Commander Philippe Verhaege, French Navy

Born in 1958.

Join the Navy 1976.

Flying school C47 Dakota 1977-1978.

BR1050 Alizé, carrier organic aircraft 1978-1982.

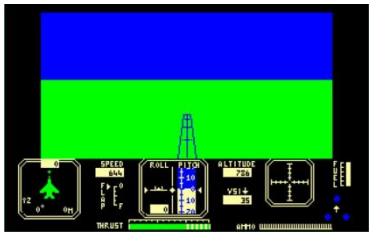
BR1150 Atlantic 1983-1992.

Atlantique 2, 1993-2002.

More than 6000 flying hours.

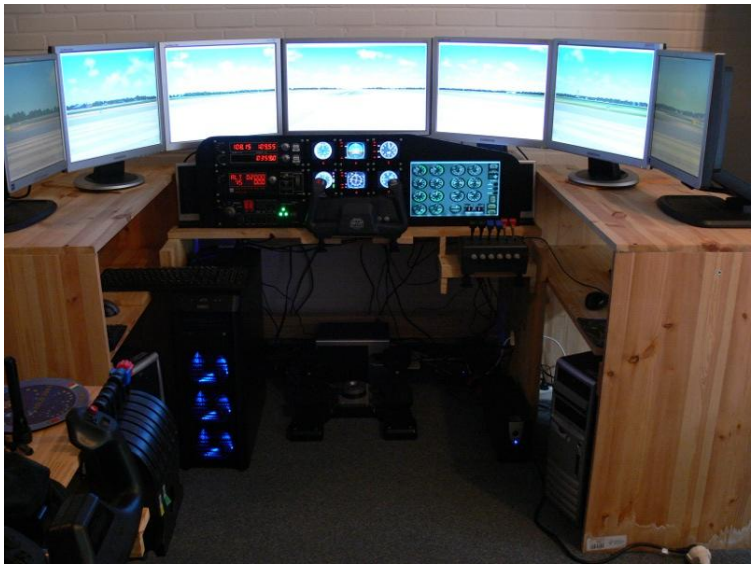
History.

My first flight simulator, in 1984 was the Fighter Pilot software running on an Amstrad CPC 464.



Then in 1989 it was F16 Fighter Pilot running on an AMSTRAD CPC6128 ,

Later I used Microsoft FS4 running on a 486 sx33 upgraded with a DX2 66 overdrive. Followed by FS5, FS 5.1, FS95, FS98, FS2000, FS2002, FS2004/FS9 and finally FSX. The last runs on a powerful computer linked with two others to let me enjoy my VFR simulator:



It was impossible for me to create a gauge when FS was made of .dll files. When the software shifted to xml files it was an unexpected pleasure to discover that files and images could be adapted for personal needs. This is when I started to create my own FS gauges and later gauges for Saitek FIP.

Creating gauges for Saitek Flight Instrument panel

by Philippe Verhaege philvge@gmail.com

To create gauges for FIP and for FSX too I use three free software and Microsoft PowerPoint which is very convenient to create accurate circular indexes and 3D objects,

The free software are The gimp, for drawing http://www.gimp.org/

Note pad ++ for the xml files, http://notepad-plus-plus.org/fr/

MB ruler to calculate angles, http://www.markus-bader.de/MB-Ruler/

Before starting you must know that not all the functions for FSX are working for a FIP, as for select/case function, the text function, visibility criteria doesn't work for all values, it works sometimes when the unit is bool like this :

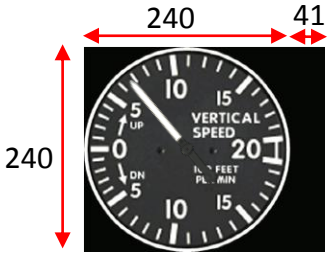
<Visible>(A:HSI GSI needle valid, bool) 1 ==</Visible>.

The syntax used in the xml file is the same as for FS9/2004. However more simple than in FSX.

What works for the FIP ? The shift and rotate functions but roughly everything can be done with these two functions. For example instead of using the select/case or the visibility functions you can use the shift function by writing that when the bitmap (this is the wording I will use to designate a picture) must not be visible then it has to move (shift) outside of the panel.

The maximum available space for a panel is 320 pixels (width) by 240 pixels (height), If you want to have some text near the FIP buttons then you have to reduce the width of the gauge.

If you want to center the gauge in the FIP you have to adapt the size of the gauge accordingly in this example the circular gauge size is 240 by 240 pixels. To have the gauge centered, the background bitmap must have the size 281 by 240 and the gauge positioned on the left side of the background bitmap. Whatever is the size of the bitmap it will be aligned on the right lower side of the FIP, therefore the number of pixels on the right lower side of the gauge will allow to move the gauge to the left side, more details to follow.



281 by 240 pixels

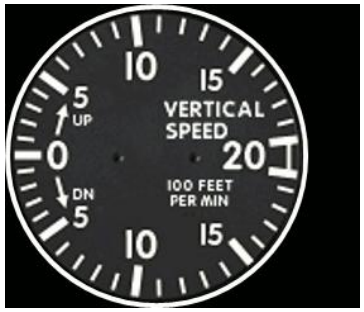
The final result will be this.

This paragraph concerns the position and identification of the back ground bitmap.

```
<?xml version="1.0" encoding="utf-8"?>
        <Gauge Name=" Baron Vertical Speed Indicator" Version="1.0">
        <Image Name="Baron_vertical_speed.bmp" ImageSizes="281,240,281,240"/>
```

We have two cases. First case the bitmap is 281 pixel wide and 240 pixel high, like this:

ImageSizes="281,240,281,240"/>

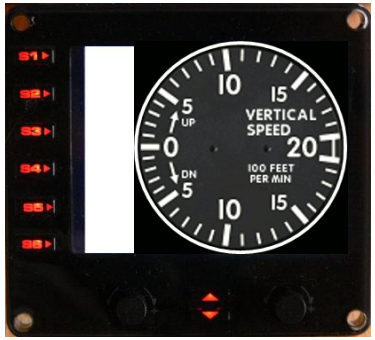


The bitmap will align with the right side of the FIP and will be centered.

Second case, the bitmap is 240 pixel wide And 240 pixel high, like this:

ImageSizes="240,240,240,240"/>

The bitmap is aligned with the right side of the FIP.



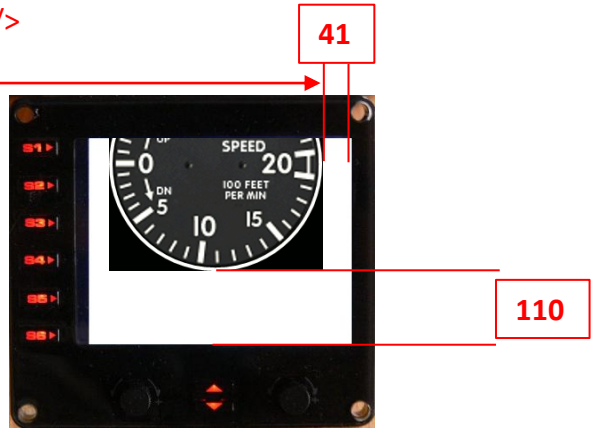
For the second case, to get the bitmap centered we change the Image size values like this
<Image Name="Baron_vertical_speed.bmp" ImageSizes="281,240,281,240"/>

The bitmap must be positioned here at 41 pixel from the right limit of the FIP.

To get the bitmap in the center and moved up we change the Image size values like this (just for a demonstration but no use at all)

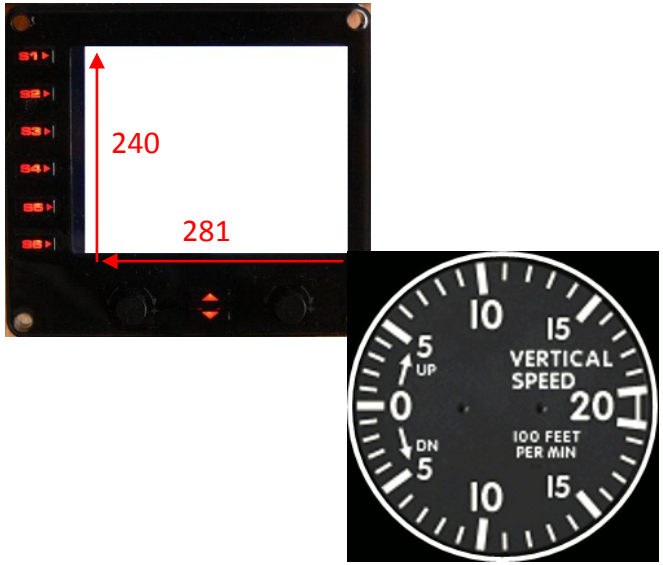
<Image Name="Baron_vertical_speed.bmp" ImageSizes="281,350,281,350"/>

The bitmap must be positioned here at 41 pixel and 110 pixel from the right lower corner of the FIP.



My conclusion is that, instead of giving the size of the image like in FSX/FS9, the ImageSize values for the first bitmap of the FIP means that the upper left side of the bitmap will align regarding the values AND from the lower right side of the FIP,

Actual size of the bitmap is 240 by 240



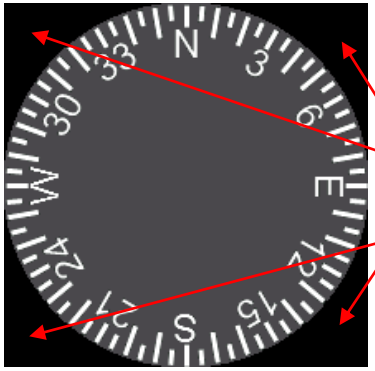
The ImageSize value of 281,240,281,240 will move the left upper corner of the bitmap 281 pixel to the left and 240 pixel up,

All combinations are valid for the background image. For the bitmaps that are positioned after the background bitmap , they will all align with the left up corner of the background bitmap.

You have to know:

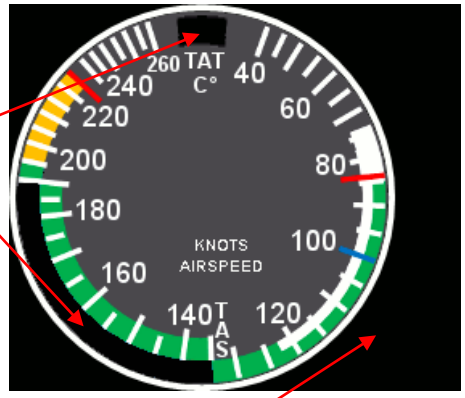
bitmaps are always a square. The bitmap is always positioned from its up left corner . The point from which it rotates could be anywhere.

The part of a mask image which let see the moving bitmap or stripe through it, is coloured with the code 010101.
The part of a bitmap that must become invisible is coloured with the code 000000.

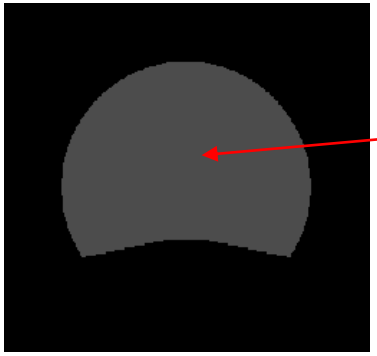


These part must be invisible and coloured 000000, this is not necessary when a Mask Image function is used.

These two parts of the AIR SPEED bitmap must allow to see the bitmaps that moves, underneath, a circle and a stripe, they are coloured 010101 because this bitmap is used as a Mask Image.



The background colour doesn't matter.



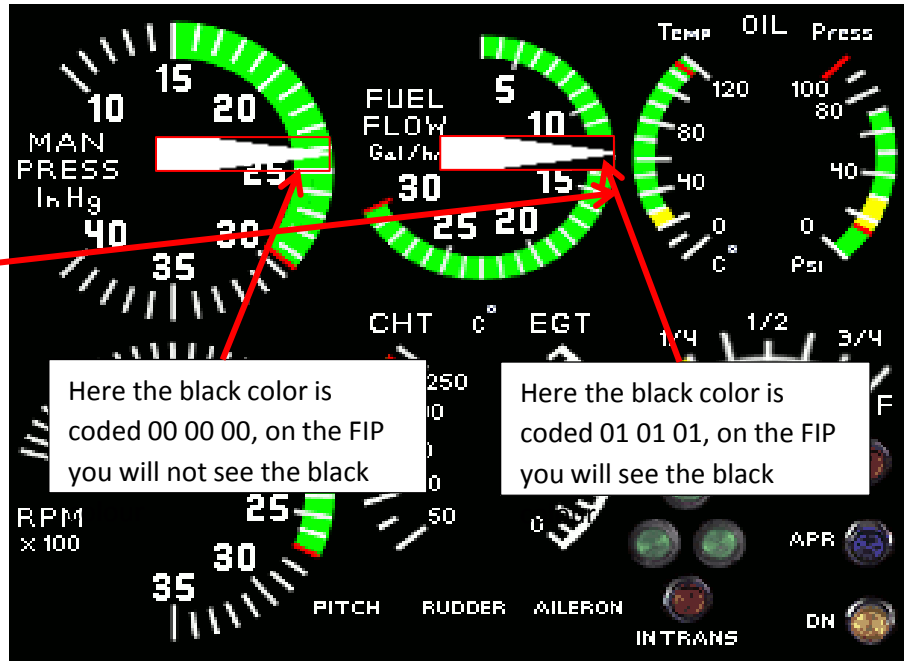
This is a Mask Image used in conjunction with another bitmap and it will let see only the part of the bitmap inside the grey portion, I coloured in grey for more visibility in this example but remember the colour code is 010101. The background could be invisible and coloured 000000, or any other near black colour like 000200, The human eye will not see any difference between all the black colours and you will see only a black bitmap.

In the following pages we will understand how to position a needle which rotates clockwise and counter clokwise, to rotate a bitmap, to use stripes instead of the FSX text function, to use masks when some part of the gauge has to be hidden, how to move a bitmap outside the FIP's available space when it must not be visible and more. To achieve this objective I use the Baron gauges I made for my own use and have the advantage to be very simple and for some more difficult.

This is an example of a needle.
This is a mask image, black colour is 010101. Associated to the mask is another bmp. On the FIP you will see only the associated bmp but inside the frame of the mask.

The goal is to have only the white part visible

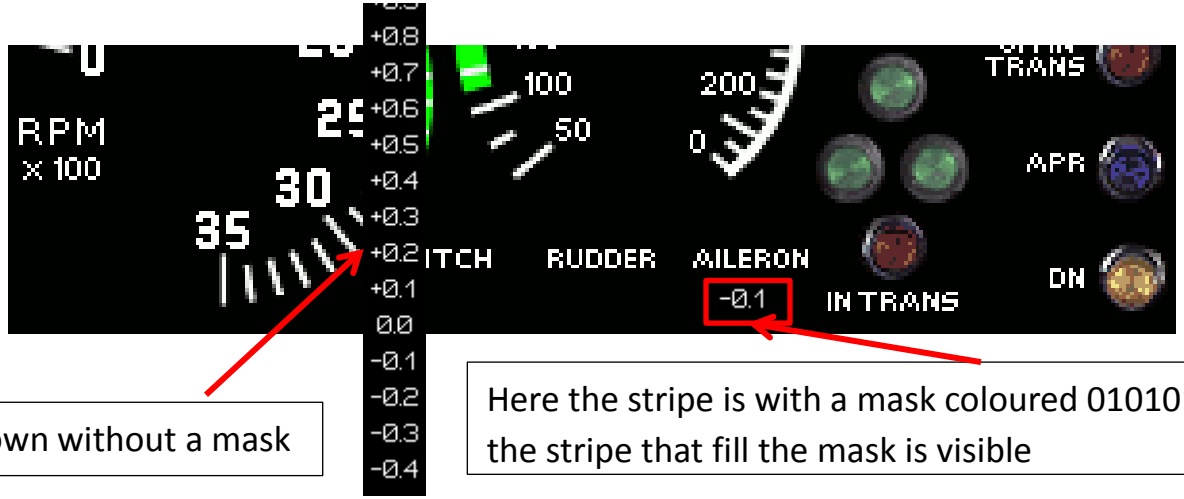
otherwise when the needle moves the black part will be seen over the gauge



Here the black color is coded 00 00 00, on the FIP you will not see the black

Here the black color is coded 01 01 01, on the FIP you will see the black

This is a mask image, black colour is 010101. Associated to the mask is another bmp. On the FIP you will see only the associated bmp but inside the frame of the mask.

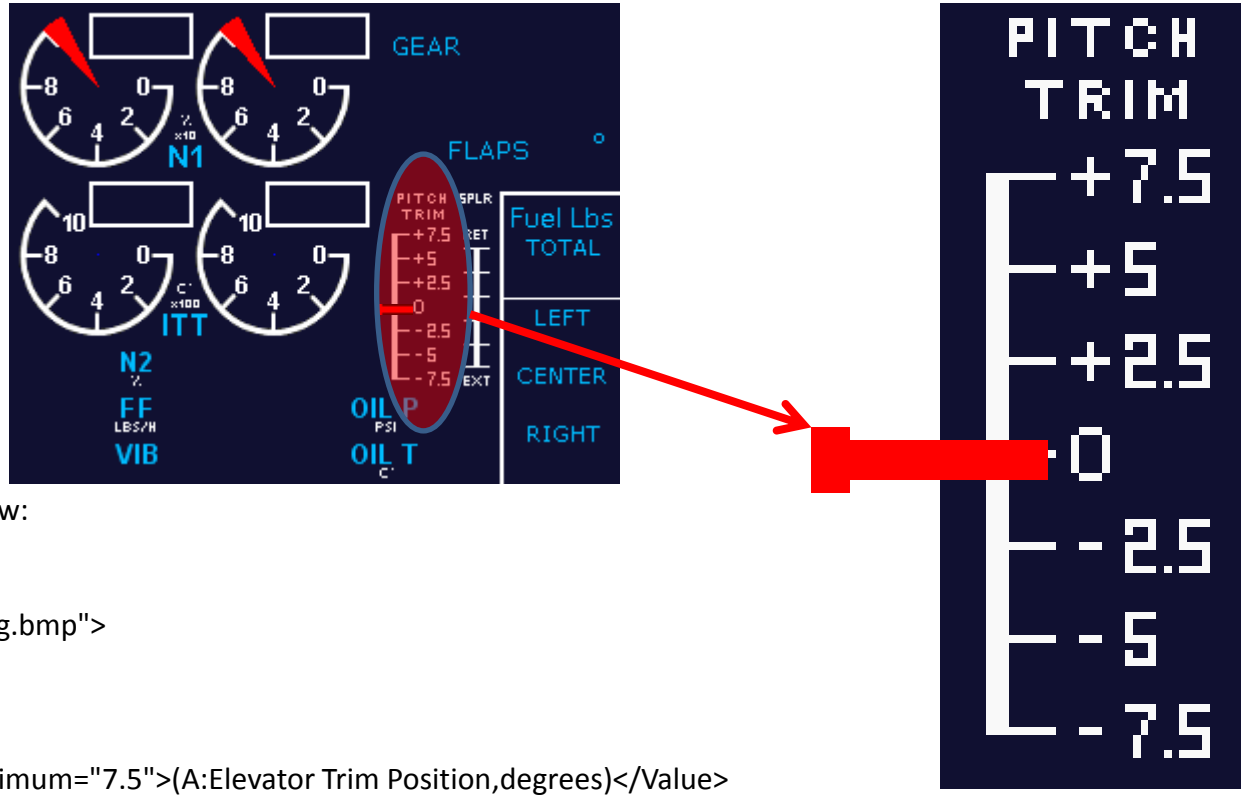


Here the stripe is shown without a mask

Here the stripe is with a mask coloured 010101, only the part of the stripe that fill the mask is visible

How to position a moving bar

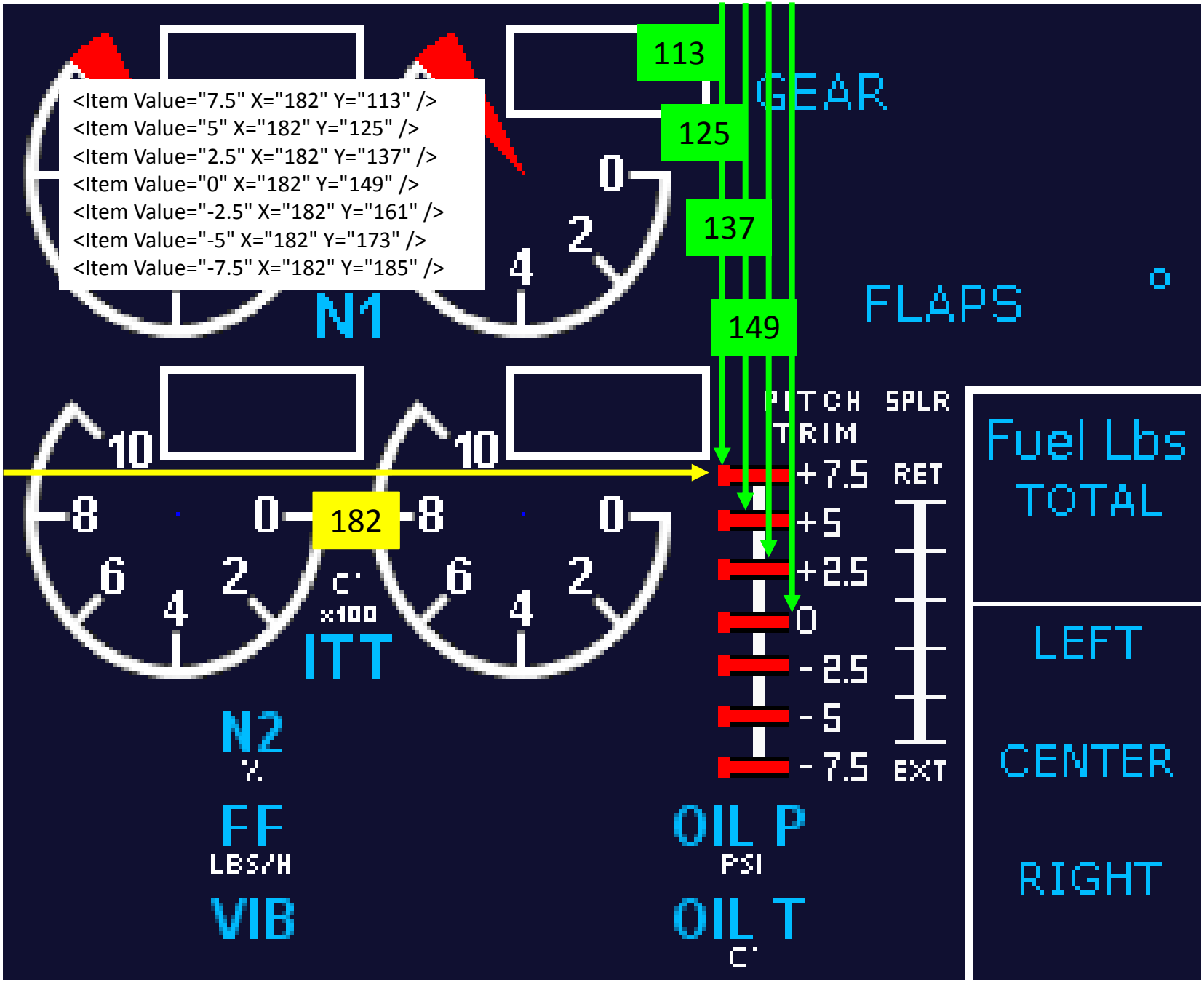
Moving bars are used to show values on a vertical or horizontal scale. For this study we will use the pitch trim bar that moves up and down to indicate the pitch trim value.



Element portion for the pitch trim is as follow:

```
<!-- PITCH TRIM DISPLAY -->
<Element>
        <Image Name="tje_Pitch_bug.bmp">
        <Axis X="0" Y="0" />
        </Image>
        <Shift>
        <Value Minimum="-7.5" Maximum="7.5">(A:Elevator Trim Position,degrees)</Value>
        <Nonlinearity>
                <Item Value="7.5" X="182" Y="113" />
                <Item Value="5" X="182" Y="125" />
                <Item Value="2.5" X="182" Y="137" />
                <Item Value="0" X="182" Y="149" />
                <Item Value="-2.5" X="182" Y="161" />
                <Item Value="-5" X="182" Y="173" />
                <Item Value="-7.5" X="182" Y="185" />
        </Nonlinearity>
        </Shift>
</Element>
```

For each value the bug will position itself at the indicated number of pixel from the left upper corner of the background bitmap. Understand X is counted from the left side and Y from the upper side.


<Item Value="7.5" X="182" Y="113" />
<Item Value="5" X="182" Y="125" />
<Item Value="2.5" X="182" Y="137" />
<Item Value="0" X="182" Y="149" />
<Item Value="-2.5" X="182" Y="161" />
<Item Value="-5" X="182" Y="173" />
<Item Value="-7.5" X="182" Y="185" />
113
125
137
149
182
GEAR
FLAPS
°
N1
ITT
x100
N2
%
FF
LBS/H
VIB
OIL P
PSI
OIL T
C°
PITCH TRIM
SPLR
+7.5
+5
+2.5
0
-2.5
-5
-7.5
RET
EXT
Fuel Lbs
TOTAL
LEFT
CENTER
RIGHT

Because the pitch scale is linear the Element portion could be as follow:

```
<!-- PITCH TRIM DISPLAY -->
<Element>
<Position X="182" Y="149" />
<Image Name="tje_Pitch_bug.bmp"/>
<Shift>
        <Value Minimum="-7.5" Maximum="7.5">(A:Elevator Trim Position,degrees)</Value>
        <Scale X="0" Y="-4.8"/>
</Shift>
</Element>
```

The left upper corner of the red bug will position at 182 pixel from the left side of the background bitmap and at 149 pixel from the upper side of the background bitmap when the value is 0.

