ACTIVEREGIONS

By: D. Barstow

ACTIVEREGIONS is a lispusers package for specifying that certain regions of a window be sensitive to button activity of the mouse. Whenever the left mouse button is held down and the mouse is within one of the active regions associated with a window, that region will be highlighted. A region is "picked" by releasing the left button when the region is highlighted. When a region is picked, the associated function is invoked. The picked region is only lowlighted by calls to SETPICKREGION. SETPICKREGION is called automatically whenever the mouse is used to pick a new region; otherwise it is the user's responsibility to call it when appropriate. The most recently picked region is saved and can be accessed by a call to GETPICKREGION. It is probably best not to try to mix this package with other menus or button activity in a window, since BUTTONEVENTFN may get confused.

SETACTIVEREGIONS(WINDOW REGIONLIST HIGHLIGHTFN LOWLIGHTFN)

REGIONLIST is a list of elements of instances of ACTIVEREGION, a record with seven fields:

REGION is relative to WINDOW

HELPSTRING is a string which will be put in the prompt window when the mouse button is held down sufficiently only.

DOWNFN is a function with three arguments: WINDOW, REGION, and DATA. It is called when either:

1. the left goes down in a region
2. a region is entered with the left button down after the left button went down in the window

UPFN is a function with three arguments: WINDOW REGION DATA it is called when the left goes up in a region

HIGHLIGHTFN, if given, overrides the highlightfn used in the call to SETACTIVEREGIONS

LOWLIGHTFN, if given, overrides the lowlightfn used in the call to SETACTIVEREGIONS

DATA is arbitrary information you may associate with the region If REGIONLIST is NIL, the active region stuff for the window wiii be disabled.

HIGHLIGHTFN and LOWLIGHTFN are functions of two arguments: WINDOW and ACTIVEREGION. LOWLIGHTFN defaults to HIGHLIGHTFN, which in turn defaults to inverting the indicated region.

FINDACTIVEREGION(WINDOW POSITION REGIONLIST/OPTIONAL)

Returns the ACTIVEREGION which contains POSITION. If there are several, returns the first in the activeregion list.

ADDACTIVEREGION(WINDOW ACTIVEREGION)

Adds ACTIVEREGION at the front of the activeregion list for WINDOW.
DELETEACTIVEREGION(WINDOW ACTIVEREGION)

Deletes ACTIVEREGION from the activeregion list for WINDOW.

SETPICKREGION(WINDOW ACTIVEREGION OLDREGION/OPTIONAL)

Lowlights is the most recent picked region of a window and sets ACTIVEREGION to be the new active region.

GETPICKREGION(WINDOW)

Returns the pick region of a window (as an ACTIVEREGION).

Examples:

If you want each active region to act like a menu item, simply put the appropriate actions as the PICKFNs of the regions in REGIONLIST. Each PICKFN should call (SETPICKREGION WINDOW NIL) unless it redisplays the entire window, in which case it should call SETACTIVEREGIONS again.

If you want to pick a region and later use it (e.g., as an argument to an item on some fixed menu), give each ACTIVEREGION a PICKFN of NIL and use (GETPICKREGION WINDOW) to determine the most recently picked region. This action would presumably want to call (SETPICKREGION WINDOW NIL) or something similar.

ALL

By: Don Cohen (DonC @ USC-ISIF.ARPA)

This file implements the "ALL" file package type, which causes the definitions of various types for a given word to be put together in a file. This is nice for people who look at listings and would like to see the documentation next to the function definition, or would like the MACROS listed in with the FNS.

