Sketch programmer's interface

Richard R. Burton

January 28, 1986

TABLE OF CONTENTS

INTRODUCTION

Sketch is an interactive illustration program that can be used to construct figures from text and graphics. It is described in the Xerox PARC report *Sketch: A Drawing Program for Interlisp-D* by R. Burton and in the Interlisp-D library packages documentation (koto release). As part of its implementation, Sketch provides a representation for and functions to manipulate two dimensional figures consisting of elements such as curves or pieces of text. This document describes a programmer's interface that allows applications to use Sketch as a front-end for displaying and manipulating graphical presentations. Using this interface, an application can create a sketch that represents its own data structures and display it in a Sketch viewer. A user can then use the standard Sketch interface to edit the representation and have the changes reflected to the underlying data structures. The interface allows elements that make up the sketch to be individually protected from modification. Additionally, the application can change the interface in a wide range of ways from globally replacing the entire Sketch menu and the set of allowable commands to locally restricting the particular values that particular elements can have. This document also describes how to get a sketch of the display of any Interlisp-D program by using sketch streams. It describes the initial sketch types.

It is envisioned that typical application of the programmer's interface to Sketch will use the following sequence of steps: create sketch elements, possibly grouping them together, add them to a sketch, open a sketch viewer onto the sketch and monitor the user's changes to them. This manual is organized roughly along these lines. Section 1 describes how to create sketch elements and group them. Section 2 describes how to create a sketch, and add and delete elements from it. Section 3 describes the various ways of opening a viewer onto a sketch as well as some other functions for dealing with viewers. Section 4 describes the hooks through which the application monitors the actions the user has requested and disallows or modifies them. At the end of section 4 some useful functions for replacing aspects of the Sketch interface are described. Section 5 describes the properties that sketch elements have and how to change them. At the end of Section 5 are specifications of Sketch's basic data structures that the application may need. Section 6 describes a collection of utility routines for dealing with Sketches. Section 7 describes sketch streams that provide a way of creating sketches using the standard display functions. Section 8 describes Sketch element types.

1.0 CREATING SKETCH ELEMENTS

The element types are TEXT, TEXTBOX, WIRE, CLOSEDWIRE, OPENCURVE, CLOSEDCURVE, CIRCLE, ARC, ELLIPSE, BOX, SKIMAGEOBJ, BITMAPELT and GROUP. The following functions are provided to create elements in a sketch. This allows user programs to create elements from their own data structures and remember the correspondence so that sketch operations can be reflected in their structures. A common way of

making the association between a sketch element and a piece of data is to save the data on the property list of the sketch element using the function PUTSKETCHELEMENTPROP. An alternative way is to put a unique identifier on the property list of the element that the application program can associate with its data. It does not work very well to keep a pointer from the application data to the sketch element because the Sketch operations replace elements with a copy rather than modifying them. Thus, only appications which prevent all Sketch operations can use this method.

(SKETCH.CREATE.TEXT STRING POSITION FONT JUSTIFICATION COLOR SCALE)

Creates a sketch text element that contains the characters in STRING. POSITION is the location of the text element's control point in sketch coordinates. FONT is the text element's font. JUSTIFICATION is a list whose first element is the horizontal justification (one of LEFT, CENTER, RIGHT) and whose second element is the vertical justification (one of TOP, CENTER, BASELINE, BOTTOM). It is used to determine the position of the control point within the area covered by the text. COLOR is the color that the text will appear in. SCALE is the scale at which the text will be exact size (1.0 if SCALE is NIL).

(SKETCH.CREATE.TEXTBOX STRING REGION FONT JUSTIFICATION BOXBRUSH BOXDASHING FILLING TEXTCOLOR SCALE)

Creates a sketch text box element whose region is REGION in sketch coordinates and that contains the characters in STRING. FONT is the text box element's font. JUSTIFICATION is a list whose first element is the horizontal position of the text within the region (one of LEFT, CENTER, RIGHT) and whose second element is the vertical position (one of TOP, CENTER, BASELINE, BOTTOM). BOXBRUSH is the brush that the box will be drawn in. BOXDASHING is the dashing pattern of the box. FILLING is the filling pattern of the part of the box not covered by text. TEXTCOLOR is the color that the text will appear in. SCALE is the scale at which the text and box brush will be exact size (1.0 if SCALE is NIL).

(SKETCH.CREATE.BOX REGION BRUSH DASHING FILLING SCALE)

Creates a sketch box element whose region is REGION in sketch coordinates. BRUSH is the brush that the box will be drawn in. DASHING is the dashing pattern of the box. FILLING is the filling pattern of the part of the box. SCALE is the scale at which the brush will be the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.WIRE POINTS BRUSH DASHING ARROWHEADS SCALE)

Creates a sketch wire element. A wire element is series of control points connected by straight lines. POINTS are the control points that are connected. BRUSH is the brush that the lines will be drawn in. DASHING is the dashing pattern of the lines. ARROWHEADS is a list of the arrowhead specifications (see description below). The first one refers to the arrowhead at the first point of the wire. The second one refers to the

arrowhead on the last point. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.CLOSED.WIRE POINTS BRUSH DASHING FILLING SCALE)

Creates a sketch closed wire element. A closed wire element is series of control points connected by straight lines and includes a line from the first point to the last one. This can also be thought of as a polygon. POINTS are the control points that are connected. BRUSH is the brush that the lines will be drawn in. DASHING is the dashing pattern of the lines. FILLING is the filling pattern of the area within the closed wire. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.OPEN.CURVE POINTS BRUSH DASHING ARROWHEADS SCALE)

