```
/* DO NOT EDIT THIS FILE.  It contains C->PostScript communication
   initializations that were automatically generated from pscan_ps.cps */

char _CPS_ps_xid[] = "%d/XLookupClient XResource send%d/XConstructID\200\002\246psend
/XLookupID\200\003 -1\246\244send ";
char _CPS_ps_setcanvas[] = "\246\252";
char _CPS_ps_synctag[] = "%d\246\311";
char                              _CPS_ps_init_transform[]                              =
"\265/Interests\246]{\246\006Name\246\320/dicttype\246K{  Name  /Damaged\246y{  Name
/Damaged\246\322}\246k}\246k\246D}\246X\265\246!\246K{\275\246S}
{\272\246\252\246\316}\246l\246q\246\022\246\223\200\002\246p\311\200\000\2-
66\306\200\001-1\246\247\275\275\275\246\223rectpath\246!\246\253{\271}\246k";
```

```
/* DO NOT EDIT THIS FILE.  It contains C->PostScript communication
   definitions that were automatically generated from pscan_ps.cps */

#define ps_xid(P__0,P__1)  pprintf(PostScript, _CPS_ps_xid, 77, P__0, P__1)
extern char _CPS_ps_xid[];
#define ps_setcanvas()  pprintf(PostScript, _CPS_ps_setcanvas, 2)
extern char _CPS_ps_setcanvas[];
#define   ps_synctag(P__0)   (   pprintf(PostScript,   _CPS_ps_synctag,   4,   P__0),
ps_checkfor(PostScriptInput, PSIO_WAIT_TAG, P__0) > 0)
extern char _CPS_ps_synctag[];
#define ps_init_transform()  pprintf(PostScript, _CPS_ps_init_transform, 141)
extern char _CPS_ps_init_transform[];
#ifndef PSMACROS_H
#include <NeWS/psmacros.h>/* standard cdef macros */
#endif
```

```
% If the canvas on the stack is also the currentcanvas,
% don't bother with gsave/grestore.
% Then reshape the canvas to its original shape in order to
% reset the canvas transform.
dup currentcanvas eq {pop false} {gsave setcanvas true} ifelse
  initclip clippath pathbbox
  2 index add % x y w <d to origin>
  0 exch translate% x y w
  1 -1 scale pop pop pop
  pathbbox rectpath currentcanvas reshapecanvas
{grestore} if
```

```
% @(#)pscan_ps.cps 1.2

% find the canvas given the clientID and objectID; leave on the stack
cdef ps_xid(int cID, int objID)

    cID /XLookupClient XResource send% get client resource DB
    objID /XConstructID 2 index send% find the contruct ID
    /XLookupID 3 -1 roll send% find the canvas

% canvas => -
cdef ps_setcanvas()
    setcanvas

% - => -
cdef ps_synctag(tag) => tag()
    tag tagprint

% canvas => -
cdef ps_init_transform()
  % REMIND
  % This code removes Damaged from the aggregate X interpreter interest
  % to work around a bug that fails to remove Damaged when ExposureMask
  % is unset.
  dup /Interests get
  {
    begin
      Name type /dicttype eq {
    Name /Damaged known {
      Name /Damaged undef
    } if
      } if
    end
  } forall
```

```
Pkg_private int
xv_psview_destroy (psview_public, status)
    PSview          psview_public;
    Destroy_status  status;
{
    PSview_info     *psview = PSVIEW_PRIVATE(psview_public);

    if (status == DESTROY_CLEANUP) {
        free ((char *) psview);
    }

    return XV_OK;
}
```

```
static char     sccsid[] = "@(#)pscan_destroy.c 1.1 ";

/** (c) Copyright 1989 Sun Microsystems, Inc. Sun design patents
 *  pending in the U.S. and foreign countries. See LEGAL NOTICE
 *  file for terms of the license.
 */
#include <xvps_private/pscan_impl.h>
#include <xview/notify.h>

Pkg_private int
xv_pscanvas_destroy (pscanvas_public, status)
    PScanvasps      canvas_public;
    Destroy_status  status;
{
    PScanvas_info   *pscan = PSCANVAS_PRIVATE(pscanvas_public);
    extern int      n_pscans;

    if (status == DESTROY_CLEANUP) {
