Programming and Examples 6

This chapter describes general programming issues relating to Solaris PEX PEXlib, and code example programs are provided at the end of the chapter.

Programming Notes

When you install Solaris PEX, the files you need for building and running PEXlib application are installed by default in subdirectories under `/usr/openwin`. Figure 6-1 shows the default locations of these files:

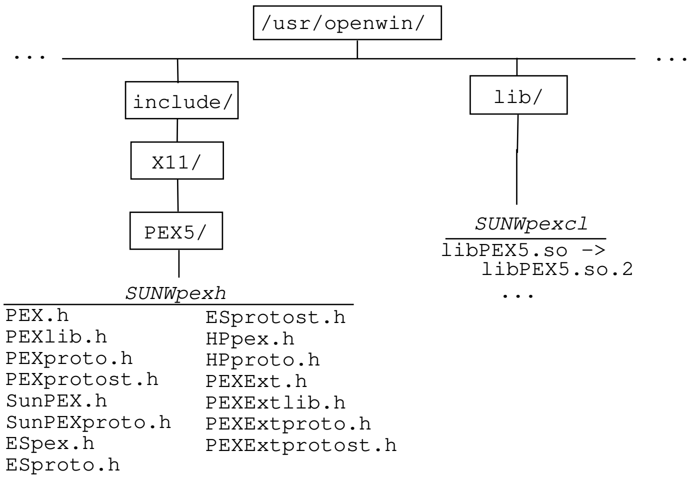

Figure 6-1 PEXlib Install Files Directory Tree

Library

You should link PEXlib programs you develop with the PEXlib shared-object library, `libPEX5.so` (as shown in Figure 6-1).

The `libPEX5.so` library is normally installed in `/usr/openwin/lib`, as shown. When you run PEXlib applications, make sure that this directory is included in your value of the environment variable `LD_LIBRARY_PATH`. For example:

```
% env | grep LD_LIBRARY_PATH
/usr/lib
% setenv LD_LIBRARY_PATH /usr/openwin/lib:$LD_LIBRARY_PATH
```

Header Files

When you write PEXlib applications, include the header files that define PEX data structures and constants. These files are installed by default in `/usr/openwin/include/X11/PEX5`:

- `ESpex.h`
- `ESproto.h`
- `ESprotost.h`
- `HPpex.h`
- `HPproto.h`
- `PEX.h`
- `PEXlib.h`
- `PEXproto.h`
- `PEXprotost.h`
- `PEXExt.h`
- `PEXExtlib.h`
- `PEXExtproto.h`
- `PEXExtprotost.h`
- `SunPEX.h`

- `SunPEXproto.h`

For simple PEXlib programs, you probably need to include only the header file, `PEXlib.h`. For programs using Escapes, GDPs, or GSEs, you might need to include one or more of the last eleven header files listed above. For more information about escapes, GDPs, and GSEs, please see Table 4-25 through Table 4-28 in Chapter 4, "Functional Description."

Online Reference Manual Pages

The Solaris PEX package includes the Solaris PEXlib man pages which describe the syntax for using the functions in the PEX Graphics Library. These man pages are located in `/opt/SUNWsdk/2.5/pex/man/man3`.

Example Programs

The Solaris PEX package includes several example programs for PEXlib, including example programs from the *PEXlib Programming Manual*. By default, these programs are installed in `/opt/SUNWsdk/2.5/pex/examples`:

