Subject: Re: Easy way to make hard copies at full printer resolution
Posted by Craig Markwardt on Mon, 09 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:
> It would be much more simple and elegant
> to solve the problem at the back end in the graphics driver. Has
> anybody ever written a graphics driver?

In the money-where-your-mouth-is department, I've had a go at writing a device driver that captures graphics commands in IDL. It uses a trick which seems to work: I install a new "X" device driver which replaces the original one, does some specialized processing, and then passes the command on to the original true X driver. You might consider it to be a "parasitic" driver since it doesn't do any real X output.

At the moment, all it does is intercept the commands, print them to the console, and then pass them down the chain. Despite this, it's still quite interesting to see how graphics calls work on IDL. See "How to activate" below to load it into your IDL session via call_external.

You are welcome to try it out and provide suggestions. While the code looks lengthy, that's just because I trap all graphics primitives. It works under Linux IDL version 5.X, though I expect it should work under any Unix version with the right compilation line. (See $IDL/external/call_external/C/callext_unix.txt for help). An intrepid Mac/Windows programmer might get it to work too.

No other practical issues, like storing commands, redirecting them to postscript, etc are addressed yet. Before I forge ahead I thought I'd see if anybody else had ideas.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

---------- Beginning xfilter.c
/*

XFILTER - device driver which captures IDL output (DRAFT)

This C code implements a graphics device "filter" for IDL. The intent
is to capture graphics commands which can be reproduced at a later
time. For example, to capture the graphics that appear on the X
windows display for printing later.

Currently this works only under X windows and Unix (Linux is tested).
At present the graphics commands are simply echoed to the IDL standard
error stream, not stored.

The method is simple. A "new" graphics device is registered. IDL's
notion of a graphics device is a jump table which implements various
drawing primitives. Each primitive is implemented below, but
trivially calls the true X device primitive. Thus the devices are
chained together. Both graphics and windowing primitives are chained.

See below for compilation instructions (for Linux) and how to activate
this within IDL.

IDL is a trademark of RSI.