        if (--n_pscans == 0)
            ps_close_PostScript();
        free ((char *) pscan);
    }
    return XV_OK;
}
```

```
    for (attr = (PScanvas_attr) avlist[0]; attr;
             avlist = attr_next(avlist), attr = (PScanvas_attr) avlist[0]) {
        switch (attr) {
          case XV_END_CREATE:
#ifdef notyet
            xid = (Window) xv_get(psview_public, XV_XID);
            fprintf (stderr, "psview_init: xid = %x\n", xid);
            wattr.bit_gravity = ForgetGravity;
            wattr.background_pixmap = None;
            XChangeWindowAttributes(xv_default_display, xid,
              CWBitGravity|CWBackPixmap, &wattr);
            XSync((Display *)xv_default_display, 0);
            pscanvas_xid(xid);
            ps_init_transform();
            ps_flush_PostScript();
#endif notyet
            break;
        default:
            xv_check_bad_attr(&xv_pscanvas_pkg, attr);
            break;

      }
    }
    return ((Xv_opaque) XV_OK);
}
```

```
                /* Make sure the window really exists on the server */
                xid = (Window) xv_get(viewzero, XV_XID);
                wattr.bit_gravity = ForgetGravity;
                wattr.background_pixmap = None;
                XChangeWindowAttributes(xv_default_display, xid,
                    CWBitGravity|CWBackPixmap, &wattr);
                XSync((Display *)xv_default_display, 0);
                pscanvas_xid(xid);
                ps_init_transform();
                ps_flush_PostScript();
                break;
            default:
                xv_check_bad_attr(&xv_pscanvas_pkg, attr);
                break;

        }
    }
    return ((Xv_opaque) XV_OK);
}

Pkg_private Xv_opaque
xv_psview_set_avlist (psview_public, avlist)
    PSview        psview_public;
    Attr_avlist   avlist;
{
    PSview_info      *psview = PSVIEW_PRIVATE(psview_public);
    PScanvas_attr    attr;
    static XSetWindowAttributes wattr;
    Window           xid;
```

```
Pkg_private Xv_opaque
xv_pscanvas_set_avlist (pscanvas_public, avlist)
    PScanvas      pscanvas_public;
    Attr_avlist   avlist;
{
    PScanvas_info    *pscan = PSCANVAS_PRIVATE(pscanvas_public);
    PScanvas_attr    attr;
    static XSetWindowAttributes wattr;
    Window          xid;
    Xv_Window       viewzero;

    for (attr = (PScanvas_attr) avlist[0]; attr;
            avlist = attr_next(avlist), attr = (PScanvas_attr) avlist[0]) {
        switch (attr) {
          case PSCANVAS_SYNC:
              pscan->sync_tag = (int) avlist[1];
              break;
          case PSCANVAS_INPUTPROC:
              pscan->input_proc = (void (*) ()) avlist[1];
              notify_set_input_func(
                  pscanvas_public, xv_pscanvas_ps_proc,
                  psio_fileno(PostScript));
              break;
          case XV_END_CREATE:
              viewzero = (Xv_Window) xv_get(pscanvas_public,
                  OPENWIN_NTH_VIEW, 0);
              xv_set(viewzero,
                  WIN_CONSUME_EVENTS, WIN_NO_EVENTS,
                  LOC_WINENTER, LOC_WINEXIT, 0,
                  0);
```

```
static char     sccsid[] = "@(#)pscan_set.c 1.5 ";

/*  (c) Copyright 1989 Sun Microsystems, Inc. Sun design patents
 *  pending in the U.S. and foreign countries. See LEGAL NOTICE
 *  file for terms of the license.
 */
#include <X11/Xlib.h>
#include <xview/xview.h>
#include <xview/xv_xrect.h>
#include <xvps_private/pscan_impl.h>

extern Display *xv_default_display;

static void
xv_pscanvas_ps_proc(client, fd)
Notify_client client;
int fd;
{
