SiliconGraphics

Pipeline

May/June 1992
Volume 3, Number 3
Bringing you the latest news and information from Silicon Graphics' Customer Support resources.

Table of Contents

2 Configuring and Using SoftPC/AT and SlavePC
3 Q&A
7 Setting Up Automount NIS Maps
19 Patch for Pre-IRIX 4.0.5 Security Problem
22 Correction

GL/X Mixed Model Programming

This is the second part of a two-part discussion on using GL rendering routines in X Window applications. The first part (in the March/April 1992 Pipeline, Vol. 3, No. 2) discussed the basic steps necessary to create mixed-model applications, mixed-model colormap applications, and using the GL widget together with the Motif widget set.

This article discusses animation using Xt work procedures, rendering to non-standard bitplanes, and GL picking. All of the discussion in this article utilize the GL widget, which simplifies much of the setup and tasks required in mixed-model programming.

The sample programs presented in this article will be available online in */usr/people/4Dgifts/examples/GLX/Pipeline* or by contacting the Technical Assistance Center.

Animation using Xt Work Procedure

In pure GL applications, animation is typically performed by checking the GL event queue using `qtest(3G)`. If there is no event waiting to be read, the code executes the next frame to be drawn. In mixed-model applications, the `qtest(3G)` function is not available. Xt toolkit has `XtAppPeekEvent(3Xt)` to test whether an event is waiting to be read from the event queue. However,

(continued on page 4)

The following article is the third of a series of articles discussing floppy access on your IRIS. The first article — "Accessing a Floppy Drive from an IRIS," published in the January/February *Pipeline* (Volume 3, Number 1) — provided a general overview of system configuration and naming conventions as well as describing how to access the floppy as an IRIX device.

The second article — "Mounting a Floppy Drive Using the `msdosd` Daemon," published in the March/April *Pipeline* (Volume 3, Number 2) — discussed `msdosd`, a daemon that Silicon Graphics provides to allow IRIS systems with floppy drives to transfer data between MS-DOS and IRIX without additional hardware or software.

This article will provide provide answers to some frequently asked questions concerning SoftPC/AT and its companion program, SlavePC along with detailed information on the capabilities of SoftPC/AT. The article focuses on using a floppy drive from SoftPC/AT.

This article assumes that you understand how to access a floppy as an IRIX device as presented in the January/February *Pipeline* article.

Answers to Common Questions

Q: What is SoftPC/AT and SlavePC?

A: SoftPC/AT is a program from Insignia Solutions, Inc. that emulates a 286 PC running MS-DOS 3.3. This emulation program allows you to run MS-DOS programs in a window on your IRIS. With some restrictions, it allows you to access floppy drives, printers and serial ports from your IRIS as though they were attached to a PC.

SlavePC, which is distributed on a 5.25-inch low density floppy with SoftPC/AT, allows you to access a floppy drive in a PC over a serial line. This is an alternative to installing a floppy drive in your IRIS.

Q: What is the current version?

A: There are two versions of SoftPC/AT currently supported for the IRIS-4D family. The first is Silicon Graphics' SoftPC/AT version 1.0.1 which displays Insignia version 2.0.1 when the program starts. The second is Silicon Graphics' SoftPC/AT version 1.1 which displays Insignia version 2.0.4 when the program starts.

Both of these versions run under IRIX 3.3.* and 4.0.*. There is no version of SoftPC/AT currently written for IRIX 4.0.*.

Q: What are the differences between the two versions?

A: There two major differences between the releases.

- Under SoftPC/AT 1.0.1, 3.5-inch floppy support was marginal. All users of SoftPC/AT that use 3.5-inch floppy drives should contact their software provider for an upgrade to SoftPC/AT 1.1.
- SoftPC/AT 1.0.1 required resetting the timer after running programs that set the timer to run at higher rates. SoftPC/AT 1.1 properly handles programs that modify the timer.

Q: What DOS software runs under SoftPC/AT?

A: Since SoftPC/AT runs MS-DOS 3.3, generally only programs that were written for MS-DOS 3.3 will work. Thus, many of the newer versions of the programs targeted for MS-DOS 4.0 or MS-DOS 5.0 will not work with SoftPC/AT. The *SoftPC/AT Release Notes* list the applications that have been tested by Insignia. If the specific version of your application does not appear on this list, it is not supported by SoftPC/AT.

Software that requires a 287 Floating Point Chip, VGA graphics, uses 80386 protected mode, or any other features that were introduced in MS-DOS 4.0 or 5.0 will probably not work with SoftPC/AT.

If you are concerned whether an application will run under SoftPC/AT, first determine if it runs under MS-DOS 3.3 on a 286-based PC (or other, relatively slow system). If the application runs on this platform, it should run under SoftPC/AT.

Keep in mind that while Silicon Graphics offers support for SoftPC/AT and SlavePC, no support is provided for MS-DOS applications.

How to Install SoftPC/AT

Like most other software packages from Silicon Graphics, SoftPC/AT is installed with the `inst` program. This is true for IRIX 3.3.* or 4.0.*.

The `inst` format has changed between IRIX 3.3.* and 4.0.*. Under IRIX 3.3.*, attempts to install a package such as SoftPC/AT that uses the new `inst` format will fail with an error indicating that it cannot read the product descriptor. Under 4.0.*, you can read the IRIX 3.3.* `inst` format as well as the IRIX 4.0.* `inst` format.

Refer to the SoftPC/AT Release Notes (IRIX 3.3.*) or the *IRIX Software Installation Guide* (4.0.*) for installation instructions.

After SoftPC/AT is installed, the next step is to make sure that your kernel is properly configured and that the UNIX floppy device files are made. If you have your *SoftPC/AT Release Notes*, refer to chapter 2 for instructions

(continued on next page)

Q: My system is not waiting on the CPU or I/O and it's memory is about 95 percent full. How can I tell that I need more memory on my system? How full is too full?

A: If you encounter one of these conditions, it's a good indication that you should add more memory or expand your swap partition:

1. **The CPU is waiting on swap I/O.** This is a performance issue.
2. **You receive processes killed due to lack of memory/swap.** This is usually fatal for the program involved.

These are separate problems. The first condition suggests that you can increase the performance of the system's operations by increasing memory. The second condition suggests that increasing swap or memory is necessary — memory is preferred, but additional swap space is far cheaper per megabyte.

Q: What commands can I use to monitor the amount of swap space and memory usage on my Silicon Graphics system?

A: `osview(1)` provides statistics about memory and swap and other system resources at regular intervals. `/etc/swap -l` provides information about swap space.

By adding the `rmem` and `swp` options to your *~/.grosview* configuration file, `gr_osview` will display real memory and swap space graphically.

Q: What is a GL widget?

A: GL widget refers to the GlxDraw (generic) and GlxMDraw (Motif) widgets that allow you to embed GL windows in Xt-based programs.

Q: I'm using the GlxMDraw widget and it doesn't seem that colors are being installed correctly. Overlays in the GlxMDraw widget are displayed in red and gray instead of the colors I specify. What is the problem?

A: The window

(continued on page 19)

Configuring and Using SoftPC/AT and SlavePC

(continued from previous page)

on configuring your kernel. If a hard copy of the Release Notes is not available, you can view them on line with the command `relnotes SoftPC 2`. Alternately, you can refer to the January/February *Pipeline* article "Accessing a Floppy Drive from an IRIS" for instructions.

SoftPC/AT does not use UNIX device names for the floppy device. Instead, it requires links to the UNIX devices. It is important to create links for both low and high density devices. If you do not create the links properly, SoftPC/AT will report an error indicating that it does not recognize the floppy drive, though `hinv` states it is present.

Using a 5.25-inch drive with SoftPC/AT 1.0.1

1. Create the links for the low and high density 5.25-inch devices with the following commands (these commands assume that the floppy drive is on unit number (SCSI id) 3). If your floppy is on a SCSI id other than 3, refer to the January/February *Pipeline* article "Accessing a Floppy Drive from an IRIS" to determine the IRIX device names.

```
# ln /dev/rdsk/fds0d3.48 /dev/pc_floppy
# ln /dev/rdsk/fds0d3.96hi /dev/pc_floppy_high
```

2. To verify that you have created the devices properly, execute the following command:

```
# ls -l /dev/*floppy*
```

You should see output similar to the following:

```
crw-rw-rw-   2 root     sys        40, 96 Dec 17
 00:38 /dev/pc_floppy

crw-rw-rw-   2 root     sys        40, 98 Dec 17
 00:38 /dev/pc_floppy_high
```

If you do not see two lines of output, you did not make both links. Repeat step 1.