Creates a sketch open curve element. An open curve element is a smooth curve that passes through a series of points. POINTS are the control points that the curve passes through. BRUSH is the brush that the curve will be drawn in. DASHING is the dashing pattern of the curve. ARROWHEADS is a list of the arrowhead specifications (see description below). The first one refers to the arrowhead at the front of the curve (at the first point). The second one refers to the arrowhead on the end of the curve (at the last point). SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.CLOSED.CURVE POINTS BRUSH DASHING WILLBEFILLING SCALE)

Creates a sketch closed curve element. A closed curve element is smooth curve that passes through a series of points and ends up back at the first point. POINTS are the control points that the curve passes through. BRUSH is the brush that the curve will be drawn in. DASHING is the dashing pattern of the curve. WILLBEFILLING is currently ignored but is planned to be the filling that the closed curve is filled with. It is provided at this time so that the curve creating functions will be symmetric with the wire creating ones. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.CIRCLE CENTERPT RADIUSPT BRUSH DASHING FILLING SCALE)

Creates a sketch circle element that has its center at CENTERPT. If RADIUSPT is a point, it will be used as the circle's second control point and the circle will pass through it. If RADIUSPT is a number, it is taken to be the radius and the second control point is (CenterX + RADIUSPT, CenterY). BRUSH is the brush that the circle will be drawn in. DASHING is the dashing pattern of the circle. FILLING is the filling pattern of the area within the circle. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.ARC CENTERPT RADIUSPT ANGLEPT BRUSH DASHING ARROWHEADS DIRECTION SCALE)

Creates a sketch arc element that has its center at CENTERPT. RADIUSPT is a point and is used as the beginning point of the arc and determines the radius. If ANGLEPT is a position, it is used as the end point of the arc (e.g. the arc will end where is meet the ray from CENTERPT to ANGLEPT) and the direction of the arc is determined by DIRECTION;

counterclockwise if DIRECTION is NIL or 'COUNTERCLOCKWISE, clockwise if it is T or CLOCKWISE. If ANGLEPT is a number, it is taken to be the number of degrees in the arc; positive if counterclockwise, negative if clockwise. In this case, DIRECTION will be ignored. In either case, the arc's third control point will be the actual end of the arc. BRUSH is the brush that the arc will be drawn in. DASHING is the dashing pattern of the arc. ARROWHEADS is a list of the arrowhead specifications (see description below). The first one refers to the arrowhead at the beginning point of the arc. The second one refers to the arrowhead on the end point. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.ELLIPSE CENTERPT ORIENTATIONPT OTHERRADIUSPT BRUSH DASHING WILLBEFILLING SCALE)

Creates a sketch ellipse element that has its center at CENTERPT. ORIENTATIONPT is a position and is used to determine the orientation of the ellipse and one of its radii, i.e., the ellipse will pass through it. OTHERRADIUSPT is a position. The distance between it and CENTERPT is the second radius. The third control point of the ellipse will be one of the points on the ellipse at the second radius. BRUSH is the brush that the ellipse will be drawn in. DASHING is the dashing pattern of the ellipse. WILLBEFILLING is ignored but is planned to be the filling pattern of the area within the ellipse. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.BITMAP BITMAP POSITION SCALE SCALECACHE)

Creates a sketch bitmap element from the bitmap BITMAP and positions its lower left corner at POSITION. SCALE is the scale at which the bitmap is the size given (1.0 if SCALE is NIL). SCALECACHE is a list of tuples (SCALE SCALEDIMAGE): the first element is a scale and the second is a bitmap. Sketch will use the SCALEDIMAGE on the cache as the image to be displayed (possibly scaled) for BITMAP when the scale of the viewer is closer to that scale than SCALE and than to any other tuple on SCALECACHE. This provides a way of specifying higher resolution images at larger scales.

(SKETCH.CREATE.IMAGE.OBJECT IMAGEOBJ POSITION SCALE)

Creates a sketch image object element from the image object IMAGEOBJ and positions its lower left corner at POSITION. SCALE is the scale at which the brush is the size given (1.0 if SCALE is NIL).

(SKETCH.CREATE.GROUP LISTOFSKETCHELEMENTS CONTROLPOINT)

Creates a sketch group element from a list of sketch elements. The elements can be group elements as well. CONTROLPOINT is the control point of the group. If not given, the control point will be the center of the rectangular region occupied by the elements. It provides a way of ensuring that the control point is on a grid point.

***May want functions for taking the difference between two states of a sketch stream and making a group element out of the difference.

2. MANIPULATING ELEMENTS IN A SKETCH

Note: Any function that takes a sketch as an argment will accept either a SKETCH structure (as returns from SKETCH.ADD.ELEMENT), a sketch viewer (as returned from SKETCHW.CREATE) or a SKETCH stream (obtained via (OPENIMAGESTREAM 'name 'SKETCH)). This is implemented by having them call the following function which is also available to user functions:

(INSURE.SKETCH SKETCH NOERRORFLG)

If SKETCH is a SKETCH structure, a sketch window (called a viewer) or a SKETCH stream (obtained via (OPENIMAGESTREAM 'name 'SKETCH)), it returns the appropriate SKETCH structure. If SKETCH is not a sketch, a sketch window or a sketch stream, it returns NIL if NOERRORFLG is non-NIL; otherwise, it causes an error.

(SKETCH.ADD.ELEMENT ELEMENT SKETCH NODISPLAYFLG)

Adds an element to a sketch. If SKETCH is NIL, a new sketch is created. SKETCH.ADD.ELEMENT returns the sketch. If ELEMENT is NIL, SKETCH is not changed but is returned. Thus, (SKETCH.ADD.ELEMENT NIL NIL) provides a way of creating an empty sketch. If NODISPLAYFLG is NIL, any windows currently displaying SKETCH will be updated to reflect ELEMENT's addition. If NODISPLAYFLG is T, the displays won't be updated.

