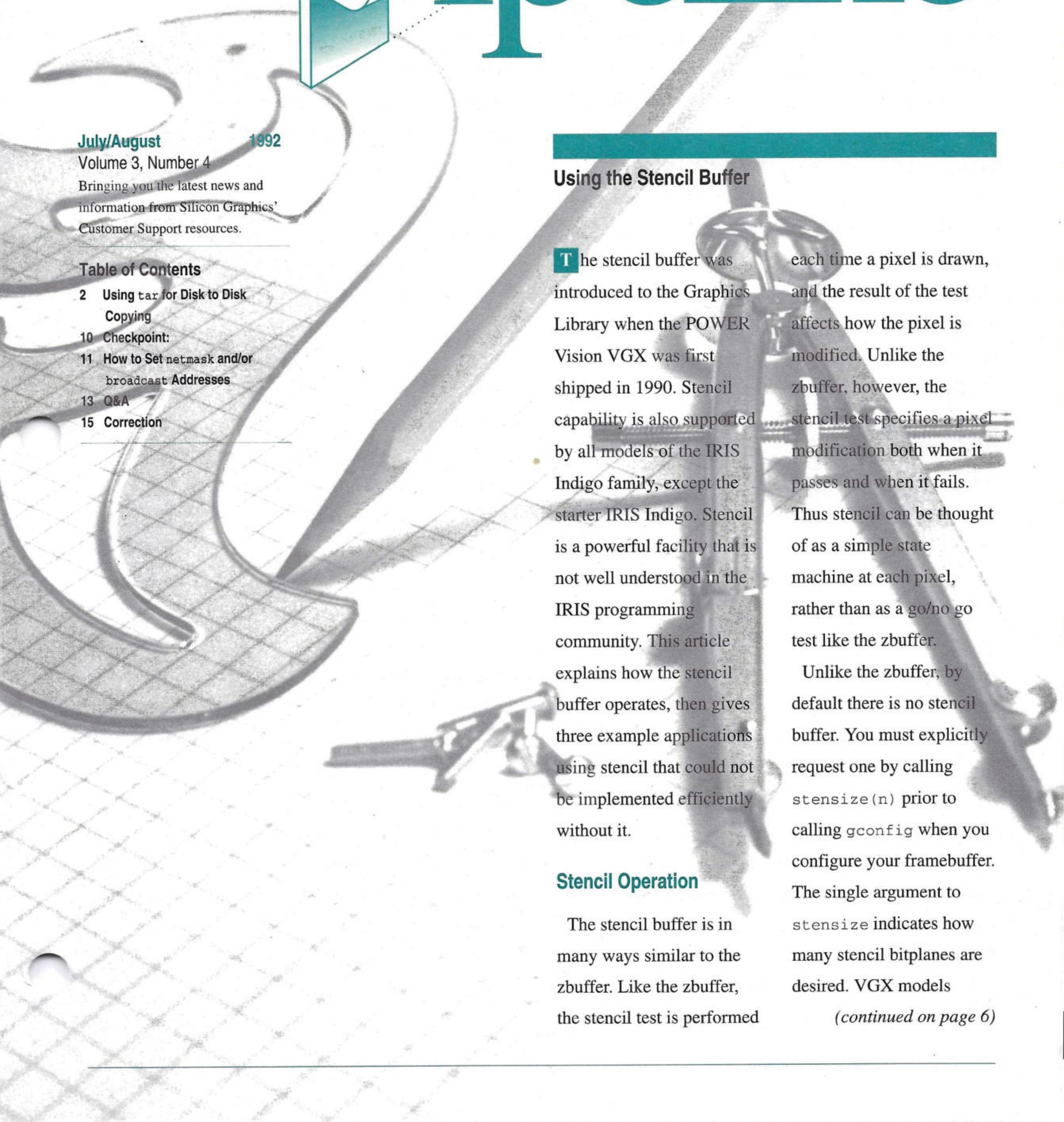
SiliconGraphics

Pipeline

July/August 1992
Volume 3, Number 4
Bringing you the latest news and information from Silicon Graphics' Customer Support resources.

Table of Contents



Using the Stencil Buffer

The stencil buffer was introduced to the Graphics Library when the POWER Vision VGX was first shipped in 1990. Stencil capability is also supported by all models of the IRIS Indigo family, except the starter IRIS Indigo. Stencil is a powerful facility that is not well understood in the IRIS programming community. This article explains how the stencil buffer operates, then gives three example applications using stencil that could not be implemented efficiently without it.

Stencil Operation

The stencil buffer is in many ways similar to the zbuffer. Like the zbuffer, the stencil test is performed each time a pixel is drawn, and the result of the test affects how the pixel is modified. Unlike the zbuffer, however, the stencil test specifies a pixel modification both when it passes and when it fails. Thus stencil can be thought of as a simple state machine at each pixel, rather than as a go/no go test like the zbuffer.

Unlike the zbuffer, by default there is no stencil buffer. You must explicitly request one by calling `stensize(n)` prior to calling `gconfig` when you configure your framebuffer. The single argument to `stensize` indicates how many stencil bitplanes are desired. VGX models

(continued on page 6)

This article provides a method and script for copying the contents of a disk to another disk. With this method, it is possible to use one disk configured with an operating system to create multiple, identical copies of the operating system to different types of disks. The method is useful for quickly creating a base software platform that can later be customized for a specific system (the systems must have the same CPU and graphics systems). The procedure loads an operating system about twice as fast as from CDROM.

In this article, the disk that contains the operating system, which is the boot device, is called the source disk. The disk that will contain the copy is called the target disk.

Note: the script contained in this article builds **identical** filesystems. (e.g. Executing the script on an IRIS Indigo Elan will create a target disk that can only be expected to support another IRIS Indigo Elan. The disk *may* support an IRIS Indigo XS or an IRIS Indigo XS24 configured similarly, but it cannot be expected nor is it recommended.)

Hardware Setup

Shutdown the system and install the target disk. Boot the system and check that the target disk is recognized by the system. `hinv` will indicate if the system recognizes the target disk and confirm its address.

Partition Changes

If a partition needs to be modified on the target disk to make it compatible with the source disk (e.g. adding more swap space), it should be changed before building the disk. If no partition modifications are necessary, skip to the **Software Setup** section.