Using a 5.25-inch drive with SoftPC/AT 1.1

1. Create the links for the low and high density 5.25-inch devices with the following commands (these commands assume that the floppy drive is on unit number (SCSI id) 3). If your floppy is on a SCSI id other than 3, refer to the January/February *Pipeline* article "Accessing a Floppy Drive from an IRIS" to determine the IRIX device names.

```
# ln /dev/rdsk/fds0d3.48 /dev/pc_floppy.525
# ln /dev/rdsk/fds0d3.96hi /dev/pc_floppy.525hi
```

2. To verify that you have created the devices properly, execute the following command:

```
# ls -l /dev/*floppy*
```

You should see output similar to the following:

```
crw-rw-rw-   2 root     sys        40, 96 Dec 17
 00:38 /dev/pc_floppy.525

crw-rw-rw-   2 root     sys        40, 98 Dec 17
 00:38 /dev/pc_floppy.525hi
```

If you do not see two lines of output, you did not make both links. Repeat step 1.

Using a 3.5-inch drive

If you do not have SoftPC/AT 1.1, contact your software provider for the upgrade.

1. For SoftPC/AT 1.1, create the links for the low and high density 3.5-inch devices with the following commands (these commands assume that the floppy drive is on unit number (SCSI id) 3). If your floppy is on a SCSI id other than 3, refer to the January/February *Pipeline* article "Accessing a Floppy Drive from an IRIS" to determine the IRIX device names.

```
# ln /dev/rdsk/fds0d3.3.5 /dev/pc_floppy.350
# ln /dev/rdsk/fds0d3.3.5hi /dev/pc_floppy.350hi
```

2. To verify that you have created the devices properly, execute the following command:

```
# ls -l /dev/*floppy*
```

You should see output similar to the following:

```
crw-rw-rw-   2 root     sys        40, 100 Dec 17
 00:38 /dev/pc_floppy.350

crw-rw-rw-   2 root     sys        40, 101 Dec 17
 00:38 /dev/pc_floppy.350hi
```

If you do not see two lines of output, you did not make both links. Repeat step 1.

Differences Between SoftPC/AT Running under IRIX 3.3.* and 4.0.*

SoftPC/AT works identically under IRIX 3.3.* and 4.0.* except for the minor differences listed below:

Under IRIX 4.0.*, the mouse can behave erratically when used under SoftPC/AT. This is caused by an increase in mouse events in IRIX 4.0.*. Refer to Known Bugs and Workarounds for more information.

Under IRIX 3.3.*, it is necessary to insert a MS-DOS formatted floppy in the drive before attaching the floppy drive. Under IRIX 4.0.* this is no longer true, SoftPC/AT will automatically attach the drive. If you are unsure that the drive is attached, refer to the Troubleshooting section for troubleshooting information.

Known Bugs and Workarounds

The SoftPC/AT product can functionally be broken down into two products: SoftPC/AT and SlavePC. This section will list known problems by product.

Indigo users (or users of any systems that do not have a hardware eject button or lever on their floppy drives) will need to use a paper clip to eject floppies. Insert the end of the paper clip into the small hole below the floppy slot and push firmly. This will activate a hardware switch that will eject the floppy.

SoftPC/AT

The current version of SoftPC/AT does not support VGA. SoftPC/AT emulates Hercules, CGA and EGA monitor types only.

When configuring SoftPC/AT to use a printer, it is recommended that you do not attempt to use a device directly. Instead, use the procedure documented on page 38 in the *SoftPC/AT User's Guide* to pipe the output to the IRIX spooler `lp(1)`. If you have difficulty printing even after using this method, follow the instructions on page 38 of the *SoftPC/AT User's Guide* to flush the COM and LPT ports. This procedure will most likely produce an error panel. It is safe to ignore this error.

If you try to run SoftPC/AT under IRIX 4.0, attempts to create the d: drive will fail. You must upgrade to IRIX 4.0.1. This is a problem with the IRIX 4.0 operating system, not SoftPC/AT.

You may experience problems controlling mouse pointer movement in SoftPC/AT under IRIX 4.0.*. This is caused by the fact that there are many more mouse events per second under IRIX 4.0.* than IRIX 3.3.*. To reset the rate, create (or modify) the *.Xdefaults* file in the home directory of the user(s) who will be using SoftPC/AT and add the following entry to the file:

```
*gl*MotionQGrowthRate: compress
```

To initialize the change, log out and log back in.

IRIX 4.0.* incorporates support for GL programs running on remote Silicon Graphics displays, however, SoftPC/AT does not support this feature. Thus, SoftPC/AT cannot be run remotely.

Software requiring on-board BASIC compilers does not run under SoftPC/AT, because SoftPC/AT does not emulate ROM-BASIC.

Software requiring hardware copy protection cannot be run on a floppy drive installed on a Silicon Graphics system. The SCSI drive does not contain hardware that supports this type of copy protection.

You might encounter error dialog boxes indicating that values for some variables may be incorrect when you run SoftPC/AT for the first time. SoftPC/AT sets the variables to reasonable values when you select the "Continue" option. If you select "Quit", SoftPC/AT terminates execution.

SlavePC

The PC containing the floppy drive used as a "slave" by SlavePC must be running MS-DOS 3.3. Some PCs

(continued on next page)

Configuring and Using SoftPC/AT and SlavePC

(continued from previous page)

running MS-DOS 4.0 will work. No PCs running MS-DOS 5.0 are known to work.

The PC containing the floppy drive used as a "slave" by SlavePC must have only one floppy drive. Using a PC with more than one drive will cause SlavePC to become confused. You must physically disconnect the second floppy drive from the bus.

SlavePC only runs at 9600 baud with cabling connected to COM1 on the PC being used as the "slave" by SlavePC.

Some PCs do not work with SlavePC. Slower, older PCs are more likely to work. Toshiba laptops do not work with SlavePC.

Troubleshooting

There are four areas where most users encounter problems when using SoftPC/AT. The areas are: printing, using SlavePC, running applications and attaching the floppy.

Printing

If a job does not print from an application running under SoftPC/AT, try printing a file from MS-DOS. If this is unsuccessful, refer to page 38 of the *SoftPC/AT User's Guide* for instructions on setting up SoftPC/AT to send files to the IRIX spooler `lp(1)` and for flushing the COM/LPT ports.

Using SlavePC

If you have problems connecting to a PC using SlavePC, refer to the *SoftPC/AT User's Guide* to check that you have the correct pinout for the serial cable. If you are still uncertain, try using a terminal emulation package such as PROCOMM to communicate with IRIX. If the terminal emulation package connects successfully, you know the cabling is correct. There are restrictions on the types of PCs that are best suited to be used as "slaves". Refer to **Known Bugs and Workarounds** for more information.

Running Applications

The SoftPC/AT Release Notes list the applications that have been tested by Insignia. If the specific version of your application does not appear on this list, it is not supported by SoftPC/AT. Refer to **Answers to Common Questions** for more information.

Attaching the Floppy

To use the floppy drive under IRIX 3.3.* type `SoftPC` at a shell prompt. After you see the DOS prompt (C:>), insert a DOS-formatted diskette into the drive, click the right mouse button in the SoftPC/AT window and select "Attach Floppy". This menu pulls to the right to reveal two additional selections. Choose the "Real" floppy and release the mouse button. This tells SoftPC/AT that you want to attach the floppy drive that is installed on the Silicon Graphics system. To do this, SoftPC/AT must reboot. Once it has rebooted, you should see the message `[Drive A:Attached]` in the title bar for SoftPC/AT, and your floppy drive will be accessible.

If you receive the error

```
Floppy drive not present or not properly configured
```

check hardware and software configuration, then:

- Check that there is a floppy that has been formatted with MS-DOS in the drive and that the floppy drive door is closed.
- Check that the kernel modification (see **How to Install SoftPC**) has been made and that the machine has been rebooted since the new kernel was built.
- Check that both of the floppy device links (see **How to Install SoftPC**) have been made.

To use the floppy drive under IRIX 4.0.* type `SoftPC` at a shell prompt. SoftPC/AT will start with the floppy

(continued on page 18)

Setting Up Automount NIS Maps

The following example demonstrates how to set up an automount file as an NIS (YP) map. This example assumes that automount is already installed, configured, and running on the servers and clients.

