Subject: Re: Shared maps and free colors, window positio
Posted by idl on Fri, 25 Mar 1994 07:17:35 GMT
View Forum Message <> Reply to Message

In article 2mpm1gINNli1@maz4.sma.ch, gga@otl.sma.ch (Gianmario Galli) writes:
> idlusers-news-gateway:
>
>
> I am using idl on a SUN SPARCstation 2 with 8bit color graphics.
> I am running idl V3.1.1 under X using the server X11.0, X11/NeWS.
>
> My questions:
>
> a) How can I get the current number of free colors (to be used in a shared
> color map) before activating the first window?

We have tried a lot of things to solve color flashing problems on using
different applications together which preserves colors cells. Especially the
Framemaker Openwinwindows Version 3.1 has an annoying bug. If the colors cells
116 - 256 are not yet preserved by another application FM 3.1 (only the
Openwindows Version!) preserves them and they are never freed -even after
exiting FM - until you restart Openwindows (The change to MOTIF will
solve this anyway ...). There is a nice tool on the net 'Colormap.c' (X contrib)
which shows whats exactly happen in the color management by the window
manager. We couldn't solve this special problem from within IDL neither
using the Xresources (Idl.colors: -10 or fixed value etc.) nor the IDL
!.d.n_colors or !d.table_size variables.
Finally I extracted and modified a part of the mentioned colormap code
to get a utility which returns the really free number of colors which can
be used by an idl application without color flashing. This it is possible at least
to inform the user if there are not enough colors available. I've included it
below as a shar archive. From inside idl it can be used this way:

IDL> spawn, ['get_nfre'], free_clrs, /NOSHELL
IDL> print, free_clrs
246 ..

I'm wondering how other users solve this problem, suggestions are welcome!

