DSPSCALE

By: Christopher Lane (Lane@Sumex-Aim.Stanford.Edu)

DSPSCALE allows a program to output to different types of streams (display, Interpress, etc.) without corrections for scaling. This module provides self-scaling graphics through two different methods: a virtual self-scaling *image stream* that overlays a regular image stream and/or new versions of the various image stream graphic manipulation functions (DRAWLINE, DSPTOPMARGIN, etc.). The goal of both methods is to make it possible to modify the normal scaling factor of an image stream without modification to the program generating the output.

VIRTUAL SELF-SCALING IMAGE STREAM

This module implements a virtual image stream type, called SCALED, which is used to overlay any other image stream and provide automatic scaling to the natural scale of the image stream or any user selected scaling factor. The function OPENIMAGESTREAM is used to overlay a scaled image stream over a regular one. For example, the following will open a scaled image stream on top of an Interpress image stream:

```
(OPENIMAGESTREAM (OPENIMAGESTREAM 'TEST.IP 'INTERPRESS) 'SCALED)
```

The only difference between the virtual stream and a normal image stream is that the SCALE argument to the DSPSCALE function is active and can be used to change the scale of the stream (multiplying the scale specified by the standard scaling factor of the stream).

SELF-SCALING GRAPHICS FUNCTIONS

As an alternative to the self-scaling image stream, self scaling versions of the various graphics functions are provided. For most of the graphic functions, self-scaling versions have been defined which have an **!** (exclamation point) at the end of their name (eg. DRAWLINE vs. DRAWLINE**!**):

```
   CENTERPRINTINREGION!        DRAWELLIPSE!          DSPSCALE!
   CHARWIDTH!                  DRAWLINE!                RELDRAWTO!
   CHARWIDTHY!                 DRAWPOINT!               RELMOVETO!
   CURSORPOSITION!             DRAWPOLYGON!             SCALEDBITBLT!
   BITBLT!                     DRAWTO!                  STRINGREGION!
   BITMAPBIT!                  FILLCIRCLE!              STRINGWIDTH!
   BLTSHADE!                   FILLPOLYGON!          DSPSPACEFACTOR!
DSPBACKUP!                     FONTPROP!             DSPTOPMARGIN!
DSPBOTTOMMARGIN!               GETPOSITION!          DSPXOFFSET!
DSPCLIPPINGREGION!          DSPLEFTMARGIN!           DSPXPOSITION!
   DRAWARC!                 DSPLINEFEED!             DSPYOFFSET!
   DRAWBETWEEN!                MOVETO!               DSPYPOSITION!
   DRAWCIRCLE!                 MOVETOUPPERLEFT!
   DRAWCURVE!               DSPRIGHTMARGIN!
```

The set includes both output *and* input functions since it is necessary when getting, for example, a mouse position, to unscale the position to put it back into the program's virtual coordinate system. By default, these functions can be used directly in place of their non-**!** counterparts and they will automatically scale their arguments to the DSPSCALE of the output stream. Some of the above functions are not identical with their non-**!** counterparts, as explained below:

(DSPSCALE! *SCALE STREAM*) [Function]

In this version of DSPSCALE, the *SCALE* argument is active and will multiply *STREAM*'s normal scaling factor. If you have a program that draws a circle, for example, to a window using the appropriate **!** functions, you can cause it to draw a different size circle (larger or smaller) by using DSPSCALE! to change the scaling factor of the window without touching the source program.

(CHARWIDTH! *CHARCODE FONT* ***STREAM***) [Function]

(CHARWIDTHY! *CHARCODE FONT* ***STREAM***) [Function]

(FONTPROP! *FONT PROP* ***STREAM***) [Function]

(STRINGWIDTH! *STR FONT FLG RDTBL* ***STREAM***) [Function]

All of the above functions have one extra argument (as compared to their non-**!** equivalents) which is the *STREAM* in question. This is necessary to do the scaling calculations.

The module also defines a couple of new stream manipulation functions:

(DSPTRANSLATE! *Tx Ty STREAM*) or (DSPTRANSLATE! *POSITION STREAM*) [Function]

Defines the amount of X and Y translation that should be added to graphic operations to *STREAM*. Similar to DSPTRANSLATE but works even if the image stream does not have an IMTRANSLATE method. The second form of the arguments is for backward (Koto) compatibility.

(DSPUNITS! *UNITS STREAM*) [Function]

Essentially the inverse of DSPSCALE!, this function lets you set how many *UNITS* (pixels or whatever) the source program generates for each unit pixel on the output stream (multiplied by the output stream's default scaling).

It is possible to use both the virtual image stream and the self-scaling graphics functions together as long as the self-scaling graphics functions are applied to the real stream, not the virtual one.

Lisp Data Type Scaling Functions

The routines below are used by the **!** functions and the virtual image stream for scaling numbers, positions, regions and other data types and are useful for defining other self-scaling functions:

(DSPSCALE.BRUSH *BRUSH STREAM*) [Function]
(DSPSCALE.DASHING *DASHING STREAM*) [Function]
(DSPSCALE.POINTS *KNOTS STREAM*) [Function]
(DSPSCALE.REGION *REGION STREAM [SmashRegion]*) [Function]
(DSPSCALE.NUMBER *NUMBER STREAM*) [Function]
(DSPSCALE.POSITION *POSITION STREAM [SmashPosition]*) [Function]
(DSPSCALE.XPOSITION *NUMBER STREAM*) [Function]
(DSPSCALE.YPOSITION *NUMBER STREAM*) [Function]
(DSPSCALE.WIDTH *WIDTH STREAM*) [Function]

(DSPUNSCALE.REGION *REGION STREAM [SmashRegion]*) [Function]
(DSPUNSCALE.POSITION *POSITION STREAM [SmashPosition]*) [Function]
(DSPUNSCALE.NUMBER *NUMBER STREAM [OFFSET]*) [Function]
(DSPUNSCALE.XPOSITION *NUMBER STREAM*) [Macro]
(DSPUNSCALE.YPOSITION *NUMBER STREAM*) [Macro]