Login to the NIS master as root and create the file */usr/etc/yp/local.make.script*. Local maps should be placed in *local.make.script* which is referenced by */usr/etc/yp/make.script* when `ypmake` is run. This script is used to include /etc/*auto.master* among the NIS maps:

```
# local.make.script
#
localall:          all       auto.master

auto.master:       auto.master.time
auto.master.time:  $(DIR)/auto.master
   -@if [ -f $(DIR)/auto.master ]; then \
           sed -e '/^#/d' $(DIR)/auto.master | \
           $(MAKEDBM) - $(YPDBDIR)/auto.master; \
           touch auto.master.time; \
           echo "Updated auto.master"; \
           if [ ! $(NOPUSH) ]; then \
                 $(YPPUSH) auto.master; \
                 echo "pushed auto.master"; \
           else \
                 : ; \
           fi \
   else \
      echo "couldn't find $(DIR)/auto.master"; \
   fi
```

The file *local.make.script* must not contain leading spaces as part of the left margin. Use tabs when editing the lines in this file. If there are leading spaces instead of tabs, when `ypmake` is run, the make may fail with errors.

A sample *auto.master* file on the NIS master might contain these lines:

```
/hosts   -hosts   -rw,hard,intr
/home    /etc/auto.home   -rw,intr
```

In this set-up, the */etc/config/automount.options* file would reference the *auto.master* file with a `-f` flag.

After setting up *local.make.script*, but before running `ypmake`, check */usr/spool/cron/crontabs/root* to see when `ypxfr`(s) are occurring. Note the warning to avoid building and pushing NIS databases at the same time the NIS slaves are pulling the maps. Check that this isn't the case, then run `/usr/etc/yp/ypmake -u auto.master` and look for the last message: **"pushed auto.master"**. This indicates the maps have been built successfully.

The files *auto.master.dir* and *auto.master.pag* should now be in */usr/etc/yp/YP_DOMAINNAME*, where *YP_DOMAINNAME* is the NIS domain name as set in */usr/etc/yp/ypdomain*. When adding new maps, the NIS slave servers will not automatically pull them, so the network manager must actively start the new map on each slave server. Do this in one of two ways:

- run `ypinit -s` on each server, or
- `rcp` the new *auto.master.pag* and *auto.master.dir* files to the */usr/etc/yp/YP_DOMAINNAME* directory on each NIS server.

To check that *auto.master* map is in place, run

`ypwhich -m|grep auto.master`

and on the NIS servers you should see

`auto.master marshmallow` *(*`marshmallow` *is the name of the NIS master that owns the map).*

Next, automount will need to be restarted on machines that want automount to pick up the *auto.master* NIS map. The best way to restart automount is to reboot. If you have NIS slave servers running, you will want to reboot the slave servers first, before rebooting the clients, in case they bind to a slave.

Although `automount` references NIS by default, an administrator might change the flags in the */etc/config/automount.options* file, which disables the NIS default. If a `-m` flag is added to *automount.options*, references to entries in NIS automount maps will fail.

For more information, see `automount(1M)`, `ypmake(1M)`. Also refer to the article "An Overview of the NFS Automount Daemon in IRIX 3.3.X and IRIX 4.0" in the July/August 1991 issue of *Pipeline*.

GL/X Mixed Model Programming

(continued from page one)

using a work procedure function provides a cleaner way to execute a function when there is nothing in the event queue. The work procedure method is preferable when full speed animation is not required. The Xt polling method may be more suitable to achieve the highest frame rate.

To register a work procedure function use `XtAppAddWorkProc(3Xt)` — this function returns the work procedure id. To explicitly remove the work procedure function when it is no longer needed, use `XtRemoveWorkProc(3Xt)`. If the work procedure function returns *True*, the function is removed after it is executed once. If the function will be called repeatedly, the function must return *False*.

The following code registers an `animate` function to be called when there is nothing in the event queue:

```
XtAppContext app_context;
Boolean animate(XtPointer);

...

XtWorkProcId work_procid =
   XtAppAddWorkProc(app_context, animate, NULL);
```

To remove the function explicitly when it is no longer needed:

```
   XtRemoveWorkProc(work_procid);
```

Glwidget2.c (see Box 1 at the end of this article) is a modified version of *Glwidget.c* (presented in the previous article). The only difference in the code is that it doesn't use an input callback function to rotate the cube. Instead, it uses the `animate` function which is registered as a work procedure. The cube will start animating when the animate button is pressed and stop when the button is pressed again.

Accessing Overlay, Underlay and Popup Planes

A bug in 4.0.* mixed-model implementation currently prevents mixed-model application to access the underlay bitplanes.

In addition to normal bitplanes, some Silicon Graphics systems provide popup, overlay and underlay bitplanes that can be used to render graphics objects. All Silicon Graphics systems that have graphics hardware have two popup planes. Higher graphics options to the systems provide 0, 2, 4, or 8 overlay and underlay bitplanes. Use `getgdesc(3G)` to determine the maximum number of bitplanes supported for the overlay, underlay, or popup framebuffer.

Rendering to non-normal bitplanes is simple using the GL. Unfortunately, the X Window system does not yet have a standard way of accessing popup, overlay, and underlay planes. For information on how to render to popup and overlay bitplanes using Xlib or Xt read the discussion in the */usr/src/X11/motif/overlay_demos/README* file.

Configuring overlay and popup planes using GlxMDraw widget

The GL widget provides several resources that can be used to access the overlay bitplanes and handle overlay expose events. For a complete listing of the GlxMDraw widget resources look at the `GlxDraw(3X)` man pages.

To configure two overlay planes, set the GLXconfig structure array as follows:

```
static GLXconfig glxConfig [] =
{     { GLX_OVERLAY, GLX_BUFSIZE, 2},
   ...
   { 0, 0, 0 }
};
```

Similarly, to configure popup planes use `GLX_POPUP` in place of `GLX_OVERLAY`.

When you create a GL widget that has overlay planes, you also need to set the `GlxNuseOverlay` resource to *True* and add a callback function to the `GlxNoverlayExposeCallback` callback list.

```
...

Arg args[ 10 ];
Widget gl_widget;
void overlayExposeCB(Widget, XtPointer, XtPointer);

...
```

```
n = 0;
XtSetArg(args[n], GlxNglxConfig, glxConfig); n++;
XtSetArg(args[n], GlxNuseOverlay, True); n++;
...
gl_widget.=
   GlxCreateMDraw(parent, "gl_widget", args, n);
...
XtAddCallback(gl_widget,
              GlxNoverlayExposeCallback,
              overlayExposeCB,
              NULL);
```

To assure that the image is rendered to the overlay bitplanes, `GLXwinset(3G)` must be called before calling other GL functions inside of `overlayExposeCB`. The window field of the `GlxDrawCallbackStruct` structure will be the correct overlay window.

```
void overlayExposeCB(Widget w,
                     XtPointer client_data,
                     GlxDrawCallbackStruct* cbData)
{
   GLXwinset(XtDisplay(w), cbData->window);

   /* Starts overlay rendering here */
   ...
}
```

Another way to get the overlay window from the GL widget is to use the `GlxNoverlayWindow` resource.

```
...
Window overlay_window;
...

XtSetArg(args[ 0 ], GlxNoverlayWindow,
 &overlay_window);
XtGetValues(gl_widget, args, 1);
```

Colormap installation

The popup and overlay planes use colormaps which are different from the normal bitplane colormap. For all of an application's colormaps to be installed when it has input focus, it must use the `XSetWMColormapWindows(3X11)` function. This routine should be called only after the GL widget has been realized, because valid X windows must exist for the widget. Otherwise `XSetWMColormapWindows` will generate an X protocol error.

```
...
Window window_list[ 3 ];
...

XtRealizeWidget( ... );
...

window_list[ 0 ] = overlay_window;
window_list[ 1 ] = XtWindow(gl_widget);
window_list[ 2 ] = XtWindow(toplevel_widget);
XSetWMColormapWindows(
   XtDisplay(toplevel_widget),
   XtWindow(toplevel_widget),
   window_list,
3);
```

Gloverlay.c (See Box 2 at the end of this article) is a modified version of *Glwidget2.c* (presented previously in this article) that draws a red grid pattern in the overlay planes. By drawing the static grid in the overlay planes, the grid does not need to be redrawn during the animation.

Picking In Mixed Model

Picking code in mixed-model applications is very similar to that of a pure GL application. This section assumes that you are capable of writing simple GL picking code. For discussion of GL picking see Chapter 12 of the *Graphics Library Programming Guide*.

In pure GL, picking happens when a user clicks a mouse button, usually the left mouse button. When this happens, the application receives a LEFTMOUSE GL event. After receiving this event, the application calls the picking routine. Using the Xt Toolkit Execution Model, a mixed-model application can capture the same event through an input callback. Inside the input callback routine, you can check whether this callback is called because of the left mouse button click.