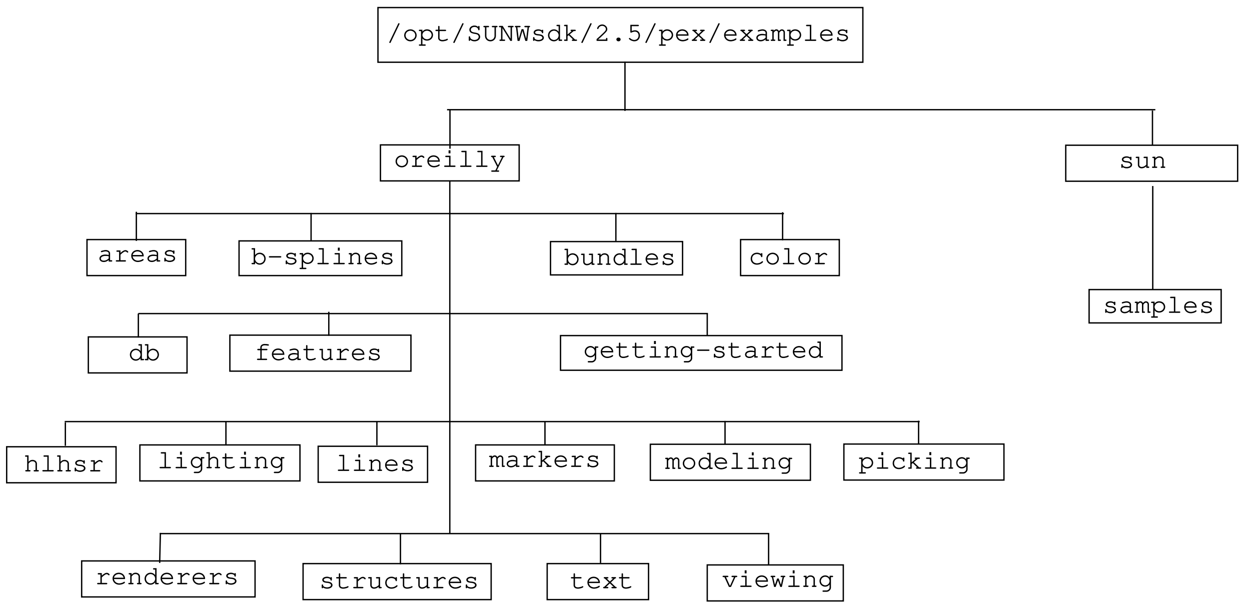

Figure 6-2 Example Programs Directory Tree

The example programs, as installed by default, are owned by the super user, root. To build these example programs, you must do either of the following:

- Make yourself "root" on your system
- Make modifications to makefiles and file protections and ownership in a copy of the directory tree or in place.

This chapter only shows how to build the example programs as root. Contact your system administrator for assistance, if necessary.

O'Reilly Examples

These are the examples from the *PEXlib Programming Manual*. To build them, `cd` to the `oreilly` directory, and issue the `make` command:

```
% cd /opt/SUNWsdk/2.5/pex/examples/oreilly
% make
```

This command makes a library of utilities, `libbook.a`, that is used by example programs under this subdirectory, and then makes the example programs.

Sun Examples

To build the Sun examples, `cd` to the `sun` directory, and issue the `make` command:

```
% cd /opt/SUNWsdk/2.5/pex/examples/sun
% make
```

Color Cube Performance Utility Example

For this example, you need only to `cd` to the `sun` subdirectory and issue the `make` command:

```
% cd /opt/SUNWsdk/2.5/pex/examples/sun
% make
```

This section discusses in detail the color cube performance utility example. This program is an example for how to work around a limitation of the Sun XGL library as used by Solaris PEX, and attain the best performance for 8-bit color windows on Sun platforms that support 8-bit visuals. To do this, the program ensures that there is a suitable colormap installed in this session on your server.

To ensure the best possible performance on 8-bit PseudoColor visuals on Sun frame buffers, you can run this example program before you run any other PEX applications. Or you can include similar code in the application you develop.

This example creates a colormap and allows this colormap to be shared with other applications running on the same server. The program is designed to accommodate two visual depths: 8-bit and 24-bit.

