Pleasing the Eye

Todd Brunhoff

Unix Review, Oct. 1989

Nowadays, computers with graphics displays are capable of putting an immense amount of information on a single screen. With a high-resolution color monitor, the average workstation can display 256 colors, selected from a palette of 16.7 million colors, on a screen that measures 1280 pixels horizontally by 1024 vertically.

All this information must be relatively static, however. The average machine can draw between 2 and 20 million pixels per second, but it can only achieve the top end of its range if the needed calculations are done ahead of time. Generally, users can see their computers drawing and scrolling text, manipulating windows, quickly painting some wireframe models, and perhaps even moving them around or shading them as solid objects. But none of these operations is executed with the degree of smoothness we see when we watch our hands butter a piece of toast in the morning. Screen activity is jerky: it clearly shows the computer going through a cycle of ''think, draw, think, draw''.

The pace of this process can be greatly increased by the addition of some custom silicon — for a price. For $50,000 to $100,000, say, you can get 10 to 30 frames per second of smoothness while animating a pulsing, three-dimensional, shaded toroid. But what you get for all that money — unlike the television set in your living room — won't be capable of showing you as much as you'll see if you watch just sixty seconds of *The Cosby Show*. Such is the state of digital workstation technology today.

The average television set in the United States, as you know, has long been in use as a means of displaying live or taped broadcasts of natural images, albeit at low resolution. TV screens in the US sport an unchanging 485 lines of vertical resolution (the horizontal scan lines) and anywhere from 200 to 500 lines of horizontal resolution. (People in the TV industry arrive at this figure by counting the largest number of individually distinguishable vertical lines.) The television signal consists of a luminance component (brightness) and a chrominance component (color and saturation), which define the appearance of each pixel along horizontal scan lines. These scan lines are grouped into frames, and each frame is composed of two fields: one field consists of even-numbered horizontal lines, and the other of odd-numbered lines. The result is 30 frames per second (fps) of constant, smooth data representation. Granted the resolution is low, but it makes for pleasant viewing anyway.

My two daughters have no problem watching a two-hour movie (no matter how bad) on TV, but they will never sit still to watch my workstation display for more than five minutes, no matter how many windows I move around.

For those who like digital comparisons, here's one: in the US, TV signals (the US standard for TV broadcasting is NTSC, which was defined by the US National Technical Standards Committee decades ago) are normally digitized at 720 pixels per line, producing about 10 megapixels of information per second (720 pixels x 485 lines x 30 fps). Each pixel requires roughly 16 bits of luminance-chrominance information, or about 22 bits of red-green-blue (RGB) information, or 30 MB per second of bandwidth, to handle it all[1]. Things a viewer can't perceive with such a TV display include an 8-point font, unflickering pixel-wide lines, and high-resolution images. For the near future, the lack of low-cost display technology and the current struggle over broadcast standards for high-definition television (HDTV) will likely prevent low-cost television from reaching into the spectrum of high resolution. Such is the state of analog television technology today.

Meanwhile, a few bright individuals have realized that these two technologies -- high-resolution graphic displays and analog television -- can aid each other nicely in the high-quality visual-display genre known as visualization. Since 1985, a smattering of workstation products have been introduced that provide a nice merging of the two. Video/graphics products for low-resolution screens, such as 640 x 480, are even more plentiful, but most manufacturers of workstations with serious graphics capabilities have expressed their tacit disapproval of that display technology by not offering it as a medium.

The new graphics/video workstations have two or more of the following features:

- *Graphics*: simply a necessary standard for a workstation.
- *Video Output*: ''output'' is defined relative to the workstation; video output is therefore the ability to capture a region of the graphics screen and encode it into a video signal, be it NTSC, PAL, or RP125.
- *Video Input*: ''input'' too is defined relative to the workstation; this is the ability to perform real-time digitization, or otherwise make a video signal displayable.
- *Mixing or blending*: video input must be merged with graphics; once a signal is digitized, you want to see it on the screen. With the advent of widely used windowing systems such as the X Window System and Sun Microsystems' NeWS, however, the mixture of graphics and video must be definable on a pixel-by-pixel basis. Furthermore, in some cases it may be interesting to define a pixel as a blend of two pixels, one graphic and one video; for example, 20 percent graphic and 80 percent video[2].

