LBX X Consortium Algorithms

Table of Contents

Introduction

The Low Bandwidth X extension allows for negotiating various algorithms used by LBX. This document describes the algorithms used in the Consortium implementation of LBX in the X11 Release 6.4.

Streaming Compression

LBX negotiates the use of a stream compressor. The consortium implementation defines a stream compressor named XC-ZLIB, which is based on the Zlib version 1.0 compression library by Gailly & Adler.

The XC-ZLIB compressor is presented with a simple byte stream - the X and LBX message boundaries are not apparent. The data is broken up into fixed sized blocks. Each block is compressed using zlib, then a two byte header is prepended, and then the entire packet is transmitted. The header has the following information:

```
out[0] (length & 0xfff) >> 8 | ((compflag) ? 0x80 : 0);
out[1] = length & 0xff;
```

If the compflag is false, then the contents of the block are not compressed.

Bitmap Compression

LBX also negotiates for bitmap compression. The consortium implementation defines a bitmap compressor named XC-FaxG42D, which uses the CCITT Group 4 2D compression algorithm.

Pixmap Compression

LBX allows for the negotiation of pixmap compression. The consortium implementation does not define a pixmap compression algorithm. (A run-length encoding algorithm was proposed, but experimentation proved it was less efficient than allowing the stream compressor to compress the image.

Colormap Algorithm

LBX negotiates for use of a colormap algorithm, used for color matching when the proxy allocates pixels in a grabbed colormap. The consortium implementation defines an algorithm named XC-CMAP. This algorithm consists of three parts, resolving to a hardware color, finding the closest existing color, and what free cell to allocate.

The XC-CMAP algorithm resolves a color to a hardware color in the following manner:

```
#define RESCALE(x, nbits) (x >> (16 - nbits)) * 65535 / ((1 << nbits) - 1)
#define GRAY(r, g, b) (30L * r + 59L * g + 11L * b) / 100

sigbits = pVisual->bitsPerRGB;

switch (pVisual->class) {
   case PseudoColor:
   case DirectColor:
   case StaticColor:
   /* rescale to rgb bits */
      *red = RESCALE(*red, sigbits);
      *green = RESCALE(*green, sigbits);
      *blue = RESCALE(*blue, sigbits);
      break;
   case GrayScale:
     /* rescale to gray then rgb bits */
     *blue = *green = *red = RESCALE(GRAY(*red, *green, *blue), sigbits);
      break;
   case StaticGray:
     /* rescale to gray then [0..limg] then [0..65535] then rgb bits */
     *blue = *green = *red = RESCALE(RESCALE(GRAY(*red, *green, *blue),
pVisual>numPixelBits), sigbits);
      break;
    case TrueColor:
      /* rescale to [0..limN] then [0..65535] then rgb bits */
      *red = RESCALE(RESCALE(*red, pVisual->numRedBits), sigbits);
      *green = RESCALE(RESCALE(*green, pVisual->numGreenBits), sigbits);
      *blue = RESCALE(RESCALE(*blue, pVisual->numBlueBits), sigbits);
      break;
  }
```

The XC-CMAP algorithm matches a color to an existing pixel in static visuals by finding the pixel with the lowest color match error, computed as follows:

```
error = errRed * errRed + errGreen * errGreen + errBlue * errBlue
```

The XC-CMAP algorithm selects a free pixel to allocate by selecting the free pixel with the lowest index from the free pixels known to the proxy. For direct visuals, it uses the lowest free or matching pixel subfield known to the proxy for each color.

Extensions

LBX allows for extensions to LBX to enable additional compression or short-circuiting. The consortium implementation does not define any extensions to LBX.