The call structure of the program is shown in Figure 6-3 as follows:

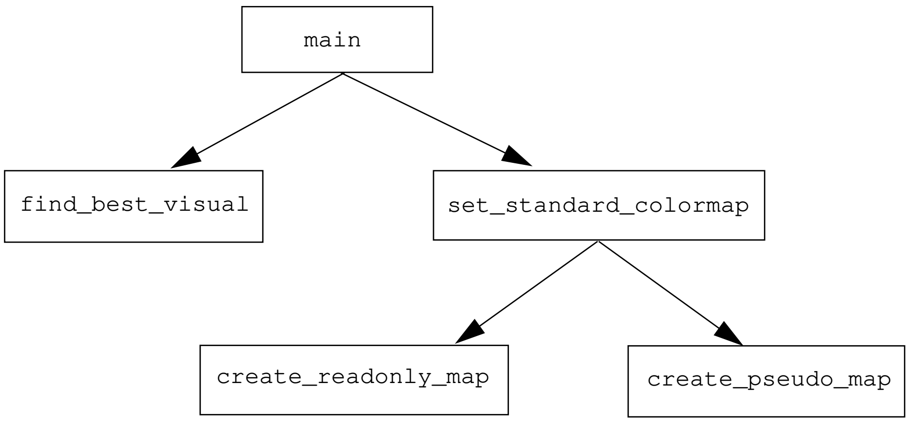

Figure 6-3 Call Structure

The command `main` determines the best visual to use by calling the subroutine `find_best_visual`. Then, it creates a color map based on that visual by calling the subroutine `set_standard_colormap`.

main

Code Example 6-1 Main Example

```
main(argc, argv)
int argc;
char *argv[];
{
    Display                     *dpy;
    XVisualInfo                 vis_info;

    dpy = XOpenDisplay();

    if ( !dpy ) exit(1);

    /* Determine the best visual to use.
     */
    if (!find_best_visual( dpy, &vis_info )) {
        fprintf(stderr, "Cannot find the best visual\n");
        exit(1);
    }

    /* create a standard colormap based on the chosen visual
     */
    if (!set_standard_colormap(dpy, &vis_info)){
        fprintf(stderr, "Cannot set standard colormap\n");
        exit(1);
    }

    /* set the close down mode to permanent so that the color
       resource will not be freed until the standard colormap
       property is deleted
     */
    XSetCloseDownMode(dpy, RetainPermanent);

    XSync(dpy);

}
```

find_best_visual

This routine is designed to handle only visuals of depth 8- or 24-bits. This routine assumes that a PseudoColor visual is best for 8-bit visuals and TrueColor for 24-bit visuals. It also assumes that `DefaultScreen` for your device has the same visual depth as you will be using; however, this may not be the case for devices with multiple visual depths. You might want to modify this subroutine to suit your own needs.

Code Example 6-2 Find the Best Visual Example

```
/* find the best visual
 */
static int
find_best_visual( dpy, chosen_vis )
    Display             *dpy;
    XVisualInfo         *chosen_vis;
{
    int                 err = 1;
    XVisualInfo         vinfo;

    /* Determine the best visual available. The best visual
       is the one with the most colors and highest capabilities
       and supported by PEX.
       This example assumes that PseudoColor visual is
       the best one on 8 bits device, and TrueColor visual is
       the best one on 24 bits device.
     */

    if (XMatchVisualInfo(dpy, DefaultScreen(dpy), 24, TrueColor,
                                &vinfo))
        *chosen_vis = vinfo;
    else if (XMatchVisualInfo(dpy, DefaultScreen(dpy), 8,
                                PseudoColor, &vinfo))
        *chosen_vis = vinfo;
    else
        err = 0;

    return (err);
}
```

set_standard_colormap

This routine ensures that there is a suitable colormap on your server during this session. For 8-bit PseudoColor visuals, this colormap should have the property `XA_RGB_DEFAULT_MAP` and contain a 6x9x4 color cube. If not, it creates a colormap containing this color cube with that property.

For 24-bit TrueColor visuals, this colormap should have the property `XA_RGB_BEST_MAP`. `set_standard_colormap` calls the subroutine `create_pseudo_map` to create the colormap for 8-bit PseudoColor visuals, and the subroutine `create_readonly_map` for 24-bit TrueColor visuals.