The following example modifies the */swap* partition (partition 1) to increase its size from 40 Megabytes (MB) (the default) to 100 MB. The space for the */swap* partition is subtracted from the */usr* partition (partition 6).

Note: this example assumes that the IRIX 4.0.* version of `fx` is being used. While earlier versions of `fx` may operate differently, the procedure is conceptually the same.

1. Calculate the number of cylinders required for the desired swap size. The formula to calculate the number of cylinders required is:

$\mathit{newcyl} = \mathit{newsize} * \mathit{oldcyl} / \mathit{oldsize}$

(where *newcyl* is the number of cylinders required, *newsize* is the desired partition size, *oldcyl* and *oldsize* are values from the `len:` line displayed with the current partition information by `fx`.) *oldsize* is reported in MB, therefore *newsize* must be specified in MB.

In this example the swap partition is being changed from 40MB (*oldsize*) to 100MB (*newsize*) with the partition's current values from the `len:` line indicating 77 cyls (*oldcyl*), the *newcyl* value is:

$\mathit{newcyl} = 100 * 77 / 40$

$\mathit{newcyl} = 192.5$

It is invalid to specify a partial number of cylinders, so the number can be rounded up to 193 (slightly more than 100MB) or down to 192 (slightly less than 100MB).

Box 1 shows an excerpt from an `fx` session changing a swap partition from 40MB to 100MB.

2. As root, run `fx -x` and select the appropriate device name (*dksc* for SCSI disks), controller number (*usually* 0, but not always) and drive number (this is the device number that `hinv` reported for the target disk). After passing the controller test, select the following options in the order listed below:

```
[l]abel<return>
[sh]ow<return>
[part]ition<return>
```

Write down the partition information for partitions 0, 1, and 6. Also record the information for any other partitions that are used (i.e. partitions 2, 3, 4, 5, etc.)

3. Go back to the `[l]abel` directory by entering `..` at the current prompt. Then select these options:

`[se]t`*<return>*
`[part]itions`*<return>*

The default partition for these operations is partition 0. Enter 1 to specify the */swap* partition and the current information for the */swap* partition is displayed. Enter the default values for the partition type (rawdata) and base cyl by pressing *<return>* at the prompts.

4. The increase in the */swap* partition must be balanced by decreasing the size of another partition. In this example, the size of partition 6 (the */usr* partition) will be reduced. Since 116 cylinders were added to the swap partition (116 = 193 - 77; *cylinders added = final number of /swap partition cylinders - initial number of /swap partition cylinders*), 116 cylinders must be removed from the */usr* partition.

To check that the correct number of cylinders have been removed from the */usr* partition, note that the number of `base:` and `len:` cylinders of the */swap* partition must add up to the number of base cylinders of the */usr* partition. In this example, there should be 226 cylinders for the */usr* partition.

Box 2 shows an excerpt from an `fx` session showing the reduction of the */usr* partition (partition 6), concluding the repartitioning.

5. At the `fx/label>` prompt, select the following options:

`[sy]nc`*<return>* (to write out the changes)
`..`*<return>* (to go up one directory)
`[exi]t`*<return>* (to exit)

Important: the output above is from IRIX 4.0.* `fx`. The cylinder information is from a 1.2GB Seagate drive. The values for other configurations will probably be different, however, the process is the same.

Software Setup

After you have partitioned your target disk, if necessary,

(continued on next page)

```
—- please choose one (? for help, .. to quit this menu)—-
 [exi]t              [d]ebug/             [l]abel/             [a]uto
 [b]adblock/         [exe]rcise/          [r]epartition/
 [f]ormat
fx> l

—- please choose one (? for help, .. to quit this menu)—-
 [sh]ow/           [sy]nc            [se]t/            [c]reate/
fx/label> se

—- please choose one (? for help, .. to quit this menu)—-
 [para]meters                [part]itions               [s]giinfo
 [g]eometry                  [m]anufacturer_params      [b]ootinfo
fx/label/set> part

Enter .. when done
fx/label/set/partitions: change partition = (0) 1
before:  type rawdata   base:    33 cyls,   35079 blks,    17 Mb
                         len:    77 cyls,   81851 blks,    40 Mb
fx/label/set/partitions: partition type = (rawdata)
fx/label/set/partitions: base cyl = (33)
fx/label/set/partitions: number of cyls (max 1898) = (77) 193
 after:  type rawdata   base:    33 cyls,   35079 blks,    17 Mb
                         len:   193 cyls,  205159 blks,   100 Mb
fx/label/set/partitions: change partition = (6)
```

Box 1. An excerpt from an `fx` session changing a swap partition from 40MB to 100MB.

Using `tar` for Disk to Disk Copying

(continued from previous page)

use the following script to copy the contents of the source disk to the target disk. Save the script file in the root partition of the source disk. Make sure that you have execute permission for the file.

The script requires a device driver file as an argument. The device driver file resides in */dev/dsk*. The device driver file name will be of the form *dks0d2s0*, *ips0d1s0*, etc. The script builds the mount point, creates new file systems on the / (root) and */usr* partitions of the target disk, copies information from the source disk to the target disk using `tar`, and loads `sash` and `fx` into the volume header.

This script must be executed from single user mode (`init 1`). *It will fail if it is run from multi-user mode.*

To use the script, follow these steps:

1. Setup the system with a target disk.
2. Boot the system in single user mode.
3. Execute the script with a device driver file as an argument.

