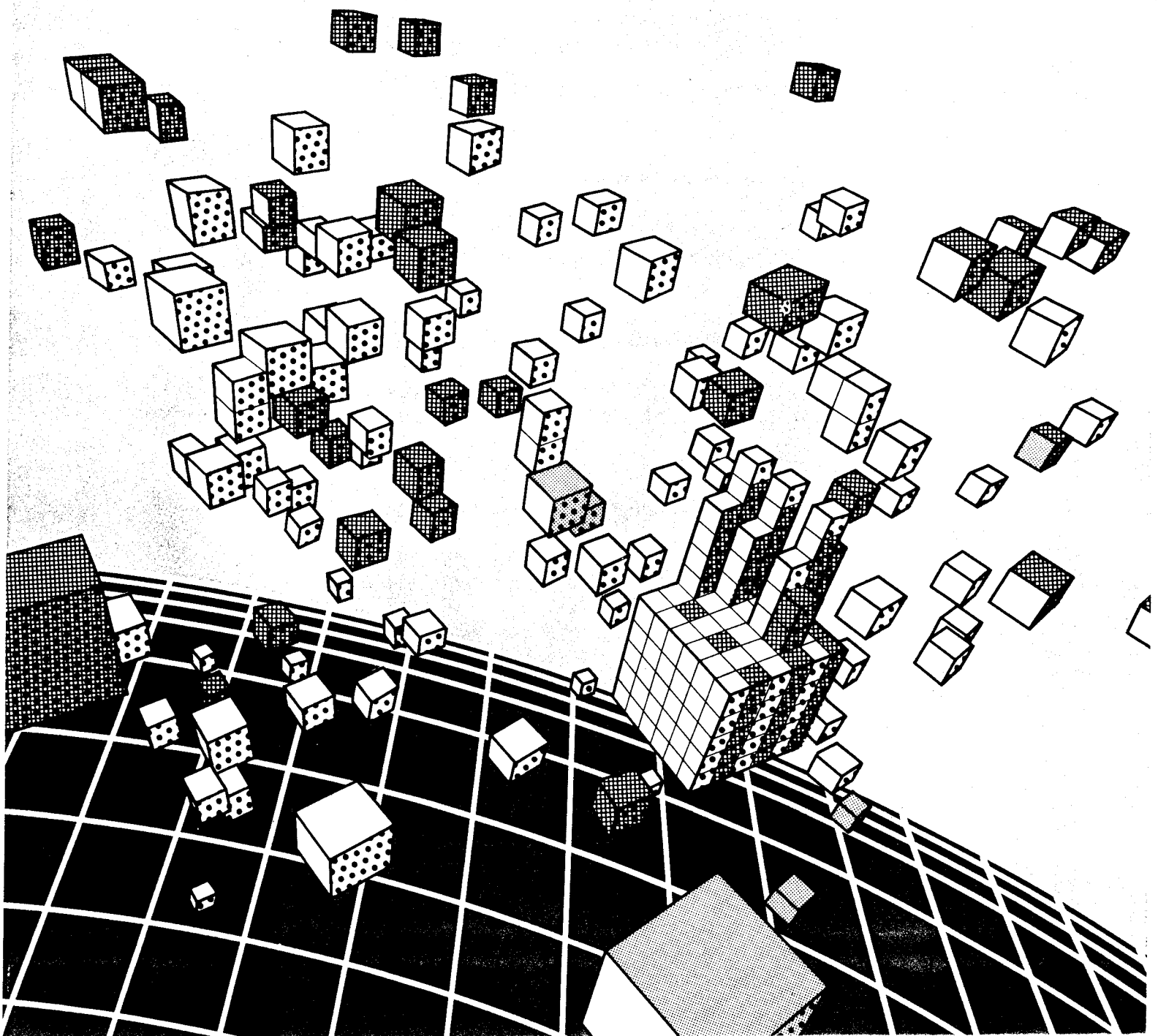
INTERGRAPHICS '83

Technical Sessions B 7



This Page Intentionally Left Blank

EXTENDED MEMORY USE IN THE ZGRASS GRAPHICS SYSTEM

Thomas A. DeFanti
University of Illinois at Chicago
Department of EECS
Box 4348 Chicago, IL 60680, U.S.A.

