

## 12. WINDOWS AND REGIONS

---

### Windows

---

Windows have two basic parts: an area on the screen containing a collection of pixels, and a property list. The window properties determine how the window looks, the menus that can be accessed from it, what should happen when the mouse is inside the window and a mouse button is pressed, and soon.

#### CREATEW

Some of the window's properties can be specified when a window is created with the function `CREATEW`. In particular, it is easy to specify the size and position of the window; its title; and the width of its borders.

`(CREATEW region title borderw'idth)`

Region is a record (named `REGION`, with the fields `left`, `bottom`, `width`, and `height`) or a list. A region describes a rectangular area on the screen, the window's dimensions and position. The fields `left` and `bottom` refer to the position of the bottom left corner of the region on the screen. `Width` and `height` refer to the width and height of the region. The usable space inside the window will be smaller than the width and height, because some of the window's region is consumed by the title bar, and some is taken by the borders.

Title is a string that will be placed in the title bar of the window.

Borderwidth is the width of the border around the exterior of the window, in number of pixels.

For example, typing:

```
(SETQ MY.WINDOW (CREATEW
  (CREATEREGION 100 150 300 200)
  "THIS IS MY OWN WINDOW"))
```

or

```
(SETQ MY.WINDOW (CREATEW
  (CREATEW ' (100 150 300 200)
  "THIS IS MY OWN WINDOW"))
```

produces a window with a default borderwidth. Note that you did not need to specify all the window's properties (see Figure 12-1).

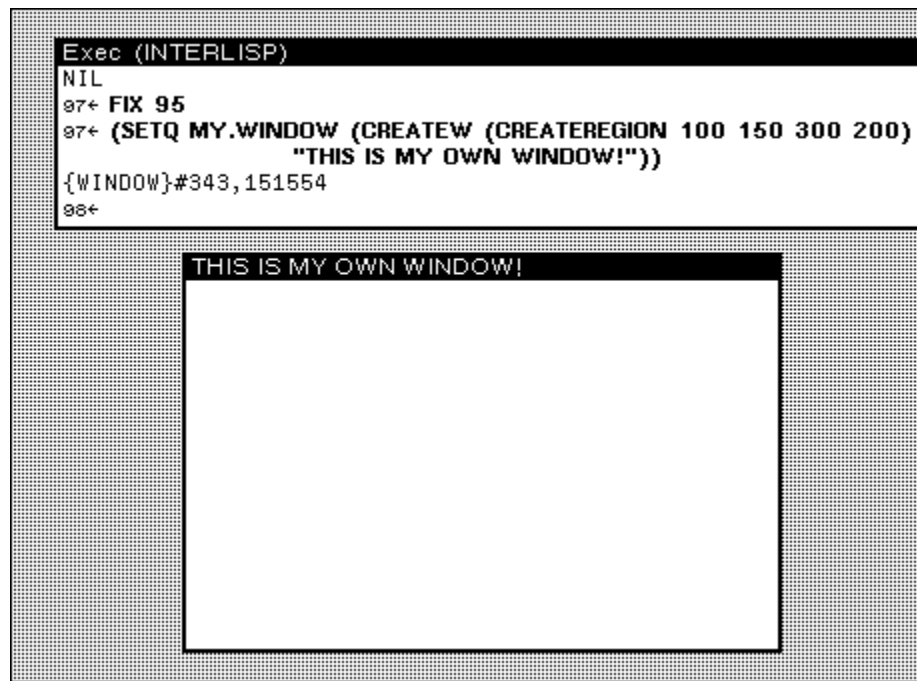


Figure 12-1. Creating a Window

In fact, if `(CREATEW)` is called without specifying a region, you will be prompted to sweep out a region for the window (see Chapter 10)

## WINDOWPROP

The function to access or add to any property of a window's property list is `WINDOWPROP`.

```
(WINDOWPROP window property <value>)
```

When you use `WINDOWPROP` with only two arguments—window and property—it returns the value of the window's property. When you use `WINDOWPROP` with all three arguments—window, property and value—it sets the value the window's property to the value you inserted for the third argument.

For example, consider the window, `MY WINDOW`, created using `(CREATEW)`. `TITLE` and `REGION` are both properties. Type

```
(WINDOWPROP MY.WINDOW 'TITLE)
```

and the value of `MY.WINDOW`'s `TITLE` property is returned, `"THIS IS MY OWN WINDOW"`. To change the title, use the `WINDOWPROP` function, and give it the window, the property title, and the new title of the window.

```
(WINDOWPROP MY.WINDOW 'TITLE "MY FIRST WINDOW")
```

automatically changes the title and automatically updates the window. Now the window looks like Figure 12-2.