The script will execute unattended.

```
#!/bin/csh

if ($#argv < 1) then
        echo "Usage: disktar
 root_device_driver"
        exit 0
endif
#
# Build a file that has entries
# for each file/directory under the
# / partition.  Then perform the
# same for the /usr partition.
#
cd /
/etc/mount /usr
/bin/ls -Al | /bin/grep -v lost | \
/usr/bin/oawk '{print $9}' > \
disktarfiles
cd /usr
/bin/ls -Al | /bin/grep -v lost | \
/usr/bin/oawk '{print $9}' > \
disktarfiles
cd /
set tmp = "`/bin/echo $1 |/usr/bin/cut -c1-6`"
set dvdisk = "/dev/dsk/$tmp""vh"
set rootname = "/dev/dsk/$1"
set usrname = "/dev/dsk/$tmp""s6"
set mntpt = "/disktar"
set usrmntpt = "/disktar/usr"

/bin/echo \
"Making file system on $rootname...Please wait"
/etc/mkfs $rootname
echo "Making file system on $usrname...Please wait"
/etc/mkfs $usrname
/bin/mkdir $mntpt
/etc/mount $rootname $mntpt

/etc/umount /usr
#
# Tar the source root partition to the target root
# partition
#
foreach i (`cat disktarfiles`)
   /bin/echo "tar -cBf - $i | (cd $mntpt; \
tar -xBf - )"
   /bin/tar -cBf - $i | (cd $mntpt; /bin/tar -xBf -)
end
#
# Tar the source usr partition to the target usr
# partition
#
/etc/mount $usrname $usrmntpt
/etc/mount /usr
cd /usr

foreach i (`cat disktarfiles`)
   /bin/echo "tar -cBf - $i | (cd $usrmntpt; \
tar -xBf -)"
   /bin/tar -cBf - $i | (cd $usrmntpt; \
/bin/tar -xBf -)
```

```
fx/label/set/partitions: change partition = (6)
before:  type efs        base:   110 cyls,  116930 blks,    57 Mb
                          len:  1821 cyls, 1935723 blks,   945 Mb
fx/label/set/partitions: partition type = (efs)
fx/label/set/partitions: base cyl = (110) 226
fx/label/set/partitions: number of cyls (max 1705) = (1821) 1705
 after:  type efs        base:   226 cyls,  240238 blks,   117 Mb
                          len:  1705 cyls, 1812415 blks,   885 Mb
fx/label/set/partitions: change partition = (7) ..

—- please choose one (? for help, .. to quit this menu)—-
 [para]meters                 [part]itions                [s]giinfo
 [g]eometry                   [m]anufacturer_params       [b]ootinfo
```

Box 2. An excerpt from an `fx` session showing the reduction of the /usr partition (partition 6), concluding the repartitioning process.

```
end
#
# Build the volume header.  Add sash and fx only.
#
/etc/dvhtool -v add /stand/sash sash $dvdisk
/etc/dvhtool -v add /stand/fx fx $dvdisk
#
# Clean up after myself
#
rm /disktarfiles /usr/disktarfiles
$rootname/disktarfiles $usrname/disktarfiles
/etc/umount $usrname $rootname /usr
/bin/rmdir $mntpt
exit 0
```

Verification

To verify that the disk to disk copy procedure was successful, follow these steps:

1. Shut the system down (`shutdown -g0 -y` or `halt` for multiprocessor systems; `init 0` for single processor systems).
2. Address the target disk such that it becomes the new boot device for the system (you may have to change boot parameters within the System Maintenance Menu).
3. Try booting the system using the target disk.

Caveats

If the target disk is not the same type as the source disk (e.g. one disk is a SCSI disk and the other is an IPI or ESDI disk), the info in */etc/fstab* may be incorrect. If */etc/fstab* uses */dev/root* and */dev/rroot* for the root partition, and */dev/usr* and */dev/rusr* for the */usr* partition, the pseudo files */dev/root*, */dev/rroot*, */dev/usr* and */dev/rusr* will be linked to the wrong device drivers for the target disk. Be sure to remove these pseudo files from the target disk and relink the correct device drivers to the pseudo files.

If */etc/fstab* has hard-coded mounts (of the form */dev/dsk/dks0d1s0*) the hard mounts on the target disk must be changed so that the target disk has the correct files (i.e. if the target disk is *ips0d0* and the source disk is *dks0d1*, the hard mount for the root partition on the target disk would be */dev/dsk/ips0d0s0*).

Checklist for Disk to Disk `tar`

Hardware Setup

_____ 1. The system is shut down.

_____ 2. The target disk is correctly addressed.

_____ 3. The target disk is installed in the system.

_____ 4. The system is powered up and `hinv` recognizes the new disk.

Partition Changes (optional)

_____ 1. The old number of blocks for the target disk equals the new number of blocks for the target disk.

_____ 2. `fx -x` has run on the target disk with the new partition size information and the new label to the target disk has been written out.

Software Setup

_____ 1. A copy of the script exists in the / directory on the system.

_____ 2. The system is in single-user mode.

_____ 3. The script has executed successfully.

Clean-Up and Verification

_____ 1. The system is shut down.

_____ 2. The source disk has been removed from the system and the target disk has been addressed as the system's boot device.

_____ 3. The prom variables are consistent with the target disk (prom variables are: `root`, `path` and `bootfile`).

_____ 4. The target disk booted successfully.

_____ 5. The system has been shut down and the target disk has been removed.

_____ 6. The prom variables are consistent with the source disk.

_____ 7. The system boots from the source disk.

Using the Stencil Buffer

(continued from page one)

support up to 8 stencil bitplanes, while Indigo models support no more than 4. Because stencil bitplanes are sometimes allocated by borrowing LSBs of the zbuffer, it is best to request no more stencil bitplanes than your application requires. Many useful applications, including the first two described in this article, can be implemented with only 1 stencil bitplane.

Like the zbuffer, the stencil buffer is used only when it is explicitly enabled. The single command

```
stencil(enable,ref,func,mask,fail,pass,zpass);
```

controls both whether the stencil buffer is enabled or not, and, when it is enabled, how it is used. Stencil is disabled by calling

```
stencil(FALSE,0,0,0,0,0,0);
```

It is enabled when called with first argument TRUE, and appropriate values for the other six arguments. The second argument, `ref`, is a reference value which is compared to the value in the stencil buffer at each pixel. The third argument, `func`, specifies the comparison function. The eight comparison functions are identical to the zbuffer functions: all eight possible combinations of less than, equal to, and greater than. All comparisons are of the reference relative to the stencil value. The fourth argument, `mask`, specifies which bits of the reference and the stencil values are actually compared. For example, a `mask` of 0x3 indicates that only the 2 least significant bits of the reference and the stencil are to be compared.