Computing has always been faced with the choice of optimizing for time (computer power) or space (memory). Real-time computer graphics is most likely the subset of computing that must choose between space and time most effectively. This paper describes in detail the evolved techniques of software memory management used by the Zgrass system to allow a slow (1.7mhz) Z-80 to be the processor for a useful computer animation workstation with at least 300K and up to 1472K bytes of memory. Conclusions will be drawn as to the applicability of these techniques to the new 16-bit microprocessors with memory management.

Background

The system being described here is formally called the Datamax UV-1. Since its software is called Zgrass, an earlier version of which is described in [1], we will refer to it as the Zgrass system. It actually started as a home computer graphics system in 1977 and later, after having been rebuilt for the cable television industry, became an artist's workstation for video production. The Zgrass system was developed to make people want to program by providing rich animated feedback with few initial barriers, yet give the user the ability to unravel layers upon layers of built-in sophistication when and if the desire arose. Rather extensive testing of the software has occurred at the University of Illinois at Chicago and the School of the Art Institute of Chicago where hundreds of art and engineering students have used it for the past several years and about 100 systems exist at this point. It is a system that encourages the writing of software tools as extensions to the command language and it makes possible the performance of real-time interactive visuals. Most of the systems are used by artists most of whom have become programmers over the years.

Difficulties of the Z-80

In 1977, the Z-80 was a rather advanced chip compared to the 8080, but in 1983, it is considered slow and primitive. In 1977, 64K bytes of memory was the limit for a personal computer; today we talk of megabyte ones. In 1977, one was lucky to have a macro assembler and a linking loader that worked. Now, assembler coding may be largely abandoned for programming in C.

The Zgrass Custom Chips

A research effort at Dave Nutting Associates, a division of Bally Corporation, produced in 1976 several custom integrated circuits to assist the Z-80 in making real-time color video at 320x201x2 bits per pixel resolution. A 1.7mhz Z-80, of course, can barely do more by itself than drive the terminal this paper is being written on if it has to generate video. These custom chips which have been used in several coin-operated video games are still in production and were the basis for the home system called the Bally Arcade. The chips have the distinct advantage that they form a true bit-mapped frame-buffer and generate rather good NTSC video, a rarity in personal graphics systems then and now. These chips were very closely tied into the Z-80 so the development of Zgrass was locked in to both the chips and the Z-80 as well. Some 20,000 lines of assembler code had been written, so going to a faster processor was not possible; faster Z-80's would not work with the custom chips. Thus, the only reasonable way to improve performance was to expand memory.

At the time of the BYTE article (1980), the Zgrass system had 16K of ROM, 16K of screen RAM, and 32K of user RAM. The ROM, later expanded to 32K, holds system code so the machine can run without booting anything from disk. The screen RAM uses 32 chips so it can be accessed 32 bits at a time in parallel to generate the video signal. The 32K user RAM is used for user programs, subpictures (called SNAPs), arrays, and so on. It is dynamically allocated and reclaimed in 4K or less blocks according to a highly tuned best fit algorithm. 32K is not much space for SNAPs, arrays and programs so the disk software was designed to be more or less an extension of user RAM.

Inexpensive 5" floppy disks are slow, however. The one we chose takes four seconds to go from track 0 to track 76. The use of the disk to store software tools and SNAPs works well but is slow since the disk has to seek quite often. Even winchester-type disks, although 20 times faster, still do an unfortunate amount of seeking. Since we were using the disk as extended RAM, it was an obvious step to load the disk into RAM and fetch out of it, so we did it. Of course, the Z-80 does not have memory management so the hardware was built rather straightforwardly and the software provided the flexibility needed.

The Current Memory Architecture of Zgrass

The system now supports 32K of ROM, 256K of screen RAM, 32K of user RAM and up to 576K of EPROM or 1152K of ROM. (An additional 32K of RAM is used to support a 64K CP/M (CP/M is a trademark of Digital Research, Inc.) system but it is not accessed by Zgrass so it will not be further discussed.) All the extra memory added is mapped into the address range 16-32K so that it can be accessed by the custom chips.

The custom chips have special features which are enabled by addressing the screen during writes to memory Ø-16K instead of 16-32K. Since there is always ROM at Ø-16K, this is not a waste of addressing space. When a write goes to Ø-16K, the data is manipulated by the chips according to an 8-bit value held in what the designers call "the magic register." The bits in the register, which you set by writing to a port, specify two extremely useful operations between the new data and the data on the screen: logical or and logical xor. They also allow the data to be shifted Ø, 1, 2, or 3 pixels