ALL [FILEPKGCOM, FILEPKGTYPE]

An ALL filepackage command looks something like this: (ALL (COMMENTS FNS MACROS VARS) (FOO1 FNS VARS) (FOO2 VARS) (FOO3 MACROS)). The second element is a list of the types that may be stored in the list, and the remaining elements are lists where the first word is the atom to be dumped and the others are the types. The form (ALL * FOOALL) is also recognized, where the value of FOOALL has the same format except that the ALL is removed. The file package command above dumps the function definition of FOO1, then the value of FOO1, then the value of FOO2 and finally the macro definition of FOO3. As a special case, some of the types may be lists of the form (PROPS p1 p2 ...) which means to put those properties in the file. When you say to store the definition of type T1 for the atom A1 on a file, the file package looks for a command in which to put it. When an ALL command is encountered, the following algorithm is used: If T1 is not in the type list then the definition will not be part of this command. If A1 is already in this command, then T1 will be inserted. The insertion attempts to maintain the order of types (and properties) in the type list. (Of course this can be subverted by the user who edits the list.) If the user specified Near: to the file package, then (A1 T1) will be inserted after the atom specified. (Of course, if that atom was not already in the ALL command, the insertion fails.) Otherwise, the list (A1 T1) is inserted in the list alphabetically. If the file package can not put the definition into one of the commands that is already in the file, it creates a new command. If T1 is one of the types in the list ALL-TYPES, then instead of creating a command like (T1 A1), the file package will create the command (ALL <ALL-TYPES> (A1 T1)).

ALL-TYPES [VALUE]
is the list of types that will be intercepted by the "ALL" package (see ALL), and is also the list of types inserted into newly created ALL filepkgcoms. Its initial value is (COMMENTS FNS MACROS VARS LISPXMACROS USERMACROS), but of course that can be set by the user.

ANIMATE

By: Dan Bobrow and Mark Stefik

This small package contains functions for moving a non-rectangular bitmap smoothly around the screen, ways of using these to get big cursors, and bitmaps for a large arrow and a hand to be used as large cursors.

(AnimateSetUp picture shadow x y saveBitMap) Uses picture (an arbitrary rectangular bit map) shadow (a bit map which is black inside outline of picture)

AnimateSetUp places picture on screen at <x,y>, leaving the background around shadow. It returns saveBitMap, which is created the same size as picture if not given. AnimateMove uses saveBitMap to compute the new screen image when the picture is moved.

(AnimateMove picture shadow saveBitMap oldx oldy newx newy) Moves a picture set up with AnimateSetUp from <oldx, oldy> to <newx, newy>

(FollowCursor picture shadow saveBitMap) Waits for a mouse button to be down, blanks out the usual cursor, and causes picture to track the mouse movement until all the mouse buttons are up.

BIGCW is a function for making a large icon which stays on the screen as a window until it is selected with the mouse when it will follow the cursor.

(BIGCW POS ICONBITMAP SHADOWBITMAP SAVEBITMAP) Creates a window which contains ICONBITMAP. When the window is selected (clicked with a mouse button) BIGCW closes the window, and causes the ICONBITMAP to Follow the mouse movement. When the button is lifted, the original window reappears.

Two sets of bit maps come with the file Hand, and HandShadow BigArrow BigArrowShado.

(BIGCW) defaults to using the Hand, allowing the user to place the window.

(BIGCW NIL BigArrow BigArrowShadow) will create a large arrow which can be used for pointing.

ARITHDECLS

By: RON KAPLAN

This file is for use with the DECL package. It defines the DECLOF properties for a variety of arithmetic functions (ADD1, QUOTIENT, RAND, ETC.)

BACKGROUND

By: Mike Bird (Bird.pasa @ Xerox)

BITBLT to windows (without **TOTOPW**-ing them) or to background. For example, this is much faster than **CHANGEBACKGROUND** for numeric and 4x4 textures.

Warning: This package uses the **WINDOW** system datatype and may, therefore, require recompilation for new releases of Interlisp-D. (The present version has been tested with Carol and Harmony).

(**BITBLTTOBACKGROUND** *SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE CLIPPINGREGION*)

This is similar to **BITBLT** except *DESTINATION* must be either an open window or NIL (which means "background"). The undocumented **BITBLT** *SOURCETYPE* **MERGE** is supported - this requires both source and texture.

Known problems:

(1) Textures are always aligned with the screen origin rather than the top/lh corner of the destination.

(2) Only numeric and 4x4 bitmap textures are supported.