Merging the technology. Workstation graphics/video products are not new. Graphics over video has been done for years with the help of a machine called a ''character generator'', which produces, for example, the credits that scroll up the screen following your evening TV news program. More recently, video-effects machines have provided real-time 2D and 3D transformations of video signals; the prelude to Johnny Carson's *Tonight Show*, which has multiple video panels flowing left to right at different speeds as if they were banners, is a good example of what these machines can do.

A workstation with a high-resolution screen is not a TV, however. Video frame-grabbers that fit into a computer have been available for years but, to my knowledge, Parallax Graphics was the first company to produce a low-cost, workstation-class video/graphics processor, introduced in 1985. Other systems capable of video input processing have appeared over the last six months, and others will follow. A relatively small number of basic designs are used, and they are sometimes augmented by extra frame-buffer memory and graphics accelerators. The design in Figure 1 places the digitized picture directly into the frame buffer right along with the graphics; each pixel is either a graphics pixel or a video pixel (usually defined by another bit-plane). Figure 2 shows the picture digitized into a separate frame buffer, which may or may not be the same size as the graphics frame buffer. The advantage of the latter design is that the pixels may be blended as described earlier

.

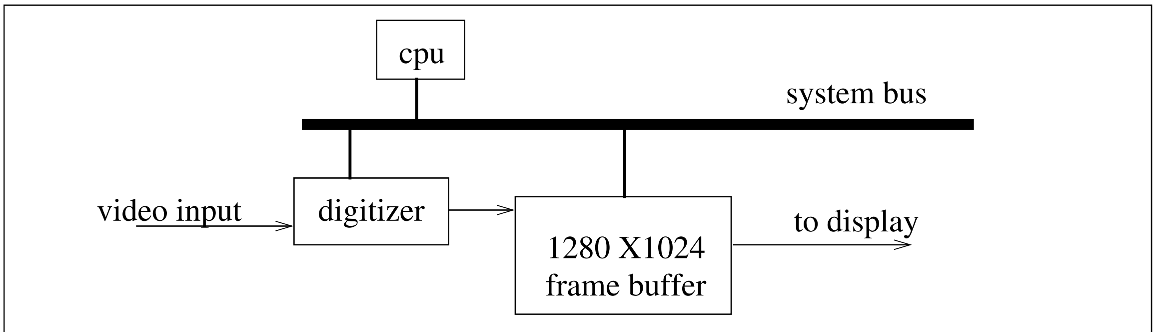


figure 1

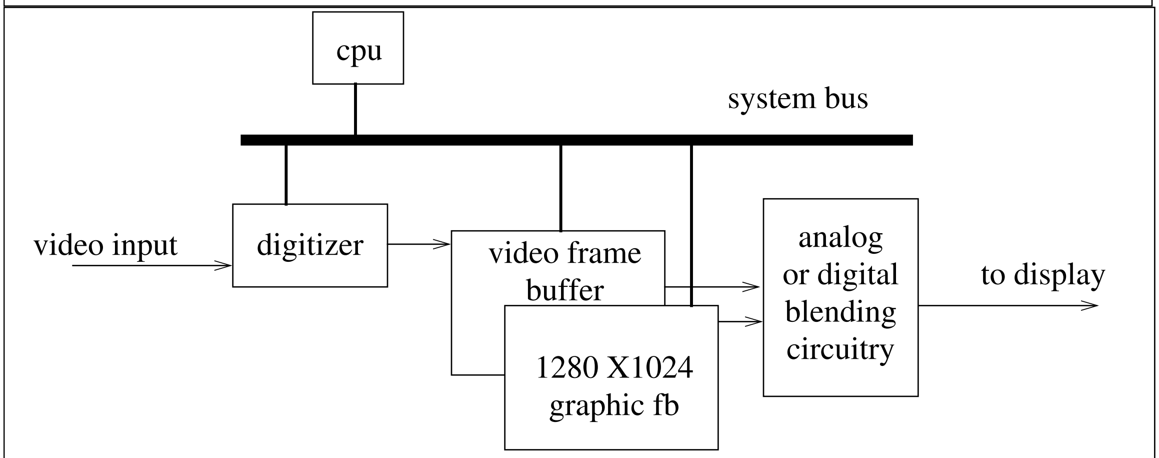


figure 2

Scan converters — video output devices — for high-resolution displays have been available for a bit longer. Before their advent, a graphics company that wanted to produce a videotaped demo of its product was forced to make it by simply pointing a video camera at the desired screen display. Over just the past year a handful of companies, including RGB Technologies, Tektronix and Fulsom, have introduced workstation-based video-output products capable of converting a high-resolution display into an NTSC-class signal.

S So what are these devices good for? What can you do with merged graphics-video capability that you can't do otherwise? Applications are numerous, in fact, and they can be conveniently grouped into five major categories: data correlation, video production, cartography and geographic information services, image processing, and animation. Each is described in the paragraphs that follow.

Data Correlation. Imagine a constructive solid-geometry model of a particular tool that is used to animate the design of the tool for the purpose of studying and improving it. Also useful might be a live video of the tool in active use. Merging these two views, either by superimposing or juxtaposing them, might reveal aberrations in one with respect to the other, and would probably make the analysis job easier.

Video Production. We have all seen pictures of video production studios with banks of monitors and control boards that dwarf a computer keyboard. But not all video production requires a studio. Artists can be intimidated by such installations and by having to rely on an engineer, no matter how talented, for the execution of their ideas. Likewise, producers may prefer to view productions in a less intimidating environment. In any case, workstations have the ability to control high-end videotape players or effects machines, most of which are driven through RS232 connections. Adding video-input capabilities to such workstations, along with a good user interface, might yield a usable (if not equally versatile) substitute for full-scale production studios.