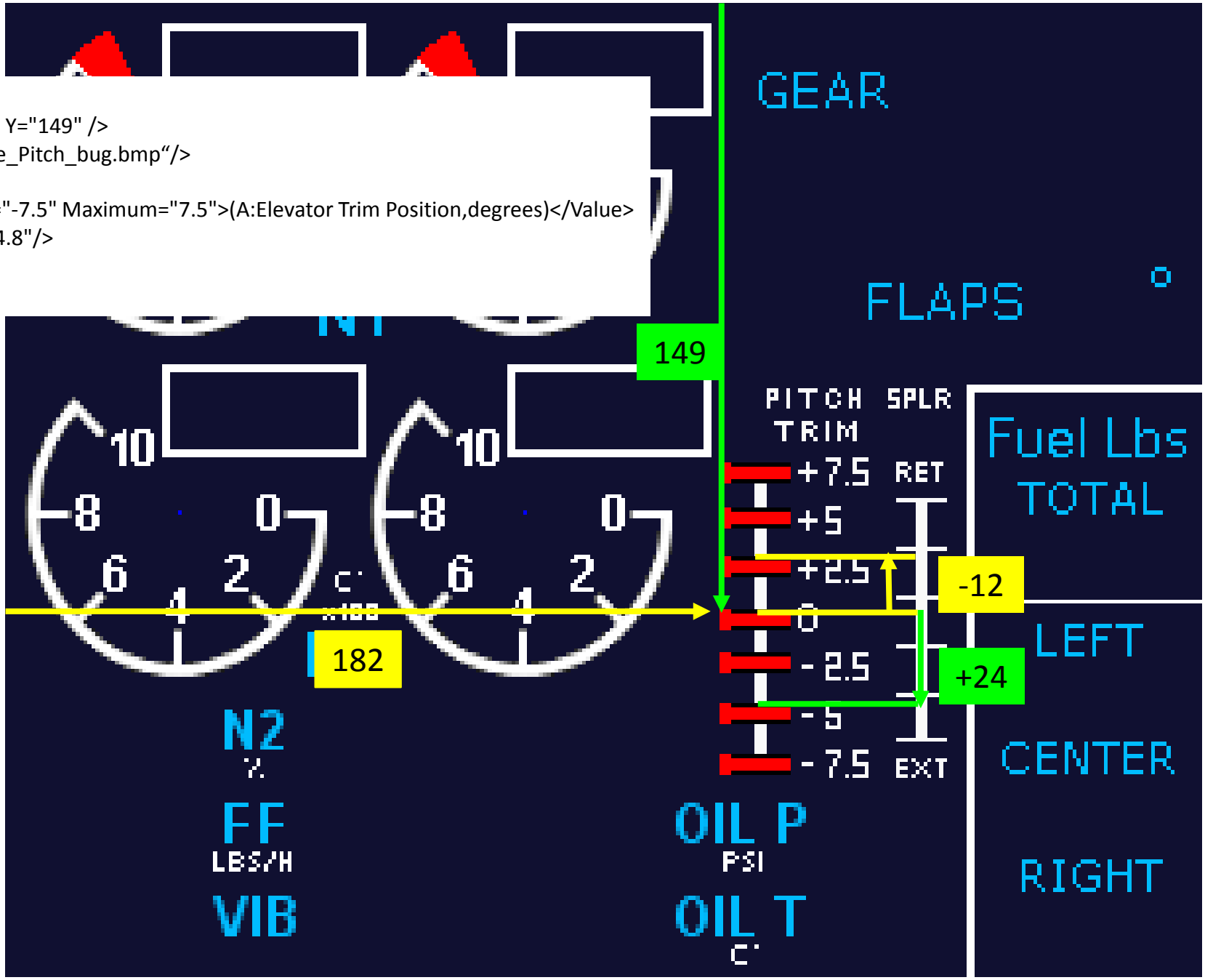
The shift function will move the bug up and down following the value of 4.8 pixel for each degree of pitch. In the scale value, the Y value is negative because when the pitch is positive the bug must move up and therefore the value of 4.8 must be subtract from 149 (Y position for the value 0). For example for a pitch value of + 5 you the bug should position at 125 pixel from the upper side of the background bitmap, If we calculate with the scale function you will have 5 x 4.8 = 24 and if you subtract 24 from 149 you will get 125.

The X value is 0 because the bug will move along the vertical line that is positioned at 182 pixels from the left side of the background bitmap.

If the bug was to move on a horizontal scale then the scale function should be like this:
<Scale X="-4.8" Y="0"/>

The scale function is sometimes useful. It calculate the values with decimals (like for the non linearity function), for example if the pitch moves of 0.1 degree then the bug will move for 0.48 pixel.

Until now I didn't see any impact on the FIP movement whatever is the number of lines in the xml file. Maybe it has for a slow computer but with really I didn't observed any delay even with my laptop.




```
<Element>
<Position X="182" Y="149" />
<Image Name="tje_Pitch_bug.bmp"/>
<Shift>
<Value Minimum="-7.5" Maximum="7.5">(A:Elevator Trim Position,degrees)</Value>
<Scale X="0" Y="-4.8"/>
</Shift>
</Element>
```

How to position a needle

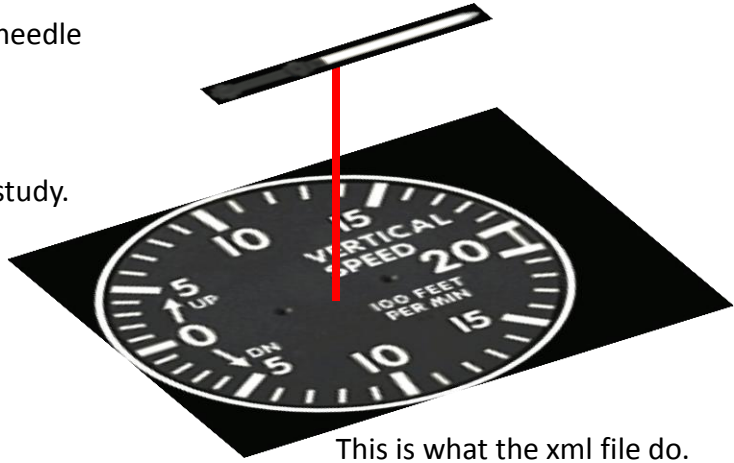
This is the final result

For this, I will use the Baron's vertical speed gauge and the Baron engine digital I made too.

1. For the Baron's vertical speed you need these two bitmaps:

Bimap 's name is: Baron_vertical_speed.bmp

Bimap 's name is: Baron_vertical_speed_needle



This is what the xml file do.

Now have a look at the xml file. Then we will go step by step for every line of this first study.

```
<?xml version="1.0" encoding="utf-8"?>
<Gauge Name=" Baron Vertical Speed Indicator" Version="1.0">
        <Image Name="Baron_vertical_speed.bmp" ImageSizes="281,240,281,240"/>
<Element>
        <Position X="120" Y="120"/>
        <Image Name="Baron_vertical_speed_needle.bmp" ImageSizes="157,19,157,19">
                <Axis X="51" Y="9.5"/>
        </Image>
        <Rotate>
                <Value Minimum="-20" Maximum="20">(A:Vertical speed,feet per minute) 100 /</Value>
        <Failures>
                <SYSTEM_PITOT_STATIC Action="Zero"/>
                <GAUGE_VERTICAL_SPEED Action="Freeze"/>
        </Failures>
        <Nonlinearity>
                <Item Value="-20" Degrees="97" />
                <Item Value="-5" Degrees="235" />
                <Item Value="0" Degrees="270" />
                <Item Value="5" Degrees="305" />
                <Item Value="20" Degrees="83" />
        </Nonlinearity>
        </Rotate>
</Element>
</Gauge>
```

```
<?xml version="1.0" encoding="utf-8"?>
<Gauge Name=" Baron Vertical Speed Indicator" Version="1.0">
        <Image Name="Baron_vertical_speed.bmp" ImageSizes="281,240,281,240"/>
<Element>
        <Position X="120" Y="120"/>
        <Image Name="Baron_vertical_speed_needle.bmp" ImageSizes="157,19,157,19">
                <Axis X="51" Y="9.5"/>
        </Image>
        <Rotate>
                <Value Minimum="-20" Maximum="20">(A:Vertical speed,feet per minute) 100 /</Value>
        <Failures>
                <SYSTEM_PITOT_STATIC Action="Zero"/>
                <GAUGE_VERTICAL_SPEED Action="Freeze"/>
        </Failures>
        <Nonlinearity>
                <Item Value="-20" Degrees="97" />
                <Item Value="-5" Degrees="235" />
                <Item Value="0" Degrees="270" />
                <Item Value="5" Degrees="305" />
                <Item Value="20" Degrees="83" />
        </Nonlinearity>
        </Rotate>
</Element>
</Gauge>
```

In red are the mandatory lines for a FIP xml file, the gauge name doesn't have any matter, it is just for your own usage and it is to identify the gauge easily.

In blue are the mandatory lines for an element (as it says) of your xml file, you can have how many elements you need for your gauge. An element can be linked with more than one function, for example to rotate and shift the bitmap in a single element. Each element and sub element start s with a < before the function and end with a /> like this : <Item Value="-20" Degrees="97" />. I f you have specific details for the sub element then the function starts with <whatever function> and end with </whatever function > like

```
        <Image Name="Baron_vertical_speed_needle.bmp" ImageSizes="157,19,157,19">
                <Axis X="51" Y="9.5"/>
        </Image>
```

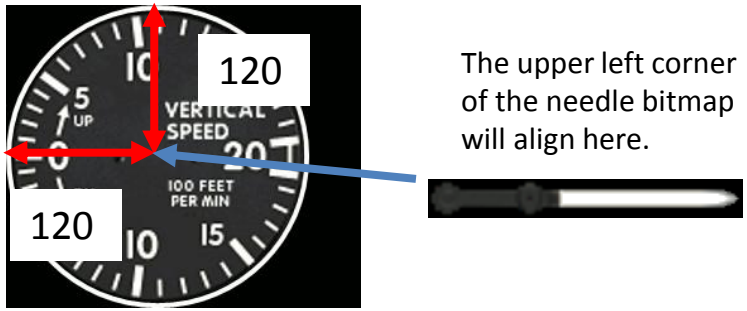
Image Name and size are self explanatory. Important for the FIP, without the size specified the bitmap will not show up.

Now look at the first part of the needle element.

```
<Element>
        <Position X="120" Y="120"/>
        <Image Name="Baron_vertical_speed_needle.bmp" ImageSizes="157,19,157,19">
                <Axis X="51" Y="9.5"/>
        </Image>
```

This where we want the needle to be positioned

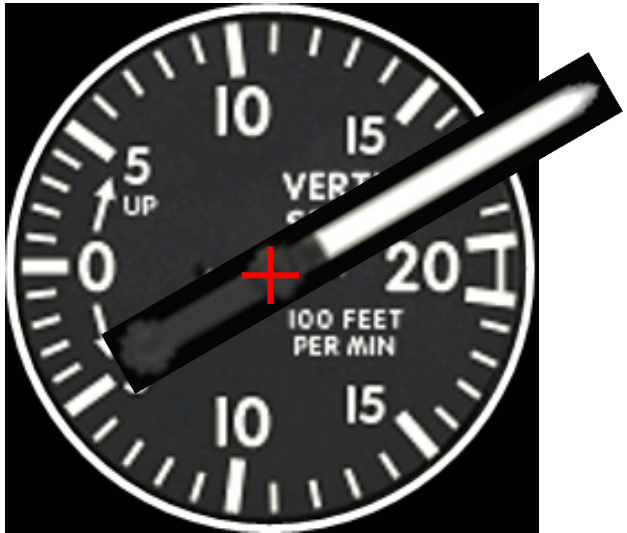
The rotation point of the needle is at 120 pixels from the left side of the background bitmap and at 120 pixels from the upper side of the background image, position will be <Position X="120" Y="120"/>
:



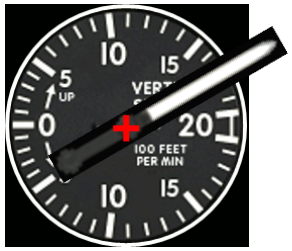
The upper left corner of the needle bitmap will align here.

I we let as it is the needle will rotate from its upper left side like this:

Instead of this:



To get the needle positioned at the right place like this:



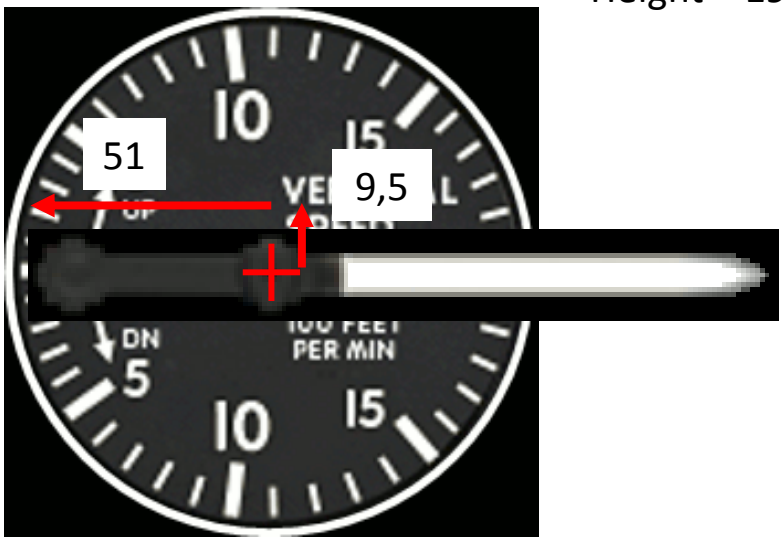
We must indicate that the needle has to be moved to the left and the middle should align with the rotation point. For this we use the line axis..

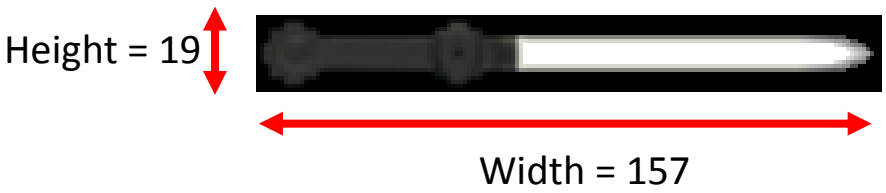
```
<Element>
        <Position X="120" Y="120"/>
        <Image Name="Baron_vertical_speed_needle.bmp" ImageSizes="157,19,157,19">
                <Axis X="51" Y="9.5"/>
        </Image>
```

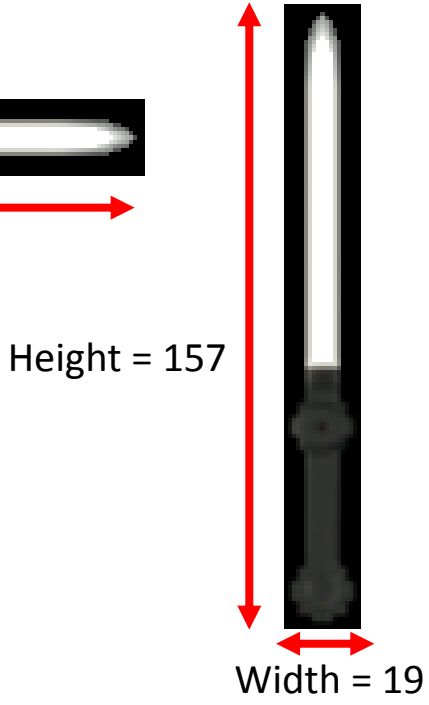
This line indicates that the needle upper left corner has to move 51 pixels to the left and 9,5 pixels up from the specified position (the center point of the rotation). Numbers are positive because it is moved up and left, it could be negatives when the bitmap must move to the right and below. 51 pixels is the range from the axis of rotation of the needle and 9,5 is half the width or height (it depends on the orientation of the needle, here the height is 19 because the needle points to the right. If the needle points to up then 19 is the width.

This is what the axis line will do:

:







No it is time to get the needle moving, for this we use the paragraph rotate that will make the needle rotate around its axis.

```
<Rotate>
        <Value Minimum="-20" Maximum="20">(A:Vertical speed,feet per minute) 100 /</Value>
<Failures>
        <SYSTEM_PITOT_STATIC Action="Zero"/>
        <GAUGE_VERTICAL_SPEED Action="Freeze"/>
</Failures>
<Nonlinearity>
        <Item Value="-20" Degrees="97" />
        <Item Value="-5" Degrees="235" />
        <Item Value="0" Degrees="270" />
        <Item Value="5" Degrees="305" />
                <Item Value="20" Degrees="83" />
</Nonlinearity>
</Rotate>
</Element>
</Gauge>
```

We can apply failures but I will not go in deep with the failure item because it is the same as for FS 9/2004 gauges, For FSX each failure depends on a specific sub element like this:

```
<FailureTable id="FailureTable">
<Failure id="Pitot Static">
<Fail_Key>SYSTEM_PITOT_STATIC</Fail_Key>
<Fail_Action>Zero</Fail_Action>
</Failure>
<Failure id="Gauge">
<Fail_Key>GAUGE_AIRSPEED</Fail_Key>
<Fail_Action>Freeze</Fail_Action>
</Failure>
</FailureTable>
```

For FIP or FS2004 xml all failures at grouped like this:

```
<Failures>
<SYSTEM_PITOT_STATIC Action="Zero"/>
<GAUGE_VERTICAL_SPEED Action="Freeze"/>
</Failures>
```

The rotate function is very similar to the shift function. Both need to know what are the limits in which the bitmap (needle) has to move. Both need values referred to what is usually called an interval variable and finally they need of how much the move is executed for each changing value.

```
<Rotate>
<Value Minimum="-20" Maximum="20">(A:Vertical speed,feet per minute)  100 /</Value>
```

In this line we indicate that the minimum and value for the gauge is 2000 feet per minute , here we write -20 because we want the interval variable to calculate for each hundred of feet level and the interval variable A:Vertical speed, feet per minutes provides values at feet level. It is why we divide the interval variable value by 100 : 100 /

Then we have to indicate of how much the needle will rotate for each value. We have two possibilities, the indexes of the circular scale have the same range between them or the scale is not linear.
The scale of this vertical speed gauge is not linear therefore we must specify the values for the specific ranges. From 0 to 5 the scale is of 7 degrees (35 degrees of range), for the range form 5 to 20 the scale is 9,2 (138 degrees of range)
For this we have the Nonlinearity paragraph where specific values are written.

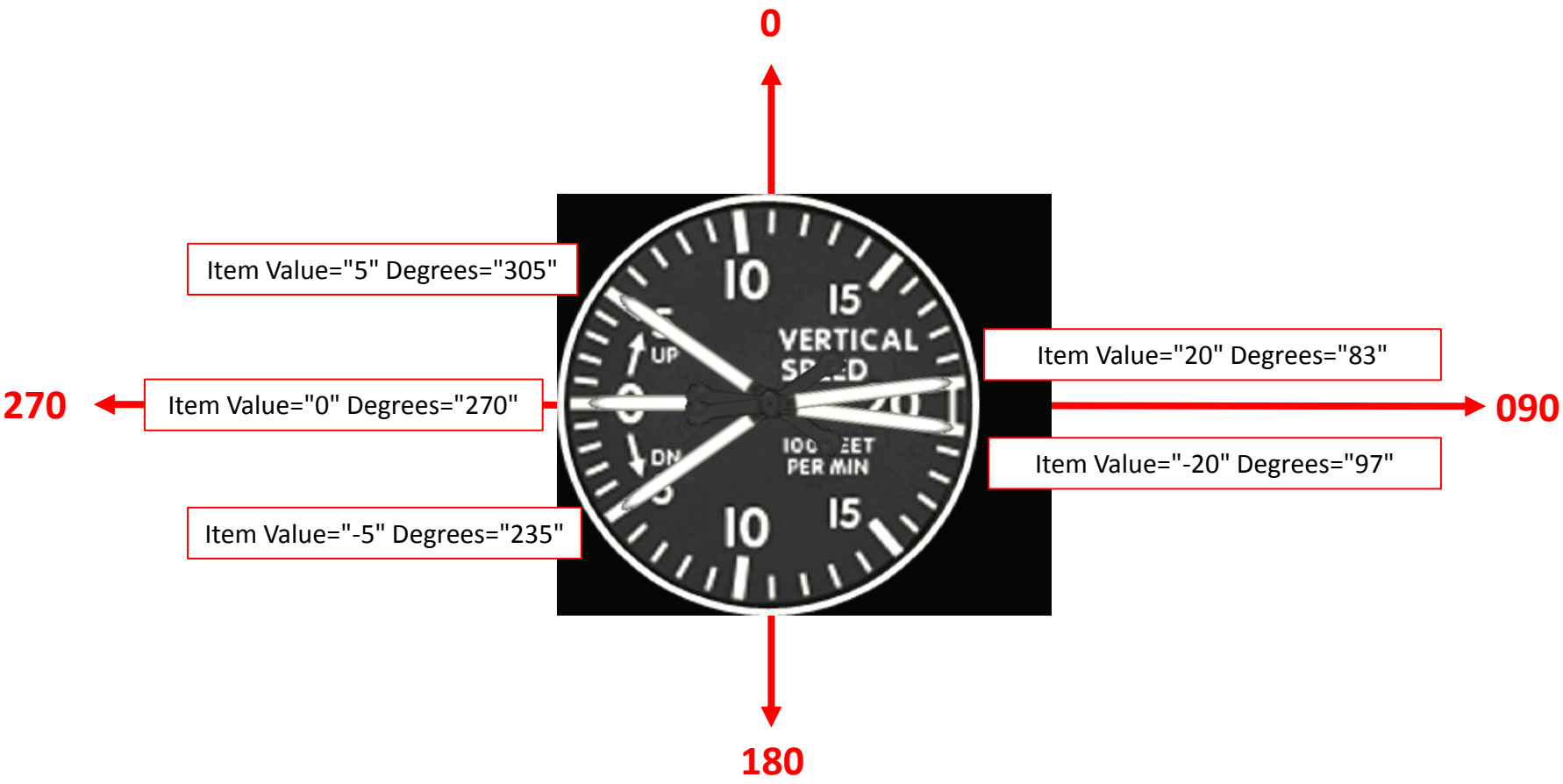
```
<Nonlinearity>
        <Item Value="-20" Degrees="97" />
        <Item Value="-5" Degrees="235" />
        <Item Value="0" Degrees="270" />
        <Item Value="5" Degrees="305" />
        <Item Value="20" Degrees="83" />
</Nonlinearity>
```

This expression is also valid:

```
<Rotate>
        <Value Minimum="-2000" Maximum="2000">(A:Vertical speed,feet per minute) </Value>
<Nonlinearity>
        <Item Value="-2000" Degrees="97" />
        <Item Value="-500" Degrees="235" />
        <Item Value="0" Degrees="270" />
        <Item Value="500" Degrees="305" />
        <Item Value="2000" Degrees="83" />
</Nonlinearity>
</Rotate>
```

Now look how angles work:

```
<Nonlinearity>
        <Item Value="-20" Degrees="97" />
        <Item Value="-5" Degrees="235" />
        <Item Value="0" Degrees="270" />
        <Item Value="5" Degrees="305" />
        <Item Value="20" Degrees="83" />
</Nonlinearity>
```



If the scale is linear then we have two possibilities, first to express the minimum and maximum values only:

```
<Nonlinearity>
        <Item Value="-20" Degrees="97" />
        <Item Value="0" Degrees="270" />
        <Item Value="20" Degrees="83" />
</Nonlinearity>
```

We must specify the 0 value otherwise the needle will move between the value from 83 to 97 degrees with the 0 value set to 90.

Second possibility is to use the scale function which I do not recommend because very difficult to settle for the rotation.

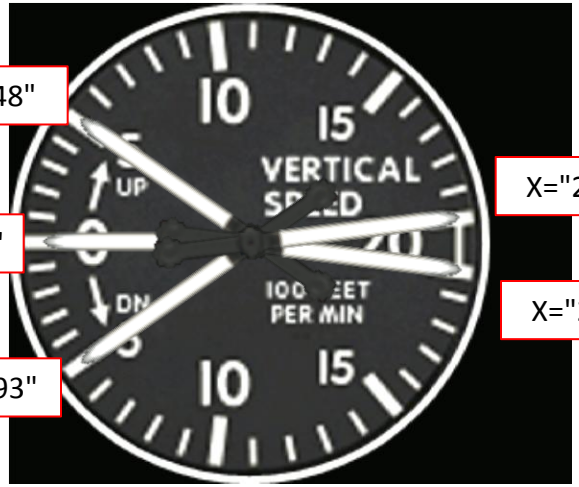
The expression could be like this:

```
<Rotate>
        <Value Minimum="-2000" Maximum="2000">(A:Vertical speed,feet per minute) </Value>
        <Scale Degrees = "0.0002 />
</Rotate>
```

But in that case the needle must point to the left like this:

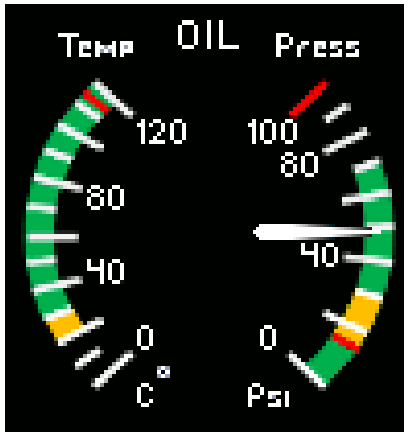
Last we can use both Degrees or X=" " and Y=" " like this:

```
<Nonlinearity>
<Item Value="-20" X="249" Y="135" />
<Item Value="-5" X="12" Y="193" />
<Item Value="0" X="0" Y="118" />
<Item Value="5" X="16" Y="48" />
<Item Value="20" X="248" Y="104" />
</Nonlinearity>
```



In this case X and Y are the positions to where points the needle.