    PScanvasps          canvas_public = (PScanvas) client;
    PScanvas_info       *pscan = PSCANVAS_PRIVATE(pscanvas_public);

    if (pscan->sync_tag != PSCANVAS_NOSYNC)
        XSync((Display *) xv_default_display, 0);
    pscan->input_proc(client, fd);
    if (pscan->sync_tag != PSCANVAS_NOSYNC)
        ps_synctag(pscan->sync_tag);
}
```

```
Pkg_private Xv_opaque
xv_psview_get_attr (psview_public, status, attr, valist)
    PSview          psview_public;
    int             *status;
    PScanvas_attr   attr;
    va_list         valist;
{
    PSview_info     *psview = PSVIEW_PRIVATE(psview_public);

    switch (attr) {
      case PSVIEW_PSCANVAS:
          return (Xv_opaque) psview->private_pscanvas->public_self;
      default:
          xv_check_bad_attr(&xv_pscanvas_pkg, attr);
          *status = XV_ERROR;
          return (Xv_opaque) 0;
    }
}
```

```
static char     sccsid[] = "@(#)pscan_get.c 1.3 ";

/*  (c) Copyright 1989 Sun Microsystems, Inc. Sun design patents
 *  pending in the U.S. and foreign countries. See LEGAL NOTICE
 *  file for terms of the license.
 */
#include <xvps_private/pscan_impl.h>

Pkg_private Xv_opaque
xv_pscanvas_get_attr (pscanvas_public, status, attr, valist)
    PScanvas         pscanvas_public;
    int             *status;
    PScanvas_attr    attr;
    va_list          valist;
{
    PScanvas_info    *pscan = PSCANVAS_PRIVATE(pscanvas_public);

    switch (attr) {
      case PSCANVAS_SYNC:
          return (Xv_opaque) pscan->sync_tag;
      case PSCANVAS_INPUTPROC:
          return (Xv_opaque) pscan->input_proc;
      default:
          xv_check_bad_attr(&xv_pscanvas_pkg, attr);
          *status = XV_ERROR;
          return (Xv_opaque) 0;
    }
}
```

```
    pscan_object->private_data = (Xv_opaque) pscan;
    n_pscans++;
    return XV_OK;
}

Pkg_private int
xv_psview_init (owner, psview_public, avlist)
    Xv_Window      owner;
    Xv_Window      psview_public;
    Attr_avlist    avlist;
{
    PSview_info*psview;
    Xv_pscanvas*pscan_object = (Xv_pscanvas *) owner;
    Xv_psview*psview_object = (Xv_psview *) psview_public;
    Window xid;

    if (! (psview = xv_alloc(PSview_info)))
        return XV_ERROR;

    psview->public_self = psview_public;
    psview->private_pscanvas = PSCANVAS_PRIVATE(pscan_object);

    psview_object->private_data = (Xv_opaque) psview;

    xid = (Window) xv_get(psview_public, XV_XID);
    if (xid) {
        XSync((Display *)xv_default_display, 0);
        pscanvas_xid(xid);
        ps_init_transform();
        ps_flush_PostScript();
    } else {
        fprintf (stderr, "psview_init: no xid yet\n");
    }

    return XV_OK;
}
```

```
static char     sccsid[] = "@(#)pscanvas.c 1.5 ";

/*  (c) Copyright 1989 Sun Microsystems, Inc. Sun design patents
 *  pending in the U.S. and foreign countries. See LEGAL NOTICE
 *  file for terms of the license.
 */
#include <xvps_private/pscan_impl.h>

int n_pscans;
extern Display *xv_default_display;

Pkg_private int
xv_pscanvas_init (owner, pscan_public, avlist)
    Xv_Window     owner;
    Xv_Window     pscan_public;
    Attr_avlist   avlist;
{
    PScanvas_info    *pscan;
    Xv_pscanvas      *pscan_object = (Xv_pscanvas *) pscan_public;

    if (!PostScript) {
        if (ps_open_PostScript() == 0)
            return XV_ERROR;
    }
    if (! (pscan = xv_alloc(PScanvas_info)))
        return XV_ERROR;

    pscan->sync_tag = PSCANVAS_NOSYNC;
    pscan->flags = 0;
    pscan->input_proc = NULL;
    pscan->repaint_proc = NULL;
    pscan->resize_proc = NULL;