The following code shows how to check for a left mouse button event within the input callback routine. When a left mouse button event occurs, the routine calls the picking routine, `pick_test()`:

```
static void inputCB(
```

(continued on next page)

GL/X Mixed Model Programming

(continued from previous page)

```
      Widget w,
      XtPointer client_data,
      GlxDrawCallbackStruct *call_data
)
{
   XEvent *e = (XEvent *) call_data->event;
   if (e->type == ButtonPress &&
       e->xbutton.button == Button1)
   {
      pick_test();
   }
}
```

The `pick_test` function used here is pure GL code (no capital Xs anywhere).

The following code fragments are parts of a demo program that draws polygons, which have sub-parts of edges and vertices.

First establish a buffer for name lists:

```
#define NAME_BUF_SIZE 50
static long nhitlists = 0;
static short name_buffer[NAME_BUF_SIZE];
```

For clarity, identify the items returned in the name lists:

```
/* pick group layout */
#define POLY_GROUP     0
#define EDGE_GROUP     1
#define VERT_GROUP     2
/* picking results */
static char *pick_group[] = {
    "poly", "edge", "vert"
};
```

Now add the functions which perform the bulk of the drawing and picking:

```
/* prototypes */
static void pick_test(void);
static void draw_scene(void);
static void draw_items(void);
static void draw_poly(int poly_index);
static void draw_hit_lists(
   short name_buffer[], long nhitlists
);

static void pick_test(void)
{
   pushmatrix();
   picksize(8, 8);
   pick(name_buffer, NAME_BUF_SIZE);
   ortho2(WIN_L, WIN_R, WIN_B, WIN_T);
   initnames();
   draw_items();
   nhitlists = endpick(name_buffer);
   ortho2(WIN_L, WIN_R, WIN_B, WIN_T);
   popmatrix();

   draw_scene();
}

static void draw_scene(void)
{
   cpack(RGB_BLACK);
   clear();
   draw_items();
   draw_hit_lists(
      name_buffer, nhitlists
   );
   swapbuffers();
}

static void draw_items(void)
{
   loadname(0);
   pushmatrix();
      draw_poly(0);
      translate(100.0, 0.0, 0.0);
      draw_poly(1);
   popmatrix();
}

static void draw_poly(int poly_index)
{
      static float poly_pt[][2] = {
         {25.0, 25.0}, {75.0, 25.0},
         {90.0, 50.0}, {50.0, 75.0},
         {10.0, 50.0}, {25.0, 25.0},
   };
   int i;

   /* name polygon */
   pushname(POLY_GROUP);
   pushname(poly_index);

   /* draw edges in blue */
   pushname(EDGE_GROUP);
   cpack(RGB_BLUE);
   linewidth(4);
   for (i=0; i<5; i++) {
      pushname(i);
      bgnline();
         v2f(poly_pt[i+0]);
         v2f(poly_pt[i+1]);
      endline();
      popname();
```

```
    }
    linewidth(1);
    popname();

    /* draw vertices in red */
    cpack(RGB_RED);
    pushname(VERT_GROUP);
    for (i=0; i<5; i++) {
        pushname(i);
        circf(poly_pt[i][0],poly_pt[i][1],2.0);
        popname();
    }
    popname();

    popname();
    popname();
}

static void draw_hit_lists(
    short name_buffer[],
    long nhitlists
)
{
    char str[50];
    int list, nitems, index;
    int tx, ty, tdy;
    int i;

    tx = 70;
    ty = 16;
    tdy = 4;

    cpack(RGB_WHITE);
    cmov2i(tx, ty);
    sprintf(str, "%ld name list", nhitlists);
    charstr(str);
    if (nhitlists != 1)
        charstr("s");
    if (nhitlists > 0)
        charstr(": ");
    index = 0;
    for (list=0; list<nhitlists; list++) {
        ty -= tdy;
        cmov2i(tx, ty);
        sprintf(str, "list %d: ", list);
        charstr(str);
        nitems = name_buffer[index++];
        index++;  /* skip... */
        nitems--; /* ...loadname(0) */
        for (i=0; i<nitems; i+=2) {
            sprintf(str, "[%s %d] ",
            pick_group[name_buffer[index]],
            name_buffer[index+1]
            );
            charstr(str);
            index += 2;
        }
    }
}
```

Summary

Silicon Graphics has announced OpenGL as a preferred, standard 3D graphics library for its graphics hardware. Mixed-model programing loosely follows the guidelines for writing OpenGL applications allowing mixed-model programs to be easily ported to OpenGL.

```
/*
  glwidget2.c -

  Ivan Hajadi
  12-June-1992

  to compile:
  cc -o glwidget2 glwidget2.c -lXirisw \
    -lgl_s -lXm -lXt -lX11_s -lPW -lsun

*/

/* Motif includes */
#include <Xm/Xm.h>
#include <Xm/Form.h>
#include <Xm/PushB.h>

/* GL widget include */
#include <X11/Xirisw/GlxMDraw.h>


GLXconfig rgb_mode[] = {
  { GLX_NORMAL, GLX_RGB, TRUE },
  { GLX_NORMAL, GLX_DOUBLE, TRUE },
  { GLX_NORMAL, GLX_ZSIZE, GLX_NOCONFIG },
  { 0,          0,         0,    }
};

XtAppContext app_context;
XtWorkProcId work_procid = 0;

void drawscene(int, int);
```

(continued on next page)

Box 1. Glwidget2.c - an example of GL widget and a work procedure used with the Motif toolkit.

(continued from previous page)

```
main(int argc, char** argv)
{
  Widget top, form, quit_but, anim_but;
  Widget gl_widget;

  /* Prototypes */

  void quitCB(Widget, XtPointer, XtPointer);
  void ginitCB(Widget, XtPointer, XtPointer);
  void exposeCB(Widget, XtPointer, XtPointer);
  void animateCB(Widget, XtPointer, XtPointer);


  Arg args[20];
  int n;

  top =
    XtAppInitialize(&app_context, "Cube",
          NULL, 0, &argc, argv, NULL, NULL, 0);
  n=0;
  XtSetArg(args[n], XmNwidth, 730); n++;
  XtSetArg(args[n], XmNheight, 445); n++;
  form = XmCreateForm(top, "form", args, n);
  XtManageChild(form);

  /* Create GL widget */

  n = 0;
  XtSetArg(args[n], XmNbottomAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNleftAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNrightAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNtopAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNleftOffset, 20); n++;
  XtSetArg(args[n], XmNrightOffset, 120); n++;
  XtSetArg(args[n], XmNbottomOffset, 40); n++;
  XtSetArg(args[n], XmNtopOffset, 40); n++;
  XtSetArg(args[n], XmNwidth, 500); n++;
  XtSetArg(args[n], XmNheight, 400); n++;
  XtSetArg(args[n], GlxNglxConfig,
       rgb_mode); n++;
  gl_widget =
    GlxCreateMDraw(form, "gl_widget", args, n);
  XtManageChild(gl_widget);

  /* Add callbacks */

  XtAddCallback(gl_widget, GlxNginitCallback,
        ginitCB, NULL);
  XtAddCallback(gl_widget, GlxNexposeCallback,
        exposeCB, NULL);

  n=0;
  XtSetArg(args[n], XmNrightAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNrightOffset, 10); n++;
  XtSetArg(args[n], XmNbottomAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNbottomOffset, 10); n++;
  XtSetArg(args[n], XmNtraversalOn, False); n++;
  quit_but =
    XmCreatePushButton(form, " Quit ", args, n);
  XtManageChild(quit_but);

  n=0;
  XtSetArg(args[n], XmNrightAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNrightOffset, 10); n++;
  XtSetArg(args[n], XmNbottomAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNbottomOffset, 50); n++;
  XtSetArg(args[n], XmNtraversalOn, False); n++;
  anim_but =
    XmCreatePushButton(form, "Rotate", args, n);
  XtManageChild(anim_but);

  XtAddCallback(quit_but, XmNactivateCallback,
        quitCB, NULL);
  XtAddCallback(anim_but, XmNactivateCallback,
        animateCB, NULL);

  XtRealizeWidget(top);
  XtAppMainLoop(app_context);
}


Boolean animate(XtPointer clientD)
{
  static int angle_x = 0, angle_y = 0;
  angle_y += 75;
  angle_x += 75;

  if ((angle_y >= 3600) || (angle_y <= -3600))
    angle_y = 0;
  if ((angle_x >= 3600) || (angle_x <= -3600))
    angle_x = 0;

  drawscene(angle_x, angle_y);

  /*
     Return False so this WorkProc
     keeps getting called.
  */
  return False;
```

