TO: Computer Technology Review
FROM: Hastings & Humble Public Relations Inc.
RE: VEX Store from Tektronix Interactive Technologies Division
DATE: 8 June 1990

X Windows and Video: Now They’re Talking

by Todd Brunhoff
principal engineer
Tektronix Inc.

VEX, the video extension for X, provides a standard interface between the popular X Window System, Version 11, and special-purpose video hardware.

When it comes to sustained data throughput and natural images, television video technology – with its effective image transmission rates of 30MB per second – takes the race. When sharp, accurate two- or three-dimensional simulations are required, computer-generated graphics – with its true-color, hi-resolution images – wins the battle. But when it comes to combining the two inside the graphics- and general-computing environment known as X, full integration has never been available. The smooth, uninterrupted motion of video requires high-speed data transmissions with real-time accuracy, while the keen manipulable image of computer graphics depends on millions of calculations completed before it can be displayed on a screen. A typical computer graphics workstation does not have a system bus that can support the throughput required by video, and a typical video display system does not have general compute facilities to support graphics. As a result, merging video and graphics, particularly with the bounds of X or other windowing systems, has been a complex task of software and hardware engineering – and, until now, a headache for systems integrators.

Yet industries from aeronautics to entertainment are increasingly using applications that depend on precisely this successful fusing of video and computer graphics. Many applications in the areas of CAD simulation, video and film special-effects production, cartography, image acquisition, motion analysis and video-database perusal yield accurate and effective results only when video and graphics function in close harmony.

Systems integrators have responded by writing specialized drivers for and array of video recording, playback, and display devices. But despite devoting hundreds of man-hours for every new device they wish to integrate with their computer graphics systems, they still have found themselves locked into particular hardware designs. They have needed a standard interface between video and computer graphics in X. VEX, the Video Extension for the X Window System, is designed to give them just that.

VEX addresses video/graphics integration issues by providing a set of basic abstractions describing the operations an X application performs with video input and output (think of input and output with respect to a video/graphic workstation: video signals comming *in* to the workstation are displayed on the screen, and video pictures produced on the workstation go *out*). These operations allow an application to request a video image to be placed on the graphic display, or to capture a portion of a graphic screen for transmission to a video recording device. The operations were also designed so that pan and zoom are an integral part.

VEX also provides an explicit mechanism for changing the connections among external video devices, such as selecting whether an input is to be a video disk player, a camera or editing equipment. Along with this it furnishes a framework for controlling the devices; e.g. searching for a specific frame. Both of these operations imply a need for controlling the resource allocation, and that need is addressed by a concept of resource ownership.

Because its design recognizes that few computers are capable of handling the video data rates without hardware assistance, VEX does not rely on simply copying pixmaps to create the illusion of video. Instead, handling video input and output is done with a level of abstraction high enough to allow specialized hardware to handle true video operations, which in turn gives systems integrators a simple and predictable basis upon which to mesh an X-based application with their video hardware. With VEX – and other extensions to X – they can avoid the work involved in integrating their application with a new piece of video hardware.

VEX and Video Input

Workstations integrated with contemporary full-motion video input equipment fall into one of three classes:

- video is digitized into a frame buffer shared with graphics

- video is digitized into its own frame buffer, and is then merged with a separate graphics frame buffer by analog or digital means

- video is not digitized at all, but simply decoded and then merged with the analog output of a graphics frame buffer

VEX allows video input to be rendered to a pixmap or a window, although the depth and visual class may be restricted to a certain set. The core X protocol names six visual classes that express different interpretations of pixels: StaticColor, StaticGray, PseudoColor, GrayScale, TrueColor and DirectColor. These represent most forms of graphic frame buffers. VEX introduces two new visual types, known as VideoGray and VideoColor, to accommodate hardware that supplies the undigitized video pixels. A system integrator attempting to incorporate video hardware into his X-based system would choose visual classes and depths that most closely match the hardware at hand.