The following functions can be used for high-performance applications requiring several **BITBLT's** to a destination within a short period of time. The operations of analyzing the destination and of **BITBLT**-ing into it are seperated. *BACKGROUNDDESTINATION* returns a description of the destination in a form acceptable to **BITBLTTODESTINATION**. This description consists of a record of datatype **\BG.DEST**. Although it is possible to rely on garbage collection to manage these records, the package maintains a freelist to which obsolete records can be posted by **DESTINATION.RELEASE** or (if RELEASEBOUNDARIES is specified) by **BITBLTTODESTINATION** itself. Since the BOUNDARY package is used internally (which has a similar freelist mechanism) it is possible to write real-time graphical display programs which run with very little garbage collection. (cf BACKGROUNDDEMO package).

Programming Notes:

(1) Users should take care to ensure that the position/shape/existence of windows does not change between the construction of a destination and its use. For example, a program shouldn't **BLOCK** during this interval.

(2) For optimum performance the *DESTINATIONLEFT, DESTINATIONBOTTOM, WIDTH* and *HEIGHT* arguments to **BITBLTTODESTINATION** should not be used unnecessarily.

(**BACKGROUNDDESTINATION** *DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT CLIPPINGREGION*)

This function constructs a **\BG.DEST** record describing the destination for subsequent use by **BITBLTTODESTINATION**. The arguments are used in the same way as the corresponding arguments to **BITBLTTOBACKGROUND**.

(**BITBLTTODESTINATION** *SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE RELEASEBOUNDARIES*)

This function performs a **BITBLT** to *DESTINATION* (which must be a **\BG.DEST** record). *SOURCEBITMAP, SOURCELEFT, SOURCEBOTTOM, SOURCETYPE, OPERATION*, and *TEXTURE* are used as in **BITBLT** itself. *DESTINATIONLEFT, DESTINATIONBOTTOM, WIDTH* and *HEIGHT* can be used to restrict the destination specified by the **\BG.DEST** record. If *RELEASEBOUNDARIES* is non-**NIL** the **\BG.DEST** record and any **BOUNDARY** records it references are released.

(**DESTINATION.RELEASE** *DESTINATION*)

The **\BG.DEST** record *DESTINATION* and any **BOUNDARY** records it references are released.

BACKGROUNDDEMO

By: Mike Bird (Bird.pasa @ Xerox)

This package serves as a demonstration and a programming example for the BACKGROUND package.

(BACKGROUNDDEMO *NUMBEROFOBJECTS WINDOWOPTION*)

This function creates a background process which bounces *NUMBEROFOBJECTS* hollow spheres in the *WINDOWOPTION* (if specified - in which case it must be an open window) otherwise the background. Any number of these processes can run simultaneously. They can be killed from *PSW*.

Examples:

BACKGROUNDDEMO(6)

(BACKGROUNDDEMO 4 (WHICHW]

BIZGRAFIX

By: Herb Jellinek (Jellinek.pa @ Xerox.ARPA)

An Interlisp-D package for creating charts and graphs.

BIZGRAFIX is a package that contains functions for creating pie charts, bar charts, line and scatter graphs.

There are three "top-level" functions in BIZGRAFIX:

(PIE *percentages chartWindow filled?*)

> Draws a pie chart on chartWindow. *percentages* is a list of dotted pairs, each of the form **(label . percent)**. The *filled?* flag determines whether the slices of the pie should be filled with texture. Filling is currently a very slow operation.

(BAR *points window label*)

> Draws a bar chart on *window*. *points* is a list of dotted pairs of the form **(xcoord . ymagnitude)**. The xcoord is ignored as we assume that each bar is of the same width, and that points are sorted in x-ascending order. *label* is an atom or string, and is used to label the chart.

(LINEGRAPH *points graphWindow wantLines hUnits vUnits*)

> Draws a line- or scatter-graph on *graphWindow*. *points* is as above, a list of dotted pairs (POSITIONs) **(x . y)**. *wantLines* controls whether lines will be drawn between successive points on the graph: if *wantLines* is **T** you get a line graph; if **NIL** you get a scattergram. *hUnits* is the label for the X-axis, and *vUnits* the Y-axis label.

All three routines give their argument windows the property of being able to be reshaped and redrawn. Reshaping a window that has been drawn on by one of the above functions causes the information it contains to be redrawn and rescaled to fit the new shape of the window.

