MapPt

Map point relative to two rectangles

#include <Quickdraw.h> **Quickdraw**

void	**MapPt(***thePoint*, *srcRect*, *destRect* **);**	
Point	***thePoint** ;	address of point to map; receives result
Rect	***srcRect** ;	address of Rect to convert from
Rect	***destRect** ;	address of Rect to convert to

MapPt maps a point within one rectangle to a similarly-located position in a different rectangle. Use this to scale individual points of an object being moved to a larger or smaller rectangle.

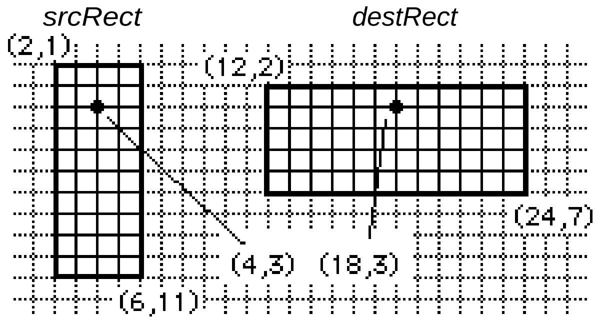
thePoint is the address of a 4-byte Point structure. On entry, it is the coordinates of a point, relative to *srcRect* that you wish to convert; upon return, it contains the coordinates of a point relative to the size and position of *destRect* .

srcRect and . . .
destRect are the addresses of two 8-byte Rect structures. For typical operations, *thePoint* is an element of an object enclosed by *srcRect* . It gets mapped to a similar position within *destRect* .

Returns: none

Notes: This function is typically used to convert individual points of an object (e.g., a "freehand" drawing) within one rectangle to similar positions within a larger or smaller rectangle. Other tools exist to scale rectangles (**MapRect**), regions (**MapRgn**), and polygons (**MapPoly**).

For instance, a corner of *srcRect* will map exactly to the corresponding corner of *destRect* ; similarly, the center of *srcRect* maps to the center of *destRect* . Other points will be positioned at distances from the edges relative to the ratio of the sizes of the rectangles.



It is OK if the two rectangles overlap, and *thePoint* need not be enclosed by *srcRect* (in that case, its remapped position will be outside of *destRect*).

This call is functionally equivalent to the long-winded:

```
h1=r1.bottom-r1.top;  h2=r2.bottom - r2.top;       /* calc heights */
w1=r1.right-r1.left;  w2=r2.right - r2.left;       /* and widths */

thePoint.h = (thePoint.h * w1) / w2;         /* apply ratio of sizes */
thePoint.v = (thePoint.v * h1) / h2;

thePoint.h += (r2.left - r1.left);          /* move to position ...*/
thePoint.v += (r2.top - r1.top);            /* ... relative to destination */
```