The rotation is normally clockwise but for some gauges the needle must rotate anti clockwise like this:



Element is this:

```
<Rotate>
<Value Minimum="0" Maximum="100">100 (A:GENERAL ENG2 OIL PRESSURE, PSI) -</Value>
<Nonlinearity>
<Item Value="0" Degrees="48" />
<Item Value="20" Degrees="64" />
<Item Value="40" Degrees="80" />
<Item Value="50" Degrees="88" />
<Item Value="60" Degrees="97" />
<Item Value="80" Degrees="115" />
<Item Value="100" Degrees="133" />
</Nonlinearity>
</Rotate>
```

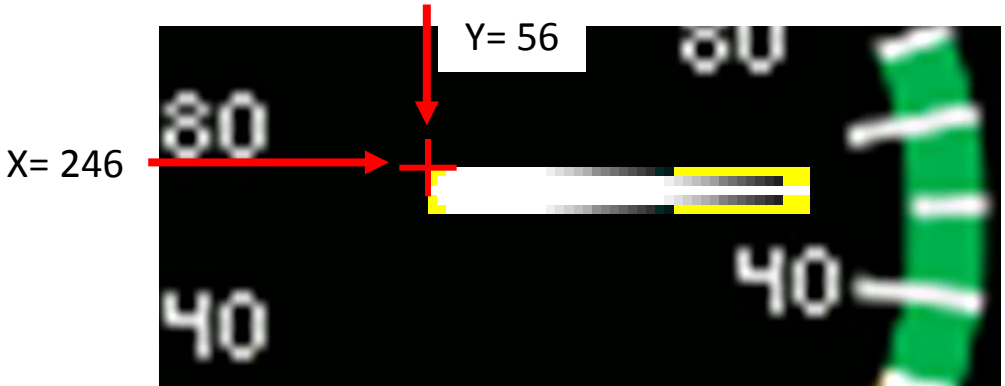
To get the needle rotating anti clockwise you have to reverse the interval variable: 100 (A:GENERAL ENG2 OIL PRESSURE, PSI) –
In fact we want the variable to produce a value that is subtracted (-) from the maximum value (100)

Then, for the nonlinearity portion you have to reverse also the value for example <Item Value="0" Degrees="48" /> when the actual value should be 133 degrees, for <Item Value="100" Degrees="133" /> when the value should be 48 degrees and so on.

I take the opportunity of this study to look at the needle parameters when the rotation point is out of the needle.
For this Oil pressure gauge the needle parameter is:

```
<Position X="246" Y="56" />
<Image Name="Baron_digit_needle_right.bmp" ImageSizes="37,5,37,5" PointsTo="East">
<Axis X="-17" Y="2.5" />
</Image>
```

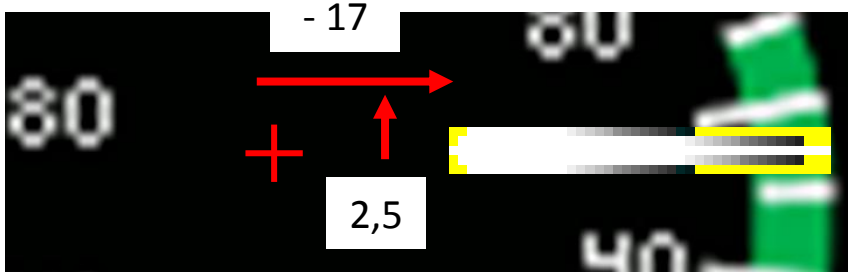
Remember the upper left corner of the bitmap should align with the rotation point:

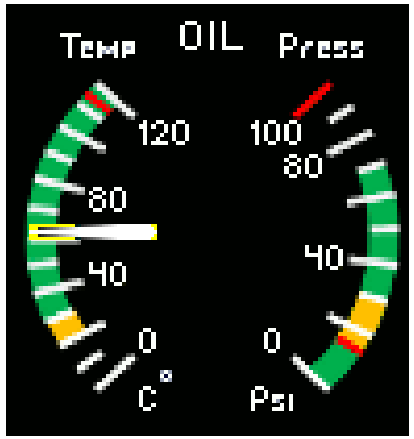


To have the needle shifted from its rotation point just write the axis portion like this:

```
<Position X="246" Y="56" />
<Image Name="Baron_digit_needle_right.bmp" ImageSizes="37,5,37,5" PointsTo="East">
<Axis X="-17" Y="2.5" />
</Image>
```

This shifts the left upper corner of the needle bitmap 17 pixels to the right
And 2,5 pixels up.





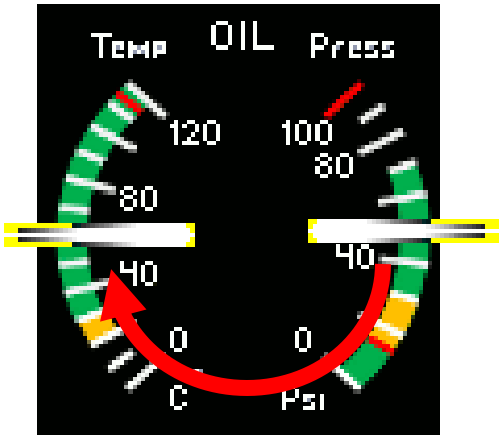
For this oil temperature needle that rotates clockwise the axis portion is also <Axis X="-17" Y="2.5" />

```
<Position X="269" Y="56" />
<Image Name="Baron_digit_needle_right.bmp" ImageSizes="37,5,37,5" PointsTo="East">
<Axis X="-17" Y="2.5" />
</Image>
```

This seems strange but it is because the original needle bitmap points to the right :

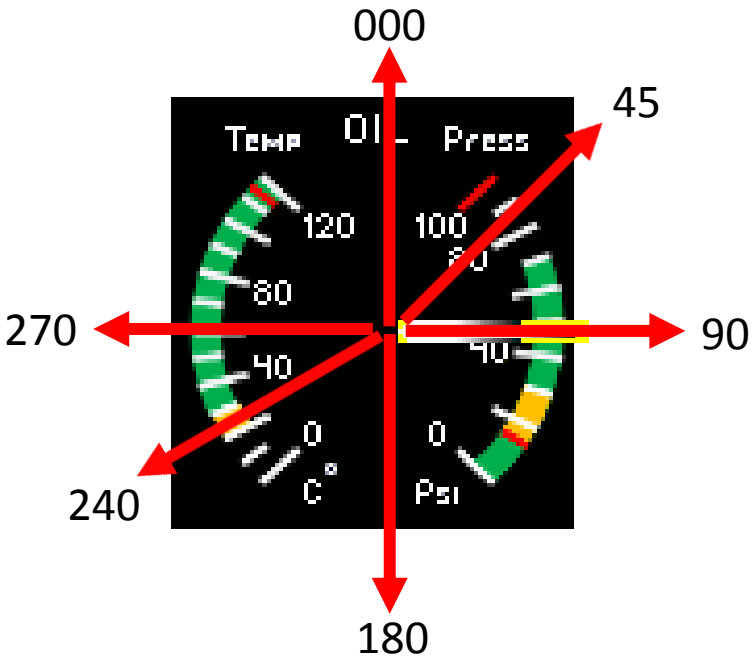
When the engine is started the needle will follow the variable values like this being in the right position.

This method is usually used to reduce the number of bitmap needle by using only one type of needle, in this case a needle bitmap that points to the right (East).



The other advantage is that when the needle points to the East (right) the degrees axis points to the north and the angles work like for a compass like this:

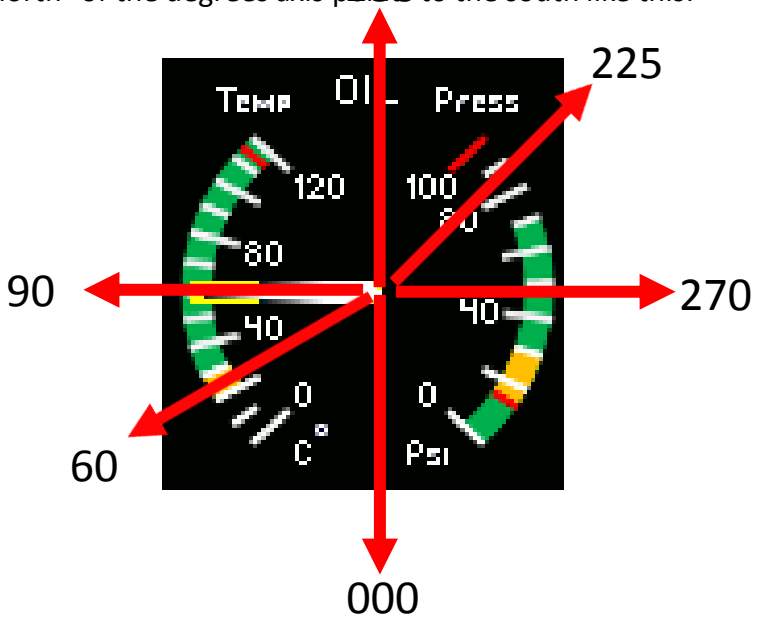
```
<Nonlinearity>
<Item Value="0" Degrees="228" />
<Item Value="40" Degrees="256" />
<Item Value="80" Degrees="284" />
<Item Value="120" Degrees="312" />
</Nonlinearity>
```



Be aware that this is only for the FIP and it doesn't work like this for FSX gauge.

If the needle points to the west , the "north" of the degrees axis points to the south like this:

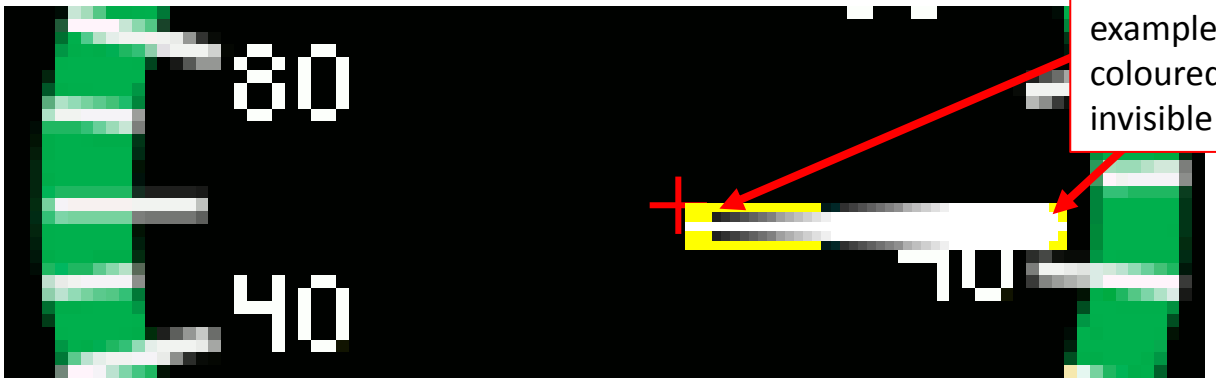
```
<Nonlinearity>
<Item Value="0" Degrees="48" />
<Item Value="40" Degrees="76" />
<Item Value="80" Degrees="104" />
<Item Value="120" Degrees="132" />
</Nonlinearity>
```



If we use a needle that points to the left (West) 
the axis portion like this:

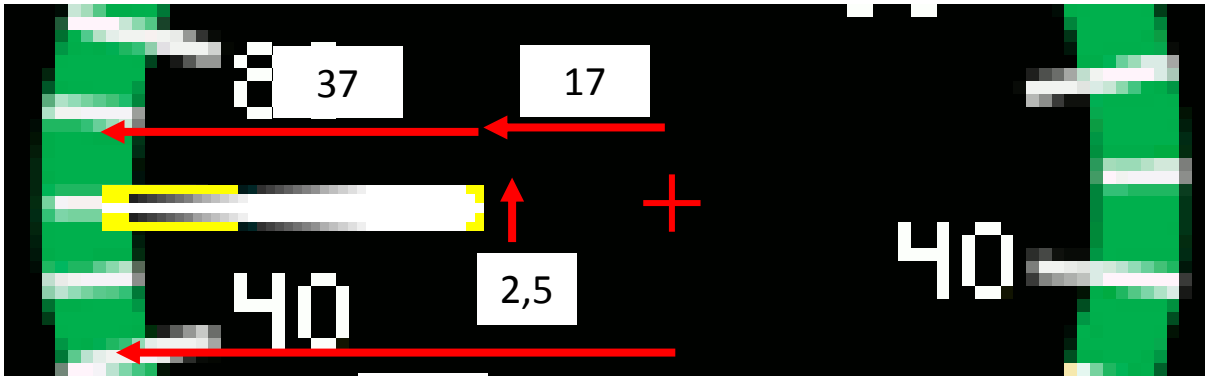
```
<Position X="246" Y="56" />
<Image Name="Baron_digit_needle_left.bmp" ImageSizes="37,5,37,5" PointsTo="West">
<Axis X="54" Y="2.5" />
</Image>
```

Remember the upper left corner of the needle bitmap will align with the rotation points.



This portion of the needle is in yellow for more visibility for this example but should be coloured 000000 to be invisible

The axis portion ; <Axis X="54" Y="2.5" /> shift the left upper corner of the needle bitmap 54 pixel to the left. In fact the 54 value includes the shift of 17 pixels and the width of the needle, 37 pixels (ImageSizes="37,5,37,5").



54

Baron Altimeter

This gauge contains these 6 bitmaps

This is the background image.

The hundred feet needle.

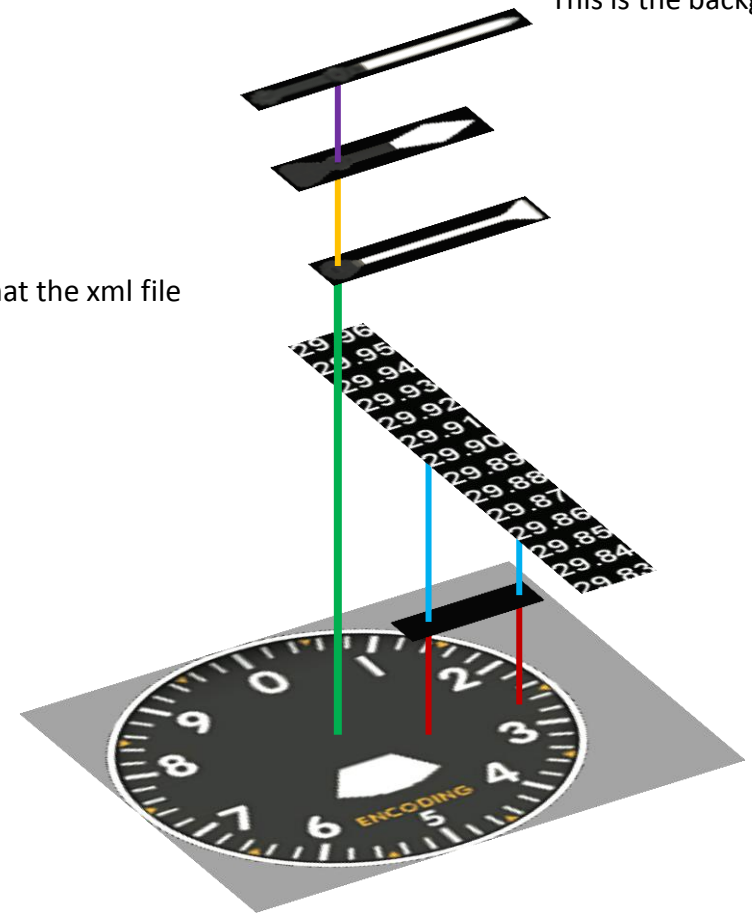
The thousand feet needle.

The ten thousand feet needle.

The mask image for the QNH setting.

This stripe contains the QNH values.

This is what the xml file provides.



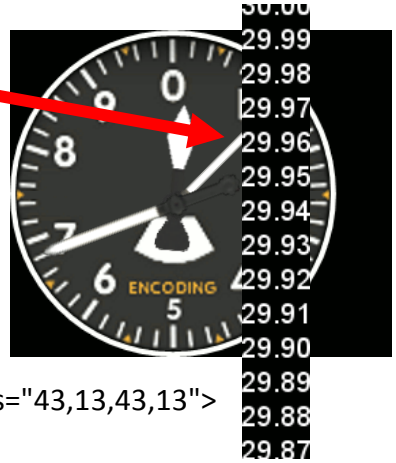
This is the final product.

I will use this gauge to show you how to shift a bitmap and how to use a mask image to get only the current value in a small window.

I will not come back on the how to position a needle, for this look at the previous chapter.

If we do not use a Mask Image the stripe will show up like this:

Now look at the xml portion for the shift and the Mask Image .

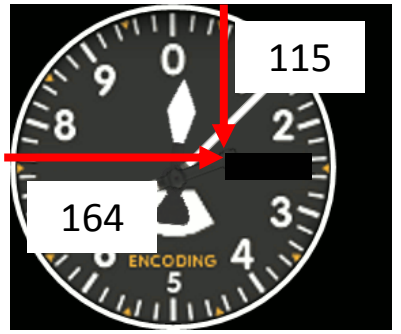



```
<!-- QNH setting -->
<Element>
		<Position X="164" Y="115"/>
		<MaskImage Name="Baron_mask_altimeter.bmp" ImageSizes="43,13,43,13">
			<Axis X="0" Y="0"/>
		</MaskImage>
		<Image Name="Baron_altimer_stripe.bmp" ImageSizes="43,14331,43,14331"/>
		<Shift>
			<Value Minimum="2560" Maximum="3210">(A:Kohlsman setting hg,inHg) 100 * %</Value>
		<Failures>
			<SYSTEM_PITOT_STATIC Action="Freeze"/>
		</Failures>
			<Nonlinearity>
			<Item Value="2560" X="0" Y="14309"/>
			<Item Value="3210" X="0" Y="9"/>
			</Nonlinearity>
		</Shift>
</Element>
```

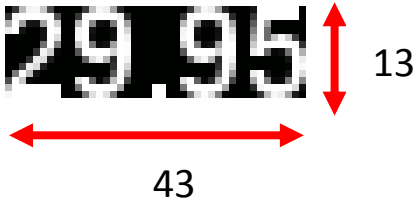
```
<!-- QNH setting -->
<Element>
        <Position X="164" Y="115"/>
```

The Mask Image will be positioned at 164 pixels from the left side of the backgroung bitmap and At 115 pixel from the upper side.

```
        <MaskImage Name="Baron_mask_altimeter.bmp" ImageSizes="43,13,43,13">
                <Axis X="0" Y="0"/>
        </MaskImage>
```

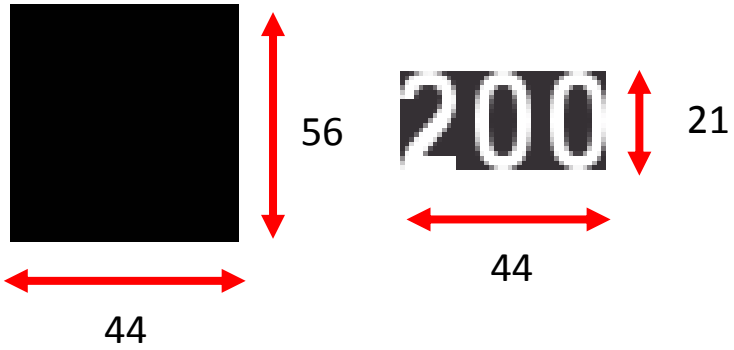


The axis value tells where the moving bitmap should align on the Mask Image.
I recommend to have the size of the Mask Image to fit with the final window to be shown.



Sometimes it is necessary to have the Mask Image bigger than the value to be shown like for this King Air altimeter.

The Mask Image size for the hundred of feet is 44 by 56 pixels when the size of the number is 44 by 21 pixels. Therefore the Mask Image let see the current value and the previous value (200) And the next value (000) pending the descent or climb of the aircraft.



The stripe has to move up and down with the setting by a knob or a key.

The portion of the element for the shift is:

```
<Image Name="Baron_altimer_stripe.bmp" ImageSizes="43,14331,43,14331"/>
<Shift>
<Value Minimum="2560" Maximum="3210">(A:Kohlsman setting hg,inHg) 100 * %</Value>
<Nonlinearity>
<Item Value="2560" X="0" Y="14309"/>
<Item Value="3210" X="0" Y="9"/>
</Nonlinearity>
</Shift>
</Element>
```

As for the rotate function we must determine the minimum and the maximum values.
<Value Minimum="2560" Maximum="3210">(A:Kohlsman setting hg,inHg) 100 * %</Value>

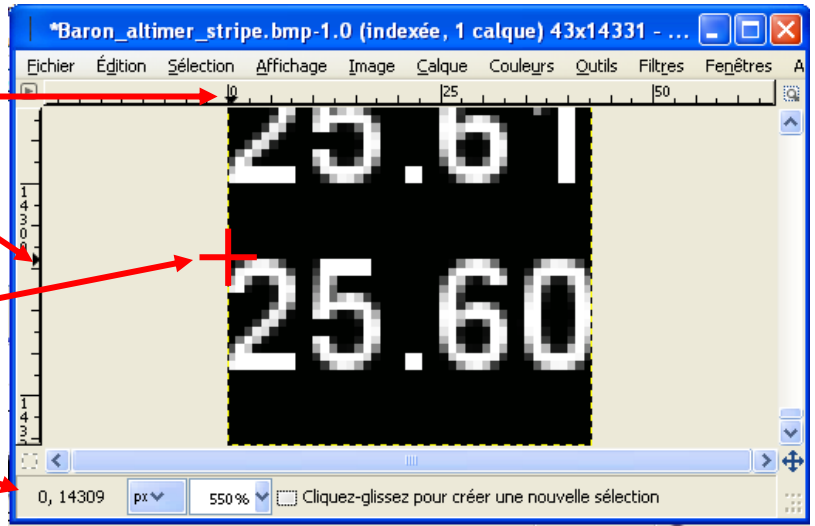
To avoid the use of decimal number just modify the value to get an integer for each value, this is why the minimum and maximum values are multiplied by 100. But due to the accuracy of the value given by FSX you must again delete the decimal after having multiplied the number and for this we use the character %.

Here I use The Gimp, move the pointer to the left upper pixel of the minimum value (2560).

The indexes show X=0 Y=14309

Pointer

This window gives instantly the X=0 and Y=14309

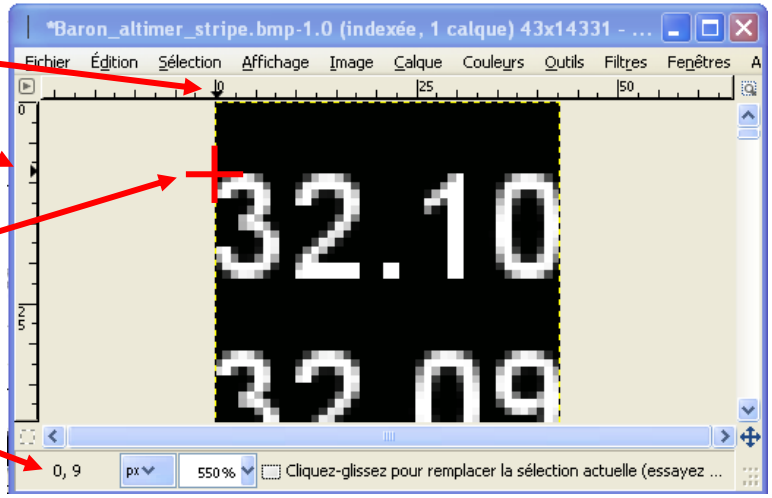


Then move the pointer to the left upper pixel of the maximum value (3210).

The indexes show X=0 Y=14309

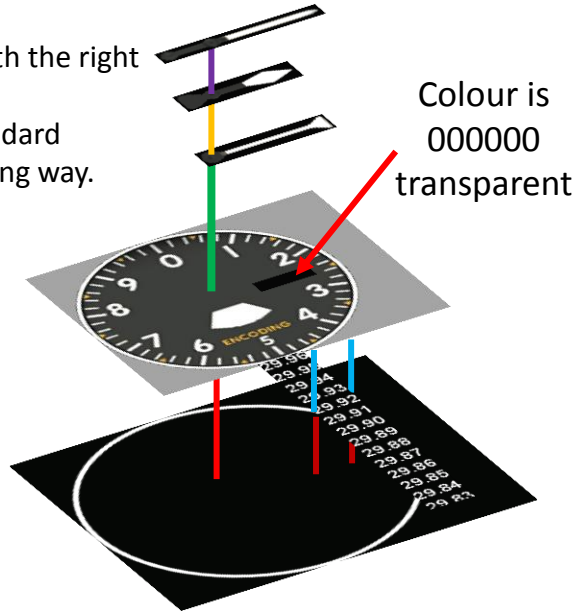
Pointer

This window gives instantly the X=0 and Y=9



There is another method but just for your information because I do not recommend to do it. The method is to make the portion of the background bitmap that will show the value transparent (coulour 000000) I call this the cover bitmap. The goal is to have a generic background image, black usually, you position the stripe, apply the function and finally recover the stripe with the cover bitmap. This an example of what the xml file must do.

The main difficulty is to find the position value for the stripe and to have the stripe moving with the right Movement. The X position doesn't create trouble but the Y position is a nightmare and as you see – 9526 doesn't follow an accurate rule to get centered the 29.91 value as for a standard when FSX starts, More when you move the altimeter setting knob the stripe moves in the wrong way. I tried to have the maximum at the end of the stripe and the minimum at the upper side but No way to get it moving correctly. Using a mask is far easier.




```
<Element>
<Position X="164" Y="-9526"/>
<<Image Name="Baron_altimer_stripe.bmp" ImageSizes="43,14331,43,14331"/>
<Shift>
<Value Minimum="2560" Maximum="3210">(A:Kohlsman setting hg,inHg) 100 * %</Value>
                <Nonlinearity>
                <Item Value="2560" X="0" Y="14309"/>
                <Item Value="3210" X="0" Y="9"/>
                </Nonlinearity>
                </Shift>
</Element>
```

```
<!-- QNH setting -->
<Element>
		<Position X="0" Y="0"/>
		<MaskImage Name="Baron_altimeter.bmp" ImageSizes="281,240,281,240">
			<Axis X="0" Y="0"/>
		</MaskImage>
		<Image Name="Baron_altimer_stripe.bmp" ImageSizes="43,14331,43,14331">
				<Axis X="-164" Y="-115"/>
		</Image>
		<Shift>
			<Value Minimum="2560" Maximum="3210">(A:Kohlsman
setting hg,inHg) 100 * %</Value>
			<Nonlinearity>
			<Item Value="2560" X="0" Y="14309" />
			<Item Value="3210" X="0" Y="9" />
		</Nonlinearity>
		</Shift>
</Element>
```

You have two possibilities for the value's stripe, either you have only one stripe that contains the full range of the values like this (this is only an extract of the bitmap which length is 14331 pixels)

either you can use several stripes with value from 0 to 9.
For this case you will need one Element paragraph for each stripe as follow.