The last three arguments specify what action is to be taken with the stencil value in each of three cases: `fail` — stencil test fails, `pass` — stencil test passes but zbuffer test fails, and `zpass` — stencil test passes and zbuffer test passes. Color and zbuffer values are, of course, updated only in the third case. One of six possible stencil actions is specified for each case:

ST_KEEP	*Make no change to the stencil value.*
ST_ZERO	*Replace the stencil value with zero.*
ST_REPLACE	*Replace the stencil value with the reference value.*
ST_INCR	*Increment the stencil value.*
ST_DECR	*Decrement the stencil value, clamping to zero.*
ST_INVERT	*Invert all bits of the stencil value.*

If the zbuffer is disabled, only the fail and pass cases are significant, and color is updated in the pass case only.

Two other commands that affect the operation of the stencil buffer are `sclear(val)`, which clears the stencil buffer to the specified value, and `smask(mask)`, which enables only the specified stencil bitplanes for writing. Note that smask is very different from the mask specified in the stencil command. The stencil mask specifies which reference and stencil bits are compared; smask enables stencil bitplanes to be written.

A Simple Application

As its name implies, stencil can be used as a 1-bit mask for all rendering, both geometric and image based. Since the stencil planes are drawn as a side effect of normal rendering, the mask can be derived from transformed geometry. The following code sequence illustrates how an image can be drawn through a stencil derived from the silhouette of the Silicon Graphics logo (as in Figure 1):

```
    stensize(1);
    doublebuffer();
    RGBmode();
    gconfig();
    sclear(0);

    while(1) {
        czclear(0,0);
        /* track the mouse and modify the matrix */
        stencil(TRUE,1,SF_ALWAYS,1,
                ST_REPLACE,ST_REPLACE,ST_REPLACE);
        /* draw the logo */

  stencil(TRUE,1,SF_EQUAL,1,ST_ZERO,ST_ZERO,ST_ZERO);
        /* draw the image */
        swapbuffers();
    }
```

Because the stencil function used when the image is drawn sets all the stencil locations to zero, it is necessary to clear the stencil buffer only once at the start of execution.

Zbuffered Images

The first example took no advantage of the differentiation between the `pass` and the `zpass` stencil cases. One way to make use of this differentiation is to compose depth/color image pairs in the framebuffer. A depth image is an array of pixels whose values are depths rather than colors. From a more practical standpoint, a depth image is the array returned by `lrectread` when the `readsource` is set to `SRC_ZBUFFER`, and the corresponding color image is the array returned by when the readsource is set to `SRC_AUTO`. Assuming that a 1-bit stencil buffer has been configured and cleared as it was in the first example, a single depth/color image pair is merged into the framebuffer with the following code sequence:

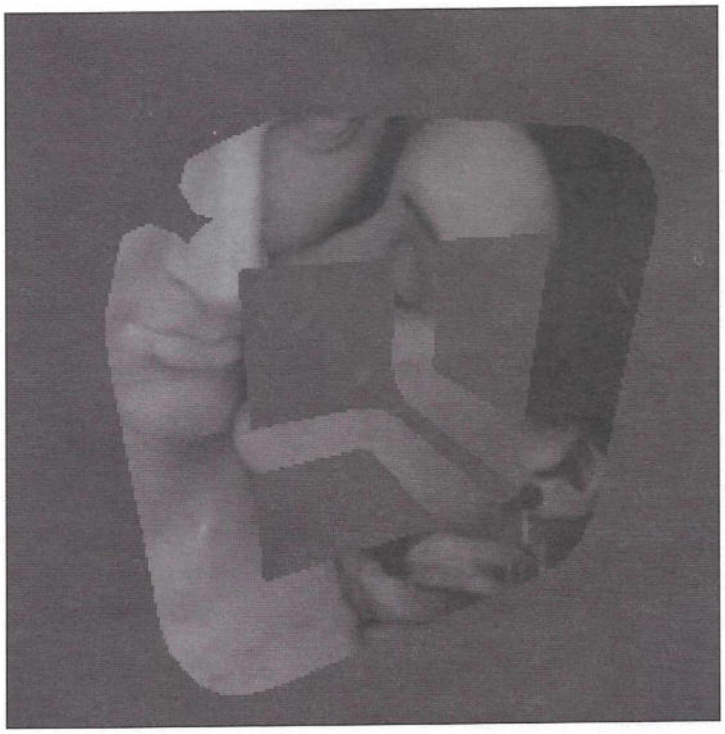


Figure 1. Image drawn "through" a stencil rendered as the silhouette of the Silicon Graphics logo.

```
pixmode(PM_ZDATA,TRUE);
wmpack(0);
zbuffer(TRUE);
stencil(TRUE,1,SF_ALWAYS,1,ST_ZERO,ST_ZERO,
        ST_REPLACE);
lrectwrite(left,bottom,right,top,depthimage);

pixmode(PM_ZDATA,FALSE);
wmpack(0xffffffff);
zbuffer(FALSE);
stencil(TRUE,1,SF_EQUAL,1,ST_ZERO,ST_ZERO,ST_ZERO);
lrectwrite(left,bottom,right,top,colorimage);
```

The first five code lines inform the GL that a depth image will be transferred, disable changes to the color bitplanes, enable the zbuffer test, configure the stencil to set the stencil buffer to 1 only where the zbuffer test passes, and transfer the depth image. The second five lines inform the GL that a color image will be transferred, enable changes to the color bitplanes, disable the zbuffer, configure the stencil to allow pixel changes only at pixels whose stencil value is 1, and transfer the color image. Notice again that the stencil buffer is left in the same state it was found in, so there is no need to clear it before the next operation is performed.

Obviously depth/color image pairs can be positioned arbitrarily in *x* and *y* prior to being composed. It is also possible, with a slight change in the code, to position the image pairs arbitrarily in *z*. All that is necessary for *z* positioning to work correctly is for the depth images to have been generated as orthographic projections (otherwise the *z* values in the images are not linear — see the article *ABCs of the Z-Buffer* in the January/February 1991 issue of *Pipeline*).

