The *XView/NeWS* Library
Preliminary Documentation

The *XView/NeWS library* is intended as cleaner/simpler method for *NeWS* canvas handling in *XView*. It is not intended to be a full C client interface for PostScript rendering. The intention is to provide *XView* developers with convenient access to *NeWS*' PostScript rendering capability. Canvases are created using the familiar `xv_create()` call. Access to canvases not created as `PSCANVASes` is available through `ps_token_from_xid()`.

Repaint and resize procs can be provided by the developer, as they can on *XView* `Canvases`. The environment is set up for each of these calls so that the rendering appears on the correct canvas (`setcanvas`) and the graphics state is not effected (`gsave/grestore`). Integration of the X damage information is provided through `PSCANVAS_CLIPRECTS`.

At the present time, all of the Red Book PostScript operators are not automatically defined by the *CPS* include files. Offerings in the future may include a more complete set of definitions, but at this time, it will still be necessary to use *CPS* to define many necessary operations.

Sample programs using this library can be found on the *OpenWindows* version 2 release tape in $OPENWINHOME/share/src/sun/xview/demos/xvps.

Attribute Summary

PSCANVAS_CLIPRECTS

Sets the clipping for an *XVie*w/*NeWS* canvas. Any subsequent *NeWS* rendering operations will be clipped to this region. Input is an *XView* rectangle list, with the special value `NULL` meaning no extra clipping is to be done.

Default: `NULL`
Type: `Xv_rectlist`
Procs: `create, set`

PSCANVAS_NEWSTOKEN

Returns the unique *NeWS* identifier for the given PSCANVAS. See `pscanvas_is_token()`, `ps_setcanvas()` in *Summary of Procedures and Macros*

Type: `NeWStoken`
Procs: `get`

PSCANVAS_REPAINT_PROC

Names a procedure called when then PSCANVAS has been damaged and must be repaired. See `pscanvas_repaint_proc()` in *Summary of Procedures and Macros*

Default: `NULL`
Procs: `create, get, set`

PSCANVAS_RESIZE_PROC

Names a procedure called when the PSCANVAS width or height changes.
See `pscanvas_repaint_proc()` in *Summary of Procedures and Macros*

Default: `NULL`
Procs: `create, get, set`

PSCANVAS_SYNC

Specifies whether *XView* will periodically synchronize *X* and *NeWS*.

Default: `TRUE`
Type: `int`
Procs: `create, get, set`

Data Types

PScanvas Handle to an opaque structure that describes a *XView/NeWS* canvas.

NeWStoken Handle to a *NeWS* object residing on the server, in *NeWS* terminology, a `usertoken`.

Summary of Procedures and Macros

This appendix lists the procedures and macros in alphabetical order and describes each of them briefly. The description includes the call that uses the procedure or macro.

pscanvas_repaint_proc()

Client-supplied callback procedure called when getting a `WIN_REPAINT` event. Function is installed using the attribute `PSCANVAS_REPAINT_PROC`.

```
void
pscanvas_repaint_proc(pscanvas, newstoken, display, xid,
                      xrects)
       PScanvas      pscanvas;
       NeWStoken     newstoken;
       Display       *display;
       Window        xid;
       Xv_rectlist   *xrects;
```

`newstoken` is a *NeWS* usertoken identifier for the pscanvas window. `display` is the display handle to the X11 server connection. `xid` is the X11 window identifier for the pscanvas. The `xrects` structure contains an array of `XRectangles` and a count that specifies the repaint area for the pscanvas. This structure can be used to set the attribute `PSCANVAS_CLIPRECTS` to use the X11 damage as a clipping region. See Attribute Summary.

pscanvas_resize_proc()

Procedure called when pscanvas width or height changes. This function is installed using the attribute `PSCANVAS_RESIZE_PROC`.

```
void
pscanvas_resize_proc(pscanvas, width, height)
       PScanvas      pscanvas;
       int           width;
       int           height;
```

pscanvas_sync()

Procedure for synchronizing the NeWS connection. This insures that the NeWS connection has “caught up.” Any errors ocurring on the NeWS side are collected and reported at this time, using `xv_error()`.

```
void
       pscanvas_sync()
```

pscanvas_flip()

Procedure for flipping over the transformation on a PSCANVAS. This macro can be called in the repaint procedure to insure that rendering will appear the correct side up.

```
void
pscanvas_flip()
```

Additional *CPS* Macros

Below are *CPS* macros defined for use with the *XView/NeWS library*. These macros supplement the existing set of *CPS* macros defined in $OPENWINHOME/include/NeWS/psmacros.h and covered in Chapter 7 of the *NeWS Programmer's Guide*.