Some video/graphic hardware has limitations; the VEX protocol recognizes this and allows for alignment constraints on the placement of video pictures within the graphics frame buffer, restrictions on the scaling of the video picture and clipping limitations. All these can be easily discovered by an X application (called a "client"), and handled with the VEX application libraries. Most important, the definition of VEX guarantees predictable results for windows that contain full-motion video or video stills. This means that the integration with the window system, the most difficult task for a system integrator, is already solved.

VEX will precipitate dramatic changes in the way some applications are implemented. Previously, only a highly integrated, and device-dependent set of software could allow an application to use both video and graphics. This extension allows a developer to write portable, device-independent applications. For one, images from a video camera can be displayed on a screen as well as digitized for quantifying, convoluting, or obtaining a histogram. In fact, there are several areas whose applications will either improve by having a standard development environment such as VEX, or are now possible for the first time because it is the first standard video extension to a standard window system. Here are a few descriptions:

An interesting problem for simulation is to merge experiment data and computed data such that the two can be compared visually. For example, a scientist may need to model the stress and flex on an aircraft wing in wind turbulence. A video image of a wing, recorded in actual use, laid over or under a three-dimensional graphics model of the wing generated from the experimental data, can let engineers clearly see abberations between reality and the chosen model. Similarly, a solid-geometry model of a tool juxtaposed with a video tape of the tool-in-use might facilitate analysis by revealing irregularities in one with respect to the other.

Integrated video-graphics systems are already providing attractive options in the entertainment industry. Today, for example, artists are able to edit video productions with a highly integrated system, producing a traditional edit-decision list, a complex set of numbers and timecodes used in edit studios to produce final copy. With the VEX extension, it is now possible to create similar applications that are portable among video-graphics systems that support VEX. Beyond that, future applications are possible that more fully utilize and integrate the video capabilities, the graphics, and the ability to control studio-quality recording and effects equipment. Workstations having this software and hardware could potentially replace large parts of video production studios at a reduced cost.

A more general VEX application might be video database perusal. Here, users can rapidly examine any of thousands of images originating from a vast range of input hardware. Two application areas subject to dramatic improvement given window system integration are cartography and geographic information systems. Today, thousands of Landsat and Spot images of Earth are generally available as large digital images stored on magnetic tape. The images are very large, having on the order of ten million to fifty million pixels, with each pixel having three to seven different spectral bands. By slicing these pictures up into NTSC-resolution images and mastering them onto standard CAV-format video disks, the video digitization hardware and VEX software can be used to tile an entire high-resolution graphics monitor with a satellite image in a fraction of a second. In contrast to the five-to-ten-minute access time of single images on magnetic tape, many tens of images can be stored on analog video media and accessed randomly in one-half second, or less.

VEX and Video Output

Several manufacturers now provide workstations capable of generating video signals, either through a process known as downconversion or by using addressable pixels or both. Downconversion, indispensable for video-graphics integration with a window system, converts all or part of a high-resolution graphics display into a low-resolution video signal such as NTSC, usually at the full 30 frames per second rate. Currently only one or two vendors market hardware capable of capturing variable subsets of a high-resolution display, but other vendors may soon offer this ability. A system using addressable pixels allows direct access to a frame buffer producing low-resolution video signals; it can implement a virtual downconverter by resampling in software. VEX serves both environments by providing a portable interface to the variety of existing video output hardware.

With the high-level abstractions provided by VEX, systems integrators can use a single application with a monitor from one vendor, a recorder from another, an editor from yet another, and so on – without having to design and implement a different driver for every piece of equipment. Like other X applications, a video application can even run on a computer separate from the video-graphics workstation and display. Normal X applications use this feature of X to achieve a level of parallelism, and VEX applications are able to do the same. A portion of the VEX implementation that will be publically available allows reconfiguration of the set of video devices through the use of an ascii description file similar to Berkeley Unix’s termcap or System V’s termio: new devices can be integrated in minutes without recompiling any software.

Two video-output applications will benefit from the use of windowing environments: video production and post-production, and scientific visualization. Today, production and post-production of video using computer-generated graphics often involves a special-purpose recording device requiring 7 to 27 seconds for each transfer of image data. Workstations with the kind of video-output capability supported by VEX can cut this transfer time to as little a one second. Scientific visualization, gaining ground in research institutions, uses computer-generated images for modeling designs, simulating natural phenomena, and viewing n-dimensional data sets. Animated output from visualization images used in research is frequently transferred to video tape for distribution to other institutions or for showing to students.