Thomas

#! /bin/sh
This is a shell archive, meaning:
1. Remove everything above the #! /bin/sh line.
2. Save the resulting text in a file.
3. Execute the file with /bin/sh (not csh) to create the files:
Makefile
get_nfre.c

```
# This archive created: Fri Mar 25 07:43:13 1994
export PATH; PATH=/bin:$PATH
if test -f 'Makefile'
then
 echo shar: will not over-write existing file "'Makefile'"
else
cat << \SHAR_EOF > 'Makefile'
# SunOS 4.x
get_nfre:


get_nfre:
 cc -O -I/usr/openwin/include -o get_nfre get_nfre.c \
       -L/usr/openwin/lib -lX11

# Using gcc on Solaris 2.x you don't have License troubles....
#Solaris 2.3:
#get_nfre:
# gcc -ansi -O -I/usr/openwin/include -o get_nfre get_nfre.c \
#        -L/usr/openwin/lib -lX11 -L/usr/ucblib -lucb
#




SHAR_EOF
fi # end of overwriting check
if test -f 'get_nfre.c'
then
 echo shar: will not over-write existing file "'get_nfre.c'"
else
cat << \SHAR_EOF > 'get_nfre.c'
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>

/* Soorce extracted and modified from colormap.c ( X-Contrib X11R5 ) */
Display     *display;
Visual      *vis;               /* Pointer to the current visual */
int         dep;                /* depth of the current visual */

Colormap    dcm, *icm;      /* Default colormap and nstalled cmaps */
Colormap    mycmap;             /* Local colormap */
XColor      colors[257];        /* Array for copying colors */
Bool        AllocatedCells[256];
Bool     MyAllocedCells[256]; /* array of my color cells */
int         dcms;               /* default colormap size */
int         numcells;           /* Number of colors in current visual */
```

```
Bool        truecolor = False;

Window      root;


Bool get_visual(depth)
    int    *depth;  /* The depth of the desired visual */
{
    XVisualInfo *visualList;
    long        vmask = VisualNoMask;
    XVisualInfo vinfo, *v;          /* visual information */
    int         big_map = 0, istat;
    Bool        success = False;
 /*---------------------------------------------------------- ----------*/
    visualList = XGetVisualInfo( display, vmask, &vinfo, &istat);

    switch (*depth) {

        case 4:
            for (v = visualList; v < visualList + istat; v++)
                if ((v->depth == 4)&&(v->class == PseudoColor)) {
                    vinfo = *v;
                    success = True;
                    break;
                }
            break;

        case 8:
            for (v = visualList; v < visualList + istat; v++)
                if ((v->depth == 8)&&(v->class == PseudoColor)) {
                    vinfo = *v;
                    success = True;
                    break;
                }
            break;

        default:
            for (v = visualList; v < visualList + istat; v++) {
                if ((v->colormap_size>big_map)&&(v->class==PseudoColor)) {
                    big_map = v->colormap_size;
                    vinfo   = *v;
                    success = True;
                }
                if (big_map >= 256) break;
            }

    }; /* end switch(depth) */
```

```
    if (success) {
        *depth   = vinfo.depth;
        vis      = vinfo.visual;
        numcells = vis->map_entries;
        return(True);
    }
    else
        return(False);

} /* end get_visual() */



int get_nfree_cols()
{
    int             i, xpos, ypos, index;
    int             cur_row, cur_col;
    int             NextNumColors, NumColors = 0, Pixels[256], Planes[16];
    Status          GotDemCells = False;
 /*------------------------------------------------------------ ----------*/
/*
 *  Assume all cells are allocated
 */
    for (i = 0; i < numcells; i++)
 AllocatedCells[i] = True;
/*
 *  Attempt to allocated as many cells as possible
 *  when we allocate then Pixels containes those cells that are free
 */
    NextNumColors = numcells;
    while (!GotDemCells) {

 NumColors = NextNumColors--;

 GotDemCells = XAllocColorCells(display, *icm, False, Planes,
          0, Pixels, NumColors);
 if (GotDemCells)
     XFreeColors(display, *icm, Pixels, NumColors, 0);

 if (NextNumColors < 1) {
     GotDemCells = True;
     NumColors = 0;
 }

    }
/*
 *  For each cell in Pixels that we were able to allocate
 *  set its slot in AllocatedCells to False
```

```
 */
   for (i = 0; i < NumColors; i++)
 AllocatedCells[Pixels[i]] = False;

   return(NumColors);

}


main( argc, argv)
   int  argc;
   char *argv[];
{
   char       cont;
   int        num;
   int        screen, index;
   char       Dashes[2];
 /*---------------------------------------------------------- ----------*/
/*
 *  Initialize the connection to the XServer and get some default
 *  values for a start
 */
   if ( ( display   = XOpenDisplay( NULL) ) == NULL ) {
      fprintf(stderr,"Couldn't open display!\n");
      exit(1);
   }

   screen    = XDefaultScreen( display);
   dcm       = DefaultColormap( display, screen);
   vis       = DefaultVisual( display, screen);
   root      = RootWindow( display, screen);
   dcms      = vis->map_entries;
   truecolor = (vis->class == TrueColor) ? True : False;
   dep       = DefaultDepth( display, screen);


   get_visual(&dep);

/*
 *  Currently, we are only dealing with either a 16 or 256 entry
 *  colortable, so rows and cols will be set to present a pretty
 *  colortable layout of 4x4 or 16x16
 */
   numcells = (numcells > 256) ? 256 : numcells;

   for (index = 0; index < numcells; index++) {
      colors[index].pixel = index;
      MyAllocedCells[index] = False;
```

```
 AllocatedCells[index] = False;

    }
/*
 *  Initialize my colormap
 */
    mycmap = XCreateColormap( display, root, vis, AllocAll);
    icm    = (Colormap *)XListInstalledColormaps( display, root, &num);
    XQueryColors( display, *icm,   colors, dcms);
    XStoreColors( display, mycmap, colors, dcms);

   printf("%d\n", get_nfree_cols() );
   XCloseDisplay( display);
}

SHAR_EOF
fi # end of overwriting check
# End of shell archive
exit 0
```