*/

```
#include <stdio.h>
#include "export.h"

/*
   Compile with:
   gcc -g -fPIC xfilter.c -shared -o xfilter.so
*/
/*
   Activate in IDL with:
   a = call_external(getenv('PWD')+'/xfilter.so', 'xfilter', 0)
   set_plot, 'x'
*/

#define oprintf(x...) fprintf(stderr, x);

extern IDL_DEVICE_DEF _IDL_x_dev; /* IDL X windows device driver */
IDL_DEVICE_DEF    xfilter_dev;    /* New device driver block */
IDL_DEVICE_CORE   *xdev, *xfdev;
IDL_DEVICE_WINDOW *xwin, *xfwin;

int cattrq = 0, ctextq = 0;  /* Flags determine when attributes are cached */
IDL_ATTR_STRUCT cattr;       /* Cached graphics attributes */
IDL_TEXT_STRUCT ctext;       /* Cached text attributes */
```

```
 /*********************************************************** **********/
/*                  Graphics primitives                        */

/* Draw line segment */
void xfilter_draw(IDL_GR_PT *p0, IDL_GR_PT *p1, IDL_ATTR_STRUCT *a)
{
  int reset = 0;
  IDL_GR_PT xp0, xp1, *pp0 = 0, *pp1 = 0;

  /* Determine if we need to recache graphics attributes */
  if (!cattrq) reset = 1;
  else if ((a->color != cattr.color) || (a->thick != cattr.thick)
    || (a->linestyle != cattr.linestyle) || (a->chl != cattr.chl))
    reset = 1;

  /* Recache */
  if (reset) {
    oprintf("ATTR: COLOR=%d, THICK=%d, LINESTYLE=%d, CHANNEL=%d\n",
     a->color, a->thick, a->linestyle, a->chl);
    cattr = *a;
    cattrq = 1;
  }

  oprintf("DRAW:");
  if (p0) {
    oprintf(" (X0=%d,Y0=%d)", p0->i.x, p0->i.y);
  }
  if (p1) {
    oprintf(" (X1=%d,Y1=%d)", p1->i.x, p1->i.y);
  }
  oprintf("\n");

  xdev->draw(p0, p1, a);   /* Call original X driver */
}

/* Draw hardware text */
int xfilter_text(IDL_GR_PT *p0, IDL_ATTR_STRUCT *a,
   IDL_TEXT_STRUCT *ta, char *t)
{
  int reset = 0;
  IDL_GR_PT xp0, *pp0 = 0;

  /* Determine if we need to recache graphics attributes */
  if (!cattrq) reset = 1;
  else if ((a->color != cattr.color) || (a->thick != cattr.thick)
    || (a->linestyle != cattr.linestyle) || (a->chl != cattr.chl))
    reset = 1;
```

```
  /* Reset cache */
  if (reset) {
    oprintf("ATTR: COLOR=%d, THICK=%d, LINESTYLE=%d, CHANNEL=%d\n",
     a->color, a->thick, a->linestyle, a->chl);
    cattr = *a;
    cattrq = 1;
  }

  /* Determine if we need to recache text attributes */
  reset = 0;
  if (!ctextq) reset = 1;
  else if ((ta->font != ctext.font) || (ta->axes != ctext.axes)
    || (ta->size != ctext.size) || (ta->orien != ctext.orien)
    || (ta->align != ctext.align))
    reset = 1;

  /* Recache */
  if (reset) {
    oprintf("TEXT_ATTR: FONT=%d, AXES=%d, SIZE=%f, ORIEN=%f, ALIGN=%f\n",
     ta->font, ta->axes, ta->size, ta->orien, ta->align);
    ctext = *ta;
    ctextq = 1;
  }

  oprintf("TEXT: ");
  if (p0) {
    oprintf(" (X0=%d,Y0=%d)", p0->i.x, p0->i.y);
  }
  oprintf(" '%s'\n", t);
  return xdev->text(p0, a, ta, t);   /* Call original X driver */
}

/* Erase graphics viewport */
void xfilter_erase(IDL_ATTR_STRUCT *a)
{
  int reset = 0;

  /* Determine if we need to recache graphics attributes */
  if (!cattrq) reset = 1;
  else if ((a->color != cattr.color) || (a->thick != cattr.thick)
    || (a->linestyle != cattr.thick) || (a->chl != cattr.chl))
    reset = 1;

  /* Recache */
  if (reset) {
    oprintf("ATTR: COLOR=%d, THICK=%d, LINESTYLE=%d, CHANNEL=%d\n",
     a->color, a->thick, a->linestyle, a->chl);
    cattr = *a;
```

```
    cattrq = 1;
  }
  oprintf("ERASE\n");
  xdev->erase(a);   /* Call original X driver */
}

/* Capture cursor information */
void xfilter_cursor(int funct, IDL_MOUSE_STRUCT *m)
{
  oprintf("CURSOR\n");
  xdev->cursor(funct, m);   /* Call original X driver */
}

/* Filled polygons.  Yuck */
void xfilter_polyfill(int *x, int *y, int n, IDL_POLYFILL_ATTR *poly)
{
  oprintf("POLYFILL\n");
  xdev->polyfill(x, y, n, poly);   /* Call original X driver */
}

/* Called when returning to interactive level; graphics state should be
   well defined after this call.
 */
void xfilter_inter_exit(void)
{
  oprintf("INTER_EXIT\n");
  cattrq = 0; ctextq = 0;
  xdev->inter_exit();   /* Call original X driver */
}

/* Flush any pending graphics output to the display */
void xfilter_flush(void)
{
  oprintf("FLUSH\n");