BIZGRAFIX depends upon the **TWOD** LispUsers package.

BLACKOUT

BY: Richard Acuff

Loading BLACKOUT.DCOM adds the item Blackout to the background menu. Selecting this item will cause the screen to go dark except for a bouncing box with the the name of the current user. The machine will stay in this state until a key is struck, thus keeping the screen from getting "burnt".

Blackout also has a secure mode in which it will demand that the user correctly type her password before ceding control. Thus, you can leave your terminal temporarily without fear of sensitive information on your screen being compromised.

The variable BLACKOUT.ASK.PASSWORD controls Blackout's mode. If it is NIL, no password will be required (non-secure mode). If it is set to MENU, a menu will pop up allowing the user to select the mode interactively. Any other value implies secure mode.

The variable BLACKOUT.ICON contains a window which is bounced around the screen when Blackout is running. It initially contains the user's name, but may be changed to some other window.

Cavets

Unfortunately, it is impossible for other processes to run while Blackout is running in secure mode, due to Lisp's process and window mechanism. However, other processes may run while non-secure Blackout is in use. They should not attempt to do keyboard input, however, or the screen will get rather confused.

Typing errors made while inputing the password cannot be corrected. You can, however, try again.

BLTDEMO

By: Michel Denber (Denber.wbst @ Xerox)

Creates a window which demonstrates BITBLT with a spinning star, a ring and a box bouncing off the sides of a window.

Call (BOUNCE x y), where x and y are velocities in the x and y directions respectively. X and y defaults to 3 if omitted. You will be prompted to open a window. The box shows whatever is near the cursor.

Interesting recursive effects can be seen if you move the cursor near the box.

BOUNDARY

By: Mike Bird (Bird.pasa @ Xerox)

Introduction

A "boundary" is defined by its left and bottom inclusive bounds and by its right and top exclusive bounds. Although equivalent to a "region" it is computationally more efficient in certain applications. For further efficiency, the datatype defined here includes a link field. This can significantly reduce CONS-ing.

Create and release functions are provided which use a freelist. This helps to reduce garbage collection.

The datatype is defined with POINTER fields rather than FIXP or SIGNEDWORD. This is computationally more efficient, especially in the important case of SMALLP values. The additional storage required for FIXP values is not significant for most applications.

*** Wherever this document refers to a list of boundaries, this signifies linkage via the NEXT.BOUND field.

BOUNDARY: DATATYPE

LEFT.BOUND - POINTER
BOTTOM.BOUND - POINTER
RIGHT.BOUND - POINTER
TOP.BOUND - POINTER
NEXT.BOUND - POINTER

BOUNDARY.CREATE(LEFT BOTTOM RIGHT TOP NEXT) FUNCTION

Creates, initialises and returns a boundary. Uses a boundary from the freelist if possible. Fields whose values are omitted are initialised to NIL, not zero.

BOUNDARY.DIFFERENCE(OUTER INNER) FUNCTION

Returns a (possibly null) list of BOUNDARY's which together comprise the difference between the OUTER and INNER boundaries.

*** Assumes no part of INNER is outside OUTER.

BOUNDARY.FREELIST VARIABLE

The freelist of available boundaries. Initially NIL.

BOUNDARY.INTERSECTION(X Y) OPENLAMBDA

If X and Y have a non-empty intersection, the boundary of that intersection is returned. Otherwise NIL is returned.

BOUNDARY.INTERSECTION*(X Y.LEFT Y.BOTTOM OPENLAMBDA
Y.RIGHT Y.TOP)

Version of BOUNDARY.INTERSECTION for use when one of the boundaries' bounds have already been unpacked from the boundary.

BOUNDARY.INTERSECTION**(X.LEFT X.BOTTOM OPENLAMBDA
X.RIGHT X.TOP Y.LEFT Y.BOTTOM
Y.RIGHT Y.TOP)

Version of BOUNDARY.INTERSECTION for use when both of the boundaries' bounds have already been unpacked.

BOUNDARY.INTERSECTP(X Y) OPENLAMBDA

Predicate for determining whether X and Y have a non-empty intersection.