(SKETCH.DELETE.ELEMENT ELEMENT SKETCH INSIDEGROUPSFLG NODISPLAYFLG)

Deletes an element from a sketch. If INSIDEGROUPSFLG is T, the element will be deleted even if it is inside a group. Otherwise it will be deleted only if it is a top level element. If NODISPLAYFLG is NIL, any viewers currently displaying SKETCH will be updated to reflect ELEMENT's deletion. If NODISPLAYFLG is T, the viewers won't be updated.

It returns ELEMENT if ELEMENT was deleted.

(SKETCH.ELEMENTS.OF.SKETCH SKETCH)

Returns the list of elements that are in SKETCH. SKETCH can be either a SKETCH structure, a sketch viewer or a SKETCH stream. If SKETCH is not a sketch, a sketch viewer or a sketch stream, it returns NIL. This can be used with sketch streams to determine the elements created by a call to a display function or series of functions by looking at the list differences; new elements are always added at the end.

(SKETCH.LIST.OF.ELEMENTS SKETCH PREDICATE INSIDEGROUPSFLG)

Returns a list of the sketch elements in SKETCH that satisfy PREDICATE. If INSIDEGROUPSFLG is T, elements that are members of a group will be considered too. Otherwise only top level objects are considered. Note: PREDICATE will be applied to GROUP elements even when INSIDEGROUPSFLG is T.

(SKETCH.REGION.OF.SKETCH SKETCH)

Returns the region occupied by SKETCH in sketch coordinates.

(SKETCH.SET.DEFAULT SKETCH SKETCHPROPERTY NEWDEFAULT)

****(not yet implemented.)

3. SKETCH VIEWERS

(SKETCHW.CREATE SKETCH SKETCHREGION SCREENREGION TITLE INITIALSCALE BRINGUPMENUFLG INITIALGRID)

opens a sketch viewer onto the sketch SKETCH. SKETCH can be NIL, in which case a new, empty sketch is used, or a sketch. SKETCHREGION, if given, is a region in sketch coordinates of the part of the sketch to be initially displayed. The lower left corner of the viewer will be mapped onto the lower left corner of SKETCHREGION. If SKETCHREGION is given and INITIALSCALE is NIL, the scale will be computed so that the height of SKETCHREGION just fits into the viewer.

If SCREENREGION is a region, it is used as the screen region for the sketch viewer. If SCREENREGION is a window, it is used as the sketch viewer.

TITLE is used as the title of the sketch viewer. If TITLE is NIL, the title "Viewer onto a sketch" is used.

INITIALSCALE, if a number is the initial scale of the sketch. If it is NIL, 1.0 is used.

If BRINGUPMENUFLG is non-NIL, the sketch command menu is brought up on the right side of the viewer. BRINGUPMENUFLG can be a menu, in which case it is brought up on the right side of the viewer. The standard Sketch command menu is the SKETCHPOPUPMENU property of the window.

INITIALGRID indicates whether to use a grid. If it is a number, the grid is set to the least power of two greater than INITIALGRID. If INITIALGRID is T, the grid is set to 8.0 (approximately half a centimeter between grid points.) If it is NIL, no grid is used.

(EDITSLIDE SKETCH)

opens a window the size of a sheet of 8 1/2 by 11 paper. This is useful for laying out slides or other full-page images.

(SKETCH.GET FILENAME VIEWER)

Reads the sketch that is stored on FILENAME and adds it to the sketch in VIEWER.

(SKETCH.PUT FILENAME sketch)

Saves the sketch sketch on the file FILENAME.

(SKETCH.RESET SKETCH)

resets a sketch and all of the viewers onto it, i.e. deletes all its elements. It also resets the grid, and positions the view at the home position. It does NOT remove any of the user set sketch properties.

(SKETCH.REGION.VIEWED VIEWER NEWREGION)

Returns the region in sketch coordinates that a sketch viewer is viewing. If NEWREGION is given, it should be a region, the atom HOME or the name of a view created by SKETCH.ADD.VIEW. If NEWREGION is a region, the view will be scaled and/or scrolled so that the viewer is viewing NEWREGION. If NEWREGION is the name of a view, the viewer will be moved to that view.

(SKETCH.ADD.VIEW SKETCH NAME SCALE CENTERPOSITION)

Adds a view to SKETCH. NAME should be an atom which will appear in the menu of views and can be passed to SKETCH.REGION.VIEWED. SCALE is the scale of the view and CENTERPOSITION is the position of the center of the region viewed.

(VIEWER.TO.SKETCH.POSITION POSITION VIEWERSCALE)

Transforms a position from viewer coordinates into sketch coordinates. VIEWERSCALE can be a scale or a viewer.

(SKETCH.TO.VIEWER.POSITION POSITION VIEWERSCALE)

Transforms a position from sketch coordinates into viewer coordinates. VIEWERSCALE can be a scale or a viewer.

(VIEWER.TO.SKETCH.REGION REGION VIEWERSCALE)

Transforms a region from viewer coordinates into sketch coordinates. VIEWERSCALE can be a scale or a viewer.

(SKETCH.TO.VIEWER.REGION REGION VIEWERSCALE)

Transforms a region from sketch coordinates into viewer coordinates. VIEWERSCALE can be a scale or a viewer.

If a sketch window has a non-NIL value for the property DONTQUERYCHANGES, the user is not asked to confirm the closing of a viewer even though changes have been made to it and not saved.

(SKETCH.DISPLAYFN SKETCHIMAGEOBJ STREAM)

Displays a sketch image object on a stream. This is the image object display function for sketch image objects but it can also be used for other purposes. The current position of STREAM will be the lower left corner of the region the sketch appears in.

(SKETCH.BITMAP.IMAGE SKETCH REGION SCALE)

Returns a bitmap that has the sketch image in it. This is called by the the image object display function for sketch image objects.

(SKETCH.ALL.VIEWERS SKETCH)

Returns a list of all the viewers on the sketch SKETCH.

