XML Gauge Programming for FS2004. Chapter 2. Interaction Sections

Version 1.0

By Nick Pike June, 2005

INDEX

1.0 <Mouse>……. Page 2
1.1 Simple Example……..... Page 2
1.2 Specified Area(s)…….. Page 3
1.3 Areas can be 'Stacked'…….. Page 4
1.4 Tooltip …….. Page 5
1.5 Repeating Clicks …….. Page 6
1.6 Simplified Click Code …….. Page 6
1.7 Left and Right Clicks ……... Page 6
1.8 Drag …….. Page 7
1.9 More Mouse Functions ……..… Page 7
2.0 HELPID……. Page 8
2.1 Deactivate Click Areas…….. Page 9
2.2 Final discussion ……Page 10

XML is a text based programming language. Therefore, the code can be written in a standard text editor. There are applications available specifically written for XML, but I have always used an enhanced shareware version of Notepad. If code is saved with a txt extension, rename with an xml extension. XML gauges usually consist of the xml file and bmp (bitmap) files, although gauges without bitmaps are quite common.

This tutorial provides a general introduction to gauge interaction sections and their functions. If you read this tutorial second, it will give some good foundation information Colours have been used to group relative information, or to allow the reader to easily find references in code or text.

1.0 <Mouse>

There are essentially two ways to interact and communicate with a gauge. Mouse clicks and variable links. This tutorial will deal with mouse click communication. Variable links are a special type of variable that cross communicate between any numbers of gauges and will be dealt with in a further tutorial. Essentially, there will be areas on the gauge where the mouse cursor becomes active, and a click will perform a function The mouse section usually resides at the end of the gauge coding, although it can be positioned anywhere in the gauge code. However, the gauge is less 'messy' if it resides at the end.
As stated in Chapter 1, XML gauges follow a common theme of being broken down into sections, each with an opening and closing instruction. The mouse section follows the same rules. The mouse section starts with <Mouse> and must end with </Mouse>. This tells FS2004 that there are areas on the gauge where the mouse cursor becomes active and will accept communication using the mouse. Let's look at some typical code.

1.1 Simple Example

We will start with a simple example and is a complete gauge. If you click on one of those little square icons that toggle a pop-up window display, the sort of thing you see if you press keys, say, shift+2, this is the sort of code that is working.

```
<Gauge Name="ExampleIcon" Version="1.0">
<Image Name="example.bmp"/>
<Update Frequency="2"/>

   <Mouse>
     <Tooltip>Pop-Up On/Off</Tooltip>
     <Cursor Type="Hand"/>
     <Click>10050 (&gt;K:PANEL_ID_TOGGLE)</Click>
   </Mouse>

</Gauge>
```

The <Mouse> </Mouse> section tells FS2004 that mouse communication can be achieved with the code in this section. The whole area of this gauge is active. You'll see why a bit later. The Tooltip line produces one of those little yellow rectangles and the wording Pop-Up On/Off will appear. The Cursor Type defines, in this case, that a little hand will appear. The <Click></Click> line defines the action taken when the gauge is clicked. Do not be concerned too much about the actual code shown in the click line at this stage. What I am attempting to do in the first tutorial and this one is to explain the various sections in an XML gauge and their layouts. Actual coding to achieve something will be given in a later tutorial.

1.2 Specified Area(s)

You may want to specify a particular single area or multiple areas that are clickable on a gauge. The next example shows how clickable areas can be defined.

The following code sets-up a single active click area.

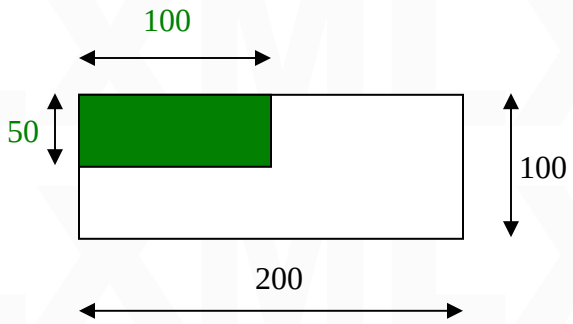
```
<Mouse>
    <Area Left="0" Right="100" Top="0" Bottom="50">
        <Tooltip ID="TOOLTIPTEXT_RADIO_ALTIMETER_FEET"/>
        <Cursor Type="Hand"/>
        <Click>code to go here to produce an event</Click>
     </Area>
</Mouse>
```

The active area is

```
<Area Left="0" Right="100" Top="0" Bottom="50">
</Area>
```

Note how it starts with <Area…… and ends with </Area> (usual thing using '/')

The first line gives the area co-ordinates. On a gauge of size 200 x 100 the active area would look like this. The active mouse click area is coloured green.