BOUNDARY.INTERSECTP*(X Y.LEFT Y.BOTTOM OPENLAMBDA
Y.RIGHT Y.TOP)

Version of BOUNDARY.INTERSECTP for use when one of the boundaries' bounds have already been unpacked from the boundary.

BOUNDARY.INTERSECTP**(X.LEFT X.BOTTOM OPENLAMBDA
X.RIGHT X.TOP Y.LEFT Y.BOTTOM
Y.RIGHT Y.TOP)

Version of BOUNDARY.INTERSECTP for use when both of the boundaries' bounds have already been unpacked.

BOUNDARY.MERGE(OLD.BOUNDARIES NEW.BOUNDARIES) FUNCTION

For each member, N, of NEW.BOUNDARIES:

While N can be combined with any member, O, of OLD.BOUNDARIES to form a single equivalent boundary; N is smashed to produce the combined boundary and O is released. Finally N is pushed onto the OLD.BOUNDARIES list.

*** Warning: Smashes NEW.BOUNDARIES.

BOUNDARY.POP(STACK) MACRO

Returns the first boundary in STACK and changes STACK to refer to the next boundary.

*** The STACK expression should be simple and have no side-effects.

*** The NEXT.BOUND of the returned boundary is not nullified.

BOUNDARY.PUSH(STACK ITEM) MACRO

Pushes the boundary ITEM onto the front of the list STACK and changes STACK to reference it.

*** Both STACK and ITEM should be simple and have no side-effects.

BOUNDARY.RELEASE(BOUNDARIES) FUNCTION

The members of the specified list of boundaries are transferred to the freelist. This smashes their NEXT.BOUND links.

*** To release a single boundary, NEXT.BOUND of BOUNDARIES must be null.

BOUNDARY.UNION(X Y) OPENLAMBDA

Returns the smallest BOUNDARY which contains the regions contained by the boundaries X and Y.

BOUNDARY.UNION*(X Y.LEFT Y.BOTTOM Y.RIGHT Y.TOP) OPENLAMBDA

Version of BOUNDARY.UNION for use when one of the boundaries' bounds have already been unpacked from the boundary.

BOUNDARY.UNION**(X.LEFT X.BOTTOM OPENLAMBDA
X.RIGHT X.TOP Y.LEFT Y.BOTTOM
Y.RIGHT Y.TOP)

Version of BOUNDARY.UNION for use when both of the boundaries' bounds have already been unpacked.

BQUOTE

By: Kelly Roach

INTRODUCTION

This package implements the "backquote convention" punctuation marks backquote "`", comma ",", and commaat ",@" used in constructing lists. To use, load <LISPUSERS>BQUOTE.DCOM. This package offers all the behaviour that <LISPUSERS>PQUOTE.DCOM has to offer. You should not load PQUOTE if you load BQUOTE.

This package has many nice features that Interlisp's nlambda BQUOTE "type-in" approach does not have. Certain unexpected breaks that are possible with Interlisp's nlambda BQUOTE do not occur with this package. Unlike Interlisp's BQUOTE, this package checks for uncaptured commas, guarding against possible user error. Backquotes in files, in breaks during reads, around dots, and around other backquotes are handled appropriately. Backquoted forms are prettyprinted for the user.

FINDING BACKQUOTE

Backquote is character 96. Normally, typing <LF> on the 1100 keyboard or typing <SAME> on the 1108 keyboard produces a line feed. Typing <SHIFT-LF> or <SHIFT-SAME> will produce the backquote character. BQUOTE makes the backquote character easier to access by switching the <LF> keyaction with the <SHIFT-LF> keyaction on the 1100 keyboard and switching the <SAME> keyaction with the <SHIFT-SAME> keyaction on the 1108 keyboard. When BQUOTE is loaded, typing <LF> or <SAME> will produce backquote and typing <SHIFT-LF>or <SHIFT-SAME> will produce line feed.

USING BACKQUOTE

The punctuation marks backquote "`", comma ",", and commaat ",@" are used to build lists. Backquote "`" is used just like quote "'" and quotes everything in an expression not preceded by a comma ",". Hence,

`(A ,B ,C) translates to (LIST 'A B C)

`(if ,C then ,F) translates to (LIST 'if C 'then F)