    pscan->public_self = pscan_public;
```

```
static char     sccsid[] = "@(#)pscan_data.c 1.2 ";

/*
 *   (c) Copyright 1989 Sun Microsystems, Inc. Sun design patents
 *   pending in the U.S. and foreign countries. See LEGAL NOTICE
 *   file for terms of the license.
 */
#include <xvps_private/pscan_impl.h>

Xv_pkg          xv_pscanvas_pkg = {
    "PScanvas", ATTR_PKG_PSCANVAS,
    sizeof(Xv_pscanvas),
    &xv_openwin_pkg,
    xv_pscanvas_init,
    xv_pscanvas_set_avlist,
    xv_pscanvas_get_attr,
    xv_pscanvas_destroy,
    NULL        /* no find proc */
};

Xv_pkg    xv_psview_pkg = {
    "PSview", ATTR_PKG_PSCANVAS,
    sizeof(Xv_psview),
    &xv_window_pkg,
    xv_psview_init,
    xv_psview_set_avlist,
    xv_psview_get_attr,
    xv_psview_destroy,
    NULL        /* no find proc */
};
```

```
typedef struct psview_info {
    PSview              public_self;/* back pointer to object */
    PScanvas_info       *private_pscanvas;
} PSview_info;

Pkg_private int xv_pscanvas_init();
Pkg_private int xv_pscanvas_destroy();
Pkg_private Xv_opaque xv_pscanvas_set_avlist();
Pkg_private Xv_opaque xv_pscanvas_get_attr();

Pkg_private int xv_psview_init();
Pkg_private int xv_psview_destroy();
Pkg_private Xv_opaque xv_psview_set_avlist();
Pkg_private Xv_opaque xv_psview_get_attr();

#endif pscanvas_impl_defined
```

```
/*  @(#)pscan_impl.h 1.6 */

#ifndef pscanvas_impl_defined
#define pscanvas_impl_defined

#ifndef FILE
#undef NULL
#include <stdio.h>
#endif FILE
#include <xview/pkg.h>
#include <xview/attrol.h>
#include <xvps/pscanvas.h>
#include <X11/Xlib.h>
#ifndef PSMACROS_H
#include <NeWS/psmacros.h>/* standard cdef macros */
#endif PSMACROS_H

/* flag values */
#define PSCANVAS_X_PAINT         0x0001

#define PSCANVAS_PRIVATE(nc)     XV_PRIVATE(PScanvas_info, Xv_pscanvas, nc)
#define PSCANVAS_PUBLIC(nc)      XV_PUBLIC(nc)

#define PSVIEW_PRIVATE(nv)       XV_PRIVATE(PSview_info, Xv_psview, nv)
#define PSVIEW_PUBLIC(nv)        XV_PUBLIC(nv)

typedef struct pscanvas_info {
    PScanvas        public_self;/* back pointer to object */
    int             sync_tag;
    unsigned int    flags;
    void           (*input_proc) ();
    void           (*resize_proc) ();
    void           (*repaint_proc) ();
} PScanvas_info;
```

```
typedef enum {
    /* Public Attributes */
    PSVIEW_PSCANVAS =    PSCANVAS_ATTR(ATTR_OPAQUE,10),
} PSview_attr;

/* Structures */
typedef struct {
    Xv_openwin           parent_data;
    Xv_opaque            private_data;
} Xv_pscanvas;

typedef struct {
    Xv_window_struct     parent_data;
    Xv_opaque            private_data;
} Xv_psview;

/* Variables */
extern Xv_pkg   xv_pscanvas_pkg;
extern Xv_pkg   xv_psview_pkg;

#endif pscanvas_defined
```

```
/*  @(#)pscanvas.h 1.4 */

#ifndef pscanvas_defined
#define pscanvas_defined

#include <xview/openwin.h>
#include <xview/win_input.h>
#include <xvps/pscan_ps.h>

#define PSCANVAS_NOSYNC  -1

#define client_id(x) ((int) ((x) & 0xfff00000) >> 20)
#define object_id(x) ((int) ((x) & 0x0007ffff))
#define pscanvas_xid(xid) ps_xid(client_id(xid), object_id(xid))

#define PSCANVAS &xv_pscanvas_pkg

#define ATTR_PKG_PSCANVAS(ATTR_PKG_UNUSED_LAST - 1)
#define PSCANVAS_ATTR(type, ordinal)\
                ATTR(ATTR_PKG_PSCANVAS, type, ordinal)

/* Types */

typedef Xv_opaque  PScanvas;
typedef Xv_opaque  PSview;

/* Enumerations */

typedef enum {
    /* Public Attributes */
    PSCANVAS_SYNC     = PSCANVAS_ATTR(ATTR_INT, 3),
    PSCANVAS_INPUTPROC= PSCANVAS_ATTR(ATTR_FUNCTION_PTR, 7),
} PScanvas_attr;
```