(continued on next page)

```
}


void animateCB(Widget w,
          XtPointer clientD,
          XtPointer callD)
{
  static Boolean started = False;
  Arg args[ 1 ];
  XmString xms;

  if (! started ) {
    Boolean animate(XtPointer);
    work_procid =
      XtAppAddWorkProc(app_context,
               animate, NULL);
    started = True;
    xms = XmStringCreateSimple(" Stop ");
  } else {
    XtRemoveWorkProc(work_procid);
    started = False;
    xms = XmStringCreateSimple("Rotate");
  }

  /* Changed the button label */
  XtSetArg(args[ 0 ], XmNlabelString, xms);
  XtSetValues(w, args, 1);
  XmStringFree(xms);
}


void ginitCB(Widget w,
         XtPointer clientD,
         XtPointer callD)
{
  GlxDrawCallbackStruct *call_data =
     (GlxDrawCallbackStruct *) callD;

  GLXwinset(XtDisplay(w), call_data->window);

  shademodel(GOURAUD);
  zbuffer(TRUE);
  subpixel(TRUE);
  lsetdepth(getgdesc(GD_ZMIN),
        getgdesc(GD_ZMAX));
  mmode(MVIEWING);
  perspective(450, 1, 1, 100);
}


void exposeCB(Widget w,
         XtPointer clientD,
         XtPointer callD)
{
  GLXwinset(XtDisplay(w), cbData->window);
  drawscene();
}


void quitCB(Widget w,
        XtPointer clientD,
        XtPointer callD)
{
  exit(0);
}


void drawscene(int angle_x, int angle_y)
{

  /* Draw smooth-shaded cube */

  static long v1[4][3] = {
    {-10, -10, 10},
    {10, -10, 10},
    {10, 10, 10},
    {-10, 10, 10},
  };

  static long v2[4][3] = {
    {-10, -10, -10},
    {-10, 10, -10},
    {10, 10, -10},
    {10, -10, -10},
  };

  static long colors1[] = {
    0x000000FF,
    0x00FF0000,
    0x0000FF00,
    0x00FFFF00,
  };
  static long colors2[] = {
    0x0000FFFF,
    0x00FF00FF,
    0x00FFFFFF,
    0x0FF00000,
  };

  register int i;

  reshapeviewport();
  czclear(0x00777777, getgdesc(GD_ZMAX));
  pushmatrix();
  polarview(80, 0, 250, 0);

  pushmatrix();
  rotate(angle_x, 'x');
  rotate(angle_y, 'y');
```

(continued on next page)

(continued from previous page)

```
bgnpolygon();
   for(i=0; i<4; i++) {
  cpack(colors1[i]);
  v3i(v1[i]);
   }
endpolygon();
bgnpolygon();
   for(i=0; i<4; i++) {
  cpack(colors2[i]);
  v3i(v2[i]);
   }
endpolygon();


bgnpolygon();
   cpack(colors1[1]);
   v3i(v1[1]);
   cpack(colors1[2]);
   v3i(v1[2]);
   cpack(colors2[2]);
   v3i(v2[2]);
   cpack(colors2[3]);
   v3i(v2[3]);
endpolygon();
bgnpolygon();
   cpack(colors1[0]);
   v3i(v1[0]);
   cpack(colors1[3]);
   v3i(v1[3]);
   cpack(colors2[1]);
   v3i(v2[1]);
   cpack(colors2[0]);
   v3i(v2[0]);
endpolygon();


bgnpolygon();
   cpack(colors1[2]);
   v3i(v1[2]);
   cpack(colors1[3]);
   v3i(v1[3]);
   cpack(colors2[1]);
   v3i(v2[1]);
   cpack(colors2[2]);
   v3i(v2[2]);
endpolygon();
bgnpolygon();
   cpack(colors1[1]);
   v3i(v1[1]);
   cpack(colors1[0]);
   v3i(v1[0]);
   cpack(colors2[0]);
   v3i(v2[0]);
   cpack(colors2[3]);
   v3i(v2[3]);
endpolygon();


popmatrix();
popmatrix();

swapbuffers();
}
```

```
/*
  gloverlay.c -

  Ivan Hajadi
  12-June-1992

  to compile:
  cc -o gloverlay gloverlay.c -lXirisw \
    -lgl_s -lXm -lXt -lX11_s -lPW -lsun

*/

/* Motif includes */
#include <Xm/Xm.h>
#include <Xm/Form.h>
#include <Xm/PushB.h>

/* GL widget include */
#include <X11/Xirisw/GlxMDraw.h>

#include <stdio.h>


GLXconfig rgb_mode[] = {
  { GLX_NORMAL, GLX_RGB, TRUE },
  { GLX_NORMAL, GLX_DOUBLE, TRUE },
  { GLX_NORMAL, GLX_ZSIZE, GLX_NOCONFIG },
  { GLX_OVERLAY, GLX_BUFSIZE, 2},
  { 0,           0,         0,    }
};

/* globals */
XtAppContext app_context;
```

(continued on next page)

Box 2. Gloverlay.c - a modified version of Glwidget2.c that draws a red grid in the overlay planes.

```
XtWorkProcId work_procid = 0;
Display *display;
Window normalWindow;
Window overlayWindow;

void drawscene(int, int);
void drawOverlayScene();
void installColormaps(Widget, Widget);

main(int argc, char** argv)
{
  Widget top, form, quit_but, anim_but;
  Widget gl_widget;
  int overlayPlns;

  /* Prototypes */

  void quitCB(Widget, XtPointer, XtPointer);
  void ginitCB(Widget, XtPointer, XtPointer);
  void exposeCB(Widget, XtPointer, XtPointer);
  void overlayExposeCB
    (Widget, XtPointer, XtPointer);
  void animateCB(Widget, XtPointer, XtPointer);

  Arg args[20];
  int n;

  top =
    XtAppInitialize(&app_context, "Cube",
          NULL, 0, &argc, argv, NULL, NULL, 0);
  n=0;
  XtSetArg(args[n], XmNwidth, 730); n++;
  XtSetArg(args[n], XmNheight, 445); n++;
  form = XmCreateForm(top, "form", args, n);
  XtManageChild(form);

  /* Create GL widget */

  /* Check if we have enough overlay planes */
  overlayPlns =
    getgdesc(GD_BITS_OVER_SNG_CMODE);

  if (! overlayPlns ) {
    fprintf(stderr, "No overlay planes.\n");
    exit(1);
  }


  n = 0;
  XtSetArg(args[n], XmNbottomAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNleftAttachment,
       XmATTACH_FORM); n++;
  XtSetArg(args[n], XmNrightAttachment,
       XmATTACH_FORM); n++;
```

```
XtSetArg(args[n], XmNtopAttachment,
     XmATTACH_FORM); n++;
XtSetArg(args[n], XmNleftOffset, 20); n++;
XtSetArg(args[n], XmNrightOffset, 120); n++;
XtSetArg(args[n], XmNbottomOffset, 40); n++;
XtSetArg(args[n], XmNtopOffset, 40); n++;
XtSetArg(args[n], XmNwidth, 500); n++;
XtSetArg(args[n], XmNheight, 400); n++;
XtSetArg(args[n], GlxNglxConfig,
     rgb_mode); n++;
XtSetArg(args[n], GlxNuseOverlay, True); n++;

gl_widget =
  GlxCreateMDraw(form, "gl_widget", args, n);
XtManageChild(gl_widget);

/* Add callbacks */

XtAddCallback(gl_widget, GlxNginitCallback,
      ginitCB, NULL);
XtAddCallback(gl_widget, GlxNexposeCallback,
      exposeCB, NULL);
XtAddCallback(gl_widget,
      GlxNoverlayExposeCallback,
      overlayExposeCB, NULL);

n=0;
XtSetArg(args[n], XmNrightAttachment,
     XmATTACH_FORM); n++;
XtSetArg(args[n], XmNrightOffset, 10); n++;
XtSetArg(args[n], XmNbottomAttachment,
     XmATTACH_FORM); n++;
XtSetArg(args[n], XmNbottomOffset, 10); n++;
XtSetArg(args[n], XmNtraversalOn, False); n++;
quit_but =
  XmCreatePushButton(form, " Quit ", args, n);
XtManageChild(quit_but);

n=0;
XtSetArg(args[n], XmNrightAttachment,
     XmATTACH_FORM); n++;
XtSetArg(args[n], XmNrightOffset, 10); n++;
XtSetArg(args[n], XmNbottomAttachment,
     XmATTACH_FORM); n++;
XtSetArg(args[n], XmNbottomOffset, 50); n++;
XtSetArg(args[n], XmNtraversalOn, False); n++;
anim_but =
  XmCreatePushButton(form, "Rotate", args, n);
XtManageChild(anim_but);

XtAddCallback(quit_but, XmNactivateCallback,
      quitCB, NULL);
XtAddCallback(anim_but, XmNactivateCallback,
      animateCB, NULL);
```

(continued on next page)