4. MONITORING A SKETCH VIEWER

Sketch performs a small number of operations on its elements: ADD, DELETE, CHANGE, MOVE, COPY and GROUP. Whenever one of these operations is performed, either by program or by user interaction, an application supplied function is called. These functions are stored as the following properties of the sketch (using PUTSKETCHPROP). Also listed are the arguments the function will be called with and the interpretations of values it returns.

(GETSKETCHPROP SKETCH PROPERTY)

Returns the value a sketch has for the property PROPERTY.

(PUTSKETCHPROP SKETCH PROPERTY VALUE)

Puts VALUE as the property PROPERTY of the sketch SKETCH. Returns VALUE.

Sketch Viewer Control Properties

WHENADDEDFN (VIEWER ELEMENT) (Note 2nd argument is a single element) If it returns the atom DON'T, nothing is added. If it returns an element, that element is added instead. Otherwise ELEMENT is added. This is called whenever an element is added or when the deletion of an element is undone.

WHENDELETEDFN (VIEWER ELEMENTS) (Note 2nd argument is a list of elements) If it returns the atom DON'T, nothing is deleted. If it returns a list of elements, only those elements are deleted. Otherwise ELEMENTS are deleted. This is called whenever an element is deleted or when the addition of an element is undone. Note: if a point is deleted from a wire or curve element, the WHENCHANGEDFN is called.

PREEDITFN (VIEWER ELEMENT) Called when the user edits a text or textbox element by typing while it is selected. ELEMENT is the text or text box being edited. This function is called before any changes are made. If it returns the atom DON'T, the edit is aborted. If not, the changes are batched (i.e., the WHENCHANGEDFN or WHENADDEDFN are called) until the user moves the caret or selects a menu command. (But see SKETCH.CLEANUP).

PREMOVEFN (VIEWER ELEMENTS ALIGNHOW) Called before the user is asked for a position. If ALIGNHOW is not NIL, the user selected an ALIGN command and ALIGNHOW will be one of LEFT, RIGHT, TOP, BOTTOM, EVENX or EVENY. If the PREMOVEFN returns a position (and ALIGNHOW is NIL), the elements will all be moved by that delta with the delta being interpreted as Sketch distances. If it returns the atom DON'T, the elements will not be moved.

The form of ELEMENTS encodes what control points of the elements are being moved too. It will be one of the following:

(T . ListOfElements) all of the elements are being moved.

(ListOfNumbers . Element) ListOfNumbers is a list of the control point numbers that are being moved in Element.

A list whose elements are of the form:

(ListOfNumbers . Element) ListOfNumbers is a list of the control point numbers that are being moved in Element.

(T . Element) the entire Element is being moved.

WHENMOVEDFN (VIEWER ELEMENTS DELTAPOSITION) Called after the user has been asked to specify an amount to move the elements by. ELEMENTS has the same form at PREMOVEFN above.

PRECOPYFN (VIEWER ELEMENTS) Called when the user has indicated that they want to copy ELEMENTS. If returns a position, the copies will be displaced by that delta from the originals. If returns DON'T, does nothing. (Not yet implemented feature: tell me if you want it. If returns an list of ELEMENTS, positions those instead of making a copy.)

WHENCOPIEDFN (VIEWER ELEMENTS DELTAPOSITION) Called after the user has indicated where to place the copy. The new position will be offset from the original one by the amount of x and y in the position DELTAPOSITION. If it returns a position, the copies will be offset by that amount instead. If it returns a list of elements, those elements are inserted instead. Note: the when added function will be called for the copy of each element on ELEMENTS that is actually placed.

WHENGROUPEDFN (VIEWER ELEMENTS) Called before a collection of elements ELEMENTS is made into a group. The whengroupedfn can return DON'T in which case the grouping will not happen; a list of elements in which case the returned elements will be grouped; or anything else in which case ELEMENTS will be grouped. Note: the when added fn will NOT be called on the newly created GROUP element. Nor are the when deleted fns called on ELEMENTS.

WHENUNGROUPEDFN (VIEWER GROUPELEMENT) Called before a group element is ungrouped. If it returns DON'T, GROUPELEMENT will NOT be ungrouped. Otherwise, it will.

BUTTONEVENTINFN (VIEWER ELEMENT) called if a button event (down transition) occurs in the ACTIVEREGION of ELEMENT.

WHENDEFAULTSETFN (VIEWER PROPERTY ChangeSpec) (*** Not yet implemented)

PRECHANGEFN (VIEWER ELEMENTS ChangeSpec) Called when the user has requested a change be made to collection of elements thru the user interface. The PRECHANGEFN can return the atom DON'T in which case the changes will not be made or a new change specification that is used instead of the proposed one.

WHENCHANGEDFN (VIEWER ELEMENT PROPERTY NEWVALUE OLDVALUE) Called whenever a property is changed. An interactive edit operation will result in a call on the WHENCHANGEDFN for each element that is changed. It is also called by the delete point command with property 'DATA and the new and old list of control points. And after interactive editing on text or textboxes with property 'DATA the new and old list of character strings.

In the case of the user adding a box around a text element, the special property HASBOX is used and NEWVALUE is the new element that is of type TEXTBOX and OLDVALUE is the old element that will have type TEXT. In the case of the user adding a box around a text element, the special property HASBOX is used and NEWVALUE is the new element that is of type TEXT and OLDVALUE is the old element that will have type TEXTBOX. In the case of a look same operation, more than one property is changed in one operation. In this case, the special property 'LOOKSAME is used with the NEWVALUE being the new text element and OLDVALUE being the old text element.