Z positioning is implemented using the PM_ADD24 option to `pixmode`. This option specifies a 24-bit signed value to be added to each pixel as it is transferred by `lrectwrite`, `lrectread`, or `rectcopy`:

```
pixmode(PM_ZDATA,TRUE);
pixmode(PM_ADD24,depth);
wmpack(0x00000000);
zbuffer(TRUE);
stencil(TRUE,1,SF_ALWAYS,1,ST_ZERO,ST_ZERO,
        ST_REPLACE);
lrectwrite(left,bottom,right,top,depthimage);
```

(continued on next page)

Using the Stencil Buffer

(continued from previous page)

```
pixmode(PM_ZDATA,FALSE);
pixmode(PM_ADD24,0);
wmpack(0xffffffff);
zbuffer(FALSE);
stencil(TRUE,1,SF_EQUAL,1,ST_ZERO,ST_ZERO,ST_ZERO);
lrectwrite(left,bottom,right,top,colorimage);
```

Using this code sequence as an inner loop, the image in Figure 2 of a molecule was drawn with 3 depth/color image pairs.

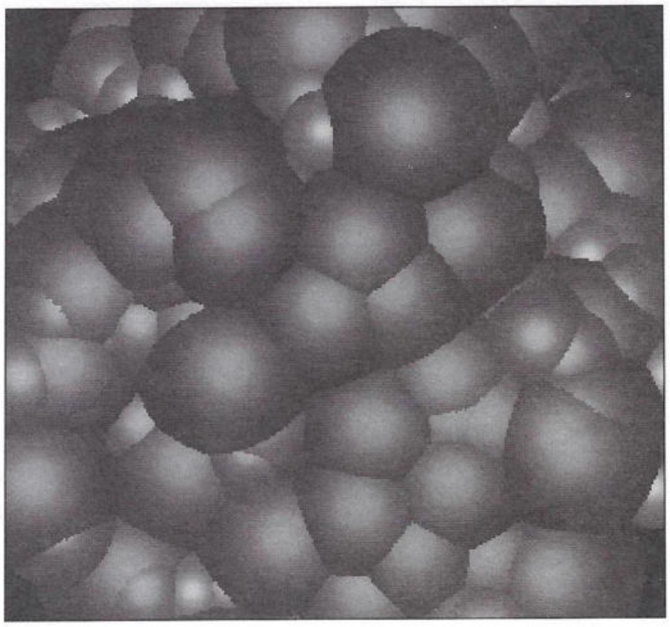


Figure 2. A molecule drawn as an orthographic composition of depth/color images.

Capping and Intersection

The final example illustrates how stencil, used with accurately point sampled polygons, can solve two well known problems in computer aided design. The first problem is *capping*. When a solid that is represented by a polygon shell is clipped by a plane other than a frustum edge, the default result is a hole through which the interior of the solid is visible (see Figure 3). A much desired result is for this hole to be capped with polygons of the correct color, shaded correctly based on their position in the scene. Of course these polygons can be computed and drawn, but such computation is expensive, and it does not leverage the graphics system in any way.

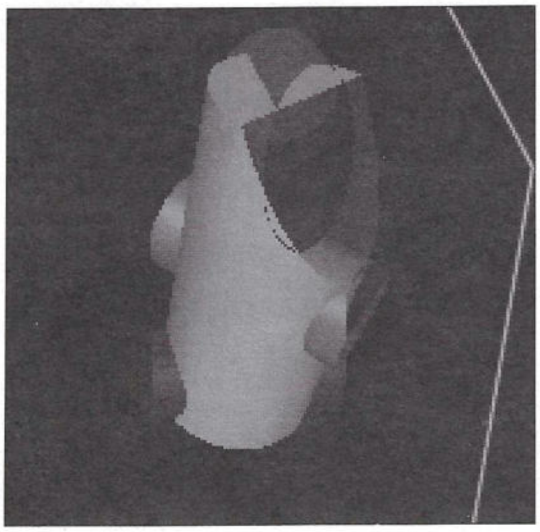


Figure 3. Four intersecting cylinders clipped without capping. Interior surfaces are visible.

A better solution is to have the graphics system somehow compute which pixels should be drawn with cap surfaces, and to render them using standard techniques. It is straightforward to use the stencil planes to compute which pixels are to be capped. All that is required is for each solid to be well defined, meaning that the polygons that represent its surface separate space into two regions: inside and outside, with no overlaps or holes.

First consider rendering a single convex volume clipped by a single clipping plane. If the viewpoint is outside the clipping volume no capping is necessary, because the resulting hole, if any, cannot be seen from the viewpoint (see Figure 4). If, on the other hand, the viewpoint is within the clipping volume, the set of pixels that need to be capped is exactly that set which are intersected once by the polygons of the surface of the convex volume. Pixels that have been

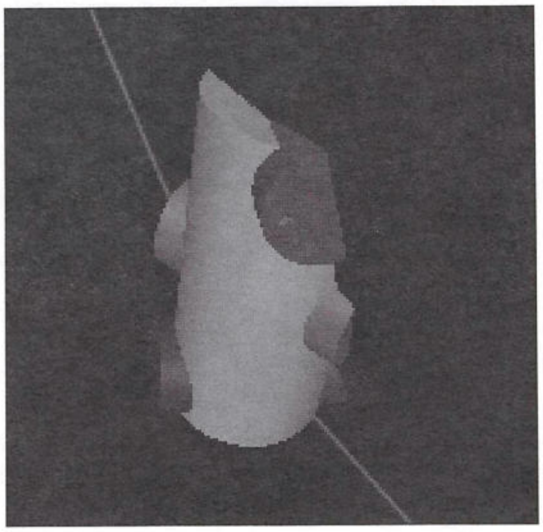


Figure 4. Viewpoint is outside the clipping volume. The resulting hole is not visible.

drawn (intersected) twice must not have been affected by the clipping plane, and pixels that have not been drawn at all cannot be a part of the object. (This reasoning clearly assumes that polygon sampling is done carefully so that pixels at the border of two adjacent polygons are neither missed nor rendered twice. Such sampling is referred to as point sampling, and it is a standard feature of Silicon Graphics rendering systems.)