Cartography and Geographic Information Services. Landsat images and aerial photographs of the Earth exist in great abundance, and most are available as 1k X 1k images stored on magnetic tape. One-time mastering could be done onto convenient CAV-format video disks for $300-500 with a video output device, using a mosaic of NTSC-resolution images for each original to preserve resolution. (Constant Angular Velocity (CAV) and Constant Linear Velocity (CLV) are two formats for commercial video disks; their names describe the movement of the read heads across the disk. CAV will allow special effects like freeze frame and frame-accurate random access but will only hold about 30 minutes per side. CLV won't allow special effects but has twice the capacity.) While hardly the best use for a real-time digitizer, video input capabilities could nonetheless be used to "tile" these images on a workstation screen, with additional graphics used to emphasize geographic features. A video disk player could provide random access to any Landsat image on the disk in two seconds or less, an impossible feat for magnetic tape, while a real-time digitizer could acquire the image an order of magnitude faster than with magnetic tape.

Image Processing. Military or security systems that perform ''friend or foe'' identification can use the ability to match periodically acquired images against a reference image to see what has changed. The development of such a feature would require both video input and CPU power for image comparison. The same general design can provide "eyes" for an industry robot. And in general, video input is nice complement to imaging applications because it offers a very convenient acquisition method.

Animation. Except for a very few, very expensive computers, the job of drawing and shading complex three-dimensional objects is too demanding for a graphics workstation to do at a smooth 30 fps. Adding video output capabilities, however, permits the leisurely capture of each laborious frame on a videotape recorder and allows for replay at a later time with the use of video input capabilities. Wavefront Technologies has demonstrated impressive results using this sort of ''animation station''.

Software concerns. When given descriptions of such systems and applications as those described above, some people worry out loud about the effect on performance of handling full-motion video in software. In my opinion, their concern is premature at the present time. If software programs become directly involved in full-motion video display, either the bandwidth of present CPU data paths must increase or the bandwidth required for video transmission must decrease. Thirty megabytes per second for a CPU data path is, as discussed earlier, not a reasonable speed to expect, because today's digital storage systems for computers — disks, tapes, or solid-state devices — are high-speed, low-capacity devices or high-capacity, low-speed devices; they are not capable of both high speed and high capacity. For example, a large main memory might record 10 seconds of a Michael Jackson video in real time, but an optical WORM jukebox would need 30 hours to record a full 10-minute music video.

