CalcCMask Calculate a mask for use in CopyMask

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

void **CalcCMask(** *srcMap*, *destMap*, *srcRect*,*destRect*, *seedRGB*, *matchProc, matchData* **);**

BitMap	***srcMap** ;	addr within a BitMap of place to start calculating
BitMap	***destMap** ;	addr within a BitMap of where to store 1s and 0s
Rect	***srcRect** ;	size, location of data from source BitMap
Rect	***destRect** ;	size, location to store data in destination
RGBColor	***seedRGB** ;	RGB value of seed
ColorSearchProcPtr	*matchProc ;*	custom matching routine; NIL = default
long	*matchData*	application specific data

CalcCMask examines a portion of a bitmap and creates a mask that can be used with **CopyMask**. It finds the outermost outline of any figure in the bitmap (as in the lasso tool of many paint programs) and creates a mask having 1s in the places where paint would "leak" were you to pour it inside the figure. The object of this procedure is the reverse of **SeedCFill** in that it protects areas from being filled with color.

srcMap is the address (srcBits) of a rectangle *inside* a BitMap or PixMap data area. **CalcCMask** examines this rectangle as it floods portions of the destination bitMap.

destMap is the address (destBits) of a rectangle *inside* a BitMap or PixMap data area. **CalcCMask** fills all or part of this rectangle with 1s.

***srcRect** and . . .
***destRect** are the rectangles within the BitMap or PixMap into which *srcMap anddestMap*, respectively point; i.e., the function will add this value to its current address pointer to move "down one line" in the BitMap.

seedRGB is the RGBvalue of the pixel from which calculations begin.

matchProc is a pointer to a custom routine to be used instead of the default searchProc. It should return 1's for RGB values that define the edges of the mask and 0's for values that do not. Pass NIL in this parameter to use the default searchProc. See **Custom Search and Complement Procedures** for more on writing a custom searchProc.

matchData holds application specific data. Pass 0 in this parameter to obtain the default behavior.

Returns: none

Notes: Use **CalcCMask** to build a mask which can be used in conjunction with **CopyMask** to implement the "lasso" tool of many paint programs. **CalcCMask** creates an area into which paint won't flow. By default, paint can leak to all adjacent pixels whose RGB values don't match that of the

seedRGB. The default behavior is obtained by setting *matchPro*c and *matchData* to NIL.

Alternatively, you could create your own version of **CalcCMask** by specifying a *matchProc* to use instead of the default searchProc. Have it return 1's for RGB values on the perimeter of the mask and 0's for all others. When *matchProc* is called the gDevice's GDRefCon field holds a pointer to a MatchRec record that has values for the red, green, and blue fields that, taken collectively, are the RGB values of the pixel at the seed location. In this structure, matchData is whatever value your application gave **CalcCMask** as the *matchData* parameter. If your application had passed a handle to a color table, you could have *matchProc* run a comparison to make sure the specified RGB matches a color in the table.

CalcCMask does not stretch or shrink the source or destination rectangles. You have to make sure they're the same size.

Calls to **CalcCMask** are not recorded in a Picture definition (See **OpenPicture**) and are not clipped to the current port.

The easiest way to work with this function is to create one or more BitMaps the size and shape of your work area or your window. See **SeedCFill** for an example of this memory-intensive technique.