Major changes in Release 5 of the X v11 Sample Server

Keith Packard

MIT X Consortium

This is an attempted summary of major changes to the sample server from release 4 to release 5. Unlike the last release, release 5 does not contain a large number of seemingly arbitrary changes. Porting an R4 based DDX to R5 should take a matter of minutes to hours, depending on how much font structure specific code your DDX contains. Naturally, this document is intended only as an aid to rather than a replacement for the server source code.

1. Font Structure Changes

This is the only area in the server which has seen major upheaval. The entire font mechanism has been redone from stem to stern. If you are fortunate to be using the mi text routines, and have simply implemented your own glyph blt routines (as both mfb and cfb have always done), the change is very minor. In prior releases, the glyphs were passed to the rendering routine in the old CharInfoRec, which contained the metrics of the glyph, an offset from the base of the bitmaps and a boolean value telling whether this glyph should be rendered at all. The offset and boolean value were packed together into a single 32 bit word, 24 bits for the offset and 8 bits for the boolean. Yuck. R5 has redone this by simply placing a pointer to the bitmap data in the structure; a NULL pointer has the obvious meaning of a non-existant glyph. So, the extra argument passed to the glyph blt routines is no longer necessary; however I envision a future change to the interface which will require an argument, so we'll leave it in for this release.

If you are not fortunate, and have implemented your own PolyText/ImageText routines, you have your work cut out for you. The font structures are all new. There are macros in include/dixfontstr.h which should help you figure out what to change. Good luck.

2. ClipNotify

There used to be a hack in mivaltree.c to register interest in the changes to a windows clip list. This hack has been legitimized and is now a Screen procedure. When specified, this procedure is called whenever the clip list to a window has changed. It is called after the new clip list has been installed on the window.

3. MI changes

MI has remained relatively constant. One change is new algorithms for rendering non-zero width arcs which are much easier to understand, and are somewhat faster. Other than that, the major changes involved integrating the font structure changes into the text helper routines.

4. CFB changes

CFB has been largely rewritten to take advantage of some new algorithms which make the source pool smaller while making rendering much faster in general. The only horrible change is that a small amount of assembly code was written to speed up polytext rendering. Assembly for sparc, mips and 68k processors exists; you should definately write routines for your hardware as the old horrible C code has been ripped out, and the current code is less than speedy. By the time R5 ships, I hope to have made CFB work on other than 8-bit displays; there is a small chance changes will be made to allow it to be compiled multiple times for different depths, allowing a poly-depth system to use cfb for all depths.

Reduced Rop is a system which allows a small number of functions to perform all combinations of rasterop, foreground pixel and planemask. By precomputing two values ("xor" and "and"), and by recompiling source files with different definitions, each raster op can run at full speed. The basic equation is simple:

(dst & and) ˆ xor

By proper computation of the two constants, this one equation performs all rasterops (for example, Copy mode is simple: and = 0, xor = fg). After computing the two values, a "reduced rop" is computed. The reduced rop can be used to select different algorithms for rendering; in particular the most common reduced rop is GXcopy - in this case, the rendering selected will avoid fetching the destination pixel. These reduced rops move the complications of discovering "equivalent" rops (like GXinverted and GXxor in many cases) into a single function.

Merged Rop is a system which performs bitblt rasterops in a single (rather large) equation, avoiding branches at the expense of a bit of boolean arithmetic. Even with the additional computation, this should run faster than the old code on any processor I can imagine. The basic equation is:

(dst & (src & and1 ˆ xor1) ˆ (src & and2 ˆ xor2))

The values and? and xor? are computed at the time the function is started; a precomputed array based on the raster op is merged with the planemask. As with the reduced rasterop code, special cases can be extracted to avoid some or all of this computation.

5. MFB changes

Little was done in MFB other than integrating the merge-rop code into the tiling and bitblt functions.

An interface change was made, however. In r4 the function mfbRegisterCopyPlaneProc was added to allow something other than miCopyPlane to be called for mismatched depths. Cfb used this function to speed up CopyPlane from PSZ to 1 bit pixmaps. In r4, calling this function was optional, a screen/depth pair which didn’t call was simply relegated to using miCopyPlane. In r5, this call has been made manditory. This removes the final reference to miCopyPlane from the server, eliminating the entire mibitblt.c file from the image. If you fail to call this routine when needed, your server will core dump the first time you copy an n-bit object to a bitmap.