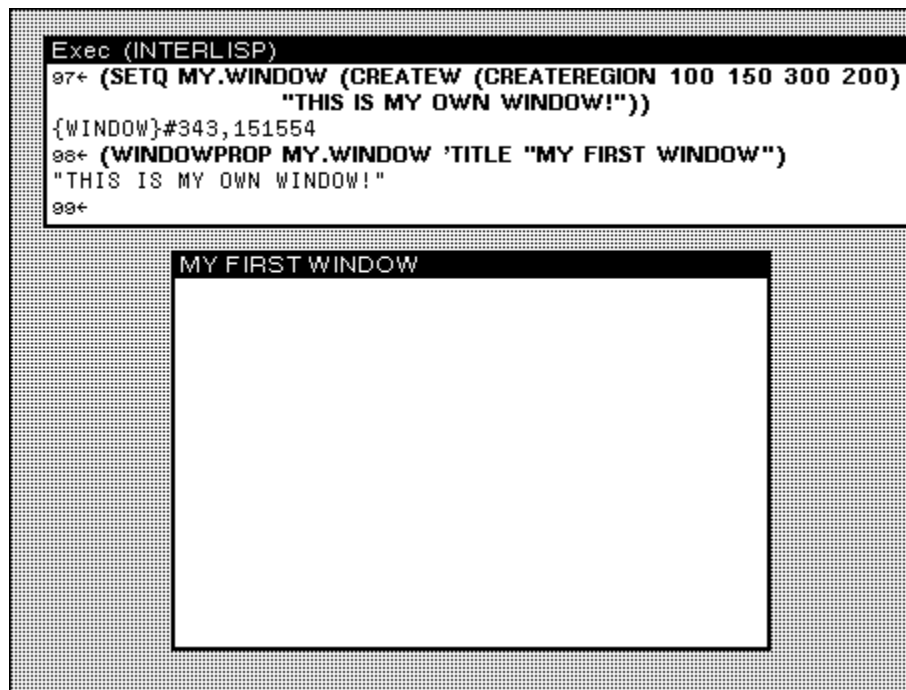


Figure 12-2. TITLE is a Window Property

Altering the region of the window, `MY.WINDOW`, is also be done with `WINDOWPROP`, in the same way you changed the title. (Changing either of the first two numbers of a region changes the position of the window on the screen. Changing either of the last two numbers changes the dimensions of the window itself.)

## Getting Windows to Do Things

Four basic window properties will be discussed here: `CURSOR1NFN`, `CURSOROUTFN`, `CURSORMOVEDFN`, and `BUTTONEVENTFN`.

A function can be stored as the value of the `CURSOR1NFN` property of a window. It is called when the mouse cursor is moved into that window.

Look at the following example:

1. First, create a window called `MY.WINDOW`. Type:

```
(SETQ MY.WINDOW
  (CREATEW
    (CREATEREGION 200 200 200 200)
    "THIS WINDOW WILL SCREAM!"))
```

This creates a window.

2. Now define the function `SCREAMER`. It will be stored on the property `CURSOR1NFN`. (Notice that this function has one argument, `WINDOWNAME`. All functions called from the property `CURSOR1NFN` are passed the window it was called from. So the value of `MY.WINDOW` is bound to `WINDOWNAME`. When it is called, `SCREAMER` simply rings bells.

```
(DEFINEQ (SCREAMER (WINDOWNAME)
  (RINGBELLS)
  (PROMPTPRINT "YAY - IT WORKS!")
  (RINGBELLS)))
```

- Now, alter that window's `CURSORINFN` property, so that the system calls the function `SCREAMER` at the appropriate time. Type:

```
(WINDOWPROP MY.WINDOW 'CURSORINFN
  (FUNCTION SCREAMER))
```

- After this, when you move the mouse cursor into `MY.WINDOW`, the `CURSORINFN` property's function is called, and it rings bells twice.

`CURSORINFN` is one of the many window properties that come with each window - just as `REGION` and `TITLE` did. Other properties include:

<code>CURSOROUTFN</code>	The function that is the value of this property is executed when the cursor is moved out of a window.
<code>CURSORMOVEDFN</code>	The function that is the value of this property is executed when the cursor is moved while it is inside the window.
<code>BUTTONEVENTFN</code>	The function that is the value of this property is executed when either the left or middle mouse buttons are pressed (or released).

Figure 12-3 shows `MY.WINDOW`'s properties. Notice that the `CURSORINFN` has the function `SCREAMER` stored in it. The properties were shown in this window using the function `INSPECT`. `INSPECT` is covered in Chapter 17.

{WINDOW} # 343,151554 Inspector	
<code>DSP</code>	<code>#&lt;Output Display Stream/354,76000&gt;</code>
<code>NEXTW</code>	<code>{WINDOW}#343,151064</code>
<code>SAVE</code>	<code>{BITMAP}#377,145344</code>
<code>REG</code>	<code>(100 150 300 200)</code>
<code>BUTTONEVENTFN</code>	<code>TOTOPW</code>
<code>RIGHTBUTTONFN</code>	<code>NIL</code>
<code>CURSORINFN</code>	<code>NIL</code>
<code>CURSOROUTFN</code>	<code>NIL</code>
<code>CURSORMOVEDFN</code>	<code>NIL</code>
<code>REPAINTFN</code>	<code>NIL</code>
<code>RESHAPEFN</code>	<code>NIL</code>
<code>EXTENT</code>	<code>NIL</code>
<code>USERDATA</code>	<code>NIL</code>
<code>VERTSCROLLREG</code>	<code>NIL</code>
<code>HORIZSCROLLREG</code>	<code>NIL</code>
<code>SCROLLFN</code>	<code>NIL</code>
<code>VERTSCROLLWINDOW</code>	<code>NIL</code>
<code>HORIZSCROLLWINDOW</code>	<code>NIL</code>
<code>CLOSEFN</code>	<code>NIL</code>
<code>MOVEFN</code>	<code>NIL</code>
<code>WTITLE</code>	<code>"MY FIRST WINDOW"</code>
<code>NEWREGIONFN</code>	<code>NIL</code>
<code>WBORDER</code>	<code>4</code>
<code>PROCESS</code>	<code>NIL</code>
<code>WINDOWENTRYFN</code>	<code>GIVE.TTY.PROCESS</code>
<code>SCREEN</code>	<code>{SCREEN}#65,147740</code>

Figure 12-3. Inspecting `MY.WINDOW` for Mouse-Related Window Properties

You can define functions for the values of the properties `CURSORMOVEDFN` and `CURSORMOVEDFN` in much the same way as you did for `CURSORMOVEDFN`. The function that is the value of the property `BUTTONEVENTFN`, however, can be specialized to respond in different ways, depending on which mouse button is pressed. This is explained in the next section.

## BUTTONEVENTFN

`BUTTONEVENTFN` is another property of a window. The function that is stored as the value of this property is called when the mouse is inside the window, and a mouse button is pressed. As an example of how to use it, type:

```
(WINDOWPROP MY.WINDOW 'BUTTONEVENTFN
  (FUNCTION SCREAMER))
```

When the mouse cursor is moved into the window, bells will ring because of the `CURSORMOVEDFN`, but it will also ring bells when either the left or middle mouse button is pressed. Notice that the right mouse button functions as it usually does, with the window manipulation menu. If only the left button should evoke the function `SCREAMER`, then the function can be written to do just this, using the function `MOUSESTATE`, and a form that only `MOUSESTATE` understands, `ONLY`. For example:

```
(DEFINEQ
  (SCREAMER2 (WINDOWNAME)
    (if (MOUSESTATE (ONLY LEFT))
      then (RINGBELLS))))
```

In addition to `(ONLY LEFT)`, `MOUSESTATE` can also be passed `(ONLY MIDDLE)`, `(ONLY RIGHT)` or combinations of these (e.g. `(OR (ONLY LEFT) (ONLY MIDDLE))`). You do not need to use `ONLY` with `MOUSESTATE` for every application. `ONLY` means that that button is pressed and no other.

If you do write a function using `(ONLY RIGHT)`, be sure that your function also checks position of the mouse cursor. Even if you want your function to be executed when the mouse cursor is inside the window and the right button is pressed, there is a convention that the function `DOWINDOWCOM` should be executed when the mouse cursor is in the title bar or the border of the window and the right mouse button is pressed. Please program your windows using this tradition! For more information, please see Chapter 28 in the *IRM*.

## Looking at a Window's Properties

`INSPECT` is a function that displays a list of the properties of a window, and their values. Figure 12.3 shows the `INSPECT` function run with `MY.WINDOW`. Note the properties introduced in `CREATEW`: `WBORDER` is the window's border, `REG` is the region, and `WTITLE` is the window's title.

## Regions

A region is a record, with the fields `LEFT`, `BOTTOM`, `WIDTH`, and `HEIGHT`. `LEFT` and `BOTTOM` refer to where the bottom left hand corner of the region is positioned on the screen. `WIDTH` and `HEIGHT` refer to the width and height of the region.

`CREATEREGION` creates an instance of a record of type `REGION`. Type:

```
(SETQ MY.REGION (CREATEREGION 15 100 200 450))
```

to create a record of type `REGION` that denotes a rectangle 200 pixels high, and 450 pixels wide, whose bottom left corner is at position (15, 100). This record instance can be passed to any function that requires a region as an argument, such as `CREATEW`, above.