A 3-pass algorithm is adequate to implement correct rendering of a single capped, convex solid (see Figure 5):

```
clipplane(0,CP_ON);
zbuffer(TRUE);
/* draw the object */
zbuffer(FALSE);
if (viewpoint is in clipping volume) {
    wmpack(0);
    stencil(TRUE,0,SF_ALWAYS,1,ST_INVERT,ST_INVERT,
            ST_INVERT)
    /* draw the object */
    wmpack(0xffffffff);
    stencil(TRUE,1,SF_EQUAL,1,ST_ZERO,ST_ZERO,
            ST_ZERO);
    clipplane(0,CP_OFF);
    /* draw a large polygon in the plane of */
    /* clipplane 0                          */
    }
```

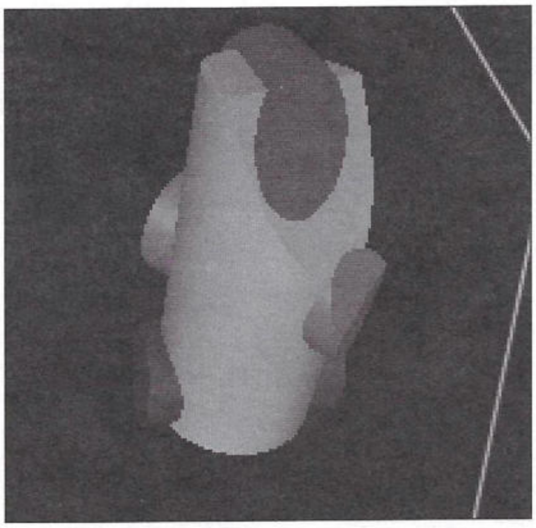


Figure 5. The 4-cylinder object clipped by a single clipping plane, rendered with capping.

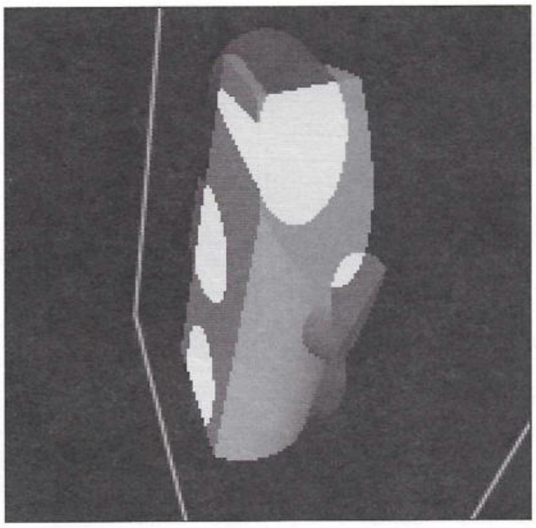


Figure 6. The 4-cylinder object clipped by two clipping planes, rendered with capping and intersection highlighting.

First the convex object is drawn, using the zbuffer to eliminate hidden surfaces, and clipped by the single clipplane 0. Then, only if a hole is potentially visible, the object is redrawn into the stencil planes, toggling the stencil value from 0 to 1 or from 1 to 0 (using `ST_INVERT`) each time a pixel is drawn. When this rendering is completed, the pixels whose stencil value is 1 are the ones that need to be capped. The cap is then drawn as a large polygon in the plane of clipping plane 0, shaded identically to the other polygons in the object, modifying color values only of pixels whose stencil value is 1. Note that clipping plane 0 is disabled when the large polygon is drawn to avoid boundary clipping failure, and that the stencil values of all the pixels in the framebuffer are returned to 0.

Because the stencil is toggled, rather than incremented, when the object is drawn into the stencil bitplanes, the same algorithm works for a single object of arbitrary convexity. (An object's convexity is the largest number of times a straight line can enter and exit the object. A toroid has a convexity of two, for example.) Multiple objects of arbitrary convexity can be rendered by repeatedly executing the same algorithm, once for each object to be rendered. The zbuffer must be enabled with `zfunction` set to `ZF_ALWAYS` when the capping plane is drawn, however, to avoid having objects overwrite previously rendered capping planes:

```
zbuffer(TRUE);
zfunction(ZF_ALWAYS);
/* draw a large polygon in the plane of */
/* clipplane 0                          */
zbuffer(FALSE);
```

(continued on page 11)

Swapping the IKON Board to use a Tektronics Phaser II PXi, Phaser III PXi or Phaser IISD Printer with some systems

Silicon Graphics has identified a problem combining the IKON board (VME based parallel interface) in the following systems with a Tektronics Phaser II PXi, Phaser III PXi printer or Phaser IISD:

IRIS systems — 4D50, 4D60, 4D70,4D80,
POWER Series, IRIS Crimson and Twin Tower

With this combination, the printer will print bands of color onto the image. *If you are experiencing other problems with your Tektronics printer please contact your support representative.*

Connecting a Tektronics Printer to an IRIS Indigo, or Personal IRIS

Connect the printer directly to the parallel port. Load the Tektronics 1.0 Drivers (included with the printer) on the system connected to the printer.

Connecting a Tektronics Printer to Other Systems

Connect the printer to other Silicon Graphics systems through a TEK 4511A (Ethernet network) interface. Purchase the TEK 4511A interface in one of these ways:

1. If you purchased the IKON board to connect your Tektronics printer to one of the IRIS systems mentioned above, contact Cynthia Brushi of Silicon Graphics at (415) 390-5256. Cynthia will provide you with information regarding the purchase, installation and support of the TEK 4511A.
2. If you own an IKON board, but it was not originally purchased to connect your Tektronics printer to one of the IRIS systems mentioned above, contact SGI Express at (800) 800-7441. SGI Express will provide you with information regarding the purchase, installation and support of the TEK 4511A.

Installing the Printer with the TEK 4511A interface

After the interface has been installed, load the Tektronics 1.0 Drivers. Then install the BSDLPR spooling system from the eoe2.sw.bsdlpr subsystem. If you need further assistance, contact Cynthia Brushi of Silicon Graphics at (415) 390-5256.

Important NetLS Update

The provisional licenses for the following software products expire on August 31, 1992:

- CodeVision 1.0
- CaseVision Workshop 1.1
- Networker 1.0

If you have one these NetLS-licensed products, please read the information below:

To receive your permanent license, send Silicon Graphics the software registration reply card that was enclosed with the software.

1. If you have returned the software registration reply card(s), you will receive a permanent license good for the life of the product before your current license expires on August 31.

2. If you have *not* sent in your software registration reply card, send it in to assure continued operation of your NetLS-licensed software product.