This is the xml portion for the Engine N1 that shows 3 integers and 1 decimal.
This is the part for the hundred value:


```
<!-- Engine 1 N1 display x100 -->
<Element>
        <Position X="43" Y="8" />
        <MaskImage Name="tje_engine_N1_mask.bmp" ImageSizes="11,17,11,17">
        <Axis X="0" Y="0" />
        </MaskImage>
        <Image Name="tje_N1_ITT_hund.bmp" ImageSizes="11,229,11,229">
                <Nonlinearity>
                        <Item Value="0" X="0" Y="205" />
                        <Item Value="9" X="0" Y="7" />
                </Nonlinearity>
        </Image>
                <Shift>
                <Value Minimum="0" Maximum="9">(A:TURB ENG1 N1, percent) 100 div</Value>
                <Scale X="0" Y="21"/>
                <Delay PixelsPerSecond="42"/>
                </Shift>
</Element>
```

The stripe start with 1 in red to have the value in red when the N1 value is equal or above 100 %.

This line tells that you want the first digit only, then you divide the value by 100. (A:TURB ENG1 N1, percent) 100 div

This is the part for the ten value:

<!-- Engine 1 N1 display white x10 -->
<Value Minimum="0" Maximum="9">(A:TURB ENG1 N1, percent) 10 div (A:TURB ENG1 N1, percent) 100 div 10 * -</Value>

This line tells that you want the second digit only, then you first divide the value by 10: (A:TURB ENG1 N1, percent) 10 div
For example the N1 % is 120 that means you will have 120 / 10 = 12 (div means you keep the integer only, with the sign / you will get the decimal)

Afterward you divide again the value by 100: (A:TURB ENG1 N1, percent) 100 div that means you will have 120 / 100 =1.2 but using the div criteria you will have 1 only.

Next you multiply the previous result by 10: 10 * you will have 1 * 10 = 10

Finally you subtract the last result from the first: - you will have 12 – 10 = 2, you get with this final result the second digit.

This is the part for the unit value:

<!-- Engine 1 N1 display white x10 -->
<Value Minimum="0" Maximum="9">(A:TURB ENG1 N1, percent) (A:TURB ENG1 N1, percent) 10 div 10 * - %</Value>

This line tells that you want the third digit only, then you first take the full value: (A:TURB ENG1 N1, percent)
For example the N1 % is 85. 7 that means you will have 85.7

Afterward you divide again the value by 10: (A:TURB ENG1 N1, percent) 10 div that means you will have 85.7 / 10 =8.57 but using the div criteria you will have 8 only.

Next you multiply the previous result by 10: 10 * you will have 8 * 10 = 80

Finally you subtract the last result from the first: - you will have 85.7 – 80 = 5.7
To delete the decimal just use this criteria at the end %, you get with this final result the third digit.

This is the part for the decimal value:

<!-- Engine 1 N1 display white x10 -->
<Value Minimum="0" Maximum="9">(A:TURB ENG1 N1, percent) (A:TURB ENG1 N1, percent) % - 10 * % </Value>

This line tells that you want the first decimal digit only (remember FSX calculates accurately with several decimals after the point, then you first take the full value: (A:TURB ENG1 N1, percent)

For example the N1 % is 120.4 that means you will have 120.4546 by FSX.

Afterward you take again the full value and you put the % criteria to delete the decimal that means you will have 120 only.

Next you subtract the last result from the first, you will have 120.4546 – 120 = 0.4546

Next you multiply the previous result by 10: 10 * you will have 0,4546 * 10 = 4.546

Finally you put the % criteria to delete the decimal, you will have 4 only, you get with this final result the first decimal digit only.

For the N1 value you will have to create 5 stripes, 3 in white color and 2 in red colour when the value is above 100. The same stripe is used for tens and units. This is quite short to create the 5 stripes but you will need seven Elements, one for each white and red value.

To get the stripe with the red value when it is above 100 just add this line at the beginning of the Element for the red value. This is an example for the decimal:

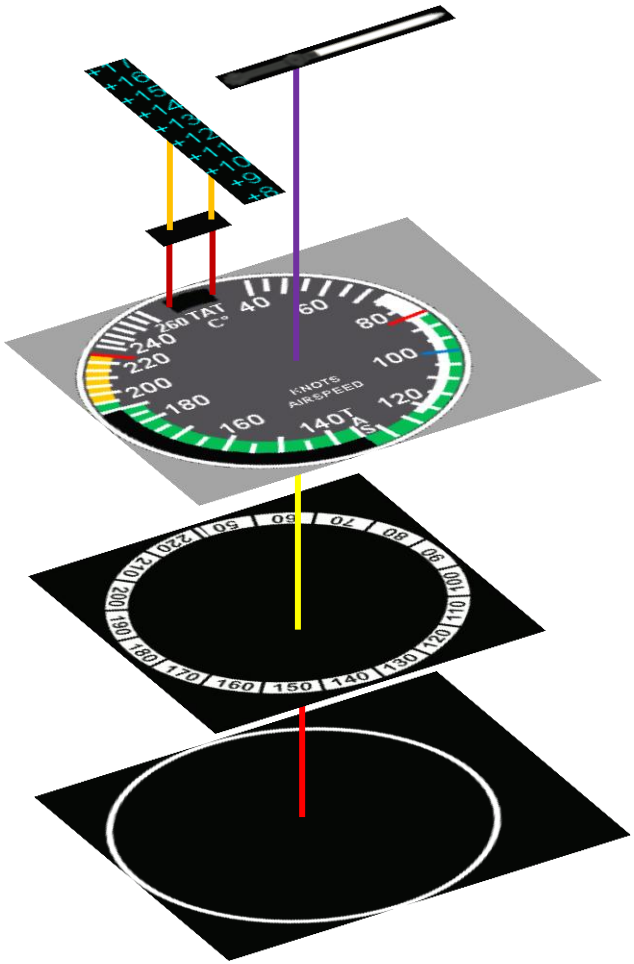
<Element>
<Visible>(A:TURB ENG1 N1, percent) 100 div 1 ==</Visible>

This means that the red stripe will appear for the decimal digit when the N1 value is divided by 100 and give 1.

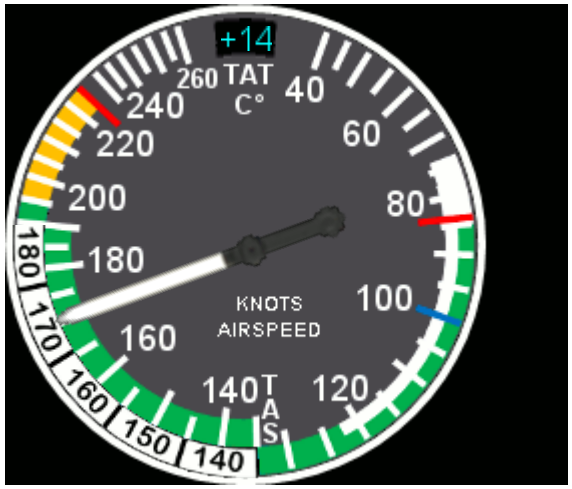
Using multiple single values permits to have smaller stripes and you can use the same stripe for several digits. For example the fuel quantity for a jet in pounds can be up to 100 000, for this I use the same stripe for the 6 digits. In this case this is the best method otherwise you have to create a stripe that runs from 0 to 100 000. I let you imagine the time that takes and the huge probability of having mistakes in the stripe.

When I started to create gauges for the FIP I used the method described above for the stripes and if it does work correctly for some stripes, as for the speed, and the compass, it create more difficulties than the use of a Mask Image. However the method with a cover does work correctly if the cover is used as a mask Image with the rotate function, To demonstrate this I will use the Baron air speed gauge.

This is what the xml file do



This is the final result

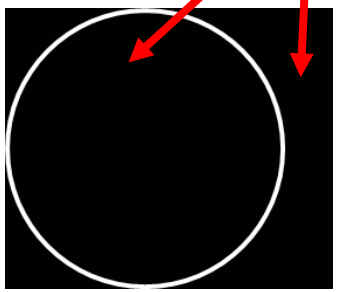


TAT stripe

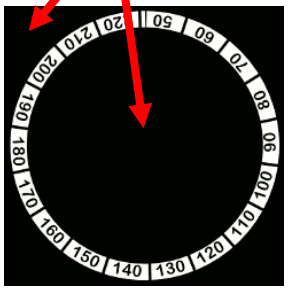
TAT Mask Image

Needle

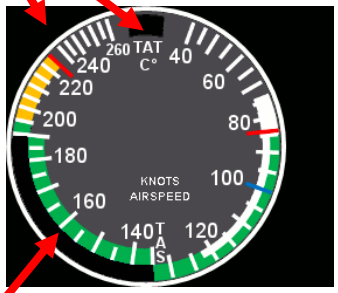
Colour is any black colour



Background bitmap



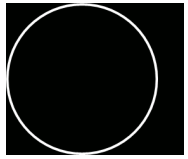
True speed bitmap

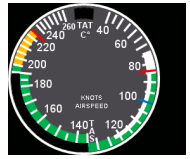


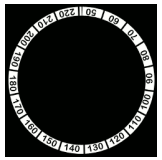
Cover bitmap

Colour is Mask colour 010101

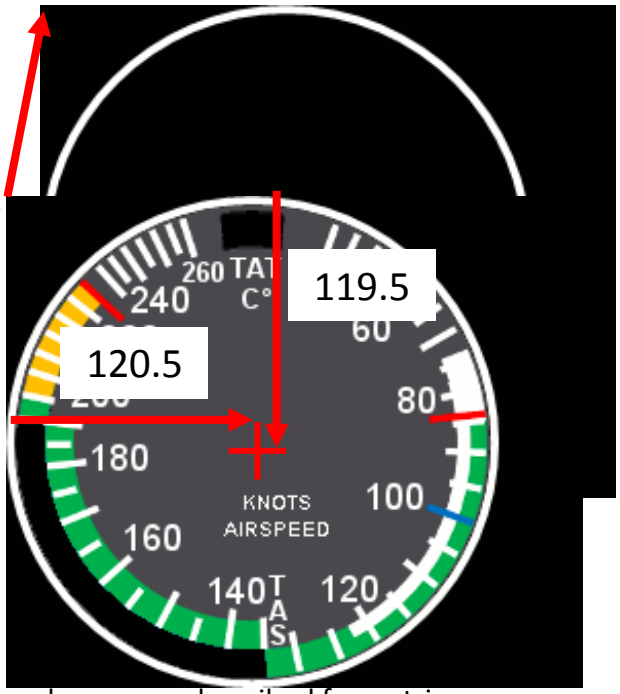
We have already studied how to position the needle and the use of a stripe with a Mask Image, for this case the Total Air Temperature. Now we have a look at the true air speed ring that follows a rotate function.







```
<Image Name="Baron_air_speed.bmp" ImageSizes="281,240,281,240"/>
<Element>
<Position X="0" Y="0"/>
<MaskImage Name="Baron_air_speed_mask.bmp" ImageSizes="281,240,281,240">
<Axis X="120.5" Y="119.5"/>
</MaskImage>
<Image Name="Baron_air_speed_TAS.bmp" ImageSizes="244,244,244,244" >
<Axis X="122.5" Y="121.5"/>
</Image>
<Rotate>
<Value Minimum="0" Maximum="100">100 (A:AIRSPEED TRUE, knots) (A:AIRSPEED INDICATED, knots) - -</Value>
<Nonlinearity>
<Item Value="0" Degrees="250" />
<Item Value="40" Degrees="330" />
<Item Value="100" Degrees="90" />
</Nonlinearity>
</Rotate>
</Element>
```



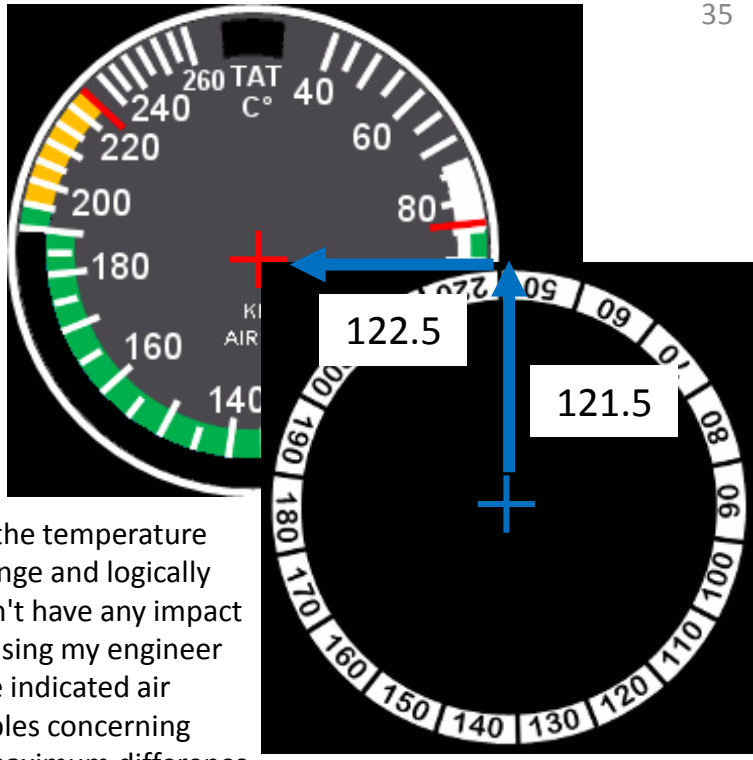
<Position X="0" Y="0"/>
With this paragraph the Mask Image left upper corner will align with the left upper corner of the background bitmap
<MaskImage Name="Baron_air_speed_mask.bmp" ImageSizes="281,240,281,240">
<Axis X="120.5" Y="119.5"/>
</MaskImage>
This paragraph indicates the rotation center position

My conclusion is that using a bitmap as a mask for a rotate function works, using it for a shift function is a nightmare, therefore you should use for the shift function the standing mask usage as described for a stripe.

<Image Name="Baron_air_speed_TAS.bmp" ImageSizes="244,244,244,244" >
<Axis X="122.5" Y="121.5"/>
</Image>
The left upper corner of the True Air speed ring will align with
the rotation center position
of the mask Image.

With the <Axis X="122.5" Y="121.5"/> we indicate that the True air speed ring
Rotation center position must move 122.5 pixels to the left and 121.5 pixels up.

Both the Mask Image and the True air speed bitmap will be aligned with their
Rotation center position. Easy !



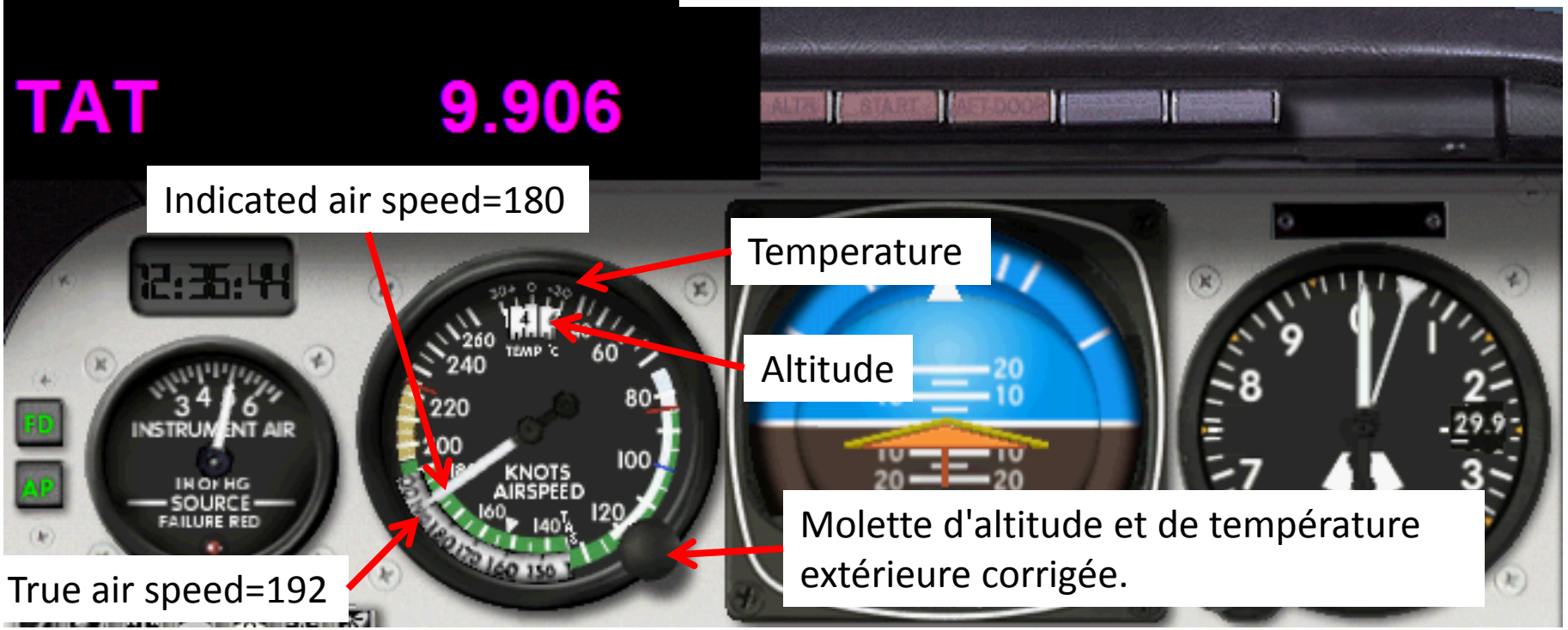
In the FSX Baron air speed gauge the True air speed ring moves in accordance with the temperature that must be change manually (with a knob) during the flight when the altitude change and logically the temperature too. This doesn't work for the FIP, changing the temperature doesn't have any impact on the True air speed ring. I was disappointed and then I had to find a solution. By using my engineer Gauge that I will show you later, I observed that in flight the difference between the indicated air Speed and the True air speed had a near linear value when using the interval variables concerning the air speed (A:AIRSPEED TRUE, knots) and (A:AIRSPEED INDICATED, knots). The maximum difference was of 40 knots, therefore I had to find a linear scale that moves in the right direction with the right value taking into account that on the ground both 140 value for True and indicated speed are aligned. After some tests I decided to have three value, 0 (zero) on the ground, -40 for the minimum and 40 for the maximum, then I had to calculate the angle corresponding to these values. Afterward to ease the process I decided to have 0 as the minimum and 100 for the maximum (80 works also). To find the right formula was not difficult, the goal was to get the difference between True air speed and indicated speed:
(A:AIRSPEED TRUE, knots) (A:AIRSPEED INDICATED, knots) –
and then to subtract the result from the maximum value (100)
100 (A:AIRSPEED TRUE, knots) (A:AIRSPEED INDICATED, knots) - -</
<Rotate>
<Value Minimum="0" Maximum="100">100 (A:AIRSPEED TRUE, knots) (A:AIRSPEED INDICATED, knots) - -</Value>
<Nonlinearity>
<Item Value="0" Degrees="250" />
<Item Value="40" Degrees="330" />
<Item Value="100" Degrees="90" />
</Nonlinearity>
</Rotate>

This example must let you understand that for the FIP nothing is always logical and for some calculation you have to go step by step, changing values or Interval Variable. For the True air speed it works correctly, I am not a mathematician, I just use my brain, trying to understand and overall I am not afraid to spend times trying to find the better solution. I would say that this is probably the most interesting part when creating gauges for FIP.

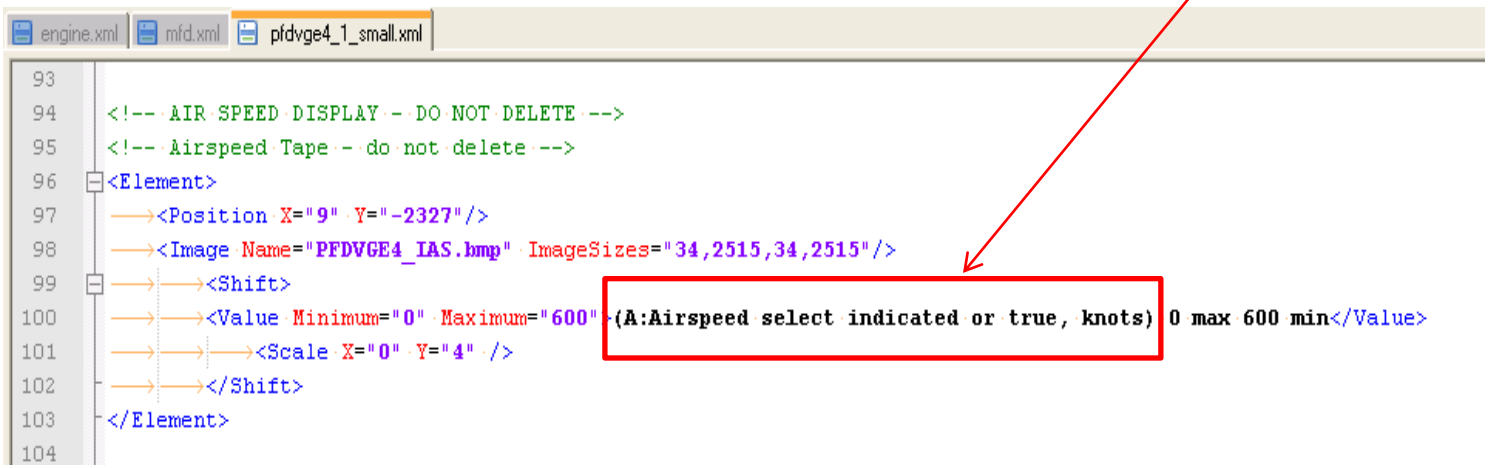
true	192.0
IAS	180.0
T-IAS	11.959
Result	88.041
TAT	9.906

How it works manually,

To get the True Air Speed (TAS) you have to set the altitude in front or the External corrected temperature with the knob. In this case the Indicated Air Speed (IAS) is 180 KTS, thre external corrected température is 9 degrees and the altitude 5000 ft . You must rotate theknob to align the altitude with the external corrected temperature, in fact the 5 with +9. It is not really accurate but the error is very small. Rotating the knob make also rotate theTrue Speed circular scale (in white) it is a simple mecanism that replace a navigation computer. >It is then important to get the external temperature. The needle indicates 180 Kts and on the white scale we can read the True Speed, 192 Kts.



There are multiple possibilities for the airspeed.
Open the xml file you are using, for air speed.
Go to the line AIR SPEED DISPLAY, the string for this item is (A:Airspeed select indicated or true, knots).



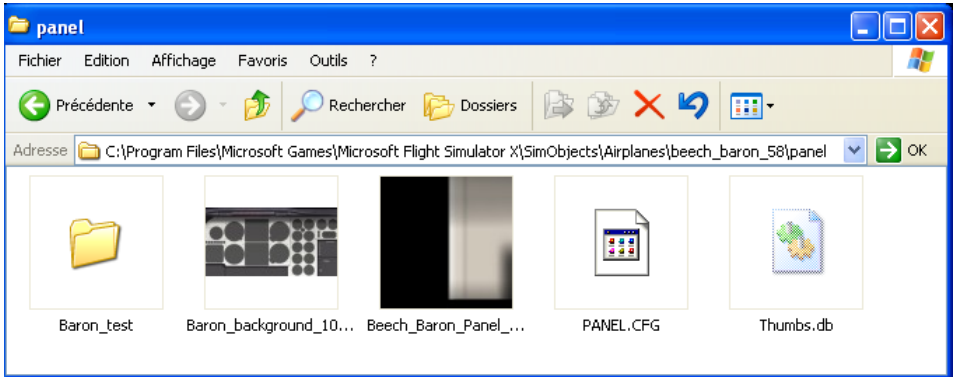
Change the string to have the speed you want to be displayed with one of these strings:
(A:airspeed true calibrate, knots)
or
(A:airspeed true, knots)
or
(A:airspeed indicated, knots)

There is no other value than these four strings for the air speed in FSX and 2004, therefore you should have the one you want by trying the above strings.

It is time to give you my secret: my engineer tool. This is a snapshot of my tool
I have created a gauge for FSX using text variable only, in this snapshot you have the values I wanted to create the air speed gauge. The first value is the true air speed. The second the indicated speed. The third the difference between the two first values. The fourth value is the final result as for
100 (A:AIRSPEED TRUE, knots) (A:AIRSPEED INDICATED, knots) - -
The last is the total air temperature.

Many of you probably know how to create a gauge for FSX but for those who are not familiar with that, I will tell you how to achieve the tool. It is very simple.

1. Create a folder in the aircraft panel folder and name it at will,
I named Baron_test and create a text gauge xml file for FSX (example Is given later)



2. Then open the PANEL.CFG file (make always a backup) and add the line like this:
Window06=TEST

Baron_test.xml | SaiFlightSimX.xml | PANEL.CFG

```
1  // Panel Configuration file
2  // Beech Baron 58
3  // Copyright (c) 2001-2003 Microso
4
5  [Window Titles]
6  Window00=Main Panel
7  Window01=Radio Stack
8  Window02=GPS
9  Window03=Throttle Quadrant
10 Window04=Compass
11 Window05=Mini Panel
12 Window06=TEST
```

3. In the PANEL.CFG file add the [Windo06] paragraph like this:

4. The engineer tool will appear when you launch the aircraft.

Baron_test.xml | SaiFlightSimX.xml | PANEL.CFG

```
148 gauge03=Beech_Baron!Horizontal Situation I
149 gauge04=Beech_Baron!Altimeter,
150 gauge05=Beech_Baron!Vertical Speed Indicat
151
152 [Window06]
153 size_mm=200,200
154 position=50, 50
155 visible=1
156 BACKGROUND_COLOR=2,2,2
157 ident=TEST
158 gauge00=Baron_test!Baron_test, 0, 0
159
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SimBase.Document Type="AceXML" version="1,0" id="Engineer tool">
<Descr>AceXML Document</Descr>
<Filename>Baron_test.xml</Filename>
<SimGauge.Gauge id="annonciator format 100 x 100" ArtDirectory=".">

<Element id="Value 1">
<FloatPosition>4.000,4.000</FloatPosition>
<GaugeText id="Value 1">
                <Bold>True</Bold>
                <Bright>True</Bright>
                <FontFace>Arial</FontFace>
                <FontColor>Yellow</FontColor>
                <FontHeight>20</FontHeight>
                <GaugeString>TRUE</GaugeString>
                <HorizontalAlign>LEFT</HorizontalAlign>
                <VerticalAlign>CENTER</VerticalAlign>
                <Size>70,22</Size>
                <Transparent>True</Transparent>
</GaugeText>
</Element>

<Element id="Value 1 Readout">
<FloatPosition>110.000,4.000</FloatPosition>
<GaugeText id="Value 1 Readout">
                <Bold>True</Bold>
                <Bright>True</Bright>
                <FontFace>Arial</FontFace>
                <FontColor>Yellow</FontColor>
                <FontHeight>20</FontHeight>
                <GaugeString>%((A:AIRSPEED TRUE, knots))%!3.1f!</GaugeString>
                <HorizontalAlign>LEFT</HorizontalAlign>
                <Size>100,22</Size>
                <Transparent>True</Transparent>
</GaugeText>
</Element>
```

This is the xml file.
On the next page you have the positions for the other values' paragraphs.