There is another way of achieving this, and a method I prefer.

```
<Area Left="0" Top="0" Width="100" Height="50">
</Area>
```

This locates the top left corner with the Left and Top values. The area size is then given by the Width and Height. I find this easier to use mentally.

This can be extended for two or more areas.

```
<Mouse>
    <Area Left="0" Top="0" Width="100" Height="50">
        <Tooltip ID="TOOLTIPTEXT_RADIO_ALTIMETER_FEET"/>
        <Cursor Type="Hand"/>
        <Click> code to go here to produce an event </Click>
      </Area>

    <Area Left="100" Top="0" Width="100" Height="50">
        <Tooltip ID="TOOLTIPTEXT_RADIO_ALTIMETER_FEET"/>
        <Cursor Type="Hand"/>
        <Click> code to go here to produce an event </Click>
      </Area>
</Mouse>
```

This now gives two areas side by side. Note: Never overlap areas.
There are other ways of doing this, but for the purpose of learning XML, stick with this method for the time being. Other ways will be shown in example gauges in later tutorials.

1.3 Areas can be ‘Stacked’

The example above can be ‘simplified’. If there are two adjacent areas, the code can be as follows. The MS default gauges tend to use a lot of these.

```
<Area Left="0" Top="0" Width="200" Height="50">
  <Tooltip>Selector</Tooltip>

      <Area Right="100">
      <Click> code to go here to produce an event </Click>
      <Cursor Type="DownArrow"/>
    </Area>

      <Area Left="100">
      <Click> code to go here to produce a different event </Click>
      <Cursor Type="UpArrow"/>
    </Area>

</Area>
```

Note how two areas are sandwiched in-between a third Area instruction. This time the width value is for the complete width of the two areas. This layout means that each sub-area can be specified by just giving the width of each sub-area. However, the <Area Right="100"> code does not mean the area on the Right. It means the area that ends at a pixel width of 100, which is the area on the left (groan!), and visa-versa for the other area. I’m always getting this wrong, but a test of the gauge will tell you if it is correct or

not. If incorrect, just change the coding around. When dealing with XML, always have the brain primed for backwards thinking.

The tooltip in this case, is in the 'sandwiching' area, meaning it will show in both sub-areas. A separate tooltip could be placed inside each sub-area if they are required to be different.

1.4 Tooltip

A tooltip is that little yellow box that shows when you hover the mouse cursor over an active area and gives some type of information. It can be text, values or a combination of these.

```
<Area Left="100" Top="0" Width="100" Height="50">
    <Tooltip ID="TOOLTIPTEXT_RADIO_ALTIMETER_FEET"/>
    <Cursor Type="Hand"/>
    <Click> code to go here to produce an event </Click>
  </Area>
```

These usually follow these formats,

a) `<Tooltip ID="TOOLTIPTEXT_RADIO_ALTIMETER_FEET"/>`
b) `<Tooltip>Radio Height (%((A:Radio height, feet))%!3d!)</Tooltip>`