(continued from previous page)

```
  XtRealizeWidget(top);

  /* Time to install the colormaps */
  installColormaps(top, gl_widget);

  /* Main Loop */
  XtAppMainLoop(app_context);
}


void installColormaps(Widget toplevel_widget,
              Widget gl_widget)
{
  Window window_list[ 3 ];

  display = XtDisplay(toplevel_widget);
  window_list[ 0 ] = overlayWindow;
  window_list[ 1 ] = XtWindow(toplevel_widget);
  window_list[ 2 ] = XtWindow(gl_widget);
  XSetWMColormapWindows(
    display,
    XtWindow(toplevel_widget),
    window_list,
    3);
}


Boolean animate(XtPointer clientD)
{
  static int angle_x = 0, angle_y = 0;
  angle_y += 75;
  angle_x += 75;

  if ((angle_y >= 3600) || (angle_y <= -3600))
    angle_y = 0;
  if ((angle_x >= 3600) || (angle_x <= -3600))
    angle_x = 0;

  GLXwinset(display, normalWindow);
  drawscene(angle_x, angle_y);

  /*
     Return False so this WorkProc
     keeps getting called.
  */
  return False;
}


void animateCB(Widget w,
           XtPointer clientD,
           XtPointer callD)
{
  static Boolean started = False;
  Arg args[ 1 ];
  XmString xms;

  if (! started ) {
    Boolean animate(XtPointer);
    work_procid =
      XtAppAddWorkProc(app_context,
                 animate, NULL);
    started = True;
    xms = XmStringCreateSimple(" Stop ");
  } else {
    XtRemoveWorkProc(work_procid);
    started = False;
    xms = XmStringCreateSimple("Rotate");
  }

  /* Changed the button label */
  XtSetArg(args[ 0 ], XmNlabelString, xms);
  XtSetValues(w, args, 1);
  XmStringFree(xms);
}


void ginitCB(Widget gl_widget,
         XtPointer clientD,
         XtPointer callD)
{
  GlxDrawCallbackStruct *call_data =
     (GlxDrawCallbackStruct *) callD;

  /* Get the overlay window */
  Arg args[ 1 ];

  XtSetArg(args[ 0 ], GlxNoverlayWindow,
       &overlayWindow);
  XtGetValues(gl_widget, args, 1);

  display = XtDisplay(gl_widget);
  normalWindow = call_data->window;
  GLXwinset(display, normalWindow);

  shademodel(GOURAUD);
  zbuffer(TRUE);
  subpixel(TRUE);
  lsetdepth(getgdesc(GD_ZMIN),
       getgdesc(GD_ZMAX));
  perspective(450, 1, 1, 100);

  /* Setup overlay colormap entries */
  GLXwinset(display, overlayWindow);
  mapcolor(1, 255, 0, 0); /* red */
}
```

(continued on next page)

```
void exposeCB(Widget w,
          XtPointer clientD,
          GlxDrawCallbackStruct* cbData)
{
  GLXwinset(XtDisplay(w), cbData->window);
  animate(NULL);
}

void overlayExposeCB(Widget w,
          XtPointer clientD,
          GlxDrawCallbackStruct* cbData)
{
  GLXwinset(XtDisplay(w), cbData->window);
  drawOverlayScene(cbData->width,
            cbData->height);
}

void quitCB(Widget w,
        XtPointer clientD,
        XtPointer callD)
{
  exit(0);
}

void drawOverlayScene(int w, int h)
{
  int xoffset = w/5;
  int yoffset = h/5;
  int x1=0,y1=0, x2=0,y2=0;
  register int i,j;

  pushmatrix();
  ortho2(-0.1, (Coord)w+0.1,
     -0.1, (Coord)h+0.1);
  reshapeviewport();
  color(0);
  clear();

  color(1); /* red */

  for (i=0;i<5;i++) {
    x2 = xoffset;
    y2 += yoffset;
    for (j=0;j<5;j++) {
      recti(x1,y1, x2,y2);
      x1 += xoffset;
      x2 += xoffset;
    }
    x1 = 0;
    y1 += yoffset;
  }
  popmatrix();
}
```

```
void drawscene(int angle_x, int angle_y)
{
  /* Draw smooth-shaded cube */

  static long v1[4][3] = {
    {-10, -10, 10},
    {10, -10, 10},
    {10, 10, 10},
    {-10, 10, 10},
  };

  static long v2[4][3] = {
    {-10, -10, -10},
    {-10, 10, -10},
    {10, 10, -10},
    {10, -10, -10},
  };

  static long colors1[] = {
    0x000000FF,
    0x00FF0000,
    0x0000FF00,
    0x00FFFF00,
  };
  static long colors2[] = {
    0x0000FFFF,
    0x00FF00FF,
    0x00FFFFFF,
    0x0FF00000,
  };

  register int i;

  reshapeviewport();
  czclear(0x00777777, getgdesc(GD_ZMAX));
  pushmatrix();
  polarview(80, 0, 250, 0);

  pushmatrix();
  rotate(angle_x, 'x');
  rotate(angle_y, 'y');


  bgnpolygon();
     for(i=0; i<4; i++) {
    cpack(colors1[i]);
    v3i(v1[i]);
     }
  endpolygon();
  bgnpolygon();
     for(i=0; i<4; i++) {
    cpack(colors2[i]);
    v3i(v2[i]);
```

(continued on next page)

(continued from previous page)

```
        }
    endpolygon();


    bgnpolygon();
        cpack(colors1[1]);
        v3i(v1[1]);
        cpack(colors1[2]);
        v3i(v1[2]);
        cpack(colors2[2]);
        v3i(v2[2]);
        cpack(colors2[3]);
        v3i(v2[3]);
    endpolygon();
    bgnpolygon();
        cpack(colors1[0]);
        v3i(v1[0]);
        cpack(colors1[3]);
        v3i(v1[3]);
        cpack(colors2[1]);
        v3i(v2[1]);
        cpack(colors2[0]);
        v3i(v2[0]);
    endpolygon();
    bgnpolygon();
        cpack(colors1[2]);
        v3i(v1[2]);
        cpack(colors1[3]);
        v3i(v1[3]);
        cpack(colors2[1]);
        v3i(v2[1]);
        cpack(colors2[2]);
        v3i(v2[2]);
    endpolygon();
    bgnpolygon();
        cpack(colors1[1]);
        v3i(v1[1]);
        cpack(colors1[0]);
        v3i(v1[0]);
        cpack(colors2[0]);
        v3i(v2[0]);
        cpack(colors2[3]);
        v3i(v2[3]);
    endpolygon();


    popmatrix();
    popmatrix();

    swapbuffers();
}
```

Configuring and Using SoftPC/AT and SlavePC

(continued from page 6)

automatically attached. If the floppy drive attached properly, you should see the the message `[Drive A:Attached]` in the title bar for SoftPC/AT, and your floppy drive will be accessible.

If you receive the error

`Floppy drive not present or not properly configured`

check hardware and software configuration, then do the following:

- Check that the kernel modification (see **How to Install SoftPC**) has been made and that the machine has been rebooted since the new kernel was built.
- Check that both of the floppy device links (see **How to Install SoftPC**) have been made.
- Check that the daemon `msdosd(1M)` is not running. `msdosd` is a daemon that monitors the floppy drive. If a floppy formatted with MS-DOS is inserted into the drive, `msdosd` mounts it and allows access to the DOS file system using IRIX commands.

To determine if `msdosd` is running, look at the system process list with the command:

```
ps -eaf|grep msdosd|grep -v grep
```

and look for the program `msdosd`. If it is running, you will need to terminate it (refer to the man page for `msdosd(1M)` for instructions), and restart SoftPC/AT.

Documentation

For more information on the installation, configuration and use of SoftPC/AT, please refer to *SoftPC/AT User's Guide*, and the *SoftPC/AT Release Notes* (also available on-line with the command `relnotes SoftPC`).

Checkpoint:

Patch for Pre-IRIX 4.0.5 Security Problem

The following is the description and a patch of a security problem found in all versions of IRIX except IRIX 4.0.5 (and later) and Trusted IRIX/B.

The problem is due to Silicon Graphics' configuration of the standard system utilities of the operating system and not a problem inherent in having an open (no password) "lp" account. Other UNIX versions derived from early S5R3 releases may or may not have similar problems.

The patch requires changing the modes of some files to remove set[gu]id bits and alter write permissions. No files need to be replaced. To implement the patch, follow the instructions below.

Problem Description

A vulnerability exists such that IRIX pre-4.0.5 systems with the basic system software (*eoe1.sw.unix*) or the system manager software (*eoe2.sw.vadmin*) installed can allow unauthorized access to the superuser account by exploiting a configuration error in standard system utilities. Due to the ease of exploiting this vulnerability and the simplicity of the corrective action, the CERT/CC urges all sites to install the patch given below.