You can change the colour at will with a plain colour name (White, Red, etc.) or with a colour code like this: #67A4C1.

This is the title of your value, here the word True.

You can change the Interval Variable of the GaugeString at will or put calculation of multiple variables. The significance of the %!3.1f! Is that you will have 3 numbers before the point and 1 after. %!6.3f! means that you will have 6 numbers before the point and 3 after.

Here are the positions of the other text values.

```
<Element id="Value 2">
<FloatPosition>4.000,50.000</FloatPosition>


<Element id="Value 2 Readout">
<FloatPosition>110.000,50.000</FloatPosition>


<Element id="Value 3">
<FloatPosition>4.000,90.000</FloatPosition>

<Element id="Value 3 Readout">
<FloatPosition>110.000,90.000</FloatPosition>

<Element id="Value 4">
<FloatPosition>4.000,130.000</FloatPosition>

<Element id="Value 4 Readout">
<FloatPosition>110.000,130.000</FloatPosition>

<Element id="Value 5">
<FloatPosition>4.000,170.000</FloatPosition>

<Element id="Value 5 Readout">
<FloatPosition>110.000,170.000</FloatPosition>
<GaugeText id="Value 5 Readout">

</GaugeText>
</Element>
</SimGauge.Gauge>
</SimBase.Document>
```

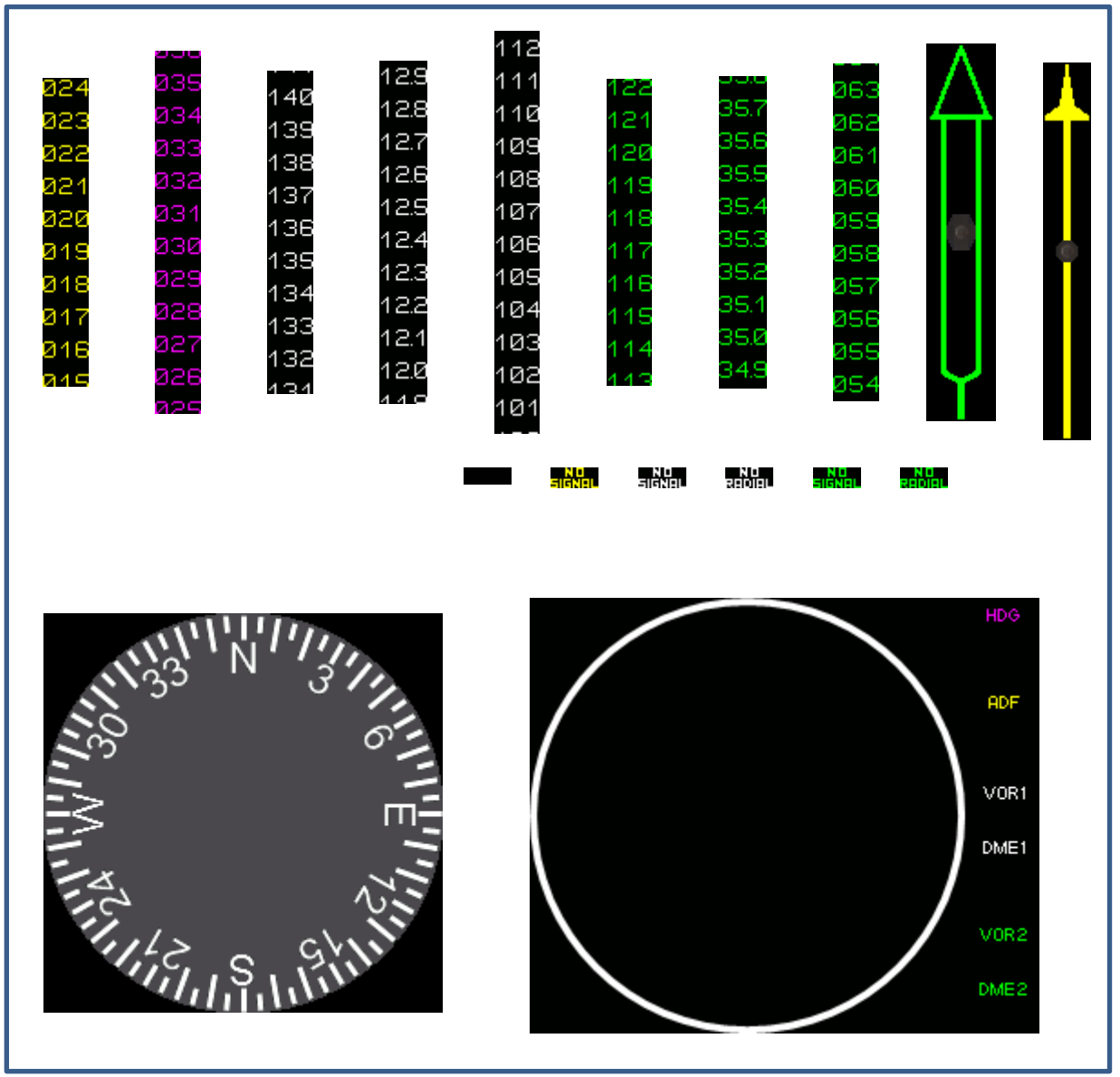
```
<GaugeText id="Value 2">
<Bold>True</Bold>
<Bright>True</Bright>
<FontFace>Arial</FontFace>
<FontColor>Yellow</FontColor>
<FontHeight>20</FontHeight>
<GaugeString>TRUE</GaugeString>
<HorizontalAlign>LEFT</HorizontalAlign>
<VerticalAlign>CENTER</VerticalAlign>
<Size>70,22</Size>
<Transparent>True</Transparent>
</GaugeText>
</Element>
```

```
<GaugeText id="Value 2 Readout">
<Bold>True</Bold>
<Bright>True</Bright>
<FontFace>Arial</FontFace>
<FontColor>Yellow</FontColor>
<FontHeight>20</FontHeight>
<GaugeString>%((A:AIRSPEED INDICATED, knots))%!3.1f!</GaugeString>
<HorizontalAlign>LEFT</HorizontalAlign>
<Size>100,22</Size>
<Transparent>True</Transparent>
</GaugeText>
</Element>
```

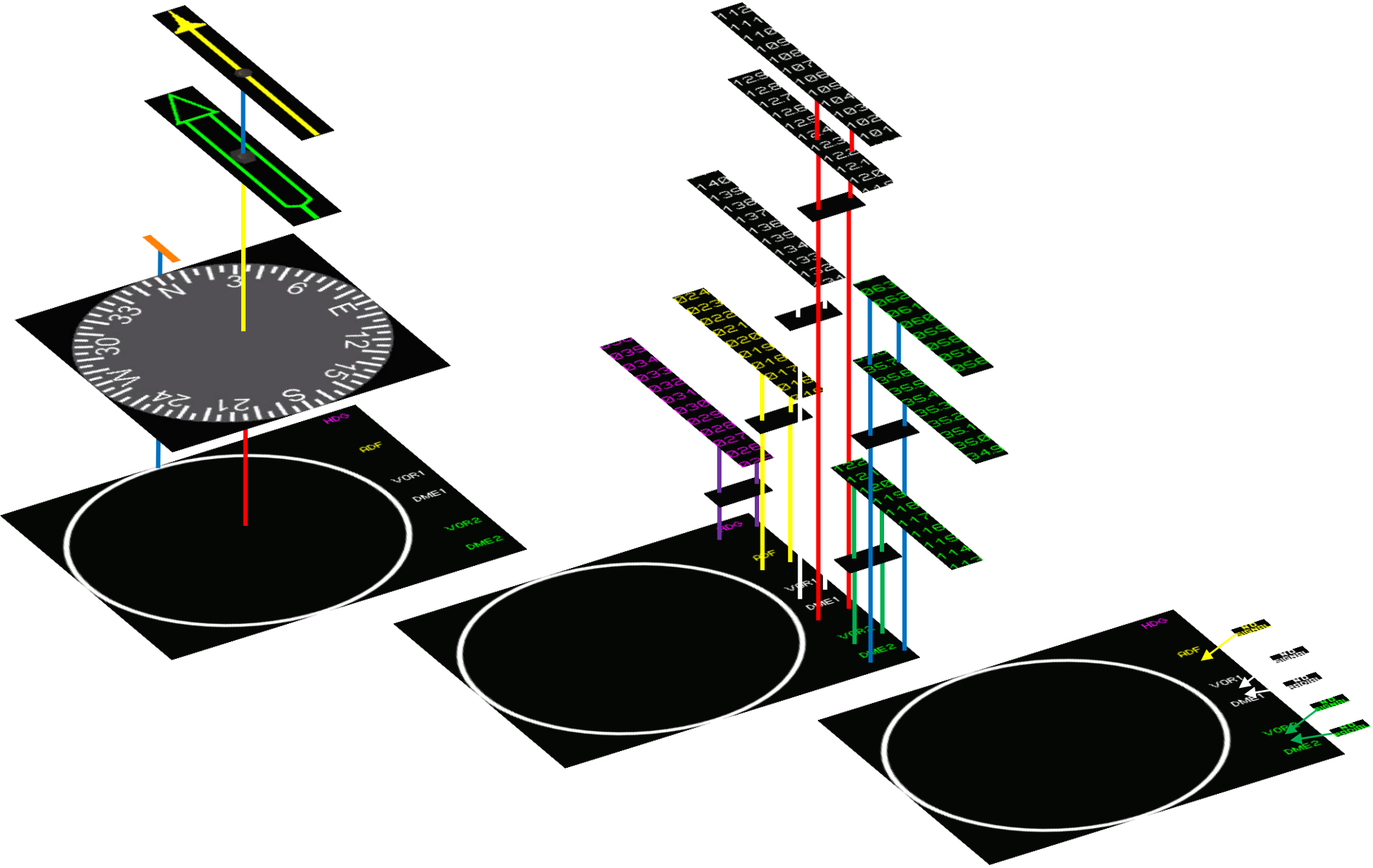
Now we will see how to make a bitmap appear or desappear, for this purpose I use my Baron RMI FIP gauge.

The final gauge is like this

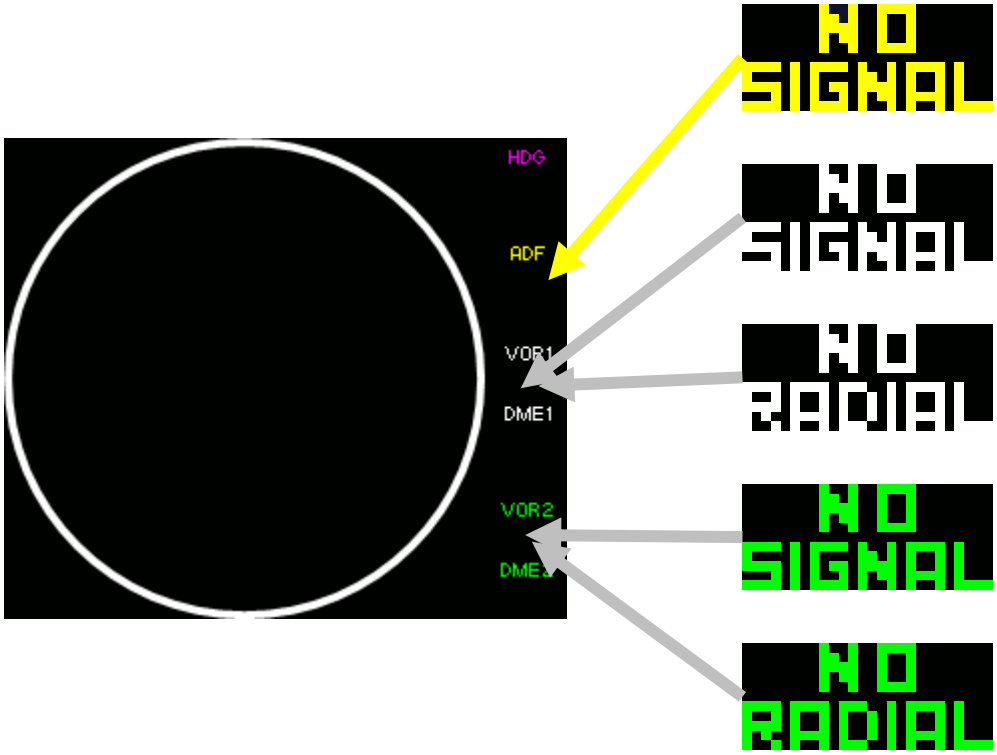
You need all these bitmaps



Baron RMI xml file will do this:



The needles and the stripes were studied in the previous pages, then for this gauge we have to make appear some text when the VOR or ADF are not in range or without signal. For this we use this portion of the xml file. When the beacon is out of range NO SIGNAL will appear, when there is no radial, like for a DME or TACAN used as DME NO RADIAL will appear but the range under the text DME will show up.



This is the xml portion for the ADF flag when the ADF signal is out of range


```
<!-- ADF signal flag display  -->
<Element>
        <Image Name="Baron_ADF_flag.bmp" ImageSizes="26,11,26,11"/>
        <Shift>
                <Value Minimum="0" Maximum="1">(A:ADF1 signal, number) 0 ==</Value>
                <Nonlinearity>
                        <Item Value="1" X="248" Y="67" />
                        <Item Value="0" X="350" Y="67" />
                </Nonlinearity>
        </Shift>
</Element>
```

When in range the string (A:ADF1 signal, number) gives 0 when not it gives 1.
Therefore to have the bitmap appear when the number is 0, the string
(A:ADF1 signal, number) 0 == is true and will give 1 (true=1 false=0).
<Item Value="1" X="248" Y="67" /> in this case when it is true the item Value is 1 and the bitmap moves to the position X=248 and Y= 67.

When out of range the string (A:ADF1 signal, number) gives 1.
Therefore to have the bitmap disappear when the number is 1, the string
(A:ADF1 signal, number) 0 == is false and will give 0.
<Item Value="0" X="350" Y="67" /> in this case when it is false the item Value is 0 and the bitmap moves to the position X=350 and Y= 67.

Remember the maximum width is 320 pixel then moving the bitmap of 350 pixel will move it in a position not visible in the FIP, 30 pixel to the left from the left side of the FIP.

This is the only solution to have a bitmap visible when the item <Visible>(A:ADF1 signal, number) 0 ==</Visible> does not work. The <Visible> criteria works for some value, mainly using the code bool as could it could be (A:ADF1 signal, bool) and which report a 1 or 0 (true or false).

Using the Visible criteria should be like this (this an example and does not work for this string):

```
<Element>
<Visible>(A:ADF1 signal, bool)</Visible>
<Position X="248" Y="67" />
<Image Name="Baron_ADF_flag.bmp" ImageSizes="26,11,26,11"/>
</Element>
```

Sometimes it works when the <Visible> criteria is written like this:

```
<Element>
<Visible>(A:ADF1 signal, bool) 1 ==</Visible>
<Position X="248" Y="67" />
<Image Name="Baron_ADF_flag.bmp" ImageSizes="26,11,26,11"/>
</Element>
```

This permits to spare some lines in the xml file but not so much, just try it and if it doesn't work, use the method with the shift function.

This is the portion for the VOR1 and it is similar to the ADF signal string. For the VOR2 just change the string with <Value Minimum="0" Maximum="1">(A:NAV2 signal, number) 0 ==</Value>

```
<!-- VOR 1 signal flag display -->
<Element>
        <Image Name="Baron_VOR1_flag.bmp" ImageSizes="26,11,26,11"/>
        <Shift>
                <Value Minimum="0" Maximum="1">(A:NAV1 signal, number) 0 ==</Value>
                <Nonlinearity>
                        <Item Value="1" X="248" Y="117" />
                        <Item Value="0" X="350" Y="117" />
                </Nonlinearity>
        </Shift>
</Element>
```

When it is a DME that means that when the VOR is in range you will not have any radial but the distance only. Therefore to get the bitmap NO RADIAL you need the paragraph below.

```
<!-- VOR 1 radial flag display -->
<Element>
		<Image Name="Baron_VOR1_rad_flag.bmp" ImageSizes="26,11,26,11"/>
		<Shift>
			<Value Minimum="0" Maximum="1">(A:NAV1 radial error, bool) 0 ==</Value>
			<Nonlinearity>
				<Item Value="1" X="248" Y="117" />
				<Item Value="0" X="350" Y="117" />
			</Nonlinearity>
		</Shift>
</Element>
```

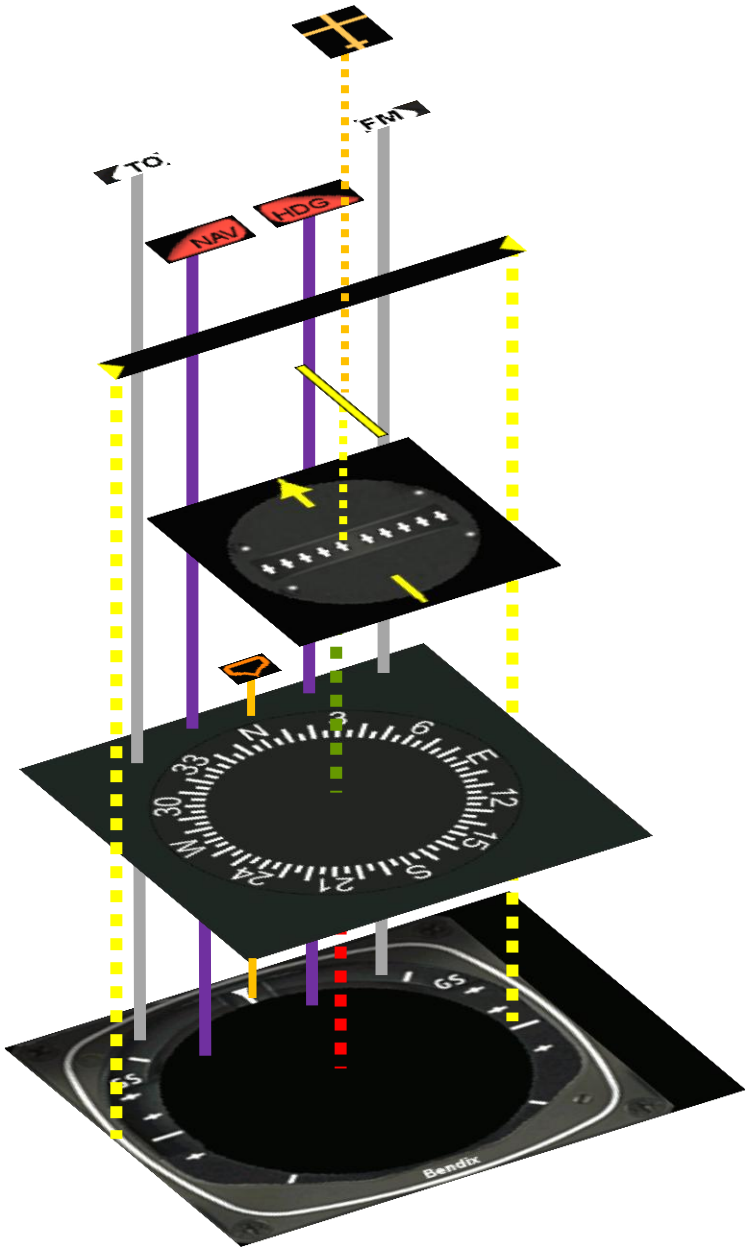
For this example the string (A:NAV1 radial error, bool) 0 == needs the bool unit. I tried with number unit but then the needle doesn't appear, don't ask me why, I am not sure but it seems that there is a direct link with the needle string and the string <Value Minimum="0" Maximum="1">(A:NAV1 radial error, number) 0 ==</Value> that seems to say that there is no radial, therefore the needle do not appear. I know it is strange but sometimes it is very difficult to find the right unit to get it working correctly. The only advice I can give you is to try different values until it works as you whish.

Although my engineer tool is very helpful it doesn't always provide the solution. Creating a FIP that works fine is a matter of patience and perseverance.

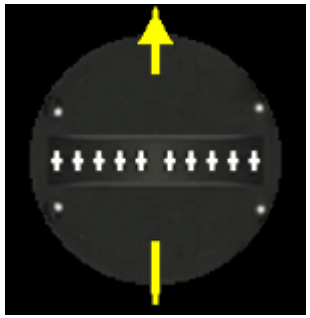
I take the opportunity of this NO RADIAL portion to highlight the first line of the paragraph:
<!-- VOR 1 radial flag display -->
Using <!-- → let you put between the two sets or characters any text and the line will not produce anything nor interfere with the element paragraph. It is very useful to remind you what about is the element concerned. I invite you to use it as many times as necessary to provide a better understanding on what is going on.

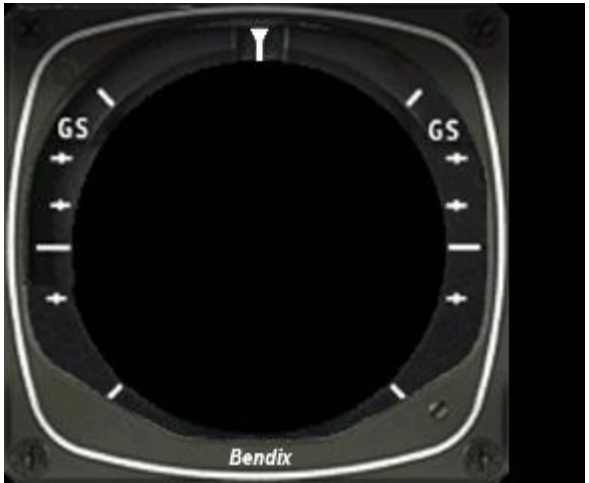
The use of the shift function to get a bitmap visible is of great interest for warning lights that lights up for a failure, for gear lights or flaps. You can find a lot of examples for the FIP I have posted on the web.

Baron HSI xml does this



Baron HSI needs all the bitmaps below and we will look at these two bitmaps.

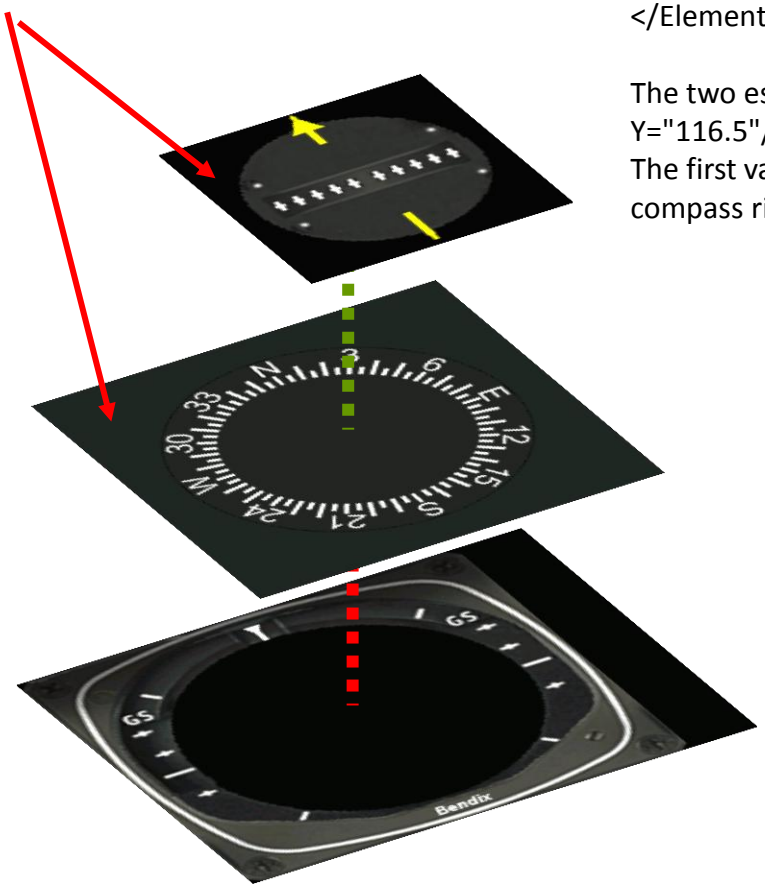




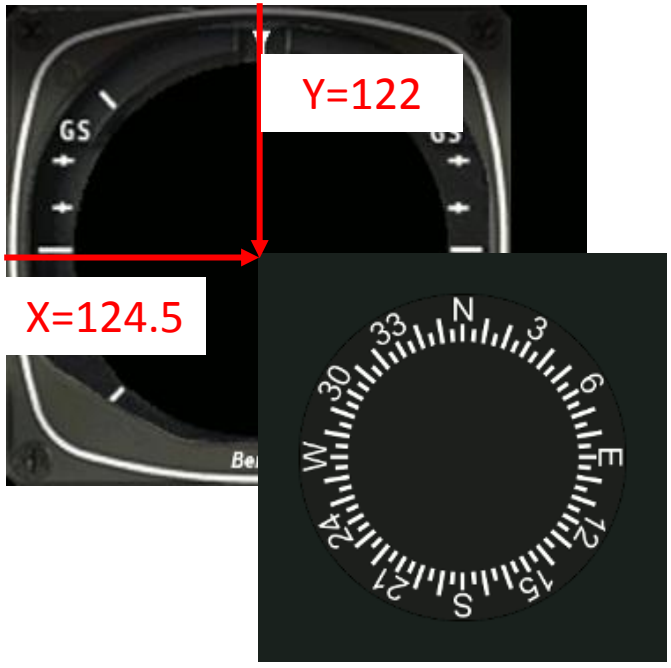
The two bitmaps are positioned above the background image but we do not need to use a mask to have the needed part visible: the compass ring and the course ring. The trick is to colour the parts that must not be visible with the colour code 000000. Then the xml portion is quite easy to create.

```
<!-- Compass card -->
<Element>
<Position X="124.5" Y="122"/>
<Image Name="Baron_HSI_Compass.bmp" ImageSizes="232,232,232,232">
<Axis X="116.5" Y="116.5"/>
</Image>
<Rotate>
<Value>(A:Plane heading degrees gyro,radians) /-/</Value>
</Rotate>
</Element>
```

000000

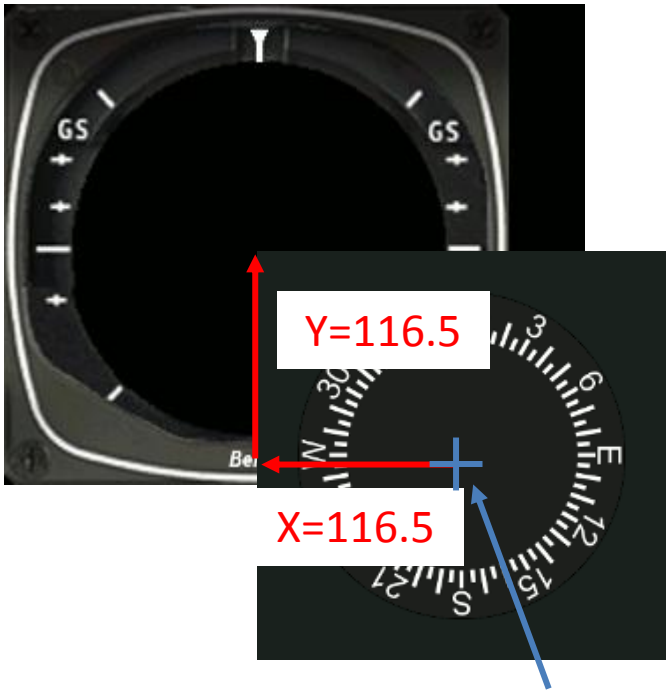


The two essential lines are <Position X="124.5" Y="122"/> and <Axis X="116.5" Y="116.5"/>
The first value <Position X="124.5" Y="122"/> moves the left upper corner of the compass ring at the position where it should rotate over the background bitmap.