A reasonable alternative approach is found in real-time data compression and decompression. Typically, however, decompression can be performed at full speed, but compression is not done in real time. And even if compression weren't the bottleneck, there are currently no compression techniques that can guarantee ''loss-less'', constant-rate compression on all data sets. Video teleconferencing involves images that lend themselves to compression because they call for little detail and very little change from frame to frame: techniques called ''temporal differencing'' and ''spatial differencing'' can achieve one or two orders of magnitude of compression[3].

Accepting these postulates, as a number of manufacturers apparently have, means no longer worrying about the impact of direct CPU involvement in the video display, because today's video input and video output solutions for computers are not designed for such involvement. Current commercial video input and output products for workstations are general-purpose products and, as such, their CPUs get involved only at a higher level, directing such things as where the video image is to appear on the screen.

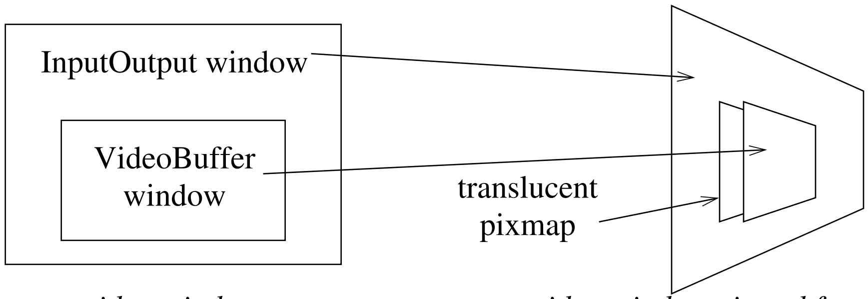
Given such hardware, it is an easy leap of faith to believe that video can be made an integral part of windowing systems such as X Windows or NeWS. And, in fact, it has been. In 1986, researchers involved in MIT's Project Athena began using the Parallax video input product along with X Windows version 10 for experiments in computerized language courses, navigation, neurobiology, and video post-production[4,5]. In June 1988, a meeting held in Red Bank, NJ, launched the X11 protocol design for a Video Extension for X. The effort, now a more mature one, is referred to as VEX.

VEX is an attempt to provide basic abstractions for the operations a client — an X program that works on behalf of the user — would want to perform with video input and output: placing a video picture on the screen, obtaining the digitized pixels from a frame of video, controlling the pixel-by-pixel blend of video and graphics, and capturing a portion of the screen for some particular video recording device. The extension also includes a basic framework for controlling external devices, but VEX leaves unspecified the semantics of each device and all its knobs and dials for the simple reason that there are potentially so many different kinds.

For VEX, video input in a window is accomplished with three components: a normal Input-

Output window that holds graphic pixels, a VideoBuffer that holds digitized video pixels, and a translucent pixmap that (partly) defines the blending or mixture of video and graphics (see Figure 3). The VideoBuffer window is a subwindow of the graphics window.

.



video window *video window viewed from the side (detail revealed)*

The translucent pixmap also has a indexed lookup table, called a \fIblendmap\fR, that functions as an indexed colormap. In the same way that a programmable colormap can allow for very fast changes affecting the entire screen, the blendmap allows an X application to accomplish a variety of fades between graphics and video very quickly.

Video output is not nearly so complicated as video input because, literally, what you see is what you get. A single VEX request allows the client to specify a region within a window, and everything within its boundaries is converted into a video output signal.

VEX permits direct control of video device connections through the use of connectivity matrices and permits such abstractions as a videotape recorder that consists of two logical devices, one for recording and one for playback. The structure of a connectivity matrix allows an X client to perceive the mutual exclusivity of the devices. Another aspect of video device control is all the ‘‘knobs’’ and ‘‘buttons’’ that may be accessible to computer control, such as a volume control or the playback controls in a videotape player. VEX handles this aspect, too, but only in a general sense. An X server with VEX may advertise a variety of controls through a generic structure called a ‘‘control’’, and will allow them to be adjusted through a structure called a ‘‘setting’’. But because the number and variety of such controls is virtually infinite, the VEX specification does not attempt to describe the semantics of any particular control. That task will instead be left to public and private convention.