In the case of a change command being applied to a group element, the following calls are made. The PRECHANGEFN is called with the group element or elements being on the list of ELEMENTS. The WHENCHANGEDFN is called with each individual subelement of the group giving the property, new and old values for that subelement. Then the WHENCHANGEDFN is called for the group element with the property DATA and the new value being the list of new, changed subelements and the old values being the previous subelements. (Any subelements that were not changed by the command are the same.) Note: If this change is undone, the only call will be WHENCHANGED with the new group element as ELEMENT, PROPERTY being DATA, NEWVALUE being the original list of subelements and OLDVALUE being the change list of subelements.

Specification of arguments to the Change operation

The third argument passed to the PRECHANGEFN is a list of two elements: the name of the property being changed and the specification for how it should change. Listed below are the possible properties and change specifications.

Line or brush properties

(SIZE {LARGER SMALLER "a number"})

(SHAPE {ROUND SQUARE VERTICAL HORIZONTAL DIAGONAL})

(DASHING {NONE, "a dashing list"})

(FILLING {NONE, "a texture"})

(BRUSHCOLOR "a color")

(FILLINGCOLOR {NONE, "a color"}

(ARROW (<WhichEnd> <ArrowChangeSpec>))
where <WhichEnd> := {FIRST, LAST, LEFT, RIGHT, BOTH}
and <ArrowChangeSpec> := {ADD, DELETE, WIDER, NARROWER, LARGER, SMALLER, OPEN, CLOSED, CURVE, SOLID or (SAME . "an arrowhead spec")}

Curve properties

(ADDPOINT (AfterPoint NewPoint))

(DELETEPOINT Point)

Text properties

(TEXT {SMALLER, LARGER, BOLD, UNBOLD, ITALIC, UNITALIC, CENTER, LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, BOX, UNBOX})

(NEWFONT "a font list")

(SETSIZE "a font list")

(SAME "individual text elt.") ** make be a global element

(FAMILY&SIZE "a font list")

Arc properties

(ANGLE "number of degrees")

(DIRECTION {CLOCKWISE, COUNTERCLOCKWISE})

Group properties

POSITION - called before the control point of a group is repositioned.

Clearing out the interactive editing change

In order to avoid calling the WHENCHANGEDFN each time a character in a text or textbox element is changed by interactive editing, the characters are "batched" until the user performs some other action (such as move the caret) which indicates that they are through with the edit. At this time, a change event is put on the undo list and the WHENCHANGEDFN is called giving all of the characters typed. In some instances, the application may want to force this closing of the interactive change event to make sure it has been notified (via the WHENCHANGEDFN) of all of the characters typed. The following function performs this action. (Note: if you find yourself using this in places where you think the system ought to have known that the user had finished editing, please let me know.)

(SKETCH.CLEANUP SKETCH)

If the user has edited (is in the process of editing) a text or textbox, it creates an undo event for the edit and calls the WHENCHANGEDFN for it. It is slightly faster if SKETCH is a viewer.

Moving Sketch Elements

Individual control points can be moved by calling PUTSKETCHELEMENTPROP with the appropriate property; e.g. 1STCONTROLPT, 2NDCONTROLPT, 3RDCONTROLPT or DATA. To move a collection of elements, you can use the following function.

(SKETCH.MOVE.ELEMENTS ELEMENTS DELTA SKETCHTOUPDATE ADDHISTORY?)

Moves the elements ELEMENTS by the amount of position DELTA (XCOORD gives x amount, YCOORD gives y delta) and updates the viewers on SKETCHTOUPDATE if it is given. It does call the elements WHENMOVEDFN. When the WHENMOVEDFN is called, if SKETCHTOUPDATE is a window, it will be passed as the viewer; otherwise an open viewer onto it, or NIL if there is not open viewer, will be passed. If ADDHISTORY? is non-NIL, information about the move event that permits the event to be undone will be put on the viewer.

*** need way to perform transformations (2 and 3 pt) on elements.

Functions to use for writing Move functions

One useful capability for some applications is to restrict the places where the user can move certain elements. An example would be restricting the placement of a circle so that the center is on a line. To implement this, the sketch should be given a PREMOVEFN property of a function that prompts the user for a point imposing the desired constraints and returns the position. A useful subfunction for this is SKETCH.TRACK.ELEMENTS (described below) that tracks an image in the sketch window according to a user provided constraint function.

(SKETCH.TRACK.ELEMENTS ELEMENTS VIEWER CONSTRAINTFN HOTSPOT PROMPTMSG CONSTRAINTDATA)

ELEMENTS is a list of sketch elements. VIEWER is a sketch viewer. HOTSPOT is a position, usually with the region occupied by ELEMENTS. SKETCH.TRACK.ELEMENTS creates an image of ELEMENTS and causes it to follow the cursor with HOTSPOT being at the cursor position. Each time the cursor moves, CONSTRAINTFN is called with the new position of HOTSPOT in sketch coordinates, VIEWER and CONSTRAINTDATA as arguments. If it returns a position, the image will be displayed at that position rather than at the cursor position. If it returns anything else, the image will be positioned at the cursor. (It is ok for CONSTRAINTFN to smash its position argument but it must also return it to have the change noticed.) If HOTSPOT is NIL, the 1STCONTROLPT of the first element is used as the hotspot. When the user releases the button, SKETCH.TRACK.ELEMENTS returns the delta between the original position of the HOTSPOT and the position of the cursor. (This is the value that the PREMOVEFN needs to return to get Sketch to do the move.)

If PROMPTMSG is non-NIL, it will be printed in the sketch viewer status window while the image is tracking.