XView Extensions (continued):

- Write find procedure --- xv_find():

```
Xv_opaque
wizzy_find(owner, pkg, avlist)
     Xv_opaque          owner;
     Xv_pkg            *pkg;
     Attr_avlist        avlist;
{
     /* initialize variables, private data structures */

     /* parse object description
      * wizzy_parse_attrs(avlist, &decoded_avlist);
      */

     /* get cache and search for object which meets this description */
     server = XV_SERVER_FROM_WINDOW(owner);  /* wizzys are cached off server object */
     if (cache = (Wizzy_cache_list *)xv_get(server, XV_KEY_DATA, wizzy_key)) {
         while (cache) {
               /* search cache for object similar to WIZZY described in avlist */
               if (found)  /* return cached WIZZY object */
                   return (old_wizzy);
         cache = cache->next_wizzy;  /* linked list of wizzy objects */
         }
     }
     /* otherwise, no WIZZY object of this type is cached, return NULL */
     return ((Xv_object) NULL);
}
```

XView Extensions(continued):

- Write destruction procedure --- xv_destroy():

```
int
wizzy_destroy(wizzy_public, status)
     Wizzy_public              wizzy_public;
     Destroy_status            status;
{
     Wizzy_private             *wizzy_private = WIZZY_PRIVATE(wizzy_public);
     int                       return_status = XV_OK;

     if (status == DESTROY_CLEANUP) {
         /* free any allocated resources, both locally and on the server */
         free(wizzy_private);
     } else if (status == DESTROY_PROCESS_DEATH) {
         /* our process is going away, do minimal cleanup */
     } else if (status == DESTROY_SAVE_YOURSELF) {
         /* if we have information that needs to be saved for the next  */
         /* time the system is restarted, do so now.                    */
     } else if (status == DESTROY_CHECKING) {
         /* if we want to "veto" the destruction request, return XV_ERROR */
     }

     return (return_status);
}
```

XView Extensions (continued):

- Write query procedure --- xv_get():

```
Xv_opaque
wizzy_get_avlist(wizzy_public, status, attr, args)
  Wizzy_public                    wizzy_public;
  int                             *status;   /* initialized by caller */
  Attr_attribute                  attr;
  Attr_avlist                     args;
{
     Wizzy_private       *wizzy_private = WIZZY_PRIVATE(wizzy_public);
     Xv_opaque            v;

     switch (attr) {
       case WIZZY_COLOR:
          v = (Xv_opaque) wizzy_private->colored;
          break;
       case WIZZY_PROC:
          v = (Xv_opaque) wizzy_private->wizzy_proc;
          break;
       default:
          if (xv_check_bad_attr(&my_wizzy_pkg, attrs[0]) == XV_ERROR)
             *status = XV_ERROR;
          v = (Xv_opaque) 0;
          break;
     }

     return (Xv_opaque) v;
}
```

XView Extensions (continued):

- Write change/update procedure --- xv_set():

```
Xv_opaque
wizzy_set_avlist(wizzy_public, avlist)
          Wizzy_public    wizzy_public;
          attr_avlist     avlist;
{
    Wizzy_private    *wizzy_private = WIZZY_PRIVATE(wizzy_public);
    Attr_attributes  *attrs;

    for (attrs = avlist; *attrs; attrs = attr_next (avlist)) {
       switch ((int) attrs [0]) {
          case WIZZY_COLOR:
            wizzy_private->colored = (int) attrs [1];
            break;
          case WIZZY_PROC:
            wizzy_private->wizzy_proc = (int (*)()) attrs [1];
            break;
          case XV_END_CREATE:
            /* if you need to know difference between create and set */
            break;
          default:
            xv_check_bad_attr(&my_wizzy_pkg, attrs[0]);
            break;
        }
   }
   /* complete creation, apply changes */
   return XV_OK; /* or XV_ERROR if you detect a problem */
}
```

XView Extensions (continued):