Line a) is the way to use pre-defined tooltips. The pre-defined tooltips automatically label the parameter, supply values and units, etc. It's a quick and dirty way of getting the tooltip information. There are a large number of these. Read the HelpIDs.doc in the Panels and Gauges SDK. It shows the text and values that Tooltips produce. Get it at, http://www.microsoft.com/games/flightsimulator/fs2004_downloads_sdk.asp#panels
Being pre-defined doesn't allow for customising tooltips. I have found through experience that most of my tooltips are customised.
Line b) allows for customisation. The code used will give the same information as line a). Why bother customising if it gives the same information? Well, this is just an example of the two ways of achieving this. Line b) uses calculation code, knowledge of which is obviously required, but being a complex part of XML, I'm saving this for a later tutorial.
Don't neglect Tooltips. Most people don't bother to read that 120 page manual that came with a complex panel. It's frustrating to be confronted with a sub-panel containing ten switches and the text on the bitmap is too small to read (another reason why I make my gauges pop-up). A tooltip for each switch solves the problem. The user can always switch tooltips off in FS2004 if they are not required.
BTW, in a very simple case, you could just have the tooltip line in the mouse section to just provide information.

1.5 Repeating Clicks

The line

<Click> code to go here to produce an event </Click>

executes an event per mouse click. This would be used to, say, activate a switch once. However, if you want to change altitude in the auto pilot, you don't want to wear out your mouse micro switch by clicking 20 or 30 times. A repeat function is available, so continuous events are achieved for as long as the mouse button is held down. The code for this is:

<Click Repeat="Yes">will repeat the code that goes here to produce an event </Click>

1.6 Simplified Click Code

The following two lines achieve the same event.

Line a) <Click>(> key code)</Click>
Line b) <Click Event="key code"/>

In Fs2004, there are 'key' codes available. I don't want to get bogged down with these at this stage, but it helps to explain the above two lines. If you look at my list of key (input) codes on the web site, you can see a whole host of code lines. These will cause an event with a mouse click. BTW, they can also be activated and cause an event with key presses, hence the term 'key' code. In Line a), I have used the code convention that I started this tutorial with. This involves modifying the key code from the list to (> key code). However, to use these codes in their simplest form, you can use the code in Line b). I only use this second type in simple cases, but if you want the click to do more complex things, then Line a) is a lot more flexible.
Incidentally, to use the repeat function with this simplified type, the code becomes,

<Click Event="key code" Repeat="Yes"/>

1.7 Left and Right Clicks

The click code shown so far will execute an event with left mouse clicks by default. However, the mouse button (left or right) to be active can be specified. For example, the altitude in the AP could be increased with the left button and decreased with the right. This is often achieved by sliding the cursor to one end of the digits where the cursor shows a negative sign and then the value can be reduced. It's more convenient to just aim for an area and make use of both mouse buttons (IMHO). This is achieved as follows.

<Click Kind="LeftSingle+RightSingle" Repeat="Yes">(M:Event) 'LeftSingle' code to go here to produce an event (M:Event) 'RightSingle' code to go here to produce a different event </Click>
</Area>

Let's break this down.
Click Kind specifies that a type or kind of click is expected.
The "LeftSingle+RightSingle" specifies that the left or right mouse button will be used to cause an event.
(M:Event) specifies that a mouse event is coming, and in the first case it is 'LeftSingle'. This means that a single left click will produce an event according to the code that follows 'LeftSingle'. The same reasoning applies to the right click.

1.8 Drag.

The drag function is often used, for example, to drag the engine throttles. You would left click on a throttle, and then move the mouse with the mouse button depressed.

<Click Kind="LeftSingle+LeftDrag"> (M:Y) (*Get Y*) code to go here to produce an event </Click>

The code <Click Kind="LeftSingle+LeftDrag"> specifies that the left mouse button will be used to produce the event.
(M:Y) defines that the relevant mouse drag movement is in the Y direction. If this was (M:X), the drag movement would be in the X direction.
Note: Wherever the word Left is used, this can be replaced by Right for the right mouse button.
BTW, notice the text (*Get Y*) which is an authors note and not part of the operational code. These can be omitted but you might find them useful. The default MS gauges tend to use these.

1.9 More Mouse Functions

<Click Kind="LeftSingle+Leave">(M:Event) 'LeftSingle' etc.

These can be seen in the default GPS (now that will make for an interesting tutorial and will be available in ten, leather bound volumes). The event is continued for as long as the mouse button is held down. For example, this may operate an animated push button.

<Click Kind="LeftSingle+MoveLeft">(M:Event) 'LeftSingle' etc.