Example of constraining moves

The following example will constrain the Move command so that whole elements can only be placed on a diagonal line in which X=Y. It will not effect the point moving commands though it will effect Move Points commands in which all of the control points of at least one element have been selected. If there were used in an actual application, it would make more sense to group all the elements so that each had only one control point.

```
(* create a sketch viewer at a fixed location of the screen with a menu.)
(SETQ SKETCHVIEWER (SKETCHW.CREATE NIL NIL '(50 0 300 400) NIL NIL T))
(* define and set up the PREMOVEFN that will read a new position from the user restricting its position.)
(PUTSKETCHPROP SKETCHVIEWER 'PREMOVEFN 'POSITION.ON.DIAGONAL.PREMOVEFN)
(DEFINEQ
 (POSITION.ON.DIAGONAL.PREMOVEFN
   (LAMBDA (VIEWER ELEMENTS)
      (* Premovefn that reads a new position from the user restricting its position to be on a diagonal.)
     (SKETCH.TRACK.ELEMENTS (SK.PICKOUT.WHOLE.ELEMENTS ELEMENTS) VIEWER
(FUNCTION ONDIAGONAL))))
 (SK.PICKOUT.WHOLE.MOVE.ELEMENTS
   (LAMBDA (MOVEELTLST)
        (* returns from a list of sketch elements that are being moved, the ones that will be completely moved.  See the discussion about PREMOVEFN for a description of the format.)
     (COND
        ((EQ (CAR MOVEELTLST) T) (CDR MOVEELTLST))
        ((EVERY (CAR MOVEELTLST) (FUNCTION NUMBERP)) NIL)
        (T (for X in MOVEELTLST when (EQ (CAR X) T) collect (CDR X))))))
 (ONDIAGONAL
   (LAMBDA (PT VIEWER)
```

```
            (* the constraint function for POSITION.ON.DIAGONAL.  It makes sure the pt is on
a diagonal.)
     (replace (POSITION XCOORD) of PT with (fetch (POSITION YCOORD) of PT))
     PT)))

(* any moves done from SKETCHVIEWER will place the control point on the diagonal.)
```

To prompt the user for element positions

Some interactive applications may want to read values for some of the element from the user possibly while providing their own special purpose feedback. The following function provides for this.

(SKETCH.GET.POSITION VIEWER CURSOR FEEDBACKFN FEEDBACKFNDATA)

Reads a position in the sketch viewer VIEWER from the user. CURSOR if a cursor will be the cursor used while reading the position. Each time the cursor moves, FEEDBACKFN is called with the new X, the new Y, VIEWER and FEEDBACKFNDATA as arguments. It is expected to XOR something to VIEWER that tells the user where his point is going. Note: the FEEDBACKFN must work in window coordinates, NOT sketch coordinates. The functions VIEWER.TO.SKETCH.POSITION and SKETCH.TO.VIEWER.POSITION provide ways of mapping from sketch to viewer coordinates. The function (WINDOW.SCALE VIEWER) returns the number to multiply a window value by to get a sketch coordinate value.

To let the user select a collection of elements

Some interactive applications may want to allow the user to specify a collection of elements. The following function is provided for this.

(SKETCH.GET.ELEMENTS VIEWER SINGLEELEMENTFLG WHICHONES)

High lights the elements in VIEWER and allows the user to select one or more of them. If SINGLEELEMENTFLG is non-NIL, the protocol will only allow the user to select a single element; otherwise, the user can select multiple elements by using the protocol described in the Sketch manual. WHICHONES effect the elements which are high lighted. If it is one of the atoms MOVE, COPY, DELETE, CHANGE, GROUP, UNGROUP, COPYSELECT, or T, only elements that do not have that atom on their PROTECTION property will be selected. If WHICHONES is a list of elements, only the elements on that list will be high lighted.

5. PROPERTIES OF SKETCH ELEMENTS

A sketch element has a set of properties. Some of these are common to all elements e.g., TYPE. Some are common to a subset of elements e.g., FONT which only exists for TEXT and TEXTBOX elements. Any properties not listed here can be used to associate information with elements. Putting a new value for the property of an element is the standard way of changing elements.

(GETSKETCHELEMENTPROP SKETCHELEMENT PROPERTY)

Returns the value of the property PROPERTY from the sketch element SKETCHELEMENT.

(PUTSKETCHELEMENTPROP SKETCHELEMENT PROPERTY VALUE SKETCHTOUPDATE)

Saves VALUE as the property PROPERTY from the sketch element SKETCHELEMENT. It returns the previous value. If SKETCHTOUPDATE is a sketch in which the element is a member, any viewers onto it will be updated.

Sometimes it is useful to make several changes to an element or elements and then change the display. The following function provides this capability.

(SKETCH.UPDATE SKETCH ELEMENTS)

Updates the display of elements ELEMENTS in SKETCH. It does this by erasing and redrawing them. If ELEMENTS is NIL, it updates the SKETCH by clearing its viewers and redrawing all of the elements. If many of the elements in a sketch have changed, it may be faster to update the whole sketch than to erase and redraw the changed elements. The application must make this determination. Another use of this function is to notify sketch that the image object in an image object element has changed.

The following properties are recognized by Sketch. Any others will be stored and returned but will not effect Sketch. Note: not all elements have all of the properties.

Control properties

TYPE One of TEXT, TEXTBOX, WIRE, CLOSEDWIRE, OPENCURVE, CLOSEDCURVE, CIRCLE, ARC, ELLIPSE, BOX, SKIMAGEOBJ, BITMAPELT or GROUP. This can not be changed.

SCALE The scale at which an element was created. For GROUP elements, this returns NIL.

REGION (***Not implemented yet. It is planned to be the region occupied by the element's image. Tell the implementor if you are in need.)

ACTIVEREGION The region the sketch's button event fn watches. The active regions of elements that are contained in a group are not watched, only the top-level group element's active region is.

POSITION (also **1STCONTROLPT**) The first point for wires and curves; center for circle, arcs, ellipses; lower left for boxes and text boxes; the control point for bitmaps, text, groups and image object elements.

2NDCONTROLPT The second point for wires and curves; radius point for circle, arcs and ellipses. NIL for any other elements.