Code Example 6-3 Create a Standard RGB Map Example *(1 of 3)*

```
/* create a standard RGB map.
 */
int
set_standard_colormap( dpy, vis_info)
    Display             *dpy;
    XVisualInfo         *vis_info;
{
    XStandardColormap   *stdcmaps, cmap_info, *s;
    int                 i, count;
    int                 st = 1, done = 0;

    switch ( vis_info->class ) {
    case PseudoColor: {

        /* first see if the property exists
         */
        if (XGetRGBColormaps(dpy, DefaultRootWindow(dpy),
                &stdcmaps, &count, XA_RGB_DEFAULT_MAP)) {

            /* find the property that matches the given visualid
             */
            for (i = 0, s = stdcmaps;
              (i < count) && (s->visualid != vis_info->visualid);
                 i++, s++)
                ;

            /* verify that the color cube is a 6x9x4 color cube.
               Sun PEX performs best with a 6x9x4 color cube on
               a PseudoColor visual
             */
            if (i != count) {
```

Code Example 6-3 Create a Standard RGB Map Example *(2 of 3)*

```
/* create a standard RGB map.
                if ((s->red_max == 5) && (s->green_max == 8) &&
                    (s->blue_max == 3) &&
                    ((s->red_mult < s->green_mult) &&
                      (s->green_mult < s->blue_mult)))
                    done = 1;
            }
            XFree((char *)stdcmaps);
        }

        /* if property doesn't exist for the given visualid
         */
        if (!done) {

            /* create a 6x9x4 color cube
             */
            st = create_pseudo_map(dpy, vis_info, 6, 9, 4,
&cmap_info);
            XChangeProperty(dpy, DefaultRootWindow(dpy),
                XA_RGB_DEFAULT_MAP, XA_RGB_COLOR_MAP, 32,
PropModePrepend,
                (unsigned char *)&cmap_info, 10);
        }
        break;
    }
    case TrueColor: {

        /* make a best_map if it does not already exist
         */
        XGetRGBColormaps(dpy, DefaultRootWindow(dpy),
                &stdcmaps, &count, XA_RGB_BEST_MAP);

        if (count == 0) {
            /* create a readonly map
             */
            st = create_readonly_map(dpy, vis_info, &cmap_info);
            XChangeProperty(dpy, DefaultRootWindow(dpy),
                XA_RGB_BEST_MAP, XA_RGB_COLOR_MAP, 32,
PropModePrepend,
                (unsigned char *)&cmap_info, 10);
        } else
            XFree((char *)stdcmaps);

        break;
```

Code Example 6-3 Create a Standard RGB Map Example *(3 of 3)*

```
/* create a standard RGB map.
    }
    default:
        break;
    }
    return st;
}
```

create_pseudo_colormap

This routine creates a colormap containing a 6x9x4 color cube, enabling the best performance for 8-bit PseudoColor visuals on Sun graphics devices.

Code Example 6-4 Create a Pseudo Colormap Example *(1 of 3)*

```
/* create a pseudo colormap
 */
int
create_pseudo_map(dpy, vis_info, nr, ng, nb, cmap_info)
    Display             *dpy;
    XVisualInfo         *vis_info;
    int                 nr, ng, nb;
    XStandardColormap   *cmap_info;
{
    int                 num_colors, st, i, j, k, p;
    unsigned long       *pixels;
    XColor              *colors, *color;
    XWindowAttributes   root_window_attrs;

    XGetWindowAttributes(dpy, DefaultRootWindow(dpy),
&root_window_attrs);

    cmap_info->colormap = root_window_attrs.colormap;
    cmap_info->visualid = vis_info->visualid;
    cmap_info->red_max = nr - 1;
    cmap_info->red_mult = 1;
    cmap_info->green_max = ng - 1;
    cmap_info->green_mult = nr;
    cmap_info->blue_max = nb - 1;
    cmap_info->blue_mult = nr * ng;

    num_colors = nr * ng * nb;
    pixels = (unsigned long *) malloc(num_colors
*sizeof(*pixels));
```

