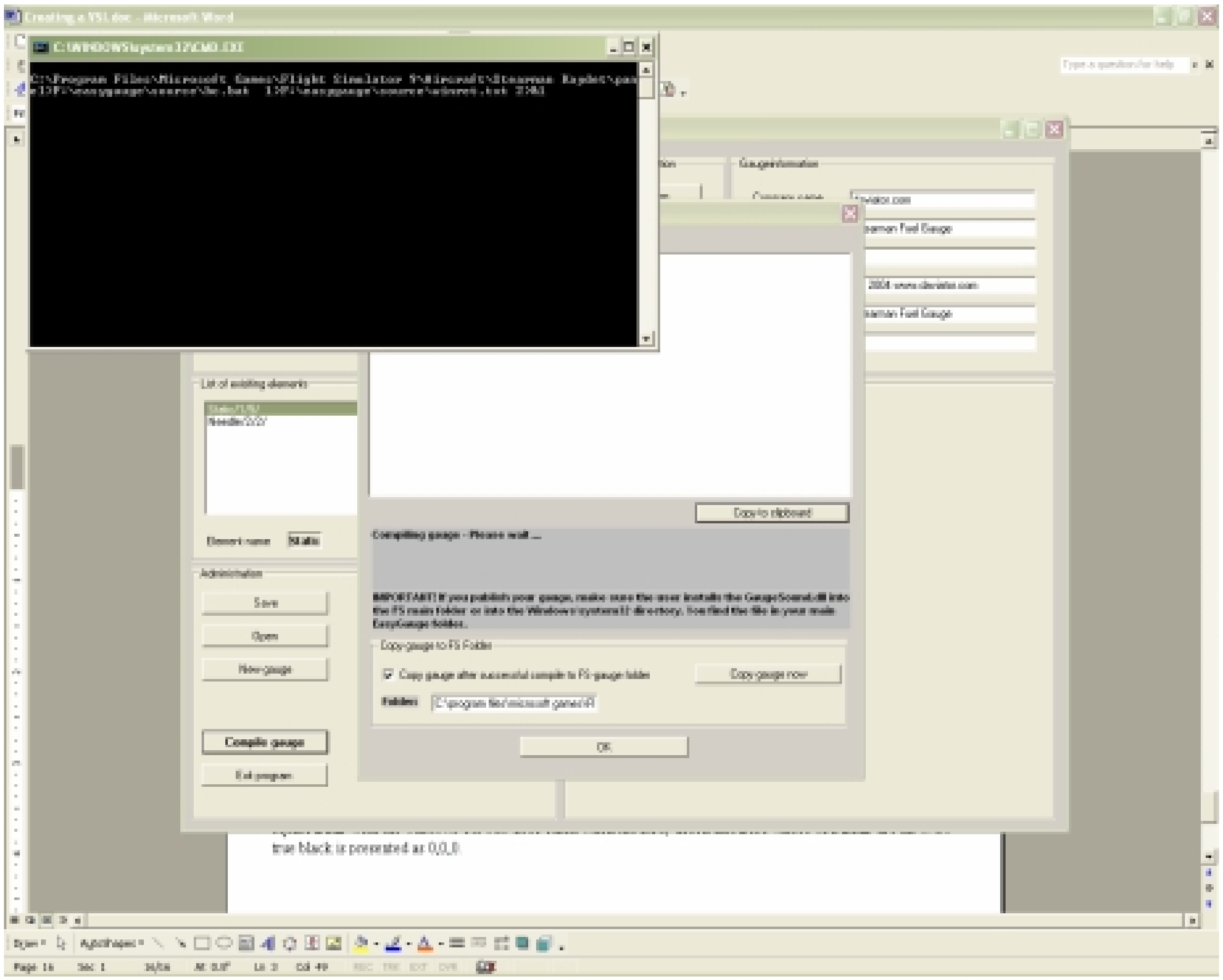
alt.esk
throttle.esk
hh43eng oil tmp.esk
hh43vsi.esk
hh43xmsn oil tmp.esk
HH43XMSN_PRESS.esk
my_vsi.esk
Shortcut to gauge engine
stear_fuel.esk
stear_starter.esk
stearas.esk
stearcomp.esk
steareng.esk
STEARRPM.esk
TB.esk
throttle2.esk

File name: my_vsi.esk
Save as type: *.esk (EasyGauge-Source)

Compiling the Gauge

Click on the **Compile gauge** button

You will see the **Compiler** window and possibly a DOS window or two appear while the gauge is being compiled.



```
C:\Program Files\Microsoft Games\Flight Simulator 9\Aircraft\[illegible]
[illegible]\easygauge\source\[illegible].bat  [illegible]\easygauge\source\[illegible].txt [illegible]
```

Compiling gauge - Please wait ...

IMPORTANT! If you publish your gauge, make sure the user installs the GaugeSound.dll into the FS main folder or into the Windows\system32 directory. You find the file in your main EasyGauge folder.

Copy gauge to FS Folder

☑ Copy gauge after successful compile to FS-gauge folder

Folder: C:\program files\microsoft games\F

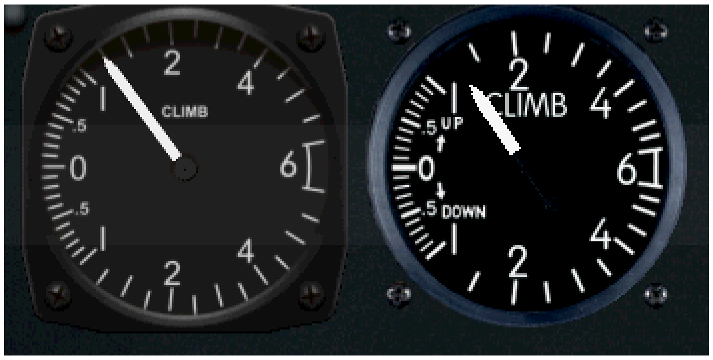
true black is presented as 0,0,0

When completed you should see "**Gauge compiled successfully"** in the grey area of the **Compiler** window. Make sure that the **Copy gauge after successful compile to FS-gauge folder** is checked ☑ and that the path to your GAUGES folder is correct.

Note: If you receive errors go back and check all of your function statements for missing items.

Add the gauge to you panel and give it a whirl. I usually use one of the default gauges next to mine to verify that I'm at least close.

Success!



Tips:

BMP Resolution and Size

As with any graphic the lower the color depth the less memory is used. More memory used in MSFS translates to lower frame rates. Whenever possible, convert imaged for gauges to 256 colors.

I keep my gauges around the 256x256 pixel size. That seems to work for me but others may have a different opinion.

Preparing the BMPs
Since BMP files are square or rectangular in order for the gauge elements to display correctly in FS some parts of the BMPs need to be made transparent. This includes but may not be limited to the outer area of the gauge face and the needle.

In the panel view in MSFS any color that is true black is rendered as transparent. This how we can have a square BMP with the windows cut out. True black is when the Red, Green and Blue values of a BMP are all 0. So true black is presented as 0,0,0. You can create black by using a color value of 1,1,1 This is indiscernible from true black to the human eye.

Save, Save, Save
Save your files as you go along. If you wish to do some major modifications to an existing gauge, do a Save As and create a second version of the gauge.

Acknowledgments
Not knowing if they care for me to use their real names I will stick with the screen names used on the Easy Gauge forum.

Special thanks to the following:
Ddawson
Frbill
Lonny
And anyone else who takes the time to answer questions on the EG or any other forum.

Marcel Burr and his team for creating EG and getting us non-programmers one step closer to the perfect flight simulator model.