

```
NewPanel/EXP=2/N=DemoPanel
GetWindow DemoPanel wsizeForControls
ListBox list0, win=DemoPanel, pos={0,0}, size={V_right, V_Bottom}, listWave=tw
```

See Also

The **SetWindow**, **GetUserData**, **MoveWindow** and **DoWindow** operations.

The **IgorInfo** function.

GetWindowBrowserSelection

GetWindowBrowserSelection (mode)

The GetWindowBrowserSelection function returns a semicolon separated string that represents the selection in the Window Browser.

GetWindowBrowserSelection was added in Igor Pro 9.00.

Parameters

mode specifies which selection should be returned:

mode=0: Returns the window names of the selected windows in the main window list.
Other values for *mode* are reserved for possible future use.

If there is no selection, GetWindowBrowserSelection returns an empty string.

If the Window Browser is not open, GetWindowBrowserSelection returns an empty string and generates an error.

The order in which the selection is returned is undefined. If the order matters to you, you must sort the list yourself.

See Also

The **Window Browser** on page II-49, **SortList**

GizmoInfo

GizmoInfo (nameStr, key)

The GizmoInfo function is used to determine if a particular name is valid and unique as a Gizmo item names.

Documentation for the GizmoInfo function is available in the Igor online help files only. In Igor, execute:

```
DisplayHelpTopic "GizmoInfo"
```

GizmoPlot

GizmoPlot

GizmoPlot is a procedure subtype keyword that identifies a macro as being a Gizmo recreation macro. It is automatically used when Igor creates a window recreation macro for a Gizmo plot. See **Procedure Subtypes** on page IV-204 and **Saving and Recreating Graphs** on page II-350 for details.

GizmoScale

GizmoScale (dataValue, dimNumber [, gizmoNameStr])

The GizmoScale function returns a scaled *dataValue* for the specified dimension. The scaled values are used to position non-data drawing objects in the Gizmo window.

Documentation for the GizmoScale function is available in the Igor online help files only. In Igor, execute:

```
DisplayHelpTopic "GizmoScale"
```

gnoise

gnoise (num [, RNG])

The gnoise function returns a random value drawn from a Gaussian distribution such that the standard deviation of an infinite number of such values would be *num*.

gnoise returns a complex result if *num* is complex or if it is used in a complex expression. See **Use of gnoise With Complex Numbers** on page V-324.

The random number generator is initialized using a seed derived from the system clock when you start Igor. This almost guarantees that you will never get the same sequence twice. If you want repeatable "random" numbers, use **SetRandomSeed**.

The Gaussian distribution is achieved using a Box-Muller transformation of uniform random numbers.

The optional parameter RNG selects one of three pseudo-random number generators used to create the uniformly-distributed random numbers providing the input to the Box-Muller transformation.

If you omit the RNG parameter, gnoise uses RNG number 3, named "Xoshiro256**". This random number generator was added in Igor Pro 9.00 and is recommended for all new code. In earlier versions of Igor, the default was 1 (Linear Congruential Generator).

The available random number generators are:

| RNG | Description |
|-----|--|
| 1 | <p>Linear Congruential Generator by L'Ecuyer with added Bayes-Durham shuffle. The algorithm is described in <i>Numerical Recipes</i> as the function <code>ran2()</code>. This option has nearly 23^2 distinct values and the sequence of random numbers has a period in excess of 10^{18}.</p> <p>This RNG is provided for backward compatibility only. New code should use RNG=3 (Xoshiro256**).</p> |
| 2 | <p>Mersenne Twister by Matsumoto and Nishimura. It is claimed to have better distribution properties and period of $2^{19937}-1$.</p> <p>See Matsumoto, M., and T. Nishimura, Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator, <i>ACM Trans. on Modeling and Computer Simulation</i>, 8, 3-30, 1998.</p> <p>This RNG is provided for backward compatibility only. New code should use RNG=3 (Xoshiro256**).</p> |
| 3 | <p>Xoshiro256** by David Blackman and Sebastiano Vigna. It has a period of $2^{256}-1$, is 4 dimensionally equidistributed and passes all statistical tests at the time of writing.</p> <p>For technical details, see "Scrambled linear pseudorandom number generators", 2019 (http://vigna.di.unimi.it/ftp/papers/ScrambledLinear.pdf).</p> |

Use of gnoise With Complex Numbers

gnoise returns a complex result if *num* is complex or if it is used in a complex expression.

```
Function DemoGNoiseComplex()
    // gnoise(rv) in a complex expression returns a complex result
    // and is equivalent to cmplx(gnoise(rv), gnoise(rv))
    Variable rv = 1 // A real variable
    SetRandomSeed 1
    Variable/C cv1 = gnoise(rv) // Real parameter, complex result
    SetRandomSeed 1
    Variable/C cv2 = cmplx(gnoise(rv), gnoise(rv)) // Equivalent
    Print cv1, cv2

    // gnoise(cv) is equivalent to cmplx(gnoise(real(cv)), gnoise(imag(cv)))
    Variable/C cv = cmplx(1,1) // A complex variable
    SetRandomSeed 1
    Variable/C cv3 = gnoise(cv) // Complex parameter, complex result
    SetRandomSeed 1
    Variable/C cv4 = cmplx(gnoise(real(cv)), gnoise(imag(cv))) // Equivalent
    Print cv3, cv4
End
```

See Also

SetRandomSeed, enoise, Noise Functions, Statistics