Impact

Anyone who can login as (or `su` to) the user "lp" can become root on any pre-IRIX 4.0.5 system. As IRIX is normally distributed, this includes any ordinary user.

Solution

As "root" execute the following commands:

```
cd /usr/lib
chmod a-s,go-w lpshut lpmove accept reject lpadmin
chmod go-ws lpsched vadmin/serial_ports
vadmin/users vadmin/disks
cd /usr/bin
chmod a-s,go-w disable enable
chmod go-ws cancel lp lpstat
```

If the *eoe2.sw.vadmin* software is not installed, you may get messages like:

```
chmod: WARNING: can't access vadmin/serial_ports
```

These messages may be ignored if they occur.

If system software should ever be reloaded from pre-4.0.5 media or from a backup tape created before the patch was applied, repeat the above procedure immediately after the software has been reloaded — before enabling logins by normal users.

Q&A

(continued from page 3)

manager must be directed to install all appropriate colormaps. Use the `XSetWMColormapWindows()` call to do this — list one window for each colormap to be installed plus the top level window. If using overlays, include the overlay window as well as the normal GL window.

If your window is TrueColor, you should still install the appropriate colormap, as Indigo TrueColor is simulated using a colormap.

Q: When I close an X client (xman, xterm) window I get the following error:

```
XIO: fatal IO error 32 (Broken pipe) on X
 server ":0.0" after 214 requests (214 known
 processed) with 0 events remaining.
The connection was probably broken by a server
 shutdown or KillClient.
```

What does this error mean? Is there something wrong with my system?

A: The Window Manager is reporting that the pipe for your X client has been broken. This is normal when an X client exits. There is nothing wrong with your system. This

(continued on next page)

Q&A

(continued from previous page)

message will also occur when the system is shut down. This is not an error condition and no core files are produced.

Q: When issuing the `cu %take` command the file is not transferred, and I receive the following diagnostics:

```
stty -echo;if test -r myfile;
then (echo '~>':myfile; cat myfile;echo '~>');
else echo cant\'t open: myfile; fi;stty echo
Badly placed ()'s.
```

What is the problem?

A: The `cu %take` command is written assuming that /bin/sh is used on the remote side. The command will work if you type `sh` on the remote system before issuing the command, and then exiting the subshell after the transfer completes.

Q: I've read that GL TIMER events in IRIX 3.3.* are not available to programs using `select()` on the queue fd under IRIX 4.0.*. I want to maintain the functionality of using `select()` and periodically wake up the program as I did with TIMER0 at a NOISE setting of around 30. What are the least invasive workarounds?

A: If you want to wake the program up periodically, you can use timeouts in `select()`. If the `select()` times out, you can perform a `qenter()` of TIMER0 before executing the event processing code.

If you want to have precise control over when the program wakes up (i.e. at a certain time), it is a bit more complicated, but straightforward. In this case, you must add bookkeeping calling `gettimeofday()`, or a similar function, to compute how large the `select()` call timeout should be.

Q: Is there a way to switch on the fly between single and double buffering within a GL widget? I would like to use double buffering while rotating an object and switch to single buffering (which produces a better image) when the rotation is halted.

A: There is no way to switch between single and double buffered mode within the same window. (In X, this would mean changing the depth, which is not allowed.)

To simulate this, you can create two GL widgets: a double buffered GL widget and a single buffered GL widget. When it is time to change the buffering mode, restack the windows so that the appropriate one is on top. If the two windows are created with a common parent, they move together when the parent is moved.

For non-mixed mode applications, the GL performs this switching when you call `gconfig()`.

Q: When we use VideoCreator to display live, uncompressed video, the image displayed is stretched vertically. What is the cause of this?

A: The problem is due to a design limitation of VideoCreator.

When VideoCreator displays in "live, uncompressed" mode, rather than passing through a 768 x 576 window of the high resolution video (as you might expect), the VideoCreator disables its vertical compression algorithm (which is implemented in digital logic), but not its horizontal compression algorithm (which is implemented as an analog filter).

The resulting scan conversion looks as though its stretched vertically, because all the pixels are compressed horizontally. If you paste a 768 x 576 image onto your high resolution screen's upper left corner and perform `vcsetmode lu`, you will see that all 576 lines of the image are being output as low resolution video, but that additional horizontal pixel information (from the high resolution screen) is contributing to the low resolution video output.

Page 1-5 of your *VideoCreator Programmer's Guide* states the following:

The VideoCreator architecture allows for a fourth mode, "live, uncompressed," although it is unlikely that this mode is useful for most applications. In live, uncompressed mode, the high resolution image is displayed one-to-one vertically and compressed horizontally.

Q: I am using NetVisualyzer 1.2.1 as a non-root user and I have modified the file */usr/etc/rpc.snoopd.auth* to include the following lines:

```
accept localhost:root
accept *:george
```

I can run `netlook`, `analyzer`, `netgraph`, `netcollect` and `netaccount`, but when I try to run `netsnoop` I get the following error message:

```
netsnoop: cannot snoop on default interface:
  Permission denied.
```

The command works if I am logged in as root. Why?

A: By default, `netsnoop` directly accesses the network interface for maximum efficiency. This can only be accomplished as root.

The */usr/etc/rpc.snoopd.auth* file is read only by `snoopd`. In order to use `netsnoop` as a user that is authorized in this file, `netsnoop` must connect with `snoopd`. To make `netsnoop` connect to `snoopd` on the local machine, use the command:

```
% netsnoop -i localhost:
```

This tells `netsnoop` to capture from the default interface on `localhost`. The full format of the `-i` flag is *<hostname>*`:`*<interface>*. In this case, `localhost` is the *<hostname>* and the *<interface>* is not specified, so the default interface will be used.

Box 1 shows an example showing capturing a packet from a remote machine with multiple interfaces (FDDI and Ethernet) via `snoopd`.

```
% rsh acadia -l guest /usr/etc/netstat -i
Name Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
ec0  1500  b9U-ng      acadia.wpd.sgi.   181947  3632     2144     0    60
ipg0 4352  wpd-fddi    fddi-acadia.wpd    45583     1     2460     4     0
lo0  32880 loopback    localhost            953     0      953     0     0
% netsnoop -i acadia:ipg0 -c 1
0000:   len  164   time 15:06:50.833
    fddi:   src 8:0:69:4:0:8a/SGI         dst 8:0:69:4:0:13/SGI
    ip:     src 192.48.198.63             dst bonnie.wpd.sgi.com
    udp:    sport 1022                    dport 2049 (sunrpc)
    sunrpc: xid 918155         direction CALL      credtype AUTH_UNIX
    nfs:    proc GETATTR

% netsnoop -i acadia:ec0 -c 1
0000:   len  566   time 15:07:01.278
    ether:  src 8:0:69:2:4:45/SGI         dst 8:0:69:6:66:84/SGI
    ip:     src england.wpd.sgi.com       dst lilac.wpd.sgi.com
    tcp:    sport 514 (shell)             dport 1020


%
```

Box 1. Example showing capturing a packet from a remote machine with multiple interfaces via `snoopd`.

Q: I am trying to port a program from Sun's XGL to GL and I want to create a triangle strip with facet normals specified. The GL specifies normals modally (with the `n3f()` and `normal()` function). This won't work with triangle strips since every point is shared by 2 or 3 triangles. Because I am building a box using one strip, I don't want common normals between faces. Thus, I cannot use vertex info and gouraud shading. How would I use the GL in this case?

A: Change the shademodel to FLAT and triangle N will be shaded entirely based on the lighting calculation of vertex N+2 (counting triangles and vertexes from 1). This approach fails if you wish to interpolate material properties, but use only a single normal, or if lighting calculations take vertex position into account.

(continued on next page)

Correction

In the March/April 1992 *Pipeline* the mixed model article described a method to obtain pointer motion events within a GL widget without the need of having a mouse button pressed. The solution provided was incorrect. The event mask for the GL widget was modified in the following way (in the GL widget initialization callback):

```
XSelectInput(
    XtDisplay(gl_widget),
    XtWindow(gl_widget),
    XtBuildEventMask(gl_widget)
    | PointerMotionMask
);
```

This adds to the events the GL widget reacts to, but there is no statement of how to react. This can be done with a translation augmentation which maps the event to invoke the standard GL widget input callback:

```
/* after gl widget creation... */
XtAugmentTranslations(
    gl_widget,
    XtParseTranslationTable(
    "<MotionNotify>: glxInput()"
    )
);
```

Both steps are needed to correctly receive MotionNotify events.

Q&A

(continued from previous page)

Q: How do I create a transparent X window?