With this, you can click and then drag around in any direction. For example, I produced a gauge that replaces the joystick. By holding down the left button, I could move the cursor

around on a square gauge which operated the rudder and ailerons according to the direction of cursor travel.

LeftSingle+LeftDrag+Wheel

This is similar to the throttle control but it also allows the mouse wheel to move the throttles as well.

2.0 HELPID

If you look at the list of Tooltips in the SDK mentioned in section 1.4, you'll also see that HELPIDs are shown in the list. These are similar to the pre-defined Tooltips and produce their own little yellow rectangles (HelpID tooltips) that give general information. The main difference is that HELPIDs produce tooltips that are text only labels, whereas Tooltips usually have label text and values. Read the HelpIDs.doc. It shows the text that HELPIDs will produce.

Example,

```
  <Mouse>
    <Help ID="HELPID_GAUGE_HSI"/>
    <Tooltip ID="TOOLTIPTEXT_HSI_HEADING_COURSE"/>

    <Area Left="10" Right="70" Top="307" Bottom="365">
      <Help ID="HELPID_GAUGE_HEADING_BUG_ADJUST"/>

      <Area Right="30">
        <Cursor Type="DownArrow"/>
        <Click Event="HEADING_BUG_DEC" Repeat="Yes"/>
      </Area>

      <Area Left="30">
        <Cursor Type="UpArrow"/>
        <Click Event="HEADING_BUG_INC" Repeat="Yes"/>
      </Area>

    </Area>
</Mouse>
```

The first HELPID and Tooltip are after the Mouse instruction, but before the first defined Area. These will produce tooltips all over the gauge. This code layout is the same as an MS default gauge. However, the first HELPID and Tooltip tend to 'swamp' the second HELPID that is in a defined area (in my experience).

I have shown the use of HELPID because you will see them used in gauge code, but I prefer to use Pre-defined or customised Tooltips in defined areas. That way you know what tooltip is seen and where.
Note: you cannot use more than one Tooltip per defined area. However, as shown, you can have a HELPID and a Tooltip in the same area. A number of Tooltips can be used, so long as the code is placed in different defined areas.

2.1 Deactivate Click Areas

You may have a clickable area that you do not want to be active until another function is operating. For example, my engine start switches cannot be operated until the fuel valves are switched on. This can be achieved using a Visible instruction (covered in Chapter 1).

In the following code, I have added a <Visible></Visible> instruction within the <Area….</Area> section. The Visible instruction is used the same as in the main body of the gauge, to turn a section on or off.

```
<!—starter control 1-->
    <Area Left="371" Top="33" Width="40" Height="40">
  <Visible>(A:General eng1 mixture lever position, percent) 50 &gt;</Visible>
    <Tooltip>Engine Starter 1</Tooltip>
       <Cursor Type="Hand"/>
    <Click>5 (starter code)</Click>
    </Area>
```

This has basically turned off the ability to click on the starter switches until the fuel valve is on, shown by (A:General eng2 mixture lever position, percent) being greater than 50%, which means it is on.
I have taken the opportunity to show that you can label a mouse section. In this case, <!—starter control 2--> as a reminder of what the section achieves.

2.2 Final discussion

Well, that's all for now. By reading the first tutorial (Chapter 1) and this tutorial (Chapter 2), you should gain an appreciation of how a complete gauge is constructed.
The best way to put the above into action is with some complete examples. See further tutorials where complete gauges are shown and their inner workings explained. There are still a few instruction code types for the interaction sections to learn about, but these are best covered by other tutorials or the examples. I hope I have shown enough code here to help the beginner to understand the more straightforward gauges.
Please bear in mind that in some cases, there are many ways of achieving the same effect. With experience, you begin to build up a portfolio of XML knowledge in your head, and the most optimised method will come to mind. However, I have found it impossible to remember every twist and turn. I often refer to previous gauges I have made to act as a foundation or template of a new gauge or to remind me what to do.
At first, this may all look a bit daunting. I actually became reasonably adept with XML after about 6 months. OK, not a five minute job, but then it takes time to learn anything properly. I also had a very good teacher, a man by the name of Arne Bartels. I have also learned useful snippets by regularly visiting the forums at Avsim.com.