  cattrq = 0; ctextq = 0;
  xdev->flush();   /* Call original X driver */
}

/* Load new color map?  Doesn't seem to be used */
void xfilter_load_color(IDL_LONG start, IDL_LONG n)
{
  oprintf("LOAD_COLOR:%d %d\n", start, n);
  xdev->load_color(start, n);   /* Call original X driver */
}

/* Read or write a pixmap to the display */
void xfilter_rw_pixels(UCHAR *data, int x0, int y0, int nx,
        int ny, int dir, IDL_TV_STRUCT *secondary)
```

```
{
  oprintf("RW_PIXELS:%d %d %d %d %d\n", x0, y0, nx, ny, dir);
  /* Call original X driver */
  xdev->rw_pixels(data, x0, y0, nx, ny, dir, secondary);
}

/* Implement DEVICE command */
void xfilter_dev_specific(int argc, IDL_VPTR *argv, char *argk)
{
  int i;
  oprintf("DEVICE: BEGIN\n");
  for(i=0; i<argc; i++) if (argv[i]->type) IDL_Print(1, &argv[i], 0);
  oprintf("DEVICE: END\n");

  xdev->dev_specific(argc, argv, argk);   /* Call original X driver */
}

/* Implement HELP, /DEVICE */
void xfilter_dev_help(int argc, IDL_VPTR *argv)
{
  xdev->dev_help(argc, argv);   /* Call original X driver */
}

/* Called once, when device driver is installed */
void xfilter_load_rtn(void)
{
  oprintf("LOAD_RTN\n");
  /* Don't call the original driver, since it expects to be called once (?) */
  /* xdev->load_rtn(); */
}

 /********************************************************** **********/
/*                    Window   primitives                          */

/* Create Window */
void xfilter_window_create(int argc, IDL_VPTR *argv, char *argk)
{
  oprintf("WINDOW_CREATE -> ");
  xwin->window_create(argc, argv, argk);
  oprintf("%d\n", _IDL_x_dev.window);

  ctextq = 0; cattrq = 0;  /* Uncache */
}

void print_int(IDL_VPTR p)
{
  switch(p->type) {
  case IDL_TYP_BYTE:       oprintf("%dB \n", (int) p->value.c); break;
```

```
  case IDL_TYP_INT :       oprintf("%d  \n", (int) p->value.i); break;
  case IDL_TYP_LONG:       oprintf("%dL \n", (int) p->value.l); break;
  case IDL_TYP_UINT:       oprintf("%dU \n", (int) p->value.ui); break;
  case IDL_TYP_ULONG:      oprintf("%dUL\n", (int) p->value.ul); break;
  default:                 oprintf("default\n"); break;
  }
}

/* Delete Window */
void xfilter_window_delete(int argc, IDL_VPTR *argv)
{
  oprintf("WINDOW_DELETE: ");
  print_int(argv[0]);

  ctextq = 0; cattrq = 0;  /* Uncache */
  xwin->window_delete(argc, argv);
}

/* Show (activate) window */
void xfilter_window_show(int argc, IDL_VPTR *argv, char *argk)
{
  oprintf("WINDOW_SHOW: ");
  print_int(argv[0]);

  xwin->window_show(argc, argv, argk);
}

/* Set new active window */
void xfilter_window_set(int argc, IDL_VPTR *argv)
{
  oprintf("WINDOW_SET: ");
  print_int(argv[0]);

  ctextq = 0; cattrq = 0;  /* Uncache */
  xwin->window_set(argc, argv);
}

/* Window menuing (?) */
IDL_VPTR xfilter_window_menu(int argc, IDL_VPTR *argv, char *argk)
{
  oprintf("WINDOW_MENU\n");
  return xwin->window_menu(argc, argv, argk);
}

/* Initialization routine */
long int xfilter(int argc, char *argv[])
{
  IDL_SYSFUN_DEF *f;
```

```
  /* Copy device driver */
  xfilter_dev = _IDL_x_dev;
  ctextq = 0; cattrq = 0;

  /* Convenience pointers, makes things easier to type above */
  xdev  = &xfilter_dev.core;
  xfdev = &_IDL_x_dev.core;
  xwin  = &xfilter_dev.winsys;
  xfwin = &_IDL_x_dev.winsys;

  /* Set up new jump table - graphics primitives */
  xfdev->draw         = &xfilter_draw;
  xfdev->text         = &xfilter_text;
  xfdev->erase        = &xfilter_erase;
  xfdev->cursor       = &xfilter_cursor;
  xfdev->polyfill     = &xfilter_polyfill;
  xfdev->inter_exit   = &xfilter_inter_exit;
  xfdev->flush        = &xfilter_flush;
  xfdev->rw_pixels    = &xfilter_rw_pixels;
  xfdev->dev_specific = &xfilter_dev_specific;
  xfdev->dev_help     = &xfilter_dev_help;
  xfdev->load_rtn     = &xfilter_load_rtn;

  /* Set up new jump table - window primitives */
  xfwin->window_create = &xfilter_window_create;
  xfwin->window_delete = &xfilter_window_delete;
  xfwin->window_show   = &xfilter_window_show;
  xfwin->window_set    = &xfilter_window_set;
  xfwin->window_menu   = &xfilter_window_menu;

  /* Add device */
  IDL_AddDevice(&_IDL_x_dev, 1);

  return 0;
}
```
