SectRect Get the intersection of two rectangles; check for overlap

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

Boolean	**SectRect(***rect1*, *rect2*, *resultRect* **);**	
Rect	***rect1** ;	addresses of two Rect ...
Rect	***rect2** ;	... structures to intersect
Rect	***resultRect** ;	receives intersection rectangle
	returns	Do the rectangles have any pixels in common?

SectRect checks to see if two rectangles share any points in common and obtains the coordinates of the intersecting rectangle

rect1 and . . .
rect2 are addresses of two 8-byte Rect structures. They should use the same coordinate system.

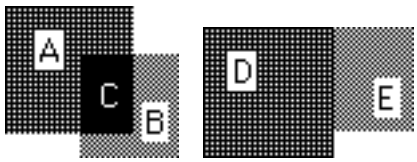
resultRect is the address of a third Rect structure. Upon return, it will contain the coordinates of the intersection of *rect1* and *rect2* . If *rect1* and *rect2* share no common points, this returns as the empty rectangle (0,0)(0,0).

Returns: a Boolean value indicating whether or not *rect1* and *rect2* share any common points. It is one of:

FALSE No overlap

TRUE Some overlap; *resultRect* is non-empty.

Notes: The figures below illustrate:



In the figure on the left, the intersection of rectangles **A** and **B** is shown in black, labeled as rectangle **C**. The call to **SectRect** will return TRUE, and *resultRect* will contain the coordinates of rectangle **C**.

The figure on the right illustrates an important concept: a shared border or a single shared point are not considered as part of the mathematical intersection of two rectangles. Rectangles **D** and **E** are right next to each other and share a common border. However, their "overlap" area is a single mathematical line, enclosing no bits of the bitMap. The call to **SectRect** would return FALSE, and *resultRect* would be set to (0,0)(0,0).

It is OK to specify *resultRect* as the same rectangle as either of *rect1* or *rect2* . For instance, the following line would find the intersection of rectangles A and B, and store its coordinates into rectangle A, overwriting the original value.

```
SectRect( &rA, &rB, &rA );
```