ps_clear_cliprects()

Sets the clipping for the current canvas back to NULL, or the original shape of the window. This is equivalent to xv_set(pscanvas, PSCANVAS_CLIPRECTS, NULL); for the appropriate PScanvas. But this can be used for routines that are used on more than one PScanvas.

```
void
ps_clear_cliprects()
```

ps_setcanvas()

Sets the current *NeWS* canvas to be the canvas pointed to by newstoken. Note: all repaint and resize procedures are bracketed with a ps_gsave, the appropriate ps_setcanvas, and a ps_grestore, so it is not necessary to do any of these during normal circumstances.

```
void
ps_setcanvas(newstoken)
        NeWStoken     *newstoken;
```

ps_token_from_xid(xid, newstoken)

xid is the X11 window identifier for a window. This macro retrieves a *NeWS* usertoken identifier for this window suitable for use with ps_setcanvas. If the value given is invalid, the routine returns FALSE and -1 is returned in the second argument.

```
Boolean
ps_token_from_xid(xid, newstoken)
        Window        xid;
        NeWStoken     *newstoken;
```

ps_token_from_code(postscript_code, newstoken)

`newstoken` is set to a *NeWS* usertoken for the PostScript object passed as the first argument (a character string). The PostScript is checked first to see if it is a single object, if it is, `newstoken` is set to a usertoken pointing at it and the routine returns `TRUE`. Otherwise the routine returns `FALSE` and `newstoken` is set to -1. This is useful for making tokens for canvases created by user-defined *CPS* procedures: `ps_token_from_code("MyCanvas",&mycanvas)`

```
Boolean
ps_token_from_code(postscript_code, newstoken)
       char          *postscript_code;
       NeWStoken     *newstoken;
```

XView/NeWS Library: Examples

Below is a sample program that uses the X*View/NeWS library* to create a PSCANVAS and display the word “NeWS” using *NeWS* PostScript rendering.

simple.c

```
/*
 * simple.c -- a short example using the XView/NeWS library
 *
 * A PSCANVAS is created, then NeWS PostScript rendering is
 * used to display the word "NeWS" at an angle.
 * The text remains centered when the window is resized.
 *
 */
#include <xview/xview.h>
#include <xvps/pscanvas.h>
#include "simple.h"
Frame frame;
void pscanvas_repaint_proc(pscanvas, newstoken, dpy, xid,
                     xrects)
 PScanvas            pscanvas;
 NeWStoken           newstoken;
 Display             *dpy;
 Window              xid;
 Xv_xrectlist *xrects;
{
 xv_set(pscanvas, PSCANVAS_CLIPRECTS, xrects, NULL);
 pscanvas_flip()
 simple_setup();
 ps_findfont("Bembo-Bold");
 ps_scalefont(97);
 ps_setfont();
 ps_show("NeWS");
}
main(argc, argv)
                     int argc;
                     char **argv;
{
 xv_init(XV_INIT_ARGS, argc, argv, 0);
 frame=xv_create(NULL, FRAME,
          FRAME_LABEL, "XView/NeWSLibrary Example",
          NULL);
 xv_create(frame, PSCANVAS,
                     PSCANVAS_REPAINT_PROC,
                     XV_VISUAL_CLASS, StaticColor,
                     pscanvas_repaint_proc,
                     NULL);
 window_fit(frame);
 xv_main_loop(frame);
}
```

Below is the *CPS* routine needed for use with the C program simple.c. The canvas is filled with white, and the coordinate space is rotated. In the C program, a string will be displayed and appear rotated.

simple.cps

```
cdef simple_setup()
          1 -1 scale                          % fllip view
          1 fillcanvas                        % clear canvas
          .8 setgray
          200 100 moveto
          22 rotate                           % 22 degrees
```

Compiling this example

To compile and test this example, run CPS against simple.cps and compile simple.c, linking with the XView/NeWS library (libxvps) and the CPS library:

```
cps simple.cps
cc -o simple simple.c -lxvps -lxview -lolgx -lX -lcps \
          -I$OPENWINHOME/include -L$OPENWINHOME/lib
```

