WHAT IS THE CALLING SEQUENCE FOR RASTEROPS?

The top-level function *pixRasterop()* should be thought of as the application programmer's interface (API) for rasterop. It is in essence a shim that accepts pointers to the PIX data structure for the images, along with a description of the rectangular areas to be operated on. It extracts the image data, decides if the operation is a unary or binary rasterop, and passes the image data to the appropriate low-level routine.

```
(9 args)    pixRasterop(PIX  *pixd,
                        INT32 dx, INT32 dy, INT32 dw, INT32 dh,
                            INT32 op,
                        PIX *pixs,
                        INT32 sx, INT32 sy);
```

If the operation is unary, the last three arguments of *pixRasterop()* are ignored, and *rasteropUniLow* is called with 10 arguments. The width is scaled and the rectangle is clipped if necessary, and the remaining 7 arguments are passed to either*rasteropUniWordAlignedLow()* or to the more general function *rasteropUniGeneralLow()*. These does the work.

If the operation is binary, all the arguments in *pixRasterop()* are used, and *rasteropLow()* is called with 16 arguments:

```
(16 args)   --> rasteropLow(UINT32 *datad,
                            INT32 dpixw, INT32 dpixh, INT32 depth, INT32 dwpl,
                                INT32 dx, INT32 dy, INT32 dw, INT32 dh,
                                INT32 op,
                                UINT32 *datas,
                            INT32 spixw, INT32 spixh, INT32 swpl,
                                INT32 sx, INT32 sy);
```

At this point, two simple operations are carried out immediately:

1. The depth of the image is set to 1 and the width is scaled appropriately. The rectangle size and left edge are also scaled. This generalizes the binary rasterop so that it can be used on an image of any depth.

2. The rectangle is clipped to both the *src* and *dest*. This is done to minimize computation and prevent any operation beyond the array bounds. Sun's rasterop macros used a bit that determined whether or not clipping was enabled, presumably because they were typically "blitting" small character images for which no overflow checking was needed and they were taking all

possible measures to save compute cycles. We ignore this flag bit and always clip the rectangle(s).

After width scaling and clipping (if necessary), the left edge alignment of the rectangle in both src and dest are compared. We distinguish three cases:

1. If they are both aligned on a 32-bit word boundary, the simple function *rasteropWordAlignedLow()* is called;
2. Else, if they have the same relative 32-bit alignment, the function *rasteropVAlignedLow()* is called;
3. Otherwise, the most general function *rasteropGeneralLow* is called.

Each of these three functions has the same 11 arguments; e.g.,

```
(11 args)      --> rasteropGeneralLow(UINT32 *datad,
                                      INT32 dwpl,
                                                  INT32 dx, INT32 dy, INT32 dw, INT32
dh,
                                                  INT32 op,
                                                  UINT32 *datas,
                                                  INT32 swpl,
                                                  INT32 sx, INT32 sy);
```

Due to the width adjustment and clipping that are performed in *rasteropLow()*, the low-level function that does all the work, *rasteropGeneralLow()*, does not need five of the arguments passed to *rasteropLow()*; namely, the width, height and depth of the*dest* and the width and height of the *src*.

Note that because *rasteropLow()* does width adjustment and clipping, it is safe to call it directly with an arbitrary image depth and un-clipped rectangular regions. It also makes it easier to call these low-level functions using a different high-level shim that uses some other packed image data structure. This separation between high-level functions that use the Pix image data structure and low-level functions that use only built-in C data types makes it much easier to port the low-level functions to applications that use other image data structures.

Note on abbreviations:

- *bpp* bits/pixel. A binary image has 1 bpp.
- *ppi* pixels/inch. Yes, this non-metric measure shows our North American provincialism.

Source: http://www.leptonica.com/rasterops.html