- Write initialization procedure --- xv_create():

```
Xv_opaque
wizzy_init (owner, wizzy_public, avlist)
     Xv_opaque         owner;
     Wizzy_public     *wizzy_public;
     Attr_avlist       avlist;
{
     /* initialize variables, private data structures */
     Wizzy_private *wizzy_private = (Wizzy_private *)xv_alloc(Wizzy_private);

     if (!wizzy_private) return XV_ERROR;

     /* link up object */
     wizzy_public->private_data = (Xv_opaque) wizzy_private;
     wizzy_private->public_self = (Xv_opaque) wizzy_public;

     /* create your object, knowing that the attrs can still be  */
     /* processed in the xv_set()                                */
     ...
     ...
     /* add this object to your cache on the server object */

     return XV_OK;
}
```

XView Extensions (continued):

- Define the new public class structure and inheritance in a source file (wizzy_data.c):

```
Xv_pkg my_wizzy_pkg = {
          "Wizzy"                       /* pkg name               */
          ATTR_PKG_WIZZY,               /* package id             */
          sizeof(Wizzy_public),         /* size of public struct  */
          &xv_generic_pkg,              /* parent class           */
          wizzy_init,                   /* create/initialize func */
          wizzy_set_avlist              /* set function           */
          wizzy_get_attr,               /* get function           */
          wizzy_destroy,                /* destroy function       */
          wizzy_find                    /* find function          */
          };
```

XView Extensions --- Creating a new XView Object Class:

- Define a new XView class and attributes in header files:

```
extern  Xv_pkg                      my_wizzy_pkg;
#define WIZZY                      &my_wizzy_pkg

#define Wizzy                      Xv_opaque
#define ATTR_PKG_WIZZY             (ATTR_PKG_UNUSED_LAST - 1)  /* your ATTR/PKG number */
#define WIZZY_ ATTR(type,ordinal)  ATTR(ATTR_PKG_WIZZY, type, ordinal)

typedef emum {
    WIZZY_COLOR                     = WIZZY_ATTR(ATTR_BOOLEAN,      1),
    WIZZY_PROC                      = WIZZY_ATTR(ATTR_FUNCTION_PTR, 2)
} Wizzy_attributes;

typedef struct {
    Xv_generic_struct               parent_data;
    Xv_opaque                       private_data;
} Wizzy_public;
```

PUBLIC
wizzy.h

```
#define WIZZY_PUBLIC(item)          XV_PUBLIC(item)
#define WIZZY_PRIVATE(i)            XV_PRIVATE(Wizzy_private_struct, Wizzy_struct, i)

typedef struct {
    Xv_opaque                       public_self;
    int                             colored;
    int                             (*wizzy_proc)();
} Wizzy_private;
```

PRIVATE
wizzy_impl.h

XView Extensions — Class Structures:

- The object's Xv_pkg structure defines it's relationship to the world

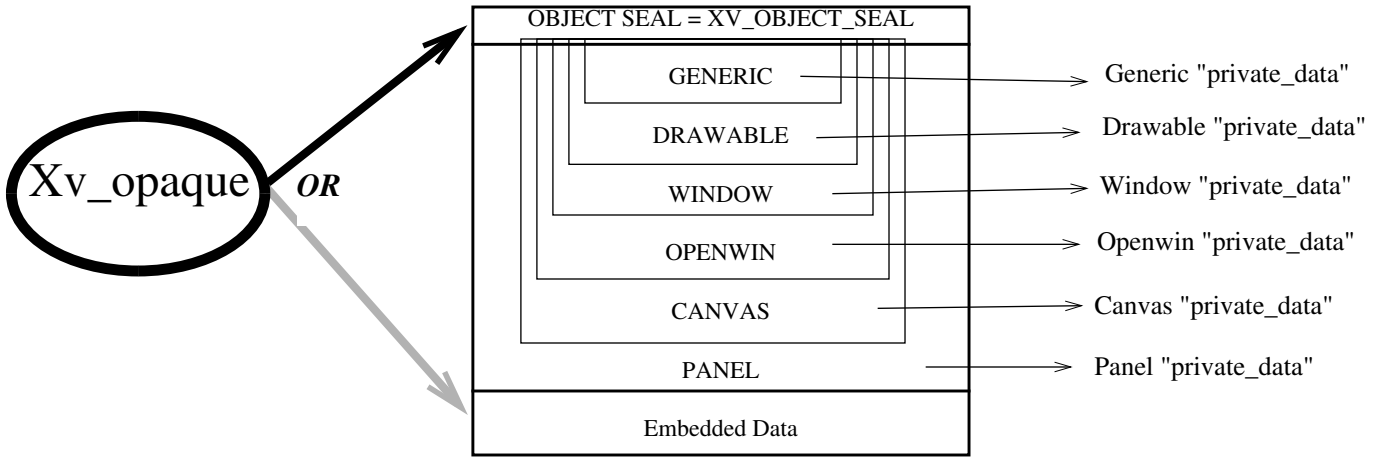
```
typedef struct _xview_pkg {
          char                  *name;
          Attr_pkg              attr_id;
          unsigned              size_of_object;
          Xv_pkg                *parent_pkg;
          int                   (*init)();
          Xv_opaque             (*set)();
          Xv_opaque             (*get)();
          int                   (*destroy)();
          Xv_object             (*find)();
} Xv_pkg;
```

