TraceFromPixel

TraceFromPixel(*xpixel*, *ypixel*, *optionsString*)

The TraceFromPixel function returns a string based on an attempt to hit test the provided X and Y coordinates. Used to determine if the mouse was clicked on a trace in a graph.

When a trace is found, TraceFromPixel returns a string containing the following `KEY:value;` pairs:

`TRACE:`*`tracename`*

`HITPOINT:`*`pnt`*

tracename will be quoted if necessary and may contain instance notation. *pnt* is the point number index into the trace's wave when the hit was detected. If a trace is not found near the coordinate point, a zero length string is returned.

Parameters

xpixel and *ypixel* are the X and Y pixel coordinates.

optionsString can contain the following:

`WINDOW:`*`winName`*`;`

`PREF:`*`traceName`*`;`

`ONLY:`*`traceName`*`;`

`DELTAX:`*`dx`*`;DELTAY:`*`dy`*`;`

Use the `WINDOW` option to hit test in a graph other than the top graph. Use the `ONLY` option to search only for a special target trace. If the PREF option is used then the search will start with the specified trace but if no hit is detected, it will go on to the others.

When identifying a subwindow with `WINDOW:`*`winName`*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

The DELTAX and DELTAY values must both be specified to alter the region that Igor searches for traces. The *dx* and *dy* values are in pixels and the region searched is the rectangle from *xpixel-dx* to *xpixel+dx* and *ypixel-dy* to *ypixel+dy*.

If DELTAX or DELTAY are omitted, the search region depends on whether PREF or ONLY are specified. If either are specified then Igor first searches for the trace using *dx* = 3 and *dy* = 3. If the trace is not identified, Igor searches again using *dx* = 6 and *dy* = 6. If the trace is still not identified, Igor gives up and returns a zero-length result string.

If neither PREF nor ONLY are specified then Igor uses tries 3, 6, 12, and 24 for *dx* and *dy* until it finds a trace or gives up and returns a zero-length result string.

See Also

The **NumberByKey**, **StringByKey**, **AxisValFromPixel**, and **PixelFromAxisVal** functions.

ModifyGraph (traces) and **Instance Notation** on page IV-20 for discussions of trace names and instance notation.

Trace Names on page II-282, **Programming With Trace Names** on page IV-87.

TraceInfo

TraceInfo(*graphNameStr*, *ywavenameStr*, *instance*)

The TraceInfo function returns a string containing a semicolon-separated list of information about the trace in the named graph window or subwindow.

Parameters

graphNameStr can be `""` to refer to the top graph.

When identifying a subwindow with *graphNameStr*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

yWaveNameStr is either the name of a wave containing data displayed as a trace in the named graph, or a trace name (wave name with "#n" appended to distinguish the nth image of the wave in the graph). You might get a trace name from the **TraceNameList** function.

If *yWaveNameStr* contains a wave name, *instance* identifies which trace of *yWaveNameStr* you want information about. *instance* is usually 0 because there is normally only one instance of a given wave

displayed in a graph. Set *instance* to 1 for information about the second trace of the wave named by *yWaveNameStr*, etc. If *yWaveNameStr* is "", then information is returned on the *instance*th trace in the graph.

If *yWaveNameStr* is a trace name, and *instance* is zero, the instance is extracted from *yWaveNameStr*. If *instance* is greater than zero, the wave name is extracted from *yWaveNameStr*, and information is returned concerning the *instance*th instance of the wave.

Details

The string contains several groups of information. Each group is prefaced by a keyword and colon, and terminated with the semicolon. The keywords are as follows:

Keyword	Information Following Keyword
AXISFLAGS	Flags used to specify the axes. Usually blank because /L and /B (left and bottom axes) are the defaults.
AXISZ	Z value of a contour level trace or NaN if the trace is not a contour trace.
ERRORBARS	The ErrorBars command for the trace, as it would appear in the recreation macro (without the beginning tab character).
RECREATION	List of keyword commands as used by ModifyGraph command. The format of these keyword commands is: *keyword*(x)=*modifyParameters*;
TYPE	The type of trace: 0: XY or waveform trace 1: Contour trace 2: Box plot trace 3: Violin plot trace TYPE was added in Igor Pro 8.00.
XAXIS	X axis name.
XRANGE	Point subrange of the trace's X data wave in "[*startPoint*,þ*endPoint* : *increment*]" format. **Note**: Unlike the actual syntax of a trace subrange specification where increment is preceded by a semicolon character, here it is preceded by a colon character to preserve the notion that semicolon is what separates the keyword-value groups. If the entire X wave is displayed (the usual case), the XRANGE value is "[*]". If an X wave is not used to display the trace, then the XRANGE value is "".
XWAVE	X wave name if any, else blank.
XWAVEDF	Full path to the data folder containing the X wave or blank if no X wave.
YAXIS	Y axis name.
YRANGE	Point subrange of the trace's Y data wave or "[*]".

The format of the RECREATION information is designed so that you can extract a keyword command from the keyword and colon up to the ";", prepend "ModifyGraph ", replace the "x" with the name of a trace ("data#1" for instance) and then **Execute** the resultant string as a command.

Note: The syntax of any subrange specifications in the RECREATION information are modified in the same way as for XRANGE and YRANGE. Currently only the zColor, zmrkSize, and zmrkNum keywords might have a subrange specification.

Examples

This example shows how to extract a string value from the keyword-value list returned by TraceInfo:

```
String yAxisName= StringByKey("YAXIS", TraceInfo("","",0))
```

This example shows how to extract a subrange and put the semicolon back: