Vex Your Hardware
VEX version 5.6

Todd Brunhoff
Tektronix, Inc.
Tek Laboratories
P.O. Box 500; MS 50-321
Beaverton, OR 97006
toddb@crl.labs.tek.com

ABSTRACT

Low-cost video signals, such as NTSC and PAL, provide on the order of thirty megabytes per second of sustained data throughput, while most magnetic disks provide about one to five megabytes per second. Television-resolution video signals may not be appropriate for all graphics applications; but some, like capturing graphics or video teleconferencing or image processing, are serviced nicely by this technology. VEX, the Video Extension for X, is a proposed standard extension for the X Window System, Version 11, and is intended to provide a standard interface to video operations from within X.

VEX INTRODUCTION

This document matches the VEX protocol document, version 5.6. An earlier version of this document describes features that have since been removed or changed [Brunhoff1-89].

The Video Extension for X, or VEX, arose out of a meeting in June of 1988 in Red Bank, New Jersey hosted by Lester Ludwig of BellCore, and attended by several parties interested in a standard X extension that would improve on available software mechanisms. Over the months that followed, I looked at the work done at Parallax [Levy87] [Picco88], Massachusetts Institute of Technology [Hodges88] [Davenport87], Bellcore [Ludwig87], and HP. With the help of these and advice from others in the field, I designed an extension for X Version 11 that handles video input, output and external video devices.

Why VEX? Why provide an extension to access special-purpose hardware? A standard extension to a window system like X provides portability, an attribute traditionally cited as a benefit for the developer and user. It guarantees that an application will work on any platform regardless of the label on the bezel. Yet a hardware vendor may benefit as well since the only difficult effort required with a new generation of hardware is a port of the X server, and perhaps a new compiler; after that, X clients begin working with little effort. A corollary to this is that applications that deal with special video hardware through an opaque X interface will also port with little effort.

VEX attempts to provide the 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: place a video picture on the screen, obtain the digitized pixels from a frame of video, and capture a portion of the screen for some video recording device. The extension also includes a basic framework for controlling external devices.

Many people, when encountering the idea of a video extension, assume that the design provides some method to move pixmaps around so that a user might quickly display them on the screen to create the illusion of video. Not so. The design of VEX recognizes that few computers are capable of handling video data rates, and therefore the level of abstraction offered by the extension must be high enough to allow specialized hardware to handle the operations.

WORKSTATIONS, WINDOWS, AND VIDEO TECHNOLOGY

First, a clarification: the term "video" is overused. It is a label for one of several plugs on the back of a television, a synonym for an arcade game, and a name for a rock music medium. Even ignoring these, it is used to describe both the signal that drives a low-resolution NTSC display, as well as the signal used to drive a high-resolution graphic display. This paper will use the term video exclusively to mean conventional television-resolution signals, such as PAL, SECAM, NTSC, CCIR 601, etc. Any reference to high-resolution video will be called "graphics display signal", "graphics signal", or just graphics.

The hardware used to accept, control and display video is typically inexpensive and is available for PC class machines, although the price spectrum extends up to several hundred thousand dollars in the television industry. Spoken from the perspective of the workstation, a video signal arriving at the workstation is called video input, a video signal created in the workstation is called video out, and anything else is, of course, an external device.

Video Input, the Technology

Video digitizers have been available for several years, but in 1985, Parallax Graphics introduced the first low-cost video-graphics processor for a workstation that allowed a user to view 30 frames-per-second of video merged with high-resolution graphics. Today, a variety of solutions are available, but in general, they fit one of the descriptions below.

The video and graphic signals may be mixed on the "hot side", a term implying that the video signal is scaled up to match the high-resolution graphic signal, and then joined in the analog domain. Other systems convert the video into digital pixels so the mixture can be done in the digital domain, or "cold side". Another possibility is to digitize the video, place it into its own high-resolution graphics frame buffer, and then do a hot side mix of two high-resolution graphic signals, one with (primarily) video pixels and one with graphics.

An important aspect of mixing graphics and video is whether it can be done pixel-by-pixel and whether it can be blended "either-or", or more interesting combinations such as 50%-50% or 60%-40%. Sometimes called "alpha channel", there exists exhaustive methods for blending [Porter84], but it is most often done with the equation

$$Result = ImageA \cdot \alpha + ImageB \cdot (1.0 - \alpha)$$

where ImageA is the graphics and Image B is the video; it is computed for each RGB primary. The capability is usually represented by 8 bits, so that alpha can vary from 0/255 up to 255/255. More often, the hardware has (effectively) a single bit meaning "all graphic" or "all video", hence alpha can be 1.0 or 0.0. This capability of mixing two pixel sets was included in an earlier design, discussed in [Brunhoff1-89] and [Brunhoff2-89]. However, the concept is interesting to a wider audience among workstation vendors that have graphic displays with overlay planes. Consequently, the blending concept was factored out of VEX, and a separate effort has begun to encapsulate it into another X extension with the working title, BLEND.

There are essentially three classes of video input hardware available for integration with high-resolution graphic workstations:

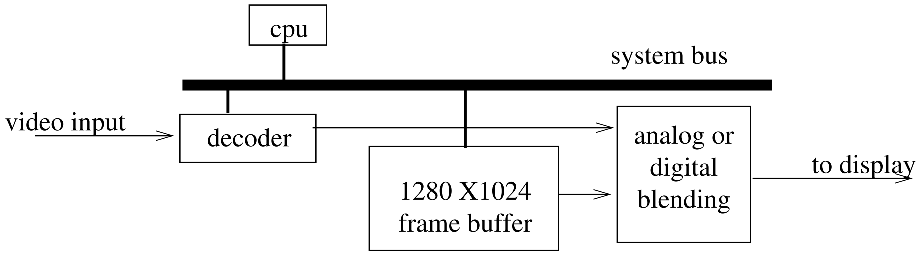

figure 1

Figure 1. Workstations with the ability to display full-motion video, but the video is never digitized. Video just "appears" on the screen when a magic pixel "key" value is placed in the workstation frame buffer, or a bit is set in a frame buffer mask, or both. The graphics and video may be blended on this hardware, but the number of levels varies. The mixture of video and graphics is done in analog, a hot-side mix.

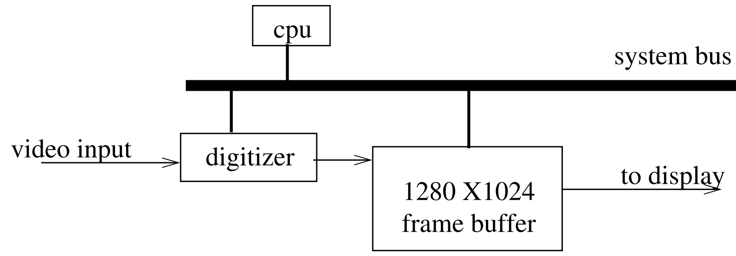

figure 2

Figure 2. Workstations with a single frame buffer that can contain graphics or video. Each pixel is either video or graphic at any point in time, therefore a a 1-bit deep alpha channel is implied.

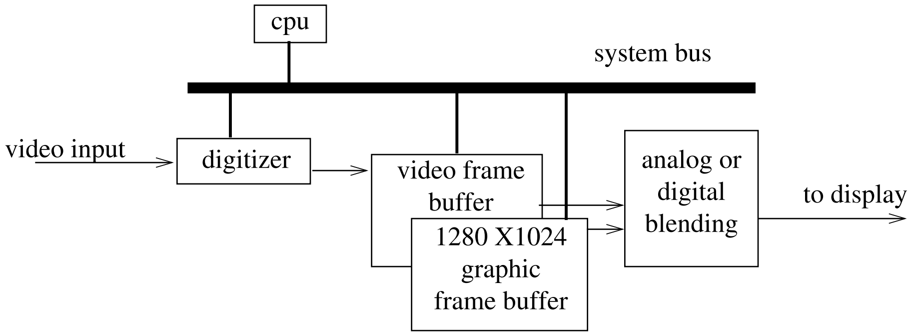

figure 3

Figure 3. Workstations with two or more frame buffers, probably one of which holds the graphics and another that can hold live video. These are likely to have one or more bits of alpha channel because both graphic and video pixels are available at once and can be blended. This architecture allows the designer to accomplish the mixture in analog or digital domains.

Video Input, the Applications

Applications are numerous for video input, and they can be conveniently grouped into five major categories: simulation, video post-production, cartography and geographic information services, image acquisition, and a catch-all class of video-in-a-window. Each is described briefly in the paragraphs that follow.

Simulation. Imagine a cargo jet with its wings flexing in wind turbulence. A scientist may study such an experiment on video tape, and then model the plane's wings with animated graphics. If it were possible to display both the experimental and computational data on the same screen, merging these two views, either by superimposing or juxtaposing them, it might reveal inconsistencies in one with respect to the other, or simply aid the scientist's understanding.

Video Post-production. Recently, the entertainment industry has become intrigued with the idea of merging the capabilities of video editing systems with graphic workstations. A computer able to provide a rich user interface combined with the ability to display video input in a window is being used in new products to create what is called a non-linear off-line editing system (non-linear because of the ability to address any frame without re-recording, and off-line because it can be used outside of the edit suite). This system allows the user to ignore the traditional "edit decision list" comprised of a complex list of numbers and timecodes, and concentrate on the artistic ideas. The concept is appealing to artists who must otherwise rely on an engineer, no matter how talented, for the execution of their ideas. Likewise, producers sometimes prefer to view productions outside of the edit suite on a personal workstation.

Cartography and Geographic Information Services (GIS). Landsat images and aerial photographs of the Earth exist in great abundance, and most are available as large images stored on magnetic tape. One-time mastering can be done onto CAV-format video disks for $300-500, using a mosaic of NTSC-resolution images for each original to preserve resolution. (Constant Angular Velocity, or CAV, is one format for commercial video disks). Video input capabilities can be used to "tile" these images on a workstation screen, filling the screen in one fifth of a second, and a video disk player (VDP) can provide random access to any of several Landsat images on each disk in two seconds or less. Current systems take five to ten minutes to read a single image from magnetic tape.

Image Acquisition. The production of natural scenes using a graphics engine uses natural textures mapped onto surfaces. Image processing applications are frequently used to work on images acquired from video. In both cases, a well integrated application might permit a user to peruse a video database or view live camera input on screen, as well as provide access to the digitized pixels.

Video-in-a-window. There are several applications that simply need a video picture on a screen. Video teleconferencing can use a workstation display to display the transmitted conversation. Also motion analysis, computer vision, video database perusal and other similar tasks can utilize video input.

Video Input and VEX

In VEX, video input may be rendered to a pixmap, or displayed in a window created with one of the core visual types: StaticColor, StaticGray, PseudoColor, GrayScale, TrueColor and DirectColor. (X uses visuals to describe the color aspects of a frame buffer. For example, PseudoColor indicates that each pixel in a frame buffer indexes a colormap for the color primaries; see [Lemke89]). As always, an X server implementor is constrained by the target hardware in the visuals and depths that may be offered by the X server with the VEX extension.

One of the instances of video hardware mentioned earlier is not capable of producing digitized pixels. For this reason, VEX introduces two new visual types, VideoGray and VideoColor that may be used for a "video-window". The semantics of these visual types make the window effectively write-only, so that you can put video in the window, but reading pixels or performing graphic operations that combine with the window's pixels have undefined results. VideoColor and VideoGray are distinguished only by whether they display a black and white picture, or color.

Besides these, video input hardware may have many other limitations on when and how video can be displayed. For example, the picture may have to be on sixteen pixel boundaries; it may be available only in one size, e.g. 640X480, or in sizes that are larger or smaller by a power of two; it may be that the video picture cannot be occluded by any graphics; the hardware may only be able to write pixels into the visible frame buffer, or only on off-screen memory. The extension design allows for these by informing the client of the capabilities and limitations through the QueryVideo request. The request provides a reference identifier for each physical video input hardware, and reports geometry, scaling, clipping and visual information for each. The reference identifier is used to create aliases for the video input with CreateVideo, which in turn can be used in parallel up to the concurrent use limit given by QueryVideo.

With this, a client has enough information to create appropriate windows or pixmaps, and place video in them with the RenderVideo request. The video picture may also be scaled in width and height, depending on published limitations. If any limitation is subsequently violated because a window manager moves a window, a client can be informed with a VideoViolation event by selecting Exposure on the destination window.

Video Output, the Technology

Workstation hardware that provides the ability to create video signals is available from several manufacturers, and possesses one or both of the following features:

- Downconversion. This is the ability to convert all or part of a high-resolution graphic display into a low-resolution video signal such as NTSC, usually at full video rates (about thirty frames per second). Only one product on the market can downconvert variable portions of the screen, an indispensable feature when integrating with a window system such as X or NeWS, but I expect this to become a trend in new products.

- Addressable pixels. A few products also allow direct access to the frame buffer that produces the low-resolution video signal. This can be used to implement a virtual variable downconverter by doing the resampling for downconversion in software.

Video Output, the Applications

The applications, from the perspective of a window system, are more limited than with video input. Video output is to a graphics display what a printer is to a traditional terminal, but with an added dimension because video is inherently capable of capturing temporal information along with the spatial.

If we consider only applications interesting in the context of window systems, there are still many programs whose windows contain things worth recording on video tape. In any case, the applications appear to lump into two camps: video production and post-production, and visualization.

Video Production and Post-Production. Many studios have high-end graphic workstations used to compose, preview and render graphic animations for television commercials, station logos and the like. The final step is to record the rendered frames on a recording medium such as D1 (digital) or one-inch (analog). Because of industry pressure for quick delivery, it makes sense to have video output capabilities directly available on the workstation. Current environments require each image to be sent across an ethernet connection to an Abekas A60, a special purpose recording device; the transfer takes seven to twenty-seven seconds. Workstations with video output capability can do the same operation an order of magnitude faster.

Visualization. Many research institutions use graphics engines to model designs, simulate natural phenomena, and view n-dimensional data. Frequently these organizations use video output capability to capture these visual animations and distribute them to colleagues with video tape or campus-wide video networks.

Video Output and VEX

Like input, video output may have restrictions on the regions of the graphics frame buffer that can be captured and encoded into a video signal: at full video rate, it may only be able to capture a fixed rectangle from the frame buffer; it may be able to capture just a subset of available visual types, or perhaps only from off-screen memory.

These limitations and capabilities are provided by QueryVideo along with a reference identifier for each video output device. A client uses this information with the CaptureGraphics request to produce a video output signal, optionally scaled. Also like video input, if a limitation is violated, perhaps because a captured window has become occluded by a window with an unacceptable visual, a client can alerted with VideoViolation events by selecting Exposure on the source window.

VEX SEMANTICS

Besides bare video input and output, there are several attendant problems that must be solved. How is a single video input resource represented to two or more clients that demand its use? What mechanisms should be available for a client to control the device that provides the inbound signal for the video input, or the device that receives the outgoing signal from the video output? How does an application know when the video picture is visible on the screen? If performing a CopyArea (a core protocol request used to copy a screen region) takes longer than a frame of video, then should the result be undefined or should it appear as an atomic operation?

Some problems are solved by providing information to the client through the informational requests, QueryVideo and QueryVideoControls; or with events, selectable through the SelectVideoEvents request. Others are addressed by defining explicit semantics (VEX defines a CopyArea from a video window to be atomic with respect to a frame). The current VEX design attempts to achieve a reasonable result with semantics that deal with known problems. The next five sections discuss the issues and how they are addressed.

Video Devices

Some contributors disagree on whether control of video devices like a VDP, belong in X. Consider image processing and 3d graphics: in image processing, the operations performed on an image in a window system is independent of the file format used to store the image on disk; with 3d graphics, the transformations and rendering of an object is independent of the structures used to represent the object when it rests in the file system. In both cases, obtaining the image or object from a file system is a separate operation from manipulations that happen within the window system. Video, on the other hand, is not separable from the source or destination. Without some external device *continually* supplying the signal for video input or recording the signal from video output, the system is useless. During a design, you must ask whether certain operations are interesting to an application written under such a system. If you review most of the applications cited above, few are independent of their source or destination. The exception might be video-teleconferencing where the remote camera is out of the user's control (although controlling which camera's picture is being displayed is still of interest).

To satisfy all parties, the protocol is designed so that the server may advertise many or few limitations on its capabilities. For example, a server may advertise that it has an input device, but that it has no control over its source: it looks like a cable TV channel. Alternatively, it can say that it is able to switch among three devices, two video cassette recorders (VCR) and one VDP.

Each device is identified by a VIDEODEVICE id provided by QueryVideo. Potential connectivity is made known by QueryVideo through adjacency matrices that represent the state of the network among video devices, and a list of potential transitions between states. Each matrix is an array of bits that represent whether the rows and columns, labeled by the video inputs, outputs and devices, are connected. The connectivity is controlled by the ChangeConnectivity request. Clients can track the state of the network with VideoConnectivityState events, and the state of devices (on-line or off-line) with VideoChange events, both by selecting DeviceNotify.

Video Controls

After connecting a device, one must provide controls for the logical knobs, dials, buttons and switches that control playback speed, direction, frame seek, recording, etc. A video input may be able to control display parameters such as color, hue, saturation and brightness, or fine tuning characteristics such as the coefficients for bi-linear interpolation (zoom). A video output may offer ability to control whether luminance alone is encoded or chrominance as well, or even the width and coefficients of the finite impulse response (FIR) filter used for downconversion.

Rather than provide access to a virtually infinite description of controls, VEX offers a raw mechanism for reading and changing these knobs and dials, called controls and settings. A list of controls is returned by QueryVideo, and each provides a name in the form of an atom (an ASCII string transformed into a 32-bit number by the server), an id that identifies the input, output or device it is associated with, the format and length for settings, and some optional data that describes fixed information about the control. Note that each input, output and device may have more than one control.

A setting consists of the name (atom), the id, and data whose length and format conform to what was returned in the control structure. The client may examine a device's current state in the setting returned by QueryVideoControls, or change it by giving a setting to ChangeVideoControls. Clients can track setting changes with the VideoControl event by selecting ControlNotify on the resource.

It is expected that convention will determine the names used for atoms, the kind of controls to expect on each device, the format and content of the data fields, ant the semantics associated with changing a setting.

Resource Ownership

The total resources required by a typical set of processes running under UNIX are likely to exceed the physical resources actually available. Compute and memory resources, for example, are represented as virtual resources; i.e. most processes do not see that the cpu is not constantly in their possession or that portions of their memory have migrated to secondary storage.

Time sharing does not extend well to video applications because in many cases, either things are happening at thirty frames per second, or things aren't happening at all. A process doing computations doesn't care whether it finishes in one second, four seconds or five; but if a video application can't produce or display full-speed video, it is unacceptable. There are a few counter examples to both, such as network packets that expire after some time; and the GIS application cited earlier that tiles the screen with a Landsat image. The problems have been recognized in video [Applebaum89] and in audio [Arons88].

The mechanisms in VEX allow direct control of a device, but don't preclude using an underlying mechanism such as Applebaum's Galatea. However, the latter has an implicit policy that gives a resource to the last client to ask for it. VEX introduces a concept of ownership for video input, output and devices.

Ownership is meant to be handled by conventions implemented in a central video resource manager analogous to a window manager. Clients should express interest in resources through properties, in the same way that clients express interest in colormaps; see ICCCM [Rosenthal89]. A video manager will use some heuristic to decide which set of clients need the video resources, and assign ownership of those devices to the respective clients using the ChangeOwnership request.

The ownership of devices is limited to a single client, and ownership of video inputs and outputs are limited by the concurrent use number given in the VIDEOGEOMETRY for that resource (VIDEOGEOMETRY is a structure returned by QueryVideo for each video input and output). This allows a video manager to distribute a video input device that can display four separate video signals at once across four clients that need one video input each. This also precipitates the same class of events that the X protocol provides window managers for intercepting window map and configure requests [Scheifler89]. A video manager can select VideoRequest events, which are generated whenever a RenderVideo or CaptureGraphics request violates ownership of a video input or output, respectively. When the video manager decides that the client should continue, it can reissue the request on behalf of the client with the information given in the event.