- Setting the "embedding data" makes the "opaque" object something that can be cast and dereferenced...
  - SERVER_IMAGE pkg objects "pretend" they are a Pixrect...

```
typedef struct {
          Xv_generic_struct   parent_data;   /* include parent's data   */
          Xv_opaque           private_data;  /* pointer to private data */
          Xv_embedding        embedding_data;
          Pixrect             pixrect;
} Xv_server_image;
```

XView Extensions — Object Instance:

- Every instance of class is also an instance of it's parent class (and so on...)
  - > PANEL is also a CANVAS is also a OPENWIN is also a .... (Xv_base)



XView Extensions — Classing Relationships:

- Everything is an "object" (ie. Xv_opaque)
  - Some objects are "hidden" — class-builder objects (drawable, openwin)
  - Some objects are "visual" — XID underlying (panel, canvas, window)
  - Some objects are object-oriented interfaces to Xlib — (fullscreen, font)
- Calling directions for init, set, get, destroy:

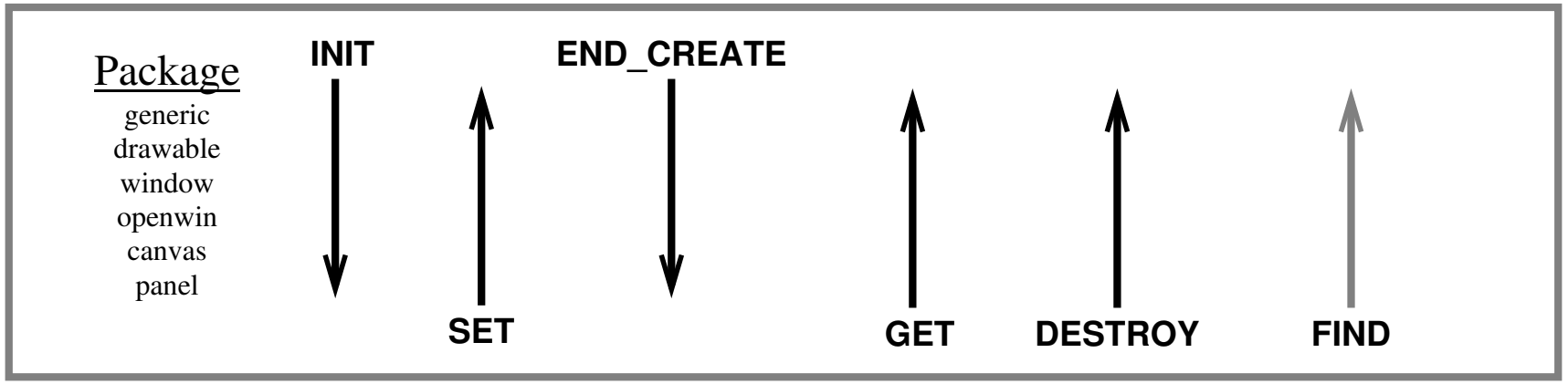


TABLE OF CALLS:

```
xv_create():  INIT, SET, END_CREATE
xv_set():     SET
xv_get():     GET
xv_destroy:   DESTROY
xv_find():    FIND .... or if not found...  INIT, SET, END_CREATE
```

Writing Extensions