Send the reply card to:

NetLS Administrator
2011 North Shoreline Boulevard
M/S 12-180
Mountain View, CA 94043

or send e-mail with the reply card information to:

(continued on page 14)

How to Set `netmask` and/or `broadcast` Addresses

The following procedure changes the default `netmask` and/or `broadcast` address of a system's network interface(s). These changes are made using `ifconfig` and associated options files at startup.

To see the current settings of a network interface, follow these steps:

1. Determine how many interfaces you have and what their interface names are. Use the `netstat` command with the `-ia` option to do this.

```
% /usr/etc/netstat -ia

Name Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
et0  1500  b12         goofball.csd.sgi.co 7227517     19  3711558     0  1033
                        224.0.2.2
                        224.0.0.4
                        224.0.0.1
                        08:00:69:02:02:9a
enp1 1500  bacbone     gate-goofball.corp.  7064883  1308   6721640    63 67104
                        224.0.0.4
                        224.0.0.1
                        02:cf:1f:e0:05:84
lo0  32880 loopback    localhost         223916      0   223916     0     0
                        224.0.0.1
```

2. In the above case, there are two interfaces (`et0` and `enp1`). The primary interface is `et0`. `lo0` is the loopback

(continued on next page)

Using the Stencil Buffer

(continued from page 9)

It is also possible to extend the algorithm to handle multiple clipping planes. In this case the conditional expression must be executed if the viewpoint is inside *any* of the clipping volumes, and clipping planes whose volumes do not intersect the viewpoint must be disabled when the object is drawn into the stencil planes. If they are not, inside/outside toggling will be incorrectly affected by clipping planes that eliminate rear-facing polygons. Note that these same errors occur if the rear clipplane of the frustum is allowed to interfere with any of the rendering during the toggle phase. It is a good idea to push to rear frustum plane to near-infinity *only* while drawing each object into the stencil buffer.

Finally, it is possible to detect and highlight intersections of objects using this same stencil technique. Simply put, the stencil buffer is configured with more than 1 bitplane, the LSB is used as a toggle for inside/outside detection, and the remaining planes are used to count the number of times that each pixel is capped. Pixels that are capped more than once are within more than one object, and therefore are intersection points. A final pass over the entire framebuffer highlights these pixels in an alternate color, and the image is complete (see Figure 6).

Even pseudo code for intersection detection is too complex for this article. A demonstration program that includes both capping and intersection detection of multiple objects clipped by multiple clipping planes has been included on the 1992 Developer's Forum compact disc. This disc is available to customers with current support agreements by calling Silicon Graphics' Customer Center.

interface and can be ignored. To see the current settings of the primary `et0` interface, use `ifconfig` as shown:

```
% /usr/etc/ifconfig et0

et0: flags=e63<UP,BROADCAST,NOTRAILERS,RUNNING,ALLMULTI,FILTMULTI,MULTICAST>
        inet 128.55.200.118 netmask 0xffffff00 broadcast 128.55.200.255
```

The information returned by `ifconfig` in this example indicates that the `et0` interface is currently configured and running with an ip address of 128.55.200.118, a `netmask` of 255.255.255.0 (or 0xffffff00 in hexadecimal) and a `broadcast` of 128.55.200.255.

To Change the Default

The following are the steps that must be taken to change the default configuration of an interface and to continue to use these new defaults on subsequent reboots.

1. Become root.

   ```
   % /bin/su
   Password:
   #
   ```

2. Edit/create the file */etc/config/ifconfig-1.options* which is the options file used for configuring the primary interface of the system. For configuring other network interfaces, the file */etc/config/ifconfig-#.options* would be used, where # is the number of the interface (Primary=1, Secondary=2, etc.).

   ```
   # vi /etc/config/ifconfig-1.options
   ```

   Options provided in the files above are retrieved by `ifconfig` at boot time to configure the corresponding interfaces.

3. EXAMPLES:

   Note: The values used in the following examples are for illustration only. Both hexadecimal and decimal cases are given for clarity.

   To assign `netmask` *to be 255.255.230.00 the contents of the /etc/config/ifconfig-#.options file would be:*

   ```
   netmask 255.255.230.00
   ```

   or

   ```
   netmask 0xffffe600
   ```

   To assign `broadcast` *to be 128.55.255.255, the contents of the /etc/config/ifconfig-#.options file would be:*

   ```
   broadcast 128.55.255.255
   ```

   or

   ```
   broadcast 0x8037ffff
   ```

   If the `netmask` *and* `broadcast` *values above were to be jointly assigned, the contents of the /etc/config/ifconfig-#.options file would be:*

   ```
   netmask 255.255.230.00 broadcast 128.55.255.255
   ```

(continued on next page)

Q: Can I program the serial port on my Personal IRIS 4D/25 to run at 38400 baud?

A: Yes. If you're programming the port, use the `TERMIO(4)` flag B38400. If you're using `STTY(1)`, use 38400. At this speed, flow control will most likely be required for a reliable connection. (Either IXON IXOFF, or using hardware flow control on the _/dev/ttyf*_ device, see `serial(7)`.)

Q: Is there a threshold below which data is sent to the graphics pipe via the CPU and above which a DMA is done? What would this threshold be for a 310 GTX?

A: The only primitive that directly relates to the above question is `lrectwrite`. Some systems use PIO for small pixel transfers. However, the GTX always uses kernel DMAs to transfer the data. Elan graphics will use kernel DMA if `xsize` > 48 or `numwords` > 1400 and most `pixmode`(s) are default.

The method of transfer varies for each system. However, with GTX, VGX, and VGXT, several of the most bandwidth intensive primitives use a "3-way" transfer mechanism that sends the data directly from main memory to graphics. All other GL primitives use the normal PIO mechanism.

Although the fast transfer mechanism on IRIS Crimson versions of the above pulls data thru the CPU instead of moving it directly, the IRIS Crimson is faster because of other factors (i.e. faster memory bandwidth).

Q: How can I turn off the banner page on printer outputs using `lp`?