Additional Notes

X and *NeWS* Color

The X11 and *NeWS* imaging models have different models for allocating and using color. Using the XView/NeWS Library, either model may be used to manage your colors. It may be more natural to use the *NeWS* StaticColor model, which allocates a *NeWS* Color Cube that all applications share. This model has more of the device-independant approach that PostScript uses. To use NeWS StaticColor, *XView* allows you to set the "Visual" or color model for specific XView objects. To do this set the `XV_VISUAL_CLASS` attribute on your `PSCANVAS`:

```
xv_create(frame, PSCANVAS,
                    PSCANVAS_REPAINT_PROC,
                    XV_VISUAL_CLASS, StaticColor,
                    pscanvas_repaint_proc,
                    NULL);
```

Using *XView* to manage your colors is also possible, using the `setpixel NeWS` operator, and the *XView* cms operations. That approach is beyond the scope of this document. See your *XView* documentation and the *NeWS 2.1 Programmer's Guide* for more infomation

Default Transformation Matrix

The current default transform for a `PSCANVAS` is the NeWS co-ordinate space, with origin in the bottom left corner. Unfortunately, this causes the rendering to appear upside down. To fix this, it is currently the developers responsibility to set up an appropriate transformation matrix in the repaint procedure. The simplest method for doing this is to call `pscanvas_flip()` in your repaint procedure.

Using the NeWS "framebuffer"

NeWS programmers may want to use the X11/NeWS root window, or "framebuffer" at some point. To do this, use XView to get a valid XID for the root window, and then use `ps_token_from_xid()` to get NeWStoken for use with `ps_setcanvas()`.

```
frame = (Frame)xv_create(NULL, FRAME,
        XV_WIDTH, 100,
        XV_HEIGHT, 100,
        FRAME_LABEL, "Base Frame",
        NULL);

screen = xv_get(frame, XV_SCREEN);
root = xv_get(screen, XV_ROOT);
xid = (XID)xv_get(root, XV_XID);
ps_token_from_xid(xid,&framebuffer);
```

Trademarks and Copyright Notices

NeWS, X11/NeWS and *XView* are trademarks of Sun Microsystems, Inc. Sun Wrokstation, Sun Microsystems and the Sun logo are registered trademarks of Sun Microsystems, Inc.

PostScript is a registered trademark of Adobe Systems. Adobe owns copyrights related to thePostScript language and the PostScript interpreter. The trademark PststScript is used herein to refer to the material supplied by Adobe or to programs written in the PostScript language as defined by Adobe.

The X Window System is a trademark of Massachusetts Institute of Technology

The *XView/NeWS* Library
Preliminary Documentation

The *XView/NeWS library* is intended as cleaner/simpler method for *NeWS* canvas handling in *XView*. It is not intended to be a full C client interface for PostScript rendering. The intention is to provide *XView* developers with convenient access to *NeWS*' PostScript rendering capability. Canvases are created using the familiar `xv_create()` call. Access to canvases not created as `PSCANVASes` is available through `ps_token_from_xid()`.

Repaint and resize procs can be provided by the developer, as they can on *XView* `Canvases`. The environment is set up for each of these calls so that the rendering appears on the correct canvas (`setcanvas`) and the graphics state is not effected (`gsave/grestore`). Integration of the X damage information is provided through `PSCANVAS_CLIPRECTS`.

At the present time, all of the Red Book PostScript operators are not automatically defined by the *CPS* include files. Offerings in the future may include a more complete set of definitions, but at this time, it will still be necessary to use *CPS* to define many necessary operations.

Sample programs using this library can be found on the *OpenWindows* version 2 release tape in $OPENWINHOME/share/src/sun/xview/demos/xvps.

Attribute Summary

PSCANVAS_CLIPRECTS

Sets the clipping for an *XVie*w/*NeWS* canvas. Any subsequent *NeWS* rendering operations will be clipped to this region. Input is an *XView* rectangle list, with the special value `NULL` meaning no extra clipping is to be done.

Default: `NULL`
Type: `Xv_rectlist`
Procs: `create, set`

PSCANVAS_NEWSTOKEN

Returns the unique *NeWS* identifier for the given PSCANVAS. See `pscanvas_is_token()`, `ps_setcanvas()` in *Summary of Procedures and Macros*

Type: `NeWStoken`
Procs: `get`

PSCANVAS_REPAINT_PROC

Names a procedure called when then PSCANVAS has been damaged and must be repaired. See `pscanvas_repaint_proc()` in *Summary of Procedures and Macros*

Default: `NULL`
Procs: `create, get, set`

PSCANVAS_RESIZE_PROC

Names a procedure called when the PSCANVAS width or height changes.
See `pscanvas_repaint_proc()` in *Summary of Procedures and Macros*

Default: `NULL`
Procs: `create, get, set`

PSCANVAS_SYNC

Specifies whether *XView* will periodically synchronize *X* and *NeWS*.