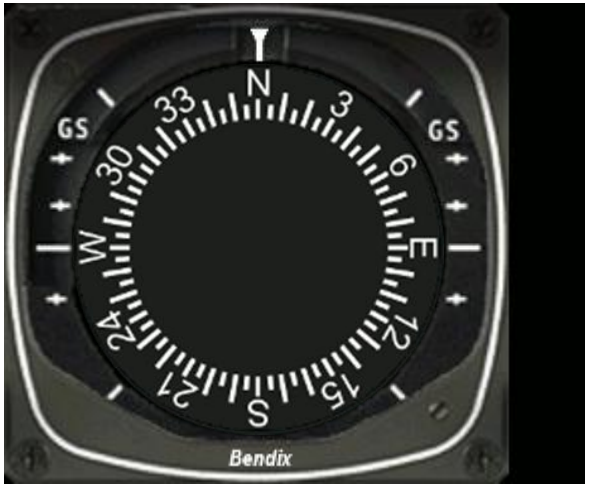


```
<!-- Compass card -->
<Element>
<Position X="124.5" Y="122"/>
<Image Name="Baron_HSI_Compass.bmp" ImageSizes="232,232,232,232">
<Axis X="116.5" Y="116.5"/>
</Image>
<Rotate>
<Value>(A:Plane heading degrees gyro,radians) /-/</Value>
</Rotate>
</Element>
```

The second value <Axis X="116.5" Y="116.5"/> moves the center of rotation of the compass ring at the position where it should rotate over the background bitmap.



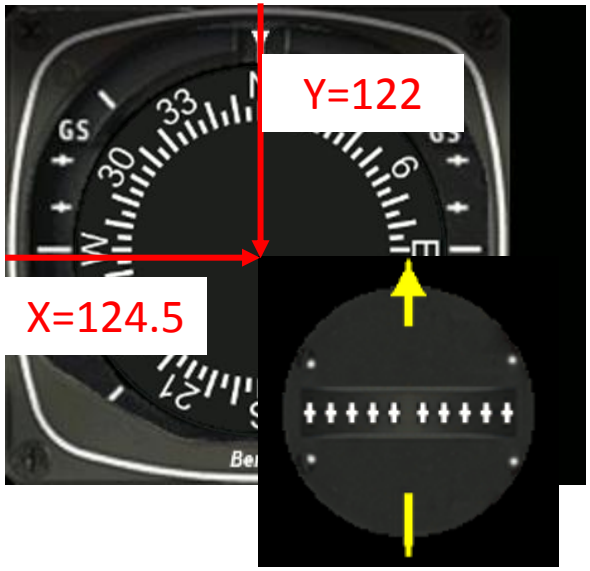
Finally you will have this:

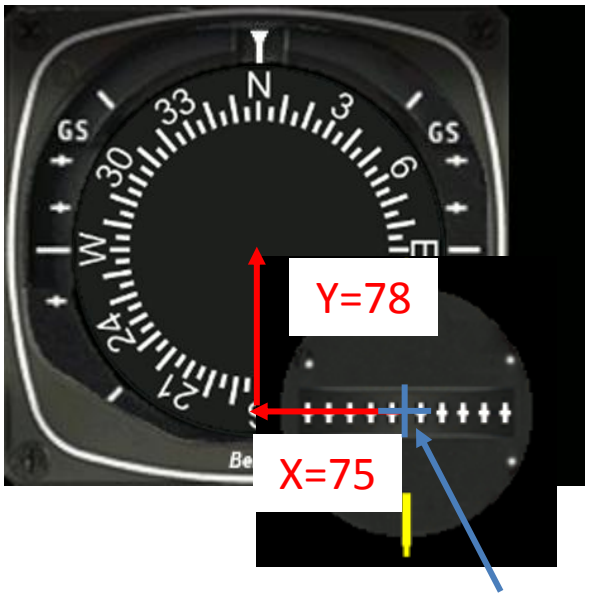


```
<!-- Course card -->
<Element>
<Position X="124.5" Y="122"/>
<Image Name="Baron_HSI_Course.bmp" ImageSizes="150,155,150,155">
<Axis X="75" Y="78"/>
</Image>
<Rotate>
<Value>(A:NAV1 OBS,radians) (A:Plane heading degrees gyro,radians) -</Value>
</Rotate>
</Element>
```

The first value <Position X="124.5" Y="122"/> moves the left upper corner of the course ring at the position where it should rotate over the background bitmap.

The second value <Axis X="75" Y="78"/> moves the center of rotation of the course ring at the position where it should rotate over the background bitmap.



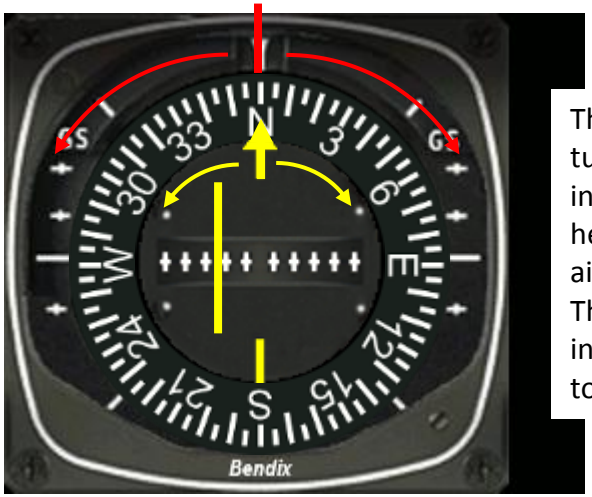


This is the final result

This method is very simple and can be used when there is no need of a Mask Image, that means the background bitmap and the next bitmap, in this case, have a perfect circle (or square) where the two other bitmaps will be visible entirely. This is not always the case and this is the topic for the next part.

Position of a single rotating bitmap for FIP

This is the second method to get the possibility to see only the rotating part. For this you need a single bitmap that contains the rotating part when what must not be seen is colored with colour code 010101. The be able to manage the colour codes, I use a very powerful drawing freeware: The Gimp, http://www.gimp.org/.



The compass ring turns right or left to indicate the heading of the aircraft,
The yellow arrow indicates the course to follow

Back ground image

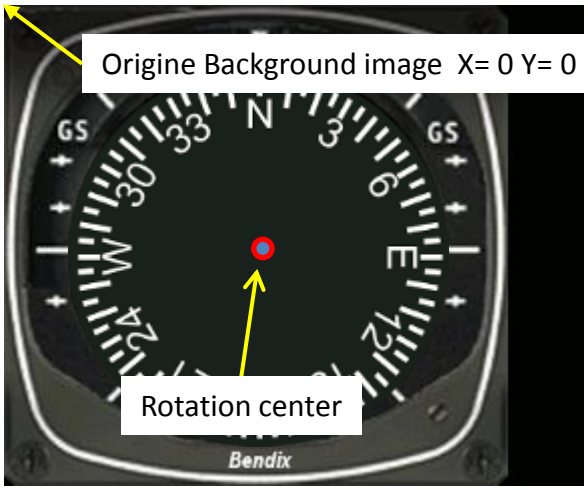
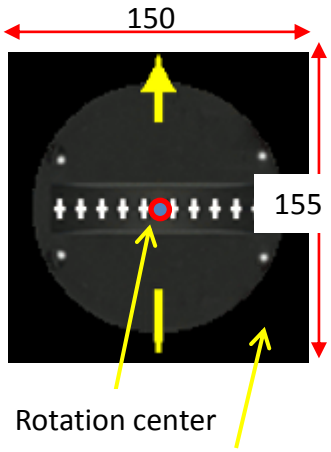
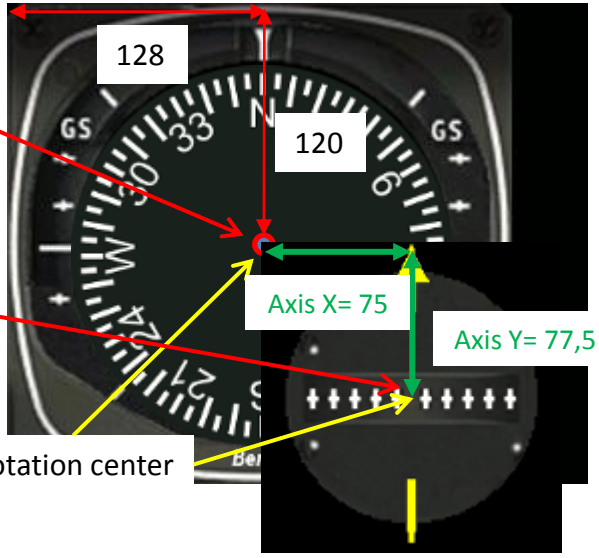


Image to move



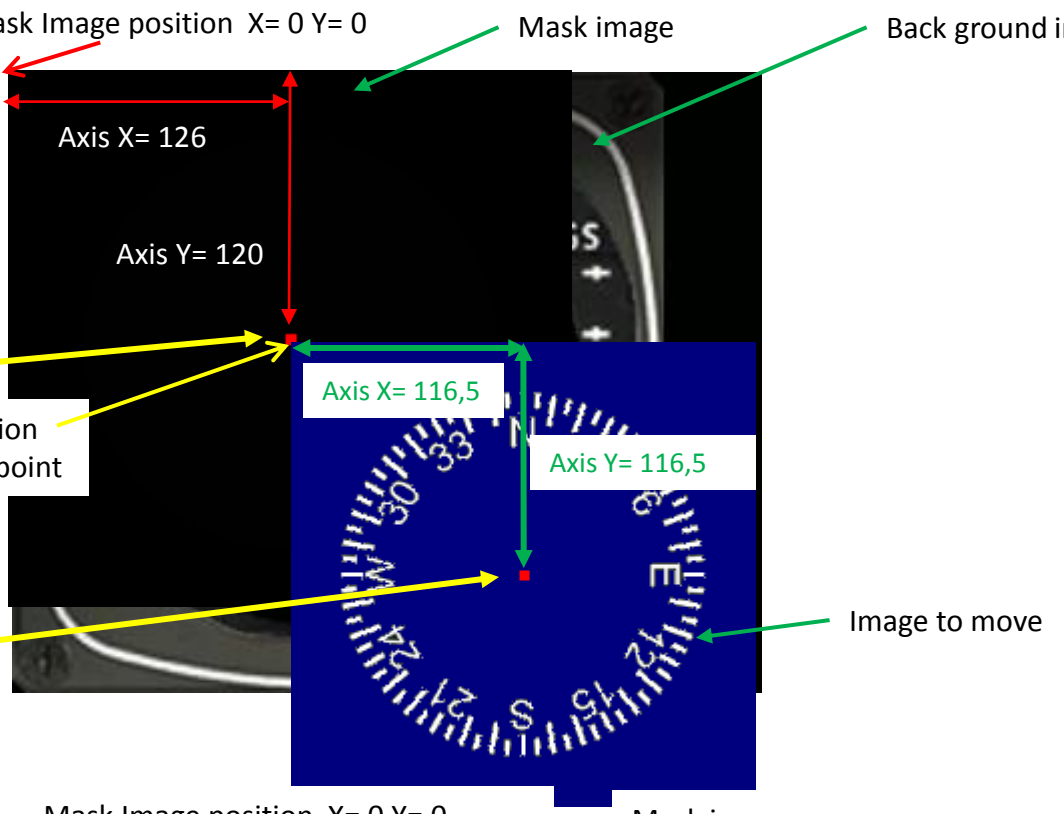
Xml portion

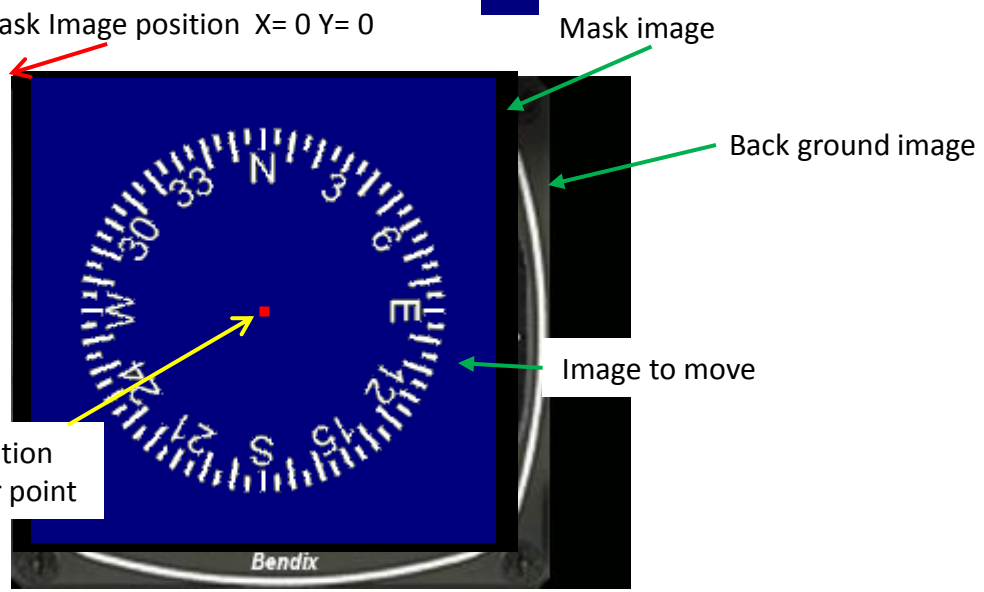
```
<Element>

<Position X="128" Y="120"/>

<Image Name=« Image.bmp"
ImageSizes="150,155,150,155">

<Axis X="75" Y="77.5"/>

</Image>
<Rotate>
<Value>(A:NAV1 OBS,radians)
(A:Plane heading degrees
gyro,radians) -</Value>
</Rotate>
</Element>
```



The image center of rotation will align with the background image rotation center,,

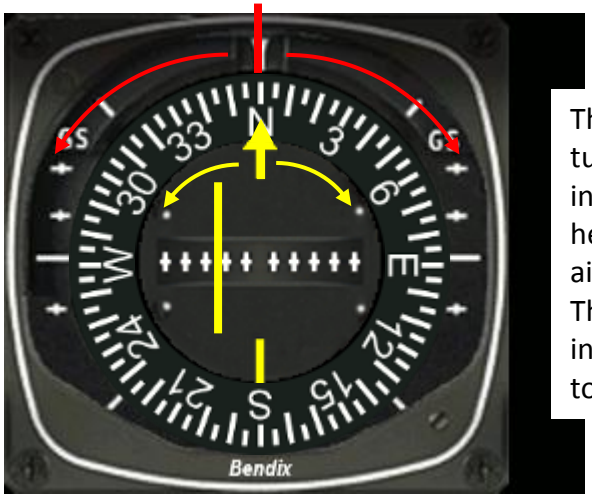
```
<Element>
<Position X="0" Y="0"/>
<MaskImage Name="mask.bmp"

 ImageSizes="252,240,252,240">

<Axis X="126" Y="120"/>

</MaskImage>

<Image Name=« Image.bmp"
ImageSizes="232,232,232,232">

<Axis X="116.5" Y="116.5"/>

</Image>

<Rotate>
<Value>(A:Plane heading degrees
gyro,radians) /-/</Value>
</Rotate>
</Element>
```





Position of a single rotating bitmap for FIP

This is the second method to get the possibility to see only the rotating part. For this you need a single bitmap that contains the rotating part when what must not be seen is colored with colour code 010101. The be able to manage the colour codes, I use a very powerful drawing freeware: The Gimp, http://www.gimp.org/.



The compass ring turns right or left to indicate the heading of the aircraft,
The yellow arrow indicates the course to follow

Back ground image

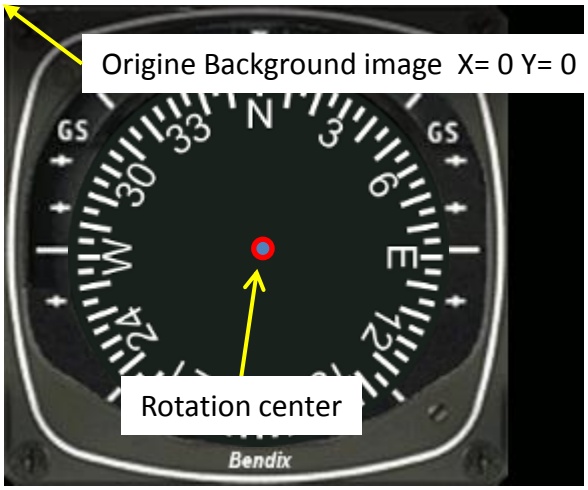
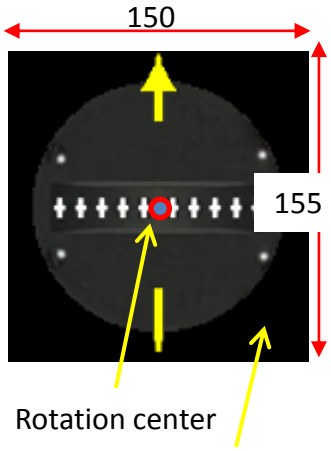


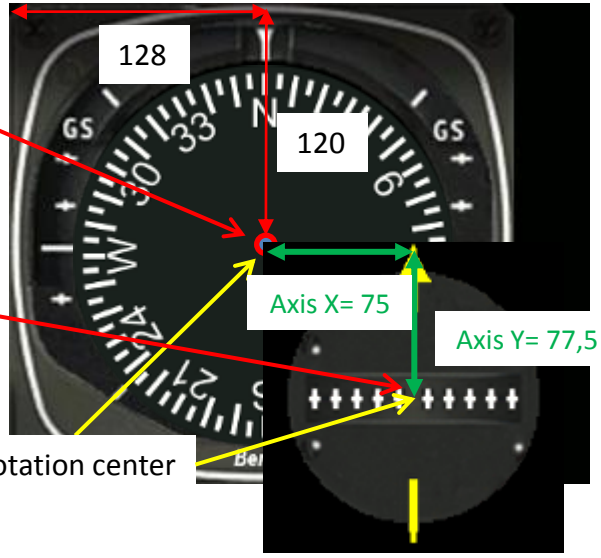
Image to move



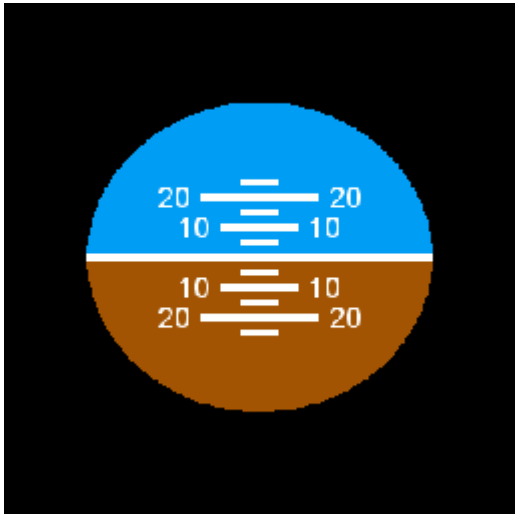
Background color is 000000 invisible

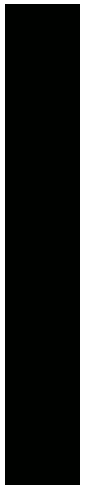
Xml portion

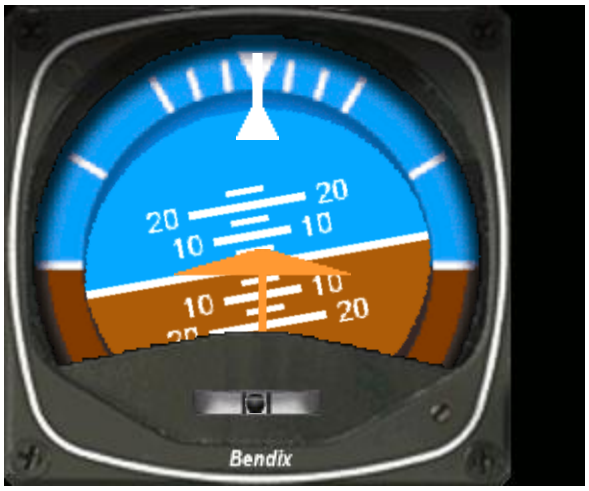
```
<Element>

<Position X="128" Y="120"/>

<Image Name=« Image.bmp"
ImageSizes="150,155,150,155">

<Axis X="75" Y="77.5"/>

</Image>
<Rotate>
<Value>(A:NAV1 OBS,radians)
(A:Plane heading degrees
gyro,radians) -</Value>
</Rotate>
</Element>
```



The image center of rotation will align with the background image rotation center,,

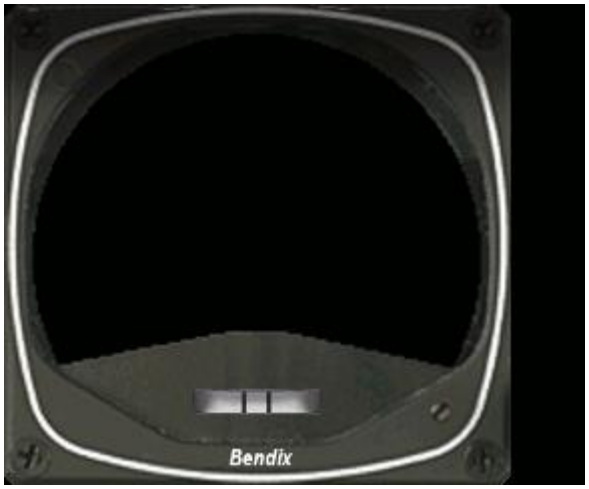




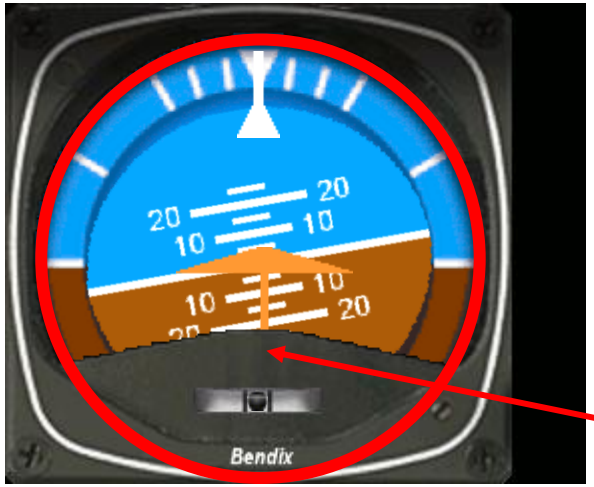


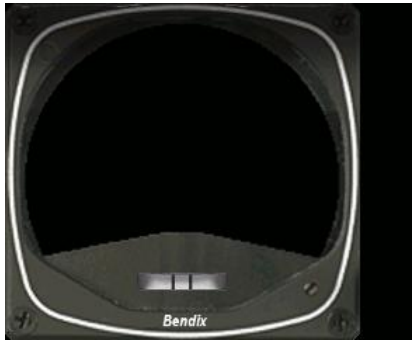
Baron VSI

Inside



Baron VSI





Inside

The method described before can't be used because the background part that should let see the moving bitmaps is not circular.

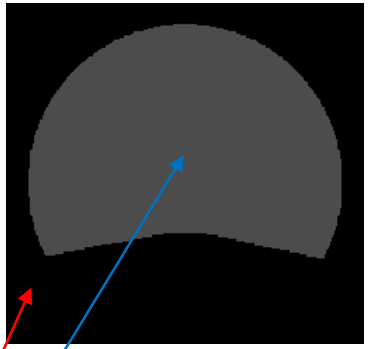
Baron_attitude_card_outside.bmp

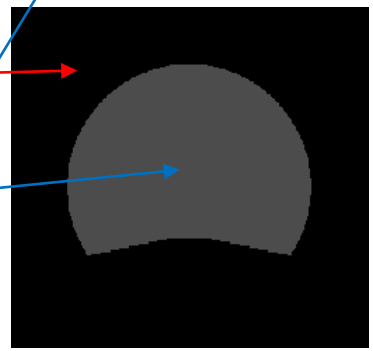
Baron_mask_attitude_card_outside.bmp

The mask seems to be full black, however it has two black colour, the code 000000, invisible that let see the parts of the background bitmap that must be visible. The other colour code is 010101 that allows the mask to show the only part of the associated bitmap to be visible.