Video input and output hardware has complex limitations and capabilities; its capabilities do not always scale in expected ways, nor do its limitations always make sense. Some video hardware, for example, is able to display video pictures on screen, but will not allow access to the digitized pixels; other hardware restricts the placement of full-motion video to the modulo-16 pixel horizontal boundary. The VEX extension is designed to recognize this and is able to provide large amounts of information about such limitations and capabilities. It is intended that an X client, through the use of convenience routines in the Xlib implementation, will be able to determine easily what it can and can’t do without incurring the overhead of a round-trip request to the server.

At the time of this writing, the VEX protocol is largely complete, and is being discussed among interested X Consortium members. VEX is currently being implemented at Tektronix, and later this year it will be submitted to the X Consortium for technical review[6,7].

The translucent pixmap also has a indexed lookup table that works like an indexed colormap, and is called a blendmap. In the same way that a programmable colormap can allow very fast changes affecting the entire screen, a blendmap allows an X application to quickly accomplish a variety of fades between the graphics and video.

Video output is not nearly so complicated as video input because, literally, what you see is what you get. A single VEX request allows the client to specify a region within a window, and everything in its boundaries are converted to a video output signal.

VEX allows direct control of video device connections through the use of connectivity matrices, and permits such abstractions as a video tape recorder that consists of two logical devices, one for recording and one for playback. The structure of a connectivity matrix allows an X client to percieve the mutual exclusivity of the devices. Another aspect of video device control are all the "knobs" and "buttons" that may be accessible to computer control, like a volume control or the playback controls in a video tape player. VEX allows this, too, but only in a general sense. An X server with this extension may advertise a variety of controls through a generic structure called a "control", and will allow it to be adjusted through a structure called a "setting". But because the number and variety of controls is virtually infinite, the VEX specification does not attempt to describe the semantics of any particular control. Instead, this will be left to public and private convention.

Video input and output hardware has complex limitations and capabilities; not always do the capabilities scale in expected ways, nor do the limitations always make sense. For example, some video hardware is able to display a video picture on the screen, but will not allow access to the digitized pixels; other hardware restricts the placement of full-motion video to modulo 16 pixel horizontal boundary. The VEX extension is designed to recognize this, and is able to provide large amounts of information about the limitations and capabilities. It is intended that an X client, through the use of convenience routines in the Xlib implementation, will be able to easily determine what it can and can't do without the overhead of a round-trip request to the server.

At the time of this writing, the protocol is largely complete, but is being reorganized for efficiency and closely examined for gaps in the logic of the specification. It is currently being implemented at Tektronix and later this year the extension will be submitted to the X Consortium for technical review.

BIO

Among UNIX kernel manipulators, Todd Brunhoff is perhaps best known for designing and writing RFS, a portable public-domain remote filesystem (not to be confused with ATT's RFS). He is now a principal engineer in the Visual Systems Lab at Tektronix, where he is overseeing the software development of a merged video and graphics architecture. While in a previous position at Tek, Brunhoff worked as a guest engineer on MIT's Project Athena and was a member of the team that produced the first release of X Windows version 11. He is currently the designated architect for VEX.

Bibliography

1. *Colour Television - Vol. 1: NTSC; Vol. 2 PAL and SECAM*; ILIFFE Books Ltd. (Wireless World), London, 1969.
2. "Compositing Digital Images"; T. Porter, T. Duff; Computer Graphics, Volume 18, Number 3, July 1984.
3. *Digital Pictures: Representation and Compression*; A. Netravali; Plemum Press, 1988.
4. "The Athena Muse User's Manual."; M. Hodges; Massachusetts Institute of Technology, Project Athena Visual Computing Group, Working Paper #12, 1988.
5. "Athena Muse: Hypermedia in Action"; Lampe; The MIT Report; February, 1988.
6. "VEX Your Hardware"; T. Brunhoff; Xhibition '89 conference proceedings.
7. "VEX: Video Extension to X" T. Brunhoff; Unpublished, but available from the author.