A: It varies by printer, but a few common switches are `-h` and `-nobanner`. Try using the `-o` switch to `lp` to pass these extra options. For example:

```
lp -dmyprinter -o"-h" myfile
```

You can also set this option in some filters. For example:

```
enscript -2r -dmyprinter -h myfile.text
```

To be sure of the proper option, look in the interface file */usr/spool/lp/interface/<printername>* on the printer host. This is a Bourne shell script which interfaces with the

(continued on next page)

How to Set `netmask` and/or `broadcast` Addresses

(continued from previous page)

or

```
netmask 0xffffe600 broadcast 0x8037ffff
```

or

```
netmask 255.255.230.00 broadcast 0x8037ffff
```

or

```
netmask 0xffffe600 broadcast 128.55.255.255
```

Note: the */etc/config/ifconfig-#.options* file should only contain one line.

4. Once the options are in the file, end the edit session, saving the */etc/config/ifconfig-#.options* file.
5. Ensure the new defaults are being applied to the interface at boottime by rebooting the system.

   ```
   # /etc/reboot
   ```

6. After the system has rebooted, check to see if the interface is configured to use the new `netmask` and/or `broadcast` by using the steps outlined in the first section of this article.

For more information regarding the `ifconfig` program and the option files it uses, see the man page on `ifconfig` and the comments in the system file */etc/init.d/network*.

Q&A

(continued from previous page)

printer and is responsible for printing the entire job, including the banner page.

Silicon Graphics has recently released Impressario, a product which contains a graphical user interface for controlling printer functions and options, including the ability to toggle banner pages on and off.

Q: The 4.0 inittab no longer uses the old programs `setupcons` and `lnsyscon`. Has this functionality been replaced by `xdm`? I know that `xdm` replaced the old `conslog` facility, but did it replace the others also?

A: The 4.0 console was completely restructured. The console driver is a pseudo-device that can utilize either *ttyd1* or the textport, which can be set with the same CPU environment variable, `console`. Valid settings are:

`G`: graphical textport with SGI logos
`g`: graphical textport without SGI logos (for VARs)
`d`: for serial port

In addition, the console output may be dynamically directed to any streams-based tty device.

This has some benefits in that console output may be directed to *any* streams-based tty device that does the somewhat standard TIOCCONS ioctl. As a result, you can use `wsh` or `xterm`, or a program of your own to display console output. The console could be displayed on a remote system across the net, or even on *ttydx* (where *x* is not necessarily 1).

The window where the console output is displayed, however, is **not** the console, but it is privy to console output, so the small `wsh` window on the screen that is labeled "console" is not actually the console.

Checkpoint

(continued from page 10)

license@csd.sgi.com

If you cannot find your reply card, request a blank reply card via e-mail to license@csd.sgi.com or fax your request to (415) 961-6502.

Please specify your current NetLS-licensed product(s) and your return address.

INFO-ACCESS coming in next Pipeline

INFO-ACCESS lists common documented solutions available via the Silicon Graphics Customer Services Engineering hotline. This new *Pipeline* section will contain a listing of highly-requested faxable and e-mailable information that has been created to provide answers to common questions and solutions to common problems. If your question or problem is listed in INFO-ACCESS, its answer or solution already exists at Silicon Graphics. Use INFO-ACCESS to find a quick fix.

LOOK FOR IT.

Q: The 4.0 inittab also has a new serial port entry *tport*. What is *tport*?

A: The textport is now controlled by a full-fledged UNIX driver called *tport*. It is implemented so that a getty is always spawned on both */dev/ttyd1* and */dev/tport*. If graphics is disabled, you can login on */dev/tport* and start graphics. (Note that your *tport* login does not go away when you start graphics. If you kill graphics, you will end up back on the textport in the same shell as you were before you started graphics.)

Q: I attempt to use find to list all my files and directories by using the commands:

```
find $home -print
find $home -depth
```

They do not print anything. How should I use these commands?

A: The command

```
find $home -depth
```

should print nothing, but just spend a while quietly searching the directory hierarchy below *$home*.

```
find $home -print
```

should print every accessible file (of any type, including directories) below your home directory.

Your first command should have worked. It may be that *$home* is a sym-link to your real home directory (*$home* isn't set to */usr/people/<login name>* or that you're not picking up `/bin/find` when you type the command `find`.

```
find $home -follow -print
```

will follow all sym-links in your directory tree.

```
/bin/find /usr/people/guest -print
```

should (presuming you have a guest account there) print all files below *~guest.*

Note: If `find` has been aliased to something like `find \!* -local`, and *$home* is NFS mounted, nothing is printed since the home directory fails the `-local` requirement).

Correction

In the May/June 1992 *Pipeline*, the article *GL/X Mixed Model Programming* discusses colormap installation with reference to overlay windows. In a mixed model program you should *always* install colormaps using `XSetWMColormapWindows`, even if it is using RGB. The sample program, *Glwidget2.c*, will not run correctly on a starter IRIS Indigo unless some other RGB program has already run and installed the RGB colormap. Further details regarding this topic will appear in the September/October 1992 *Pipeline*.

SiliconGraphics

Pipeline

Editor: Peter Lonsky

Contributors: Kurt Akeley, Susan Berning, Robert Cummins, Dave Duis, Scott Henry, Paul Jackson, Brian McClendon, Joyce Richards, Matt Robinson, Michael Sanchez, Mark Stadler, Joe Yetter

Published bi-monthly by the Customer Support Division of Silicon Graphics, Inc. for customers with support agreements. Copyright © 1992, Silicon Graphics, Inc.

The *Pipeline* welcomes your comments. Please contact us with any questions or suggestions for the newsletter. Call a telephone number at right for subscription inquiries. Send article and Q&A topics along with other correspondence to:

Silicon Graphics, Inc.
P.O. Box 7311, M/S 12-180
Mountain View, CA 94039-7311
Attn: Susan Berning

e-mail address:
pipeline@sgi.com

In the U.S. and Canada, to subscribe to *Pipeline* or change your current mailing address, for technical assistance and other support services, and, to reach Customer Education, 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

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 913
SAN JOSE, CA

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 or log a call (see previous page).***

C

This newsletter is printed on recycled paper using soy ink.

Customer Education Training Schedules

			Starting Dates												
			October				November					December			
		Length	5	12	19	26	2	9	16	23	30	7	14	21	28
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													
	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.)*