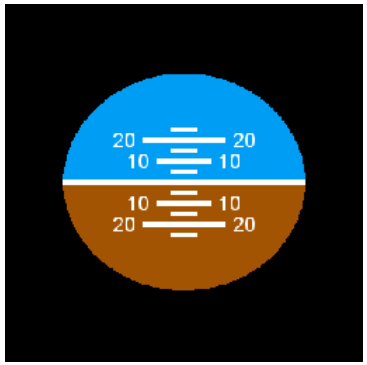
For this study I coloured in black the code 000000

and in grey the black colour code 010101.



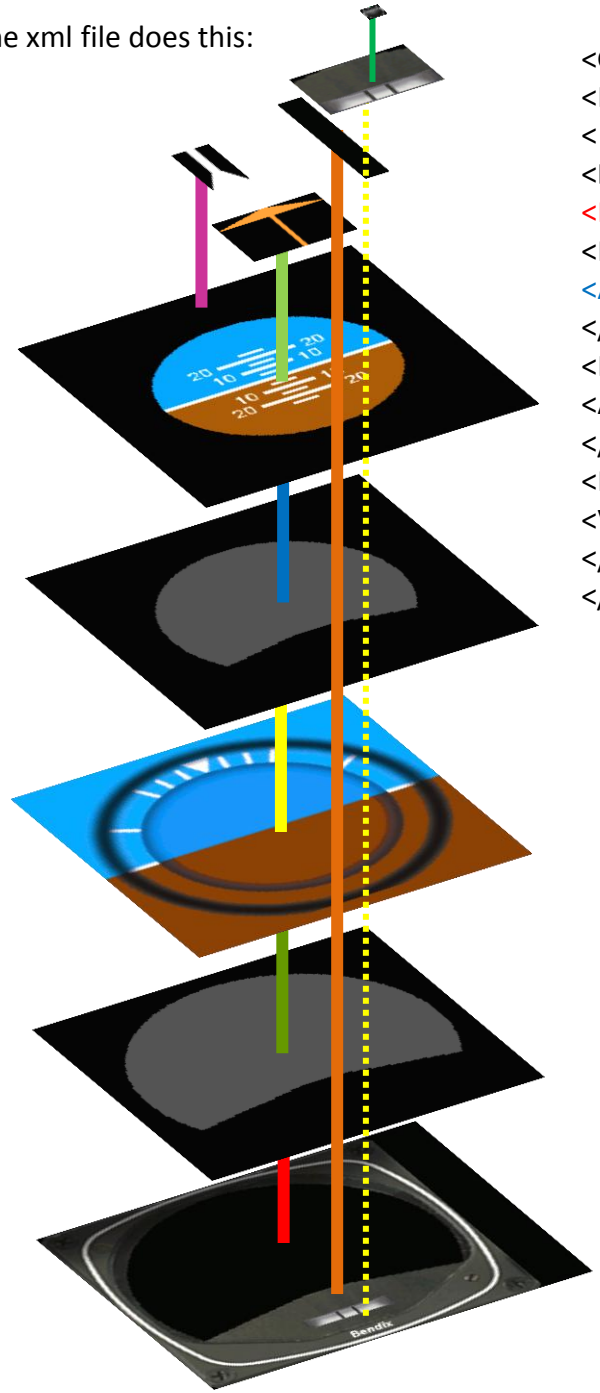


Baron_attitude_card_inside.bmp

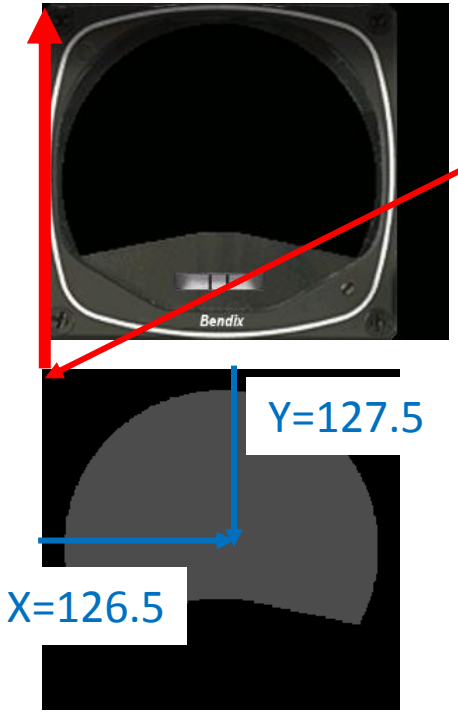


Baron_mask_attitude_card_inside.bmp

The xml file does this:

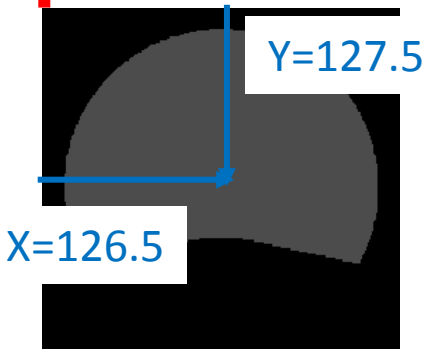


```
<Gauge Name="Baron attitude Indicator" Version="1.0">
<Image Name="Baron_VSI.bmp" ImageSizes="290,240,290,240"/>
<!-- Bank indicator -->
<Element>
<Position X="0" Y="0"/>
<MaskImage Name="Baron_mask_attitude_card_outside.bmp" ImageSizes="255,243,255,243">
<Axis X="126.5" Y="127.5"/>
</MaskImage>
<Image Name="Baron_attitude_card_outside.bmp" ImageSizes="255,255,255,255">
<Axis X="126.5" Y="127.5"/>
</Image>
<Rotate>
<Value>(A:Attitude indicator bank degrees,radians)</Value>
</Rotate>
</Element>
```

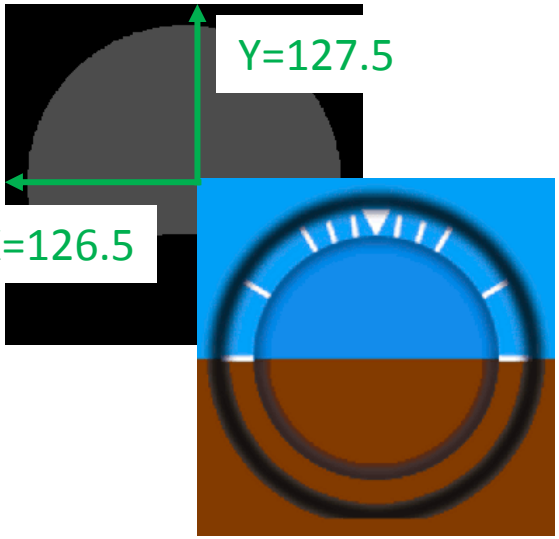


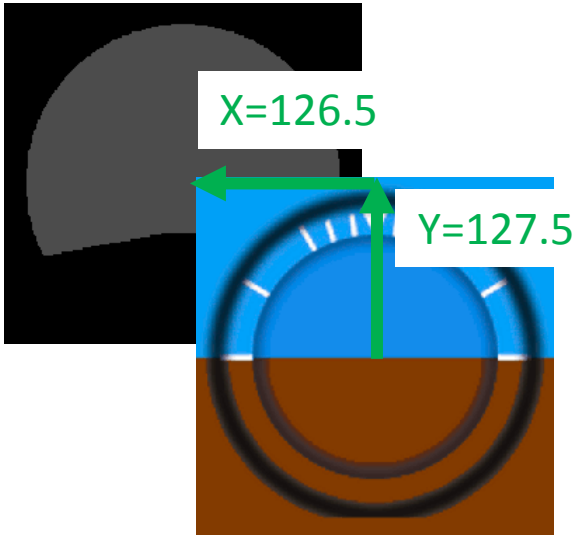
<Position X="0" Y="0"/> indicates that the left upper corner of the Mask Image will align with the left upper corner of the background bitmap

<Axis X="126.5" Y="127.5"/>
Indicates the position of the rotation center in the Mask Image




```
<Gauge Name="Baron attitude Indicator" Version="1.0">
<Image Name="Baron_VSI.bmp" ImageSizes="290,240,290,240"/>
<!-- Bank indicator -->
<Element>
<Position X="0" Y="0"/>
<MaskImage Name="Baron_mask_attitude_card_outside.bmp" ImageSizes="255,243,255,243">
<Axis X="126.5" Y="127.5"/>
</MaskImage>
<Image Name="Baron_attitude_card_outside.bmp" ImageSizes="255,255,255,255">
<Axis X="126.5" Y="127.5"/>
</Image>
<Rotate>
<Value>(A:Attitude indicator bank degrees,radians)</Value>
</Rotate>
</Element>
```





The image align its left upper corner with the Mask Image axis position
<Axis X="126.5" Y="127.5"/>

<Axis X="126.5" Y="127.5"/> indicates that the left upper corner of the Mask Image will move up 126.5 pixels and left 127.5 pixels. In another hand you can consider that the center of rotation of the Image will align with the center rotation points of the Mask Image. This is exactly the same. I prefer to think moving the center.
Be aware that the value X and Y seams to be identical but it is due only by the fact that Mask Image and the Image have the same size, values could be different.

I recommend to paint in black the useless part of the bitmap (transparent black 000000) to avoid coloured parts visible outside of the Mask Image range.

The coloured parts is visible

The coloured parts is not visible

If you look at the rotate function there is no value given for the movement as for Linearity or scale.

<Rotate>
<Value>(A:Attitude indicator bank degrees,radians)</Value>
</Rotate>

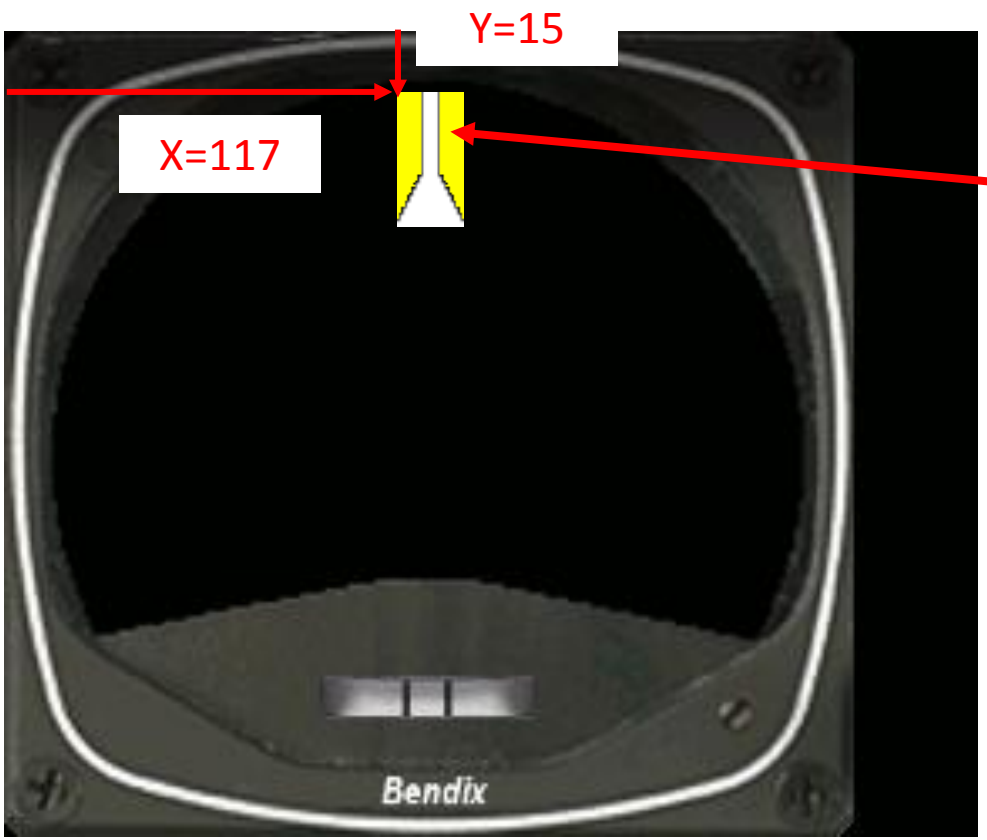
For this particular case there is no need to provide Minimum, Maximum, Linearity or scale because the movement act as a compass do and moves degree per degree in both direction. Note that the unit must be radians because Degrees doesn't work for this function and for the FIP. I can't explain why but I discovered this by chance.

For the other Mask Image and associated image just follow the same process. For the other bitmaps that are fixed, you only have to put the position values where the upper left corner of the bitmap should go.

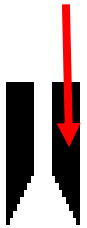
```
<!-- Index arrow -->
<Element>
        <Position X="117" Y="15"/>
        <Image Name="Baron_attitude_arrow.bmp" ImageSizes="21,43,21,43"/>
</Element>
```

For the other Mask Image and associated image just follow the same process. For the other bitmaps that are fixed, you only have to put the position values where the upper left corner of the bitmap should go.

```
<!-- Index arrow -->
<Element>
<Position X="117" Y="15"/>
<Image Name="Baron_attitude_arrow.bmp" ImageSizes="21,43,21,43"/>
</Element>
```



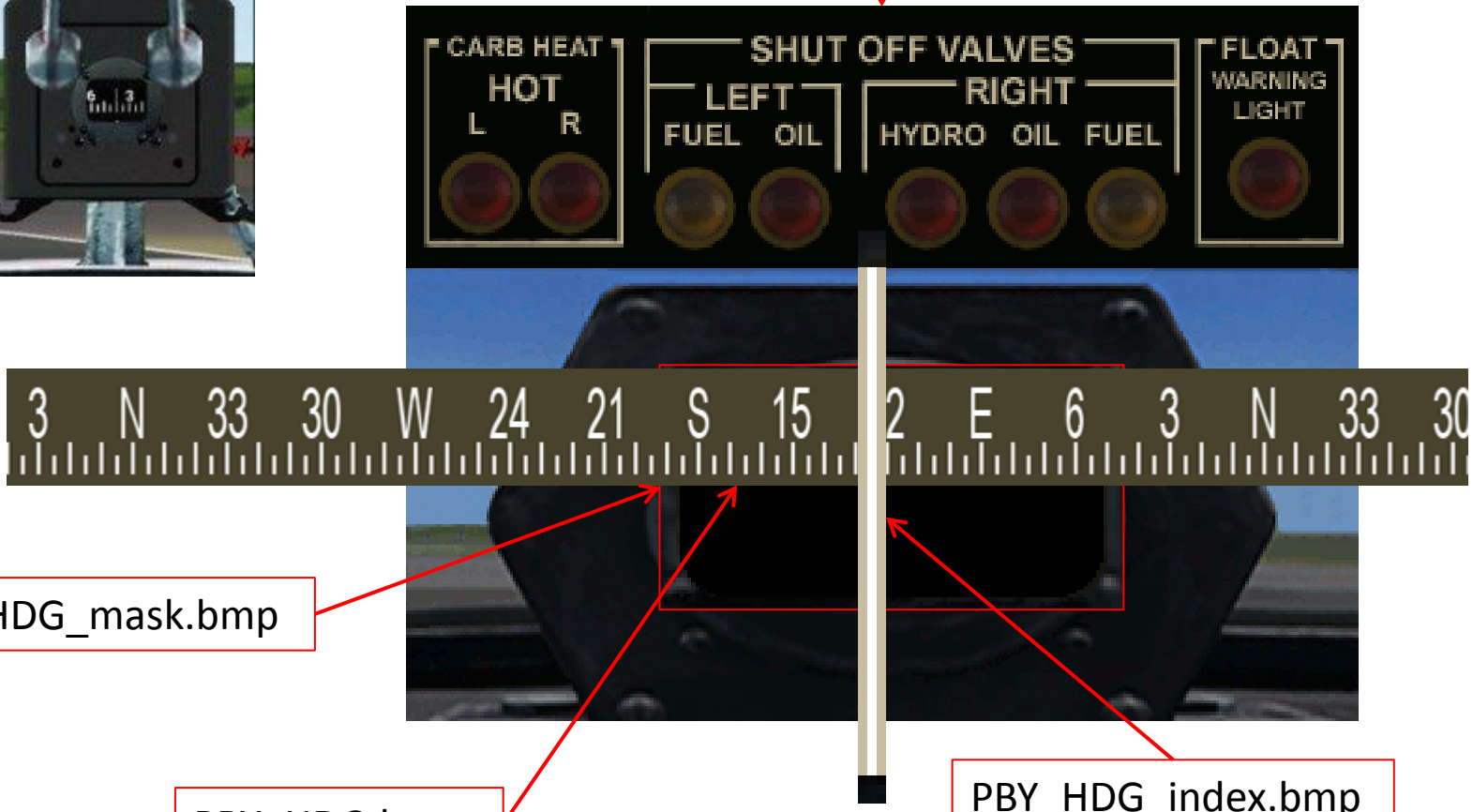
I coloured in Yellow to have a better view of the bitmap. The colour should be black transparent 000000.



The order of the bitmaps follows the position of the <Element> portions in the xml file, therefore the first <Element> will be the background bitmap, the second element will be above the background bitmap, the third will be above the second <Element> and so on. Its is mandatory to have a good understanding of what must be seen otherwise a bitmap could be hidden by the following <Element> portion.

How to create a magnetic compass

We wiill look at a PBY magnetic compass I made for a Canadian friend, The main difficulty is to calibrate the heading stripe to be sure it indicates the right heading when you fly.



```
<?xml version="1.0" encoding="utf-8"?>
<Gauge Name="Catalina boussole" Version="1.0">
<!-- Background -->
<Image Name="PBY_Bousole_bckgrd.bmp" ImageSizes="320,240,320,240" />
<!-- Boussole -->
<Element>
            <Position X="102" Y="142" />
    <MaskImage Name="PBY_HDG_mask.bmp">
       <Axis X="58" Y="0"/>
     </MaskImage>
            <Image Name="PBY_HDG.bmp" ImageSizes="779,62,779,62"/>
            <Shift>
            <Value Minimum="0" Maximum="360">(A:Wiskey compass indication
degrees,degrees) dnor</Value>
                        <Nonlinearity>
                                    <Item Value="0" X="670.5" Y="0" />
                                    <Item Value="360" X="67.5" Y="0" />
                        </Nonlinearity>
            </Shift>
</Element>
<!-- Boussole index-->
<Element>
            <Position X="159.5" Y="142" />
            <Image Name="PBY_HDG_index.bmp" ImageSizes="3,62,3,62"/>
</Element>
</Gauge>
```

If you make a mistake and write <Item Value="0" X="67.5" Y="0" /> <Item Value="360" X="670.5" Y="0" /> The heading will move in the wrong way eg. You will read 350 when you are heading 010.

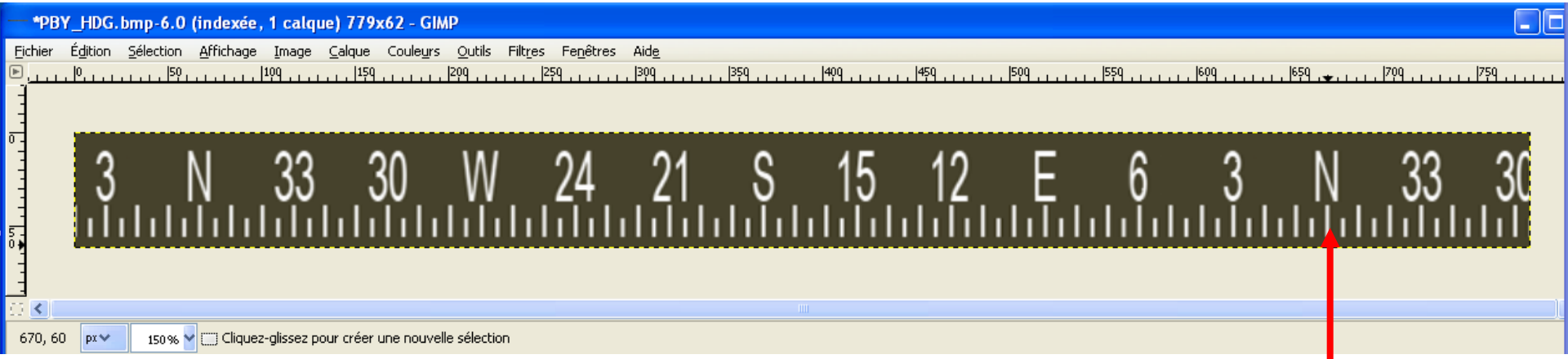
Let study this portion to understand values

<Nonlinearity>

<Item Value="0" X="670.5" Y="0" />

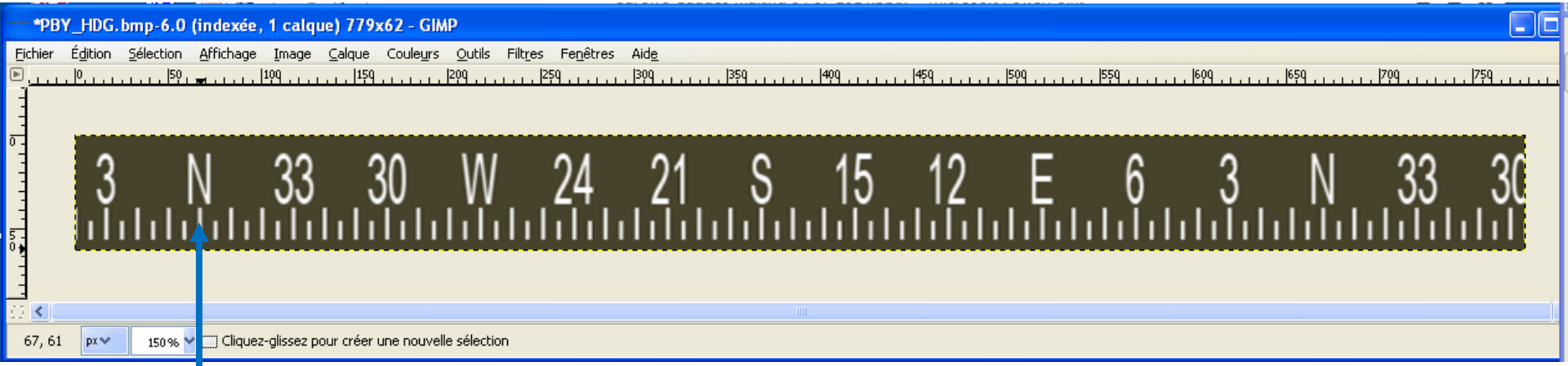
<Item Value="360" X="67.5" Y="0" />

</Nonlinearity>



<Item Value="0" X="670.5" Y="0" />

X=670,5 is where the red arrow points, 670,5 pixels on the right from the left border of the bmp.



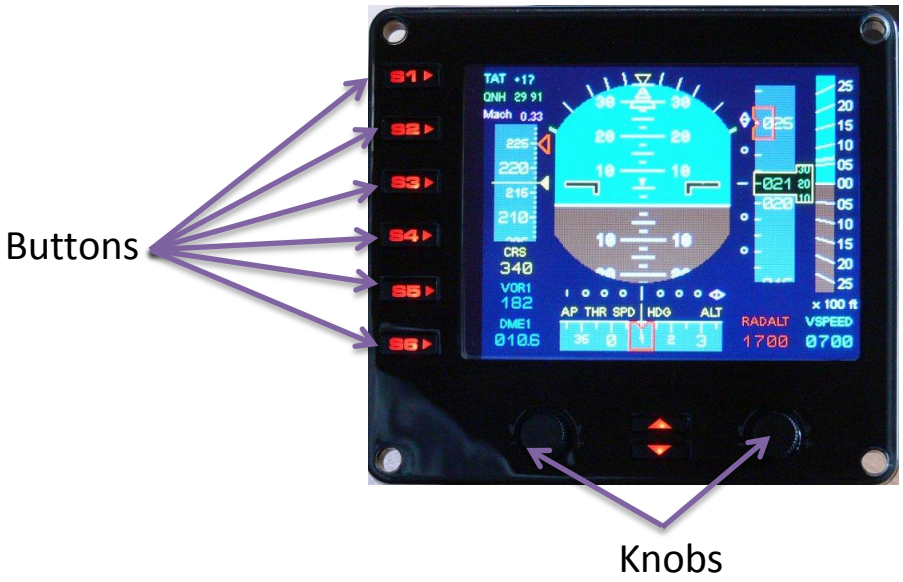
<Item Value="360" X="67.5" Y="0" />

X=67,5 is where the blue arrow points, 67,5 pixels on the right from the left border of the bmp,

FIP management

TUTORIAL FOR SAITEK PRO FLIGHT INSTRUMENT PANEL BUTTONS AND KNOBS

This tutorial explains how to manage buttons and knobs for the FIP. Six buttons are available on the left side of the FIP and two rotating knobs on the lower side.



How it works.

Buttons are like any buttons on add-on instruments, keyboard or joystick. When pressed they send an instruction which is understood and translated as a command to FSX and to the FIP if this command acts within the FIP. For example, I select Master Autopilot command for button 1, when pressed the autopilot engages and the yellow AP appears in the FIP above. There is a list of available commands provided by fs2x.com, also listed in the Key events.txt file provided and which can be found there: http://www.fs2x.com/Tutorials_files/FS2004_Click_Events.txt

Original commands.