Computer animation, already available on several workstations and a variety of software packages, will also benefit from a standard interface with a window system, because it will allow a wider selection of related software to be used in parallel. Currently, most animation programs do not cooperate with X, and consequently cannot "share" the computer display. This extension will allow the animation program to work on the display along with other pertinent X applications, and in addition, the animated frames can be recorded with video output hardware and later displayed with video input hardware, all with the control of VEX.

VEX and Video Resources

Systems integrators know that solving problems of video input and output is only the beginning of what they must consider when integrating video and graphics into a windowing system. Other problems must be solved such as video resource allocation, connectivity among these resources, and controlling non-X aspects like current tape frame number and audio volume. Two of these, connectivity and resource allocation, are addressed by specific mechanisms in the VEX protocol. On the other hand, controlling device "settings" – like volume – is allowed through a basic, but extensible framework.

VEX allows clients to remain informed of the connectivity among video devices through the use of connectivity matrices and the delivery of events that mark a change in a matrix. The information in the matrices is expressed as mutual exclusions. This can be used to describe a 10-to-1 demultiplexer, or to view a videotape recorder as two devices: one for recording and one for playback.

On and X station or workstation, resource allocation typically is handled by the operating system, using so-called "virtual resources", which dictate a time-sharing agenda among different system components. Because of the binary nature of tasks involving video − an activity is either occurring or it is not − the concept of a virtual video device is not practical. So, VEX introduces a concept of ownership for video inputs, outputs and external video devices. This means that a non-owning client can use a resource when its owner does not need it, but if the owning application requests the device, the non-owning application loses it. Similar to an X window manager, it is intended that a "video manager" handle the allocation of ownership, which is explicitly assigned through one of the VEX requests.

Given this ability for a client to own a device and connect it up to something, the device may not be interesting unless you can instruct it to "play" or "rewind". Similarly, there may be other devices that must work in tandem with video equipment such as audio players or amplifiers whose volume or tone level must be set. And finally, while certain aspects of video input and output hardware − such as color maps − can be controlled by the core X server, there may be other aspects of the hardware that it cannot control; e.g. hue, brightness and saturation.

Each of these non-X aspects of video devices may be critical to an application’s proper operation, but the number and complexity of these controls is virtually unlimited. Consequently, the VEX extension defines handles for "controls" that may be set or read, but the precise meaning and structure of each control is left unspecified by the VEX protocol so that the list of controls can be expanded as new video devices are developed that can be computer-controlled. From a practical standpoint, many of these controls are already defined by convention and are described in a document that accompanies the VEX protocol.

The Status of VEX

The VEX protocol, the server implementation, and the library interface are complete. At the time of this publication, the software will have completed alpha release and entered beta release. During this beta test, it will be reviewed by the MIT X Consortium members and by the general public. If it is accepted, the extension and the software will become an X Consortium standard, and available free of charge. Other pending extensions include XINPUT, an extension that defines the use of other input devices such as data gloves and valuator dials. Other accepted extensions include PEX, which brings ANSI-standard PHIGS+ into the X arena.

VEX provides a sorely needed mechanism for integrating graphics and video, an integration that has begged to happen since low-cost high-resolution graphic displays became commonplace in the early 1980's. The X Window System has been successful because the engineering public subscribes to this public process, and that same process will make VEX successful. VEX is not a panacea. But it does offer solutions such as portability and easy integration.

###

Todd Brunhoff is a principal engineer in the Visual Systems Lab at Tektronix Inc. (Beaverton, OR). Previously, he was a member of the original version 11 X development team at MIT, and has since authored a variety of X software. Brunhoff is the designated architect for the VEX protocol, and directs the VEX software development at Tektronix. For more information, you can contact Mr. Brunhoff at (503) 627-1121, or preferably, via internet electronic mail at `toddb@crl.labs.tek.com`.