VEX's concept of ownership is lenient: if a resource is not being used by its owner(s), then any other client may use it; but when an owner requires the resource, the non-owning client loses it.

The SyncNotify event tells a client when a RenderVideo or CaptureGraphics request is active, and a VideoOverride says when a request is stopped because of ownership conflict. A stressed environment has yet to be observed, but I believe a video manager that utilizes these mechanism will allow clients to coexist. The clients can survive this environment by making their render and capture requests, and doing all activity between pairs of SyncNotify and VideoOverride events. In addition, simple environments that do not have contention for video resources should be able to exist without a video manager.

STATUS

At the time of this writing the protocol is complete after a swirl of activity that resulted in blending semantics being extracted and placed in a separate extension called BLEND. Implementation is being done at Tektronix with a Tek 4301 compute station and a Parallax 1280 video graphics processor. Later this year it will be ported to Tek's XD88/10 and XD88/30 using Tek's own video hardware. The two software ports should give the VEX implementation some level of generality. The software is about 85% complete and the Xlib and server code will be generally available without charge through the MIT X Consortium sometime in mid- to late- 1990. Members of the Consortium will have access to early versions of the software by the end of 1Q90.

ACKNOWLEDGMENTS

The VEX design has received useful input from several parties. The primary contributors are Tim Glauert, Olivetti; Robert Scheifler, Director, MIT X Consortium; Keith Packard, MIT X Consortium; Dave Carver, Digital Equipment Corp.; and Scott Hennes, Tektronix;

The invited reviewers who provided useful input are Dan Applebaum, MIT Media Labs; Martin Levy, Parallax Graphics; Gloriana Davenport, MIT Media Labs; Matt Hodges, MIT Project Athena; and Barry Arons, Olivetti.

Special thanks go to Scott Hennes, Tektronix, for constant feedback, pervasive input for fine tuning the protocol, and a large portion of the implementation.

REFERENCES

[Applebaum89] "Galatea, Version 2.0"; D. Applebaum; Massachusetts Institute of Technology Media Laboratory; 1989.

[Arons88] "The Vox Audio Server"; B. Arons, C. Binding, K. Lantz and C. Schmandt; 1988.

[Brunhoff1-89] "VEX Your Hardware"; T. Brunhoff; Xhibition '89 conference proceedings.

[Brunhoff2-89] "Pleasing the Eye"; T. Brunhoff; Unix Review, October 1989; pp. 65-72.

[Davenport] "Software Considerations for Multimedia Video Projects"; G. Davenport; Massachusettes Institute of Technology, Media Laboratory; 1988.

[Hodges88] "The Athena Muse User's Manual."; M. Hodges; Massachusetts Institute of Technology, Athena Project; 1988.

[Hodges89] "A Construction Set for Multimedia Applications"; M. Hodges, R. Sasnett, M. Ackerman; IEEE January 1989.

[Lampe88] "Athena Muse: Hypermedia in Action"; D. Lampe; The MIT Report; February, 1988.

[Lemke89] "Visualizing X11 Clients"; D. Lemke, D. Rosenthal; Winter 1989 Usenix Conference; pp. 125-138.

[Levy87] "X Video 'C' Programming Interface"; M. Levy; Parallax Graphics Company; Unpublished.

[Ludwig87] "Overview of the Integrated Media Architecture Laboratory", L. Ludwig, Dunn; ACM SIGSOM 1987.

[Picco88] "Live Digital Video in a Windowing Environment."; M. Picco; 1988 NCGA Conference Proceedings, v.3, pp. 643-663.

[Porter84] "Compositing Digital Images"; T. Porter, T. Duff; ACM Computer Graphics, Volume 18, Number 3, July 1984.

[Rosenthal89] "Inter-Client Communication Conventions Manual"; D. Rosenthal; MIT X Consortium Standard.

[Scheifler89] X Window System; R. Scheifler, J. Gettys, R. Newman; Digital Press; ISBN 1-55558-012-2; 1988.