Buttons' commands are located in the file SaiFlightSimX.xml in the folder C:\Program Files\Saitek\DirectOutput. The file shows this:

```
<?xml version="1.0" encoding="utf-8"?>
<GaugeList>
  <X52Pro>
    <ComNav radiol="true"/>
    <ComNav radiol="false"/>
    <Adf/>
    <Dme/>
    <Xpndr/>
    <Autopilot/>
    <VorHdg/>
  </X52Pro>
  <Gauge Timeout="100" RootFolder="Relative" File="Gauges\Altimeter.xml">
    <Button Id="1" Name="Map" Event="FLIGHT_MAP"/>
    <Button Id="2" Name="Main Panel" Event="PANEL_1"/>
    <Button Id="3" Name="Radios" Event="PANEL_2"/>
    <Button Id="4" Name="GPS" Event="PANEL_3"/>
    <Button Id="5" Name="Panel 4" Event="PANEL_4"/>
    <Button Id="6" Name="Panel 5" Event="PANEL_5"/>
  </Gauge>
```

Commands are located in the paragraph <Gauge> </Gauge>.
The first line indicates which gauge you want to load and from where. In this example the xml file will show altimeter gauge and is located in the Gauges folder from C:\Program Files\Saitek\DirectOutput.
The six remaining lines concern the buttons and associated commands. Take the first.
<Button Id=”1” Name=”Map” Event=”FLIGHT_MAP”/>

Button Id=”1” tells that following instruction concerns the first button, named S1 on the FIP.

Name=”Map” tells what text will appear on the FIP beside the first button, here you should read: Map.

Remark:
For my xml files I do not put a name because it takes space that could be available for a larger gauge, see my pfdvge gauge as sample. However it is your choice to have or not a text on your FIP.

Event=”FLIGHT_MAP” tells what command will be sent when you push the button. You can put any command available for FSX. Commands are listed in Key Events.txt file.

Example of my SaiFlightSimX.xml file for my pfdvge gauge.

```
15
16  <Gauge Timeout="100" RootFolder="Relative" File="Gauges\pfdvge.xml">
17      <Button Id="1" Name="" Event="AP_MASTER"/>
18      <Button Id="2" Name="" Event="AP_ALT_HOLD"/>
19      <Button Id="3" Name="" Event="AP_HDG_HOLD"/>
20      <Button Id="4" Name="" Event="AP_APR_HOLD"/>
21      <Button Id="5" Name="" Event="AP_NAV1_HOLD"/>
22      <Button Id="6" Name="" Event="AP_AIRSPEED_HOLD"/>
23  </Gauge>
24
```

In this above example when I push the button S1 the autopilot master moves ON. When I push again it moves OFF.
For button S2 the altitude function of the autopilot moves ON or OFF. And so on……..
As you can see it is very easy and I am sure you will enjoy your own selection of commands.

Knobs.

How it works.

Knobs' commands are located in the file yourgaugename.xml in the folder C:\Program Files\Saitek\DirectOutput\Gauges. Commands for knobs are listed as mouse command at the very end of the file. The first mouse command concern the right knob, the second concerns the left knob.

For example in my pfdvge.xml the file shows this:

```
<!-- Right knob controls QNH -->
<Mouse>
<Area>
    <Area>
        <Cursor Type="DownArrow" />
        <Click Event="KOHLSMAN_DEC" Repeat="Yes" />
    </Area>
    <Area>
        <Cursor Type="UpArrow" />
        <Click Event="KOHLSMAN_INC" Repeat="Yes" />
    </Area>
</Area>
</Mouse>

<!-- Left knob moves VOR 1 course -->
<Mouse>
<Area>
    <Area>
        <Cursor Type="DownArrow"/>
        <Click Event="VOR1_OBI_DEC" Repeat="Yes"/>
    </Area>
    <Area>
        <Cursor Type="UpArrow"/>
        <Click Event="VOR1_OBI_INC" Repeat="Yes"/>
    </Area>
</Area>
</Mouse>
</Gauge>
```

Take the first Mouse group.
Cursor Type=”DownArrow” tells that the command will be sent when the knob is moved anti clockwise (to the left).
Cursor Type=”UpArrow” tells that the command will be sent when the knob is moved clockwise (to the right).
Click Event=”KOHLSMAN_DEC” tells that when you move the knob to the left it will send the command to decrease QNH pressure.
Click Event=”KOHLSMAN_INC” tells that when you move the knob to the right it will send the command to increase QNH pressure.
Repeat=”Yes” means that the command will be sent continuously for each turn as for real knob.

```
794  <!-- Right knob controls QNH -->
795  <Mouse>
796  <Area>
797      <Area>
798          <Cursor Type="DownArrow" />
799          <Click Event="KOHLSMAN_DEC" Repeat="Yes" />
800      </Area>
801      <Area>
802          <Cursor Type="UpArrow" />
803          <Click Event="KOHLSMAN_INC" Repeat="Yes" />
804      </Area>
805  </Area>
806  </Mouse>
807
808  <!-- Left knob moves VOR 1 course -->
809  <Mouse>
810  <Area>
811      <Area>
812          <Cursor Type="DownArrow"/>
813          <Click Event="VOR1_OBI_DEC" Repeat="Yes"/>
814      </Area>
815      <Area>
816          <Cursor Type="UpArrow"/>
817          <Click Event="VOR1_OBI_INC" Repeat="Yes"/>
818      </Area>
819  </Area>
820  </Mouse>
821  </Gauge>
```

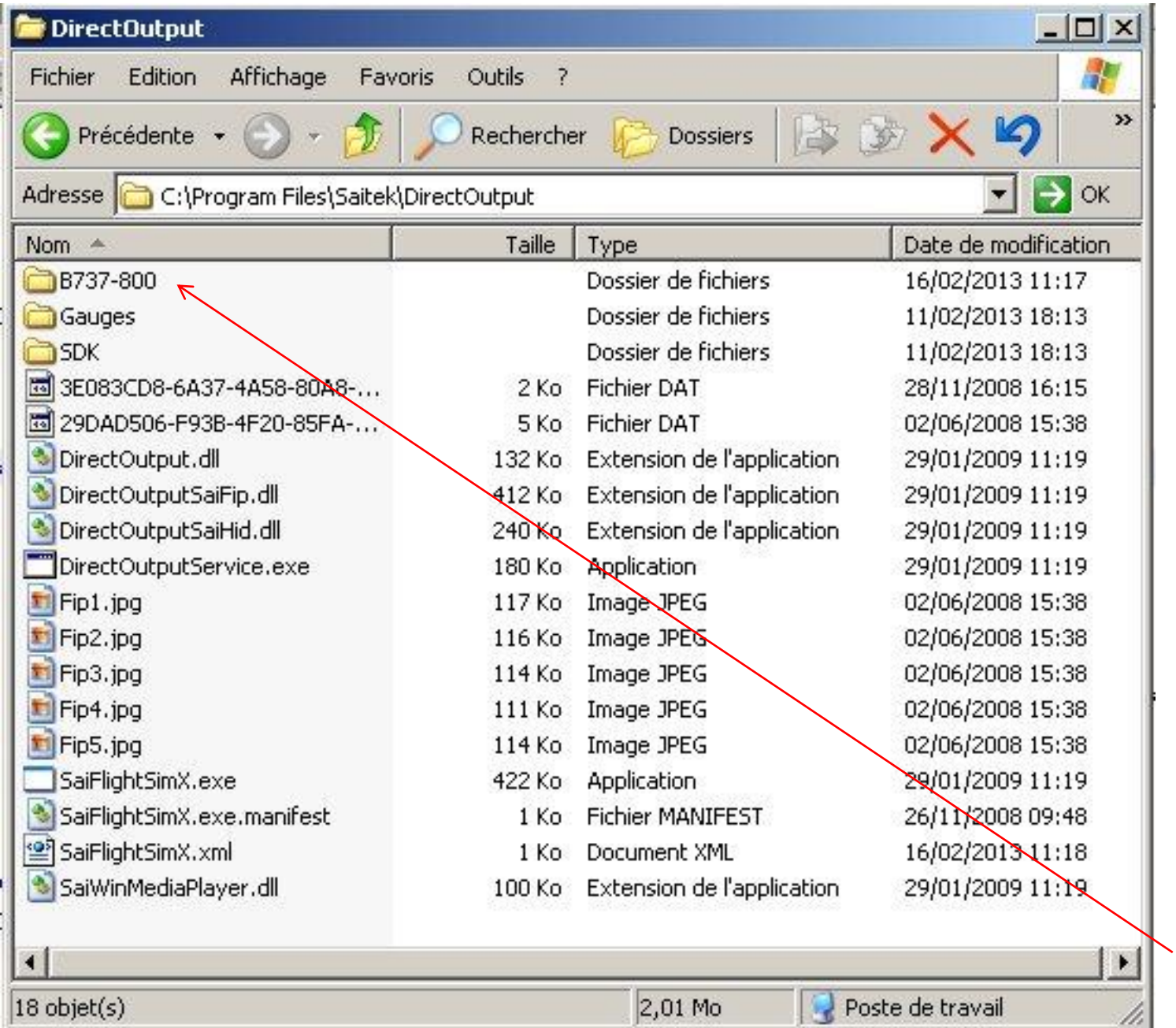
How to program.

Just write the full paragraph at the end of your xml file, all fields are mandatory. Then change the command Click Event= at will from the Key events.txt file or from fs2x.com click events file. As a reminder the first group concern the right knob and the second the left knob.
In the file above if I wish to move heading index for the autopilot I should type:

```
<Area>
<Cursor Type=”DownArrow”/>
<Click Event=”HEADING_BUD_DEC” Repeat=”Yes”/>
</Area>
<Area>
<Cursor Type=”UpArrow”/>
<Click Event=”HEADING_BUD_INC” Repeat=”Yes”/>
</Area>
```

How to use multiple folders for FIP 1/5

By Philippe Verhaege – philvge@gmail.com

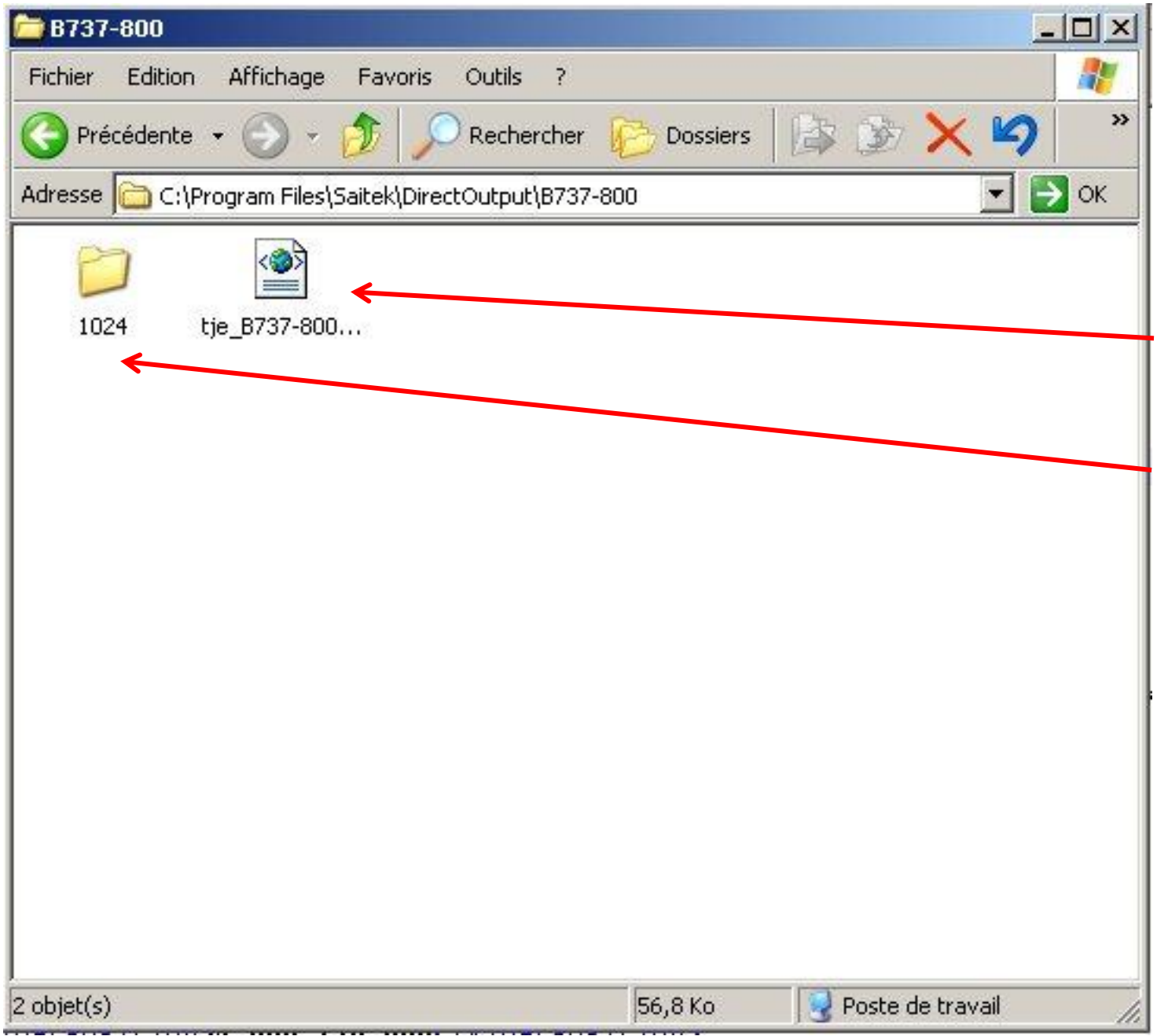


Usually people put all their FIP files in the Gauges folder that exists already in the Saitek\DirectOutput folder.

The goal is to have a cleaner way to organize the FIP folders to be able to manage them more accurately. If you use the Gauges folder then you will have all the files of your FIP in the same folders which is a little bit difficult to identify what document belongs to what FIP.

First you have to create a folder in the DirectOutpout folder in the Saitek folder. Name it for a better identification. In this example the folder is named B737-800.

How to use multiple folders for FIP 2/5

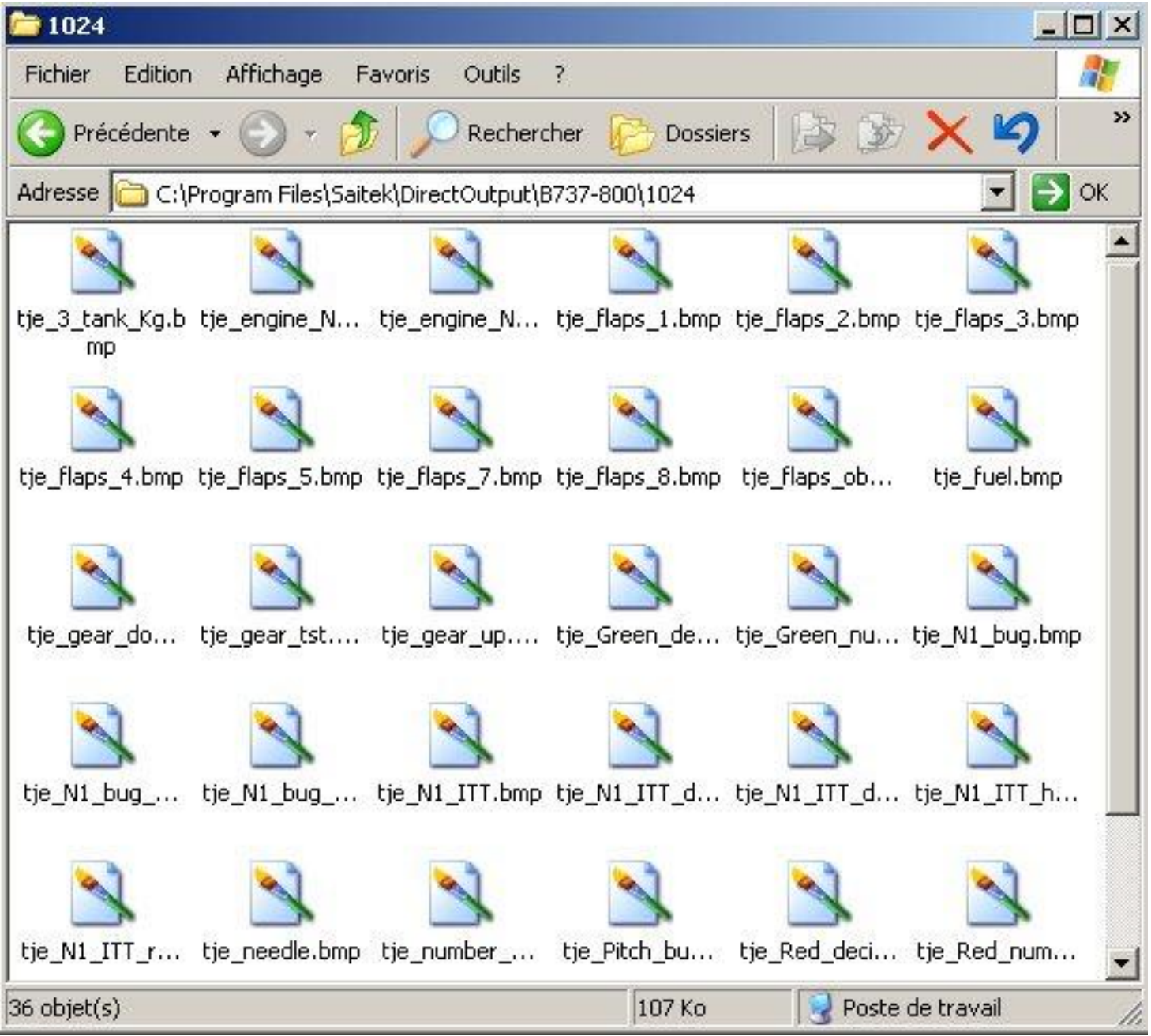


Copy and paste the xml file in the root folder B737-800.

In the B737-800 folder you must create another folder called 1024.

Open the 1024 folder.

How to use multiple folders for FIP 3/5



In the B737-800\1024 folder paste all the bmp files that belong to your FIP.

How to use multiple folders for FIP 4/5

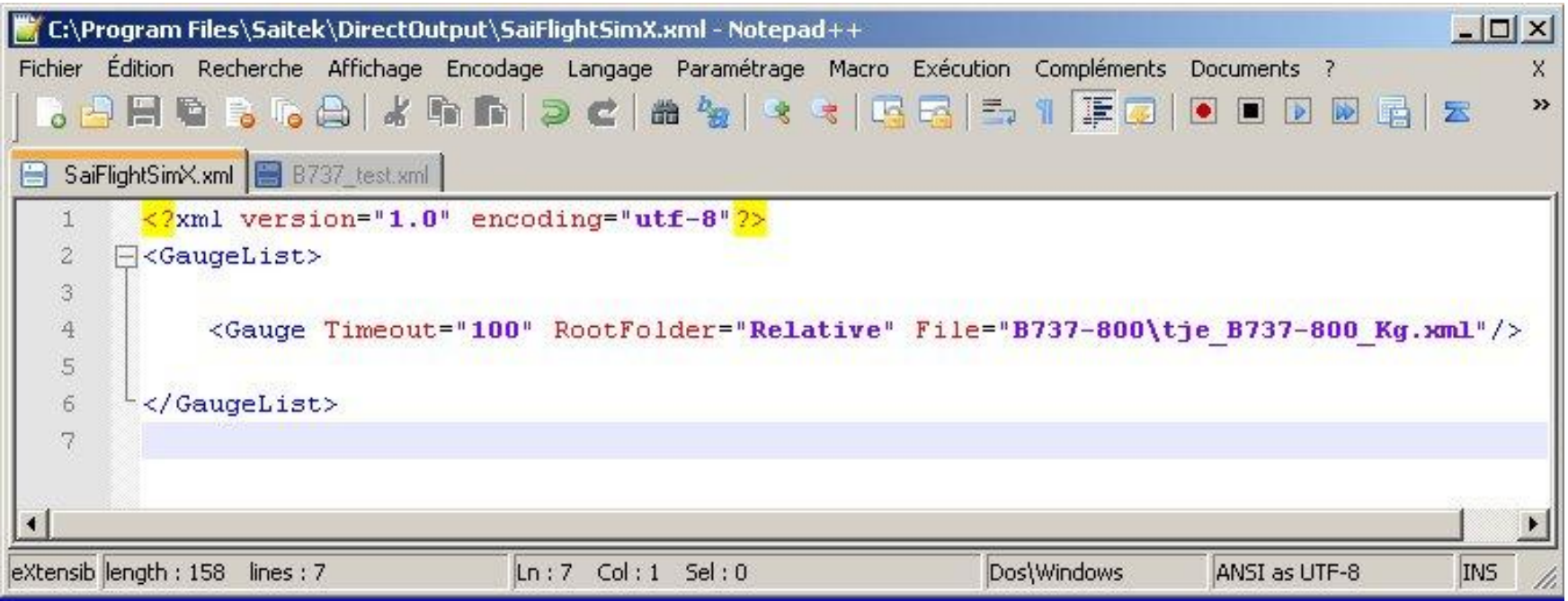
Open the SaiFlightSimX.xml file In the Saitek\DirectOutput folder and type the name of your FIP folder after File=". For this example we type B737-800. This will indicate the path where you copied the xml file and the bmp files.

After File="B737-800" type \ and then the name of your xml file, here File="B737-800\tje_B737-800_Kg_xml"

If you do not use the buttons on the left of the FIP gauge just type:
<Gauge Timeout="1003 RootFolder"Relative" File="B737-800\tje_B737-800_Kg_xml"/>

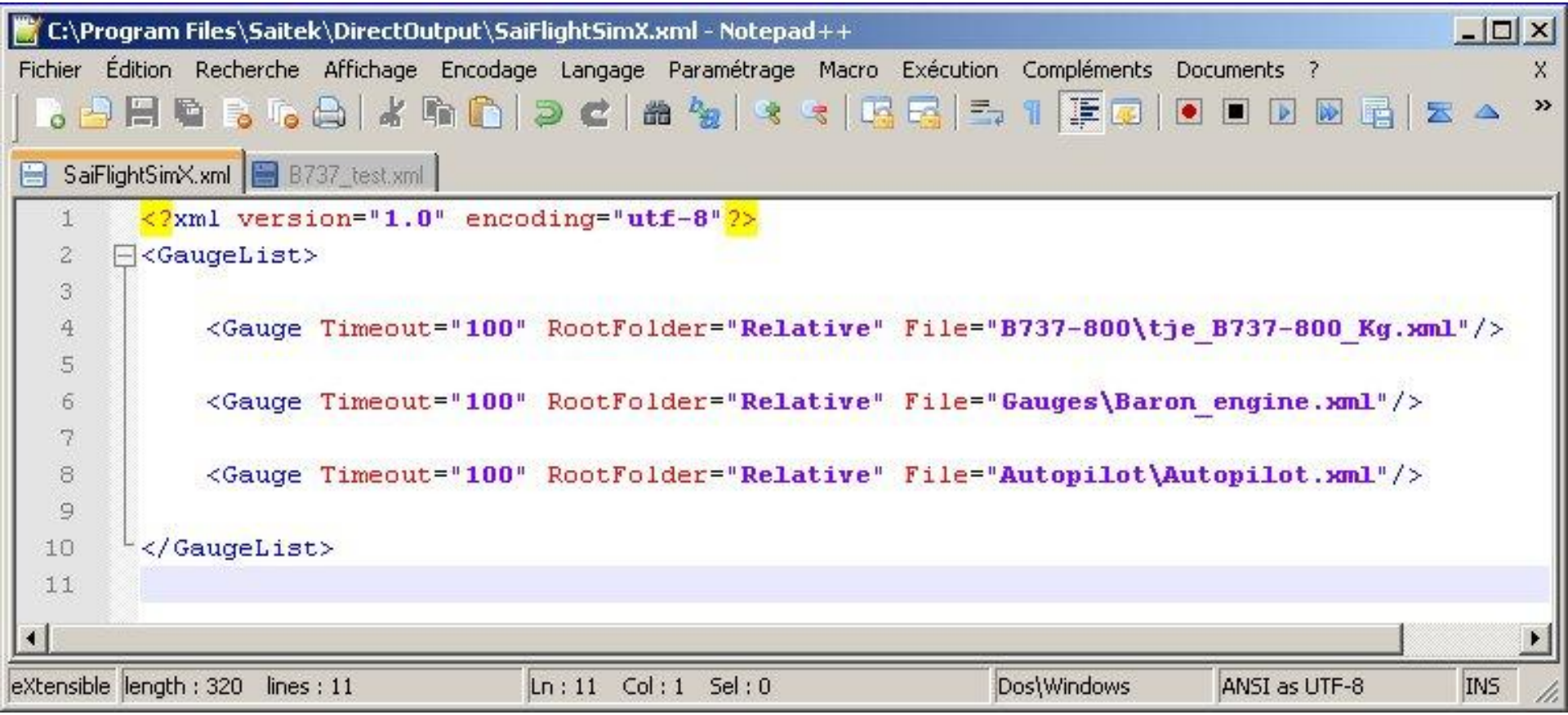
If you use the buttons then the paragraph could be as follow:

```
<Gauge Timeout="100" RootFolder="Relative" File="B737-800\tje_B737-800_Kg_xml">
   <Button Id="1" Name="Map" Event="FLIGHT_MAP"/>
   <Button Id="2" Name="Main Panel" Event="PANEL_1"/>
   <Button Id="3" Name="Radios" Event="PANEL_2"/>
   <Button Id="4" Name="GPS" Event="PANEL_3"/>
   <Button Id="5" Name="Panel 4" Event="PANEL_4"/>
   <Button Id="6" Name="Panel 5" Event="PANEL_5"/>
 </Gauge>
```



How to use multiple folders for FIP 5/5

You can use as many folders as you want. I recommend to use a different name for each FIP, then it will give you the possibility to update each folder at will without having the risk to delete the wrong file or to modify a file that belongs to multiple FIP.



FIP problem

Some FIP users reported that when they installed my gauges there was a white shade following the needle.

I have observed this too and solved the issue by connecting the FIP to a powered USB Hub.
Some computers mainly laptop do not have the right power for their USB plug, therefore using a POWERED usb Hub solves the issue.

Beside this, if you have more than one FIP connected, the first USB powered by the computer will show the first gauge from your SaiFlightSimX.xml file, the next powered USB plug will show the second gauge and so on.
You can change theFIP using the lower button but if you want to have all the time in the same order.

The best solution is to by a powered USB hub with On/Off switches. Then when you switch ON a USB plug it will show the first FIP, when you switch On another USB plug it will show the second FIP from your SaiFlightSimX.xml fil