A: Under IRIX 4.0.*, you can open X windows in the overlay bitplanes, where color index 0 is transparent.

Q: What are the exact sampling frequencies on the audio port? Does 48K = 48000 or 48 * 1024?

A: The seven sampling frequencies supported by SGI hardware and software are:

- 8000 samples/second
- 11025 samples/second
- 16000 samples/second
- 22050 samples/second
- 32000 samples/second
- 44100 samples/second (CD rate)
- 48000 samples/second

Q: I am a system administrator using automount under IRIX 3.3.* to mount users' home directories. When visuallogin checks a user's home directory for a picture, it causes that home directory to be mounted via automount. During login, each time someone clicks More the visuallogin re-reads all of the home directories. Is there a way of disabling this feature without disabling visuallogin?

A: Under IRIX 4.0.1, you may disable searching for icons if the home directory starts with a particular string. The specification resides in */etc/passwd.sgi*. There is no way to do this under IRIX 3.3.

For example, the following lines:

```
/hosts:ignore
/home:ignore
```

will flag visuallogin not to search for pictures (and perhaps more importantly, no accesses will be made to the home directories) if the home directory pathname starts with either */hosts* or */home*.

Also, no regular icon will be displayed for all users whose home directories start with either of the strings. You can still type those users' names into the text field.

You can also disable the display of an icon (of either type) on a per-user basis with the following form; this method works in IRIX 3.3 as well.

```
daemon:noshow
```

See `man pandora` for details.

Q: I timed how long it took to do an `fx`/exercise on the

SCSI disks of a 4D/35 and a 4D/240 with an IO2 and a 4D/25. The SCSI performance on the 4D/35 was about twice as fast as the 4D/240 and the 4D/25 was somewhere in-between. Does anybody have the real SCSI I/O performance figures?

Do the results of my test mean that a 4D/35, or even a 4D/25, would make a better NFS server than a POWER Series system?

A: The peak transfer rate on the 4D/25 and 4D/35 should be 5 MB(Megabyte)/sec. On systems with an IO2, it is approximately 1 MB/sec. On systems with an IO3 it is 4 MB/sec. into memory (disk read) and 5 MB/sec. out of memory (disk write). On VME-SCSI (jaguar) systems, 01D Firmware is 4 MB/sec. and 01J Firmware is 4.72 MB/sec.

The above are peak transfer rates and assume that the VME board is a fast one. Since VME is asynchronous REQ-ACK-NREQ-NACK protocol, board speed is just as important as the bus speed on the Silicon Graphics system.

IO2 and IO3 are the I/O boards on POWER Series systems (4D/100 and up). The IO3 is not tested on 4D/100 systems. Crimson systems use an IO3, but not an IO2. The VME-SCSI is unsupported on the 4D/25 or 4D/35 systems.

Since Ethernet is quite a bit slower than all of the speeds above (except for the IO2), SCSI bandwidth should not be a factor in NFS performance. If you are using FDDI, you want to use an IO3, since its VME performance is much better than other systems' as shown here:

- VME on IO3 maxes out at approx. 32 MB/sec.
- VME on IO2 maxes out at approx. 10 MB/sec.
- VME on 4D/35 maxes out at approx. 8 MB/sec.
- VME on 4D/25 maxes out at approx. 3 MB/sec.

In addition, much of NFS activity is in relatively small reads and writes, where disk seek time and latency may be more of an issue than SCSI bus maximum performance.

SiliconGraphics

Pipeline

Editor: Peter Lonsky

Contributors: Kurt Akeley, Brett Bainter, David Blythe, Brendan Eich, Ivan Hajadi, John Hallesy, Jeremy Higdon, Ronald Jacoby, Phil Karlton, Gints Klimanis, Dave Olson, Joyce Richards, Kathy Roy, Michael Sanchez, Robert Skinner, Joel Tesler, Donna Derby Yobs

Published bi-monthly by the Customer Support Division of Silicon Graphics, Inc. for support contract customers.

The *Pipeline* welcomes your comments. Please contact us with any questions or suggestions you have regarding the newsletter. Send your correspondence to:

Silicon Graphics, Inc.
P.O. Box 7311, M/S 12-190
Mountain View, CA 94039-7311
Attn: Peter Lonsky

e-mail address:
pipeline@sgi.com

In the U.S. and Canada, for technical assistance and other support services, Customer Education and customer information call the Silicon Graphics Customer Center at:
(800) 800-4SGI

For customer support overseas, call the Silicon Graphics office nearest you:

Australia — Brisbane
Phone: (61) 7 - 257 .11.94
Australia — Melbourne
Phones: 008 - 335 430
(61) 3 - 882.82.11 (in Melbourne)
Australia — Sydney
Phones: 008 - 251 073
(61) 2 - 879.95.00 (in Sydney)
Belgium — Brussels
Phone: (32) 2 - 675.21.10
Denmark — Copenhagen
Phone: (45) 31 - 95.00.88
Finland — Helsinki
Phone: (358) 0 - 4354.2071
France — Paris
Phone: (33) 1 - 34.65.96.85
Germany — Berlin
Phone: (49) 30 - 823.10.42
Germany — Karlsruhe
Phone: (49) 721 - 96.20.70
Germany — Köln-Niehl
Phone: (49) 221 - 71.52.40
Germany — Munich
Phone: (49) 89 - 461.08.0
Holland — De Meern
Phone: (31) 34.06 - 21.711
Hong Kong
Phone: (852) 5 - 25.72.37
Italy — Milan
Phone: (39) 2 - 57.51.01.08
Israel — Tel Aviv
Phone: (972) 3 - 49.21.91
Japan — Kawasaki
Phone: (81) 44 - 812.60.60
Latin America — Pompano Beach
Phone: (1) 305 - 785.28.62
FAX: (1) 305 - 781.95.79
Norway — Oslo
Phone: (47) 2 - 73.20.15
Singapore
Phone: (65) 776.0970
Spain — Madrid
Phone: (34) 1 - 572.03.60
Sweden — Stockholm
Phone: (46) 8 - 33.07.05
Switzerland — Lausanne/Vidy
Phone: (41) 21 - 25.94.12
Switzerland — Zurich
Phone: (41) 1 - 731.10.70
Taiwan — Taipei
Phone: (886) 2 - 393.41.88
United Kingdom — Reading
Phone: 0734 - 30.63.63 (s/w ?)
0734 - 30.60.40 (other ?)

SiliconGraphics
Computer Systems
P.O. Box 7311, M/S 12-134
Mountain View, CA 94039-7311

Address correction requested

If you have received this newsletter, but do not have a Silicon Graphics workstation, please pass it along to a member of your organization that does. Send inquiries (in the U.S. and Canada) to the address listed on the previous page. ***Send address corrections to the address above.***

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 913
SDS
MILPITAS, CA

C

This newsletter is printed on recycled paper using soy ink.

Customer Education Training Schedules

Starting Dates

			August					September				October			
		Length	3	10	17	24	31	7	14	21	28	5	12	19	26
4D Western Education Center *(Mountain View, CA)*	**Graphics Library Programming I**	4.5 days	●									●			
	Graphics Library Programming II & PowerVision	4.5 days		●									●		
	Motif Programming	4 days								●					
	Realtime Programming	4.5 days							●						
	Parallel Programming	4.5 days													
	Mastering IRIX	4.5 days			●				●					●	
	System Administration	4.5 days				●				●					●
	Advanced System Administration	4.5 days					●								
	Network Administration	4.5 days													
	IRIS Inventor	4.5 days		●								●			
	IRIS Explorer	4.5 days				●									●
	System Maintenance (Power Series)	10 days								●					●
	End User Fundamentals	2 days													
4D Eastern Education Center *(Bethesda, MD)*	**Graphics Library Programming I**	4.5 days												●	
	Graphics Library Programming II & PowerVision	4.5 days													●
	Motif Programming	4 days					●								
	Realtime Programming	4.5 days											●		
	Parallel Programming	4.5 days									●				
	Mastering IRIX	4.5 days													
	System Administration	4.5 days							●						
	Advanced System Administration	4.5 days								●					
	Network Administration	4.5 days										●			
	System Maintenance (Power Series)	10 days	●												
4D Southern Education Center *(Dallas, TX)*	**Graphics Library Programming I**	4.5 days							●						
	Graphics Library Programming II	3 days								●					
	Motif Programming	4 days													
	Mastering IRIX	4.5 days	●										●		
	System Administration	4.5 days		●										●	
	Network Administration	4.5 days			●										
	IRIS Inventor	4.5 days													

To register for one of these courses, or get additional information on training programs and policies, please call Customer Education at (800) 800-4SGI (U.S. and Canada) or an overseas Silicon Graphics office. *(Schedule subject to change.)*