Code Example 6-4 Create a Pseudo Colormap Example *(2 of 3)*

```
/* create a pseudo colormap

    /* Try to allocate the color cells from the default
       colormap first. It it doesn't fit, then create a private
       colormap
     */
    st = XAllocColorCells(dpy, cmap_info->colormap, True,
                 (unsigned long *)NULL, 0, pixels, num_colors);
    if (st == 0) {
        cmap_info->colormap = XCreateColormap(dpy,
            RootWindow(dpy, DefaultScreen(dpy)),
            vis_info->visual, AllocNone);
        st = XAllocColorCells(dpy, cmap_info->colormap, True,
                 (unsigned long *)NULL, 0, pixels, num_colors);
        if (st == 0) {
            free(pixels) ;
            return 0;
        }
        cmap_info->killid = ReleaseByFreeingColormap;
    } else {

        /* ICCCM requires an arbitrary resource id for the
           the killid if the colormap resource is the default
colormap
         */
        cmap_info->killid = XCreateWindow(dpy,
                RootWindow(dpy, DefaultScreen(dpy)),
             1,1,1,1,0,0, InputOnly, vis_info->visual, 0, NULL);
        XDestroyWindow(dpy, cmap_info->killid);
    }

    cmap_info->base_pixel = pixels[0];
    free(pixels);

    colors = (XColor *) malloc(num_colors * sizeof(*colors));
    p = cmap_info->base_pixel;
    for (i = 0, color = colors; i < nr; i++) {
        for (j = 0; j < ng; j++) {
            for (k = 0; k < nb; k++) {
                color->flags = DoRed | DoGreen | DoBlue;
                color->pixel = cmap_info->base_pixel +
                                i * cmap_info->red_mult +
                                j * cmap_info->green_mult +
                                k * cmap_info->blue_mult;
```

Code Example 6-4 Create a Pseudo Colormap Example *(3 of 3)*

```
/* create a pseudo colormap

                color->red = i * 65535 / cmap_info->red_max;
                color->green = j * 65535 / cmap_info->green_max;
                color->blue = k * 65535 / cmap_info->blue_max;
                ++color;
            }
        }
    }

    XStoreColors(dpy, cmap_info->colormap, colors, num_colors);
    free(colors);

    return 1;
}
```

create_readonly_map

This routine creates a colormap for 24-bit TrueColor visuals.

Code Example 6-5 Create a Readonly Map Example *(1 of 2)*

```
/* create a readonly map
 */
int
create_readonly_map(dpy, vis_info, cmap_info)
    Display             *dpy;
    XVisualInfo         *vis_info;
    XStandardColormap   *cmap_info;
{
    cmap_info->colormap = XCreateColormap(dpy,
            RootWindow(dpy, DefaultScreen(dpy)),
            vis_info->visual, AllocNone);

    cmap_info->base_pixel = 0;
    cmap_info->visualid = vis_info->visualid;

    cmap_info->red_max = vis_info->red_mask;
    cmap_info->red_mult = 1;
    while (!(cmap_info->red_max & 0x01)) {
        cmap_info->red_max >>= 1;
        cmap_info->red_mult <<= 1;
    }
    cmap_info->green_max = vis_info->green_mask;
```

Code Example 6-5 Create a Readonly Map Example *(2 of 2)*

```
/* create a readonly map
    cmap_info->green_mult = 1;
    while (!(cmap_info->green_max & 0x01)) {
        cmap_info->green_max >>= 1;
        cmap_info->green_mult <<= 1;
    }
    cmap_info->blue_max = vis_info->blue_mask;
    cmap_info->blue_mult = 1;
    while (!(cmap_info->blue_max & 0x01)) {
        cmap_info->blue_max >>= 1;
        cmap_info->blue_mult <<= 1;
    }

    cmap_info->killid = ReleaseByFreeingColormap;
    return 1;
}
```