3RDCONTROLPT The third point for wires and curves; angle point for arcs; minor radius point for ellipses. NIL for any other elements.

DATA The bitmap or image object for bitmap and image object elements, the list of points for wires and curves, the text for text and textboxes, list of elements for groups.

PROTECTION (MOVE COPY DELETE CHANGE GROUP UNGROUP COPYSELECT) or T = protect against everything except COPYSELECT. (COPYSELECT T) will protect against everything. Note: having the token on the property list means the operation will NOT be allowed. FROZEN has the same effect as T and is used by the sketch commands Freeze and Unfreeze so that is doesn't interfere with the user set value T. Thus, if you want the element originally frozen but want to allow the user to be able to unfreeze it, use the atom FROZEN. If a text or textbox element is protected against CHANGE, pressing the left button in its characters will not be recognized as a selection.

Graphic properties

BRUSH The brush for elements that have one. NIL for other elements.

FILLING The filling for an element that has an enclosed area. (See the description of a filling in the section on the Specifications of data structures for Sketch properties.) Fillings for ellipses or closed curves are not implemented yet.

DASHING The dashing list for elements that have one.

ARROWHEADS For elements that have arrowheads (WIRE OPENCURVE ARC), a list of two arrowhead specifications: the first is for the first end of the element, the second is for the last end of the element.

Text properties

FONT For text and textbox elements, the font that the text appears in.

JUSTIFICATION For textbox elements, the justification that the text has within the box. For text elements, the justification that the text has relative to its control point.

TEXTCOLOR The color that the text is displayed in.

Angle properties

DIRECTION The direction property of arc elements.

Specifications of data structures for Sketch properties

A **brush** is defined by the record BRUSH. It has fields BRUSHSHAPE, BRUSHSIZE and BRUSHCOLOR. Shape is one of ROUND, SQUARE, HORIZONTAL, VERTICAL or DIAGONAL. ROUND is the default. Size is a number in units of screen points. The color is a legal color, i.e. color name, RGB triple, or HLS triple. The BRUSH argument to any of the element creation functions has the following interpretations. NIL is taken to be SK.DEFAULT.BRUSH. A number is a round brush of that size.

A **dashing** is a list of numbers interpreted as number of points on, number of points off, etc.

A **font** to sketch is a list of (family size face).

A **filling** is a list of (texture color). Texture is either a small number, a 16 by 16 bitmap or NIL. Color is a legal color or NIL. (At present, if color is non-NIL, texture is ignored. It is expected that a standard way of displaying colors as textures on the black and white screen and allowing textures on the color screen will be developed.) The FILLING argument to any of the element creation functions has the following interpretations. NIL means texture of SK.DEFAULT.TEXTURE (originally NIL meaning no texture) and a color of SK.DEFAULT.COLOR (originally NIL meaning WHITE). A texture (i.e. TEXTUREP returns T) is taken to be texture of the given value and SK.DEFAULT.BACKCOLOR as a color. A texture that is a color is taken to be texture of SK.DEFAULT.TEXTURE and a color of the given value.

An **arrowhead** specification is defined by the ARROWHEAD record and has fields ARROWTYPE, ARROWANGLE, and ARROWLENGTH. T means use an arrowhead with the default characteristics. ARROWTYPE is one of CURVE, LINE, SOLID or CLOSEDLINE, ARROWANGLE is the number of degrees and ARROWLENGTH is the length of the sides. The default is controlled by the variables SK.DEFAULT.ARROW.LENGTH (initially 8), SK.DEFAULT.ARROW.ANGLE (initially 18.0) and SK.DEFAULT.ARROW.TYPE (initially CURVE).

A **justification** specification is a list whose first element is the horizontal position of the text within the region (one of LEFT, CENTER, RIGHT) and whose second element is the vertical position (one of TOP, CENTER, BASELINE, BOTTOM).

A **direction** specification is one of atoms CLOCKWISE or COUNTERCLOCKWISE.

6. MISCELLANEOUS INFORMATION ABOUT SKETCHES

To save a sketch on a file

A sketch can be saved on a file by making it an image object and either adding the image object to the COMS of a file using an UGLYVARS command or HPRINTing it directly.

Changing the Sketch command menu

(SKETCH.COMMANDMENU ITEMS)

creates a sketch command menu that contains ITEMS.

(SKETCH.COMMANDMENU.ITEMS ADDFIXITEM ELEMENTTYPES)

Returns the list of items used in the standard Sketch menu. If ADDFIXITEM is non-NIL, the list will contain an item which fixes the menu at the right side. If ELEMENTTYPES is NIL, no element adding commands will be included in the list. If ELEMENTTYPES is T, a list of all known element types, SKETCH.ELEMENT.TYPE.NAMES, is used. Otherwise, ELEMENTTYPES should be a list of the names of the element types that are included in the menu.

Replacing the menu: If the BRINGUPMENUFLG argument to SKETCHW.CREATE is a menu, it is used as the command menu. The function SKETCHW.SELECTIONFN is the Sketch WHENSELECTEDFN.

(SKETCHW.SELECTIONFN ITEM MENU)

It performs some housekeeping for Sketch and then applies the second element of ITEM to the sketch window. More specifically, if the CADR of ITEM is an atom, it is called with the sketch window as its first argument. If It is a list, it is evaluated with the sketch window being added as another argument on the end. For example, if the ITEM is (Run (DO.RUN 'FOREVER) "help string"), DO.RUN will be called with a first argument of FOREVER and a second argument of the sketch window.

To make an image object from a sketch

(MAKE.IMAGE.OBJECT.OF.SKETCH SKETCH REGION SCALE GRIDSIZE)

Returns a sketch image object. REGION is the region in sketch coordinates that the image object will show. SCALE is the scale at which it will be shown. GRIDSIZE is the grid size of the sketch. If SKETCH is a viewer, any of the other arguments that are NIL will be filled in from the values in the viewer. If SKETCH is a sketch, REGION defaults to the extent of the sketch, SCALE defaults to 1.0 and GRIDSIZE defaults to 8.0.

