AIREGIONS

(Active Irregular Regions)

By: Greg Wexler (Wexler.pasa@Xerox)
and
By: Jim Wogulis (Wogulis@ICS.UCI.EDU)
New Owner: James Turner (Turner.Lexington@Xerox.com)
Uses: FILLREGION, AIREGIONS-DEMO

INTRODUCTION

The purpose of this package is to provide menu-like operations on irregularly shaped regions within a window and make available general functions that allow users to create their own applications using irregularly shaped active regions. An added feature of AIRegions is that multiple IREGIONs may be activated by selecting the intersecting area of those IREGIONs. (Throughout this document an irregularly shaped region will be referred to as an IREGION).

DESCRIPTION

Virtually all of the features of menu selection have been implemented in this package: ease of menu creation, item-selected shading, quick response to selection, and execution of an associated function. Yet, this package adds one additional feature without any degradation to the quality and efficiency of menu implementation: the selection of any irregularly shaped region from any point within that region, and without any unsightly cosmetic change.

In describing the package by means of an example, picture a map of the world, or better yet, of a particular country broken up into its individual states and/or provinces. Suffice it to say that these regions are not square but irregular in shape and that they are bordered by solid lines, as they are on a common map. Unlike the menu package or ACTIVEREGIONS package, AIRegions allows you to select any of these pre-set states/provinces just as if your are making a menu selection of an item. One of the nice aspects of this package lies in the fact that the package does NOT make any costmetic changes to the irregularly shaped region, like providing some small box within the region to button in. Simply button your mouse within the solidly bordered region, anywhere in the region, and it will shade it to your particular shade and execute your defined function.

Functionality provided:

The functions in this package allow the user to work with familiar concepts: creating and implementing windows and menus. The examples provided within this documentation should be sufficient for the user to begin setting up irregularly shaped regions.

(CREATEIR *window shade buttoneventfn helpstring region poslist*) [Function]

window: the window which will contain the irregular region.

shade: can be either a number between 0 and 65535 for a 4 by 4 shading or a 16 by 16 bitmap (if *shade* is NIL then the default is black, 65535).

buttoneventfn: the function called when the region is selected. The arguments that are passed to the function are: the window containing the IREGION, the IREGION record itself, and the button which selected the IREGION.

helpstring: the string that is placed in the PROMPTWINDOW when the mouse is held over the item for a few seconds.

region: if specified, will be the region relative to *window* in which the IREGION can be found. (If *region* is NIL, the user will be prompted to sweep out a region within *window*.)

poslist: If specified, will be either a position or list of positions relative to *window* that are the starting points for the FILLREGION routine (i.e. a point within the desired IREGION). (if *poslist* is NIL, the user will be prompted for a position until he/she selects outside of *region*.)

Description of use: This is the first function that is called when actually setting up an irregularly shaped region to become sensitive to button activity. If the *region* argument is not set, then the cursor changes its shape and prompts for a region to completely surround the IREGION within the desired active window.(Note: That it is best to surround the desired IREGION as close as possible since this will save on execution time and memory useage.) A thin box will appear temporarily where the IREGION was scanned. If *poslist* is NIL, then the cursor changes into a TARGET symbol. The user should left-button mouse within desired active IREGION. Note: the IREGION must be surrounded by a border that FILLREGION can use to define the active area. Any gaps in the IREGION will cause the next routine to fill the region and anything outside with the shade provided. Mistakes can be corrected by using the REMOVE.IREGION function described below and PAINTing in the gap to retry. After left-buttoning within the desired active IREGION, the cursor continues to remain in its TARGET state. If the IREGION is split up into many different parts, those parts may be selected with the left-button also making them all active concurrently. However, when one is finished activating that one IREGION, then she/he should left-button outside of *region*. This function must be called for each desired IREGION.

Examples:

```
(CREATEIR  window  21930  'myfunction  "This is the helpstring")

(CREATEIR (WHICHW) 1234 'MY.SELECTED.FN "This is the helpstring"
'(0 0 20 30) '((12 . 15)(2 . 29))
```

(SURROUNDIR *window shade buttoneventfn helpstring poslist inside.pos*) [Function]

window: the window which will contain the irregular region.

shade: can be either a number between 0 and 65535 for a 4 by 4 shading or a 16 by 16 bitmap (if *shade* is NIL then the default is black, 65535).

buttoneventfn: the function called when the region is selected. The arguments that are passed to the function are: the window containing the IREGION, the IREGION record itself, and the button which selected the IREGION.

helpstring: the string that is placed in the PROMPTWINDOW when the mouse is held over the item for a few seconds.

poslist: If specified, a list of positions relative to *window* that are the edge points for the FILLREGION routine. If NIL, the user will be prompted to define the outer border of the region desired to be active. Holding the SHIFT key will define the last point used in defining the edge. If this field is non-nil, *Inside.pos* must be specified.

Inside.pos: If specified, this would be the inside position in which the Fillregion routine would begin filling from. If *poslist* is non-nil, then this field must be specified.

Description of use: Like the CREATEIR function, this function creates IREGIONS. However, the functionality of this routine is quite different. There are times when you do not care what is within a particular region. Say, for example, you have a map of some country and you wish to surround a particular region of the country with an IREGION as you wish to denote an area rich in some mineral deposit or some other characteristic. Such a characteristic is oblivious of the borders of the country's states or provinces, streams, rivers,etc., yet you would like to make active a very general area. Upon calling this function, you are prompted to button around the area of interest. And so, in viewing the crosshairs cursor, you begin buttoning about specifying the border of the area you wish to make active, independant of what is inside it. To stop being prompted for the next edge, simply hold the SHIFT key on the keyboard, (either one will do), as you make your last button selection. At this point, the lisp DRAWCURVE function will take effect and draw the closed region you've defined. Note that the first and last points do not have to touch as the DRAWCURVE routine will connect them for you. You will also be prompted to button within the region you've marked. It is here that the Fillregion routine will begin filling your region from. When complete, this function adds the IREGION to the window and returns the iregion added.

Examples:

```
(SURROUNDIR  window  21930  'myfunction  "This is the helpstring")

(SURROUNDIR (WHICHW) 1234 'MY.SELECTED.FN "This is the helpstring"
'((5 . 5)(6 . 50)(50 . 50)(50 . 7)) '(10 . 10)))
```

(ADD.IREGION *window iregion*) [Function]

window: the window to which the iregion is to be added.

iregion: the IREGION to be added to window.

Description: This function will add iregion to window which will then allow mouse selection of that IREGION.

(REMOVE.IREGION *window iregion*) [Function]

window: the window in which the *iregion* exists.

iregion: the IREGION you wish to remove from *window*.

Description: This function removes the region from a list of active irregular regions which is stored as a window property of the window. The list of irregular active regions can be found by evaluating: (ALL.IREGIONS *window*)).

(INTERSECTING.IREGIONS? *window flg*) [Function]

window: a window.

flg: either T or NIL

Description: This function sets up window to allow selection of intersecting iregions. If two or more iregions overlap and this function had been called with flg = T, then when the overlapping region is selected, all of those iregions will be high-lighted and each IREGIONs BUTTONEVENTFN will be called. If flg is set to NIL, then the last IREGION created in that intersection of iregions will be selected. (Please be aware that intersecting iregions *might generate effects that you do not wish to have*. That is, if you leave the iregion "ON" (the exact same thing you see when you hold the

mouse button down on the iregion, done by inverting that iregion) and create another iregion intersecting with the first, then the mask of the second would have a partial image of the first. At this point, buttoning in an area where both regions interesect might show everything but the intersection of those regions. Sometimes, it all depends on the *order* that they are created and what iregion's mask is left on or off. Shades that are "negatives" or "equals" of one another might make matters more complex than necessary when they are intersected. It is recommend that you play with this function in order to understand how it actually works so that when you work it into your application you'll have a better idea of the functionality and end-results). If this becomes a problem, an EDIT.MASK function has been provided so that you may edit the mask of the iregion by hand. Currently, there are no programmatic methods for doing this.

(ALL.IREGIONS *window*) (Function)

window: a window containing IREGIONS.

Description: This function returns a list of all the IREGIONS attached to *window*.

(DOSELECTED.IREGION *window iregion button*) (Function)

window: the window associated with iregion.

iregion: the iregion to be activated

button: the button which selected iregion.

Description: Applied iregions BUTTONEVENTFN to window, iregion and button. This provides a programmatic way of activating a given IREGION. This does not invert the iregion.

(EDIT.MASK *iregion*) (Function)

iregion: the IREGION whose mask you want to edit.

Description: This function is provided for buttoning in places where the MASK is not set. More explicitly, TARGETing a region (while creating the regions) specifies the places where the FILLREGION routine is to create a mask. For example, if a US state contains many rivers one pixel wide, the FILLREGION routine will fill around the river, but not the river itself. This means that when the mouse is positioned on the river, the region will not shade because the mask does not have that bit turned on. However, if the mask is edited and the rivers filled in, buttoning on those rivers will activate the IREGION.

(INVERT.IREGION *window iregion*) (Function)

window: the window in which the *iregion* exists.

iregion: the IREGION targeted for shading.

Description: This will highlight the *iregion* with that *iregion*s shade. Calling it a second time will low-light it **.**

(IREGIONP *iregion*) (Function)

iregion: the IREGION to be tested.

Description: This function returns NIL if *iregion* is not an IREGION datatype and returns *iregion* if it is an IREGION.

(IREGIONPROP iregion prop newvalue) (Function)

iregion: the region of which you are setting/requesting the property.

prop: the property in which you are interested.

newvalue: the new value to be assigned to *prop*.

Description: As with WINDOWPROP, if newvalue is not specified, it will return the current value of the *iregion's* property. If newvalue is specified, then the property will be reassigned with that value. If a prop name is not one of the fields of an IREGION record, it will be stored in property-list format on the USERDATA field of the *iregion* record.

IREGION fields:

BUTTONEVENTFN - function called when *iregion* is selected.
USERDATA - property list format for user properties (similar to WINDOWPROP).
REGION - region relative to the window that surrounds the *iregion*.
MASK - a bitmap the same size of REGION that is blackened where the *iregion* is active.
SHADE - the shade number or bitmap used to shade the region.
HELPSTRING - the string that is printed in the *PROMPTWINDOW* when a region is held.

Examples:

```
(IREGIONPROP  iregion  'SHADE)      -- returns shade of iregion
(IREGIONPROP  iregion  'SHADE  21930) - assigns new shade to iregion.
```

(SHOW.ALL.IREGIONS *window shade delay*) (Function)

window: the window in which the IREGIONs exist.

shade: the shade with which the iregions will be shown.

delay: the time (in milliseconds) between which each IREGION is displayed . (if *delay* is NIL, then a default of 500 is used.)

Description: This function will shade and unshade in *shade* (black is used if *shade* is NIL), each IREGION that has been created in the particular window. This is especially useful when the user has lost track of the number of IREGIONS within a window.

(WHICH.IREGIONS *window posorx y*) (Function)

window: the window in which the IREGIONs lie. (if *window* is NIL, default is window to which mouse points).

posorx, y: the location within the window where the IREGIONs can be found. These points must be local to the window's coordinates...not the screen. (if *posorx* is a position, then it will be used, otherwise if x or y are not numbers then the current mouse position is used.)

Description: Will return either NIL or the list of IREGIONs found in *window* and specified by *posorx, y*.

Examples:

```
(WHICH.IREGIONS)
(WHICH.IREGIONS  MY.WINDOW  50  23)
(WHICH.IREGIONS  MY.WINDOW  '(50 . 23))
```

Saving IRegions

IREGIONS can be saved on a file by setting a variable to be the value returned by ALL.IREGIONS. This variable can be saved by using the file package command, UGLYVARS.

Example:

```
(SETQ IRS (ALL.IREGIONS (WHICHW)))
(SETQ SAVEIRSCOMS '((UGLYVARS IRS)))
(MAKEFILE 'SAVEIRS)
```

The file SAVEIRS can be loaded and IRS will be set. You can then add IRS to a window by doing:

```
(WINDOWPROP (WHICHW) 'IREGIONSLIST IRS)
(WINDOWPROP (WHICHW) 'BUTTONEVENTFN 'IN.CURSOR.REGION)
```

Caution: Some properties on the USERDATA field of an IREGION might not be saved correctly such as a window which can not be saved on a file.

Window images can be saved on a file by creating a bitmap the same size as the window, BITBLT from the window to the bitmap, and then saving the bitmap with the file package command VARS.

Example use of the AIRegions package:

1. Open a window...about 1/4 of a screen.
2. Use the paint function provided when you right-button in the window and paint a picture.

3. With your mouse in this painted window, type in:

```
(CREATEIR (WHICHW) 21930)
```

4. The cursor changes shape and prompts for creating a region similar to the prompt for creating a window. In this case, span a region that contains California.
5. When you are done, and the mouse button is released, the region spanned will remain temporarily on the screen. The cursor changes into a target and now prompts for a left-button within the region.

Select somewhere in California. When done, left-button the mouse outside and away from the temporarily blocked off region. (If you want to continue selecting areas of the same irregular region, in this example, the upper left corner of California, then button that area within the squared off region. As you can see, your irregular region does not necessarily have to be connected).

6. To test it out, simply button anywhere in California and it will fill to a nice shade of grey, as we have just set it up to do:

7. To create more active irregularly shaped regions, follow steps 3 through 5 above. If you want to set the selection of one of the regions to activate the execution of some function that calls RINGBELLS, and have the region shade to black upon selection, type in the following in the top level typescript window keeping the mouse within the painted window.

```
(CREATEIR (WHICHW) 65535 'IR.TESTFN)

(DEFINEQ (IR.TESTFN (LAMBDA (WINDOW IREGION BUTTON)
      (If (EQ (QUOTE LEFT) BUTTON)
        then (RINGBELLS 2)))))
```

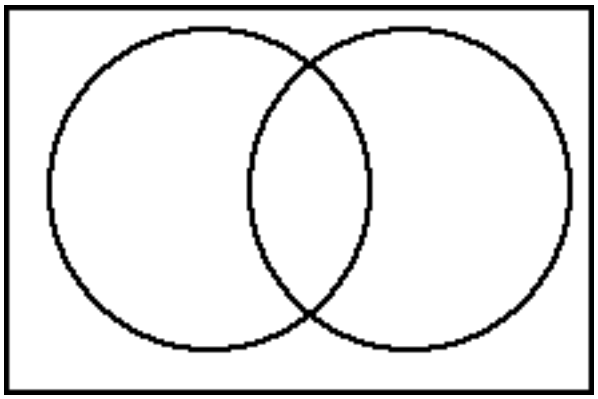
Span the cursor out over another state/region and repeat steps 3-5 above. When you button in this IREGION, the IREGION will temporarily shade black, and call the RINGBELLS function. Note that like menu selection, the function is called only when you release the button within the region. If the mouse button is held down and you move over the created IREGIONs, they will shade and unshade as you enter and exit them.

Note: if you wish to create your own shades but don't know what shades correspond to which numbers, call the function (EDITSHADE) and begin selecting points that you want shaded. When you are done, the function will return the appropriate shade number. You can also use 16x16 bitmaps for the shade of an IREGION (try (EDITBM (BITMAPCREATE 16 16)))

DEMO PACKAGE: To run the demo package, load AIRegions-Demo.

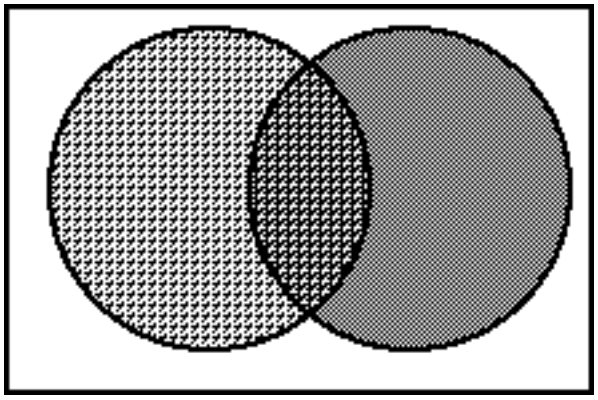
Intersecting Iregions

1. Create a window and paint in the following:



2. Now call CREATEIR passing in this window and a shade of 4747 and surround the left circle and select inside that circle and also in the intersecting area for the area fill. Repeat this for the right circle but use a different shade (say 42405).

3. Now, with your mouse in the window, call the function (INTERSECTING.IREGIONS? (WHICHW) T). When you button in the intersection of the two circles, you should get:



4. When the mouse is released inside of the intersecting region, both IREGIONs BUTTONEVENTFN will be called.

Comments and suggestions are welcome.