Understanding fbdev's cmap

These notes explain how X's dix layer uses fbdev's cmap structures.

*. example of relevant structures in fbdev as used for a 3-bit grayscale cmap

```
struct fb_var_screeninfo {
        .bits_per_pixel = 8,
        .grayscale      = 1,
        .red =          { 4, 3, 0 },
        .green =        { 0, 0, 0 },
        .blue =         { 0, 0, 0 },
}
struct fb_fix_screeninfo {
        .visual =       FB_VISUAL_STATIC_PSEUDOCOLOR,
}
for (i = 0; i < 8; i++)
        info->cmap.red[i] = (((2*i)+1)*(0xFFFF))/16;
memcpy(info->cmap.green, info->cmap.red, sizeof(u16)*8);
memcpy(info->cmap.blue, info->cmap.red, sizeof(u16)*8);
```

*. X11 apps do something like the following when trying to use grayscale.

```
for (i=0; i < 8; i++) {
        char colorspec[64];
        memset(colorspec,0,64);
        sprintf(colorspec, "rgb:%x/%x/%x", i*36,i*36,i*36);
        if (!XParseColor(outputDisplay, testColormap, colorspec, &wantedColor))
                printf("Can't get color %s\n",colorspec);
        XAllocColor(outputDisplay, testColormap, &wantedColor);
        grays[i] = wantedColor;
}
```

There's also named equivalents like gray1..x provided you have an rgb.txt.

Somewhere in X's callchain, this results in a call to X code that handles the colormap. For example, Xfbdev hits the following:

xc-011010/programs/Xserver/dix/colormap.c:

```
FindBestPixel(pentFirst, size, prgb, channel)

dr = (long) pent->co.local.red - prgb->red;
dg = (long) pent->co.local.green - prgb->green;
db = (long) pent->co.local.blue - prgb->blue;
sq = dr * dr;
UnsignedToBigNum (sq, &sum);
BigNumAdd (&sum, &temp, &sum);
```

co.local.red are entries that were brought in through FBIOGETCMAP which come directly from the info->cmap.red that was listed above. The prgb is the rgb that the app wants to match to. The above code is doing what looks like a least squares matching function. That's why the cmap entries can't be set to the left hand side boundaries of a color range.