Default: `TRUE`
Type: `int`
Procs: `create, get, set`

Data Types

PScanvas Handle to an opaque structure that describes a *XView/NeWS* canvas.

NeWStoken Handle to a *NeWS* object residing on the server, in *NeWS* terminology, a `usertoken`.

Summary of Procedures and Macros

This appendix lists the procedures and macros in alphabetical order and describes each of them briefly. The description includes the call that uses the procedure or macro.

pscanvas_repaint_proc()

Client-supplied callback procedure called when getting a `WIN_REPAINT` event. Function is installed using the attribute `PSCANVAS_REPAINT_PROC`.

```
void
pscanvas_repaint_proc(pscanvas, newstoken, display, xid,
                      xrects)
       PScanvas      pscanvas;
       NeWStoken     newstoken;
       Display       *display;
       Window        xid;
       Xv_rectlist   *xrects;
```

`newstoken` is a *NeWS* usertoken identifier for the pscanvas window. `display` is the display handle to the X11 server connection. `xid` is the X11 window identifier for the pscanvas. The `xrects` structure contains an array of `XRectangles` and a count that specifies the repaint area for the pscanvas. This structure can be used to set the attribute `PSCANVAS_CLIPRECTS` to use the X11 damage as a clipping region. See Attribute Summary.

pscanvas_resize_proc()

Procedure called when pscanvas width or height changes. This function is installed using the attribute `PSCANVAS_RESIZE_PROC`.

```
void
pscanvas_resize_proc(pscanvas, width, height)
       PScanvas      pscanvas;
       int           width;
       int           height;
```

pscanvas_sync()

Procedure for synchronizing the NeWS connection. This insures that the NeWS connection has “caught up.” Any errors ocurring on the NeWS side are collected and reported at this time, using `xv_error()`.

```
void
       pscanvas_sync()
```

pscanvas_flip()

Procedure for flipping over the transformation on a PSCANVAS. This macro can be called in the repaint procedure to insure that rendering will appear the correct side up.

```
void
pscanvas_flip()
```

Additional *CPS* Macros

Below are *CPS* macros defined for use with the *XView/NeWS library*. These macros supplement the existing set of *CPS* macros defined in $OPENWINHOME/include/NeWS/psmacros.h and covered in Chapter 7 of the *NeWS Programmer's Guide*.

ps_clear_cliprects()

Sets the clipping for the current canvas back to NULL, or the original shape of the window. This is equivalent to xv_set(pscanvas, PSCANVAS_CLIPRECTS, NULL); for the appropriate PScanvas. But this can be used for routines that are used on more than one PScanvas.

```
void
ps_clear_cliprects()
```

ps_setcanvas()

Sets the current *NeWS* canvas to be the canvas pointed to by newstoken. Note: all repaint and resize procedures are bracketed with a ps_gsave, the appropriate ps_setcanvas, and a ps_grestore, so it is not necessary to do any of these during normal circumstances.

```
void
ps_setcanvas(newstoken)
        NeWStoken     *newstoken;
```

ps_token_from_xid(xid, newstoken)

xid is the X11 window identifier for a window. This macro retrieves a *NeWS* usertoken identifier for this window suitable for use with ps_setcanvas. If the value given is invalid, the routine returns FALSE and -1 is returned in the second argument.

```
Boolean
ps_token_from_xid(xid, newstoken)
        Window        xid;
        NeWStoken     *newstoken;
```

ps_token_from_code(postscript_code, newstoken)

`newstoken` is set to a *NeWS* usertoken for the PostScript object passed as the first argument (a character string). The PostScript is checked first to see if it is a single object, if it is, `newstoken` is set to a usertoken pointing at it and the routine returns `TRUE`. Otherwise the routine returns `FALSE` and `newstoken` is set to -1. This is useful for making tokens for canvases created by user-defined *CPS* procedures: `ps_token_from_code("MyCanvas",&mycanvas)`

```
Boolean
ps_token_from_code(postscript_code, newstoken)
       char          *postscript_code;
       NeWStoken     *newstoken;
```

XView/NeWS Library: Examples

Below is a sample program that uses the X*View/NeWS library* to create a PSCANVAS and display the word “NeWS” using *NeWS* PostScript rendering.

simple.c

```
/*
 * simple.c -- a short example using the XView/NeWS library
 *
 * A PSCANVAS is created, then NeWS PostScript rendering is
 * used to display the word "NeWS" at an angle.
 * The text remains centered when the window is resized.
 *
 */
#include <xview/xview.h>
#include <xvps/pscanvas.h>
#include "simple.h"
Frame frame;
void pscanvas_repaint_proc(pscanvas, newstoken, dpy, xid,
                     xrects)
 PScanvas            pscanvas;
 NeWStoken           newstoken;
 Display             *dpy;
 Window              xid;
 Xv_xrectlist *xrects;
{
 xv_set(pscanvas, PSCANVAS_CLIPRECTS, xrects, NULL);
 pscanvas_flip()
 simple_setup();
 ps_findfont("Bembo-Bold");
 ps_scalefont(97);
 ps_setfont();
 ps_show("NeWS");
}
main(argc, argv)
                     int argc;
                     char **argv;
{
 xv_init(XV_INIT_ARGS, argc, argv, 0);
 frame=xv_create(NULL, FRAME,
          FRAME_LABEL, "XView/NeWSLibrary Example",
          NULL);
 xv_create(frame, PSCANVAS,
                     PSCANVAS_REPAINT_PROC,
                     XV_VISUAL_CLASS, StaticColor,
                     pscanvas_repaint_proc,
                     NULL);
 window_fit(frame);
 xv_main_loop(frame);
}
```

Below is the *CPS* routine needed for use with the C program simple.c. The canvas is filled with white, and the coordinate space is rotated. In the C program, a string will be displayed and appear rotated.

simple.cps

```
cdef simple_setup()
          1 -1 scale                          % fllip view
          1 fillcanvas                        % clear canvas
          .8 setgray
          200 100 moveto
          22 rotate                           % 22 degrees
```

Compiling this example

To compile and test this example, run CPS against simple.cps and compile simple.c, linking with the XView/NeWS library (libxvps) and the CPS library:

```
cps simple.cps
cc -o simple simple.c -lxvps -lxview -lolgx -lX -lcps \
          -I$OPENWINHOME/include -L$OPENWINHOME/lib
```

Additional Notes

X and *NeWS* Color

The X11 and *NeWS* imaging models have different models for allocating and using color. Using the XView/NeWS Library, either model may be used to manage your colors. It may be more natural to use the *NeWS* StaticColor model, which allocates a *NeWS* Color Cube that all applications share. This model has more of the device-independant approach that PostScript uses. To use NeWS StaticColor, *XView* allows you to set the "Visual" or color model for specific XView objects. To do this set the `XV_VISUAL_CLASS` attribute on your `PSCANVAS`:

```
xv_create(frame, PSCANVAS,
                    PSCANVAS_REPAINT_PROC,
                    XV_VISUAL_CLASS, StaticColor,
                    pscanvas_repaint_proc,
                    NULL);
```

Using *XView* to manage your colors is also possible, using the `setpixel NeWS` operator, and the *XView* cms operations. That approach is beyond the scope of this document. See your *XView* documentation and the *NeWS 2.1 Programmer's Guide* for more infomation

Default Transformation Matrix

The current default transform for a `PSCANVAS` is the NeWS co-ordinate space, with origin in the bottom left corner. Unfortunately, this causes the rendering to appear upside down. To fix this, it is currently the developers responsibility to set up an appropriate transformation matrix in the repaint procedure. The simplest method for doing this is to call `pscanvas_flip()` in your repaint procedure.

Using the NeWS "framebuffer"

NeWS programmers may want to use the X11/NeWS root window, or "framebuffer" at some point. To do this, use XView to get a valid XID for the root window, and then use `ps_token_from_xid()` to get NeWStoken for use with `ps_setcanvas()`.

```
frame = (Frame)xv_create(NULL, FRAME,
        XV_WIDTH, 100,
        XV_HEIGHT, 100,
        FRAME_LABEL, "Base Frame",
        NULL);

screen = xv_get(frame, XV_SCREEN);
root = xv_get(screen, XV_ROOT);
xid = (XID)xv_get(root, XV_XID);
ps_token_from_xid(xid,&framebuffer);
```

Trademarks and Copyright Notices

NeWS, X11/NeWS and *XView* are trademarks of Sun Microsystems, Inc. Sun Wrokstation, Sun Microsystems and the Sun logo are registered trademarks of Sun Microsystems, Inc.

PostScript is a registered trademark of Adobe Systems. Adobe owns copyrights related to thePostScript language and the PostScript interpreter. The trademark PststScript is used herein to refer to the material supplied by Adobe or to programs written in the PostScript language as defined by Adobe.

The X Window System is a trademark of Massachusetts Institute of Technology