

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.

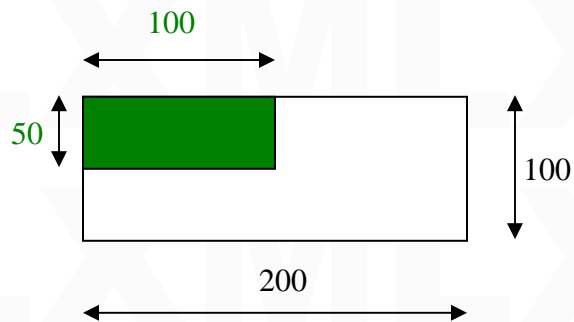
```
<Mouse>  
  <Area Left="0" Right="100" Top="0" Bottom="50">  
    <Tooltip ID="TOOLTIPTEXT_RADIO_ALTIMETER_FEET"/>  
    <Cursor Type="Hand"/>  
    <Click>code to go here to produce an event</Click>  
  </Area>  
</Mouse>
```

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