Note: SKETCHW.CREATE will take an image object as its sketch argument and use the region, scale and grid fields from it unless they are provided in the function call.

6. SKETCH STREAMS

Sketch streams provide a way of creating a sketch from the display output of a program. This may be useful if you want to include the graphic output of a program in a document but do not need to monitor any changes the user may make to the sketch. Their advantage is that no changes need be made to the program producing the image. To use them, load the file SKETCHSTREAM. A sketch stream is a type of image stream and can be used as the stream argument for printing and drawing functions. On the display, it will look like another window. The difference is that a sketch is being constructed so that the image can be edited using the standard Sketch interface and/or copy selected into a document.

To open an image stream

(OPENIMAGESTREAM 'name 'SKETCH options)

Opens a window on the screen and returns a sketch stream that displays in that window. If the options *REGION* is given, it will be used as the region of the screen where the window is opened. If *REGION* is not given, the user will be prompted for a region. If the option *SKETCHREGION* is given, it will be used as the region of the sketch that will be visible in the window. If it is not given, the sketch region will be *(0 0 WindowWidth WindowHeight)*. By giving a *SKETCHREGION* that is larger or smaller than *REGION*, the displayed image can be scaled.

Examples:

```
(OPENIMAGESTREAM 'dummy 'SKETCH)
```

prompts the user for a screen region, opens a window there and returns a sketch stream that displays in that window.

```
(OPENIMAGESTREAM 'mysketch 'SKETCH '(REGION (100 200 300 200)))
```

opens a sketch window in the screen region (100 200 300 200) and returns a sketch stream that displays in it.

```
(OPENIMAGESTREAM 'enlarged 'SKETCH
          '(REGION (50 50 200 200) SKETCHREGION (0 0 100 100)))
```

Opens a sketch window in the screen region (50 50 200 200) and returns a sketch stream onto it. Anything displayed on the sketch stream will appear on the screen at approximately twice size. (It is only approximate because REGION is the exterior of the window while SKETCHREGION is the region of the sketch that maps into the interior of the region.)

Limitations: BITBLT and BLTSHADE ignore the operation and clipping region arguments.

To get the editing menu for the sketch window created by OPENIMAGESTREAM, middle button in the title bar and select the item "Fix Menu".

8. ADDING NEW SKETCH ELEMENT TYPES

********** This section is incomplete and out of date; see me if you need this.

This section describes for adventurous people, how to add new sketch element types. Few applications will need to do this as most capabilities are more easily provided by groups existing types or defining image objects.

Sketch element types

The initial element types (stored on the list SKETCH.ELEMENT.TYPE.NAMES) are TEXT, WIRE, CLOSEDWIRE, OPENCURVE, CLOSEDCURVE, CIRCLE, ELLIPSE, ARC, BOX, TEXTBOX, BITMAP, SKIMAGEOBJ and GROUP.

An element is broken into two basic collections of data: a global part which contains all of the information about an element in the sketch and a local part that contains all of the information about an element displayed in a viewer. The global part contains: a type, a minimum scale, a maximum scale, a property list and an individual global part that has data that depends upon the type. The minimum and maximum scale give the range of scales in which this element will be displayed. The global and local parts depend on the element type. The global part is a TYPERECORD that describes the element in terms of a global coordinate system. The local part contains information particular to the displaying of the element in a particular window. This is broken down into three fields: HOTSPOTS, the local coordinates of parts of the element that can be selected, LOCALHOTREGION, the active part of the element, and OTHERLOCALINFO, which contains other information such as the local radius of a

circle. The information stored in the local part is intended solely for efficiency and is recreateable from the global part.

The following function is used to create new sketch element types:

CREATE.SKETCH.ELEMENT.TYPE(SketchType Label DocStr DrawFn ExpandFn Obsolete ChangeFn InputFn InsideFn RegionFn TranslateFn UpdateFn ReadChangeFn TransformFn TranslatePtsFn GlobalRegionfn)

Creates a new sketch element of type SketchType. There should be a TYPERECORD of type SketchType.

Label is used as the label in the sketch command menu that when selected will create a new element (by calling its InputFn - see below). It can be a string or a bitmap.

DocStr is the help string put on the item in the sketch command menu.

DrawFn is the elements drawing function. It is called with three arguments: the element (an instance of SCREENELT), the sketch window, and the region of the window to be displayed.

ExpandFn is a function that creates the local field values for an element. It is called with two arguments: the global element and the sketch window. It should return an instance of the record LOCALPART that contains the local information to display element in the sketch window. One of the fields of a LOCALPART is HOTSPOTS, a list of points in the local coordinate system of window that can be used to select this element. The positions on this list will be passed back to the elements "move function" (see below) so the element needs to know the mapping between these points and its internal structure; ie that the first point is the center and the second point is the radius point. Each element should have at least one point on HOTSPOTS so it can be selected for deletion. The LOCALHOTREGION will be filled in by the system. The OTHERLOCALINFO part can be whatever information is useful for this element type.

InputFn is an inputting function. It is called with one arguments: the sketch window. It should read the information necessary to create an instance of the element from the user via mouse, keyboard, etc. and return the global part of the element (instance of GLOBALPART).

Changefn: A changing function (element window). Typically gives the user a menu of things about element that can be changed and makes the changes. It returns a list of lists of the form: the new element, the property changed, the new value of the property and the old value of the property.

InsideFn: An extent predicate INSIDEP(globalElement globalRegion) which returns T if globalElement is inside of the region globalRegion.

RegionFn:

TranslateFn:

UpdateFn:

ReadChangeFn:

TransformFn:

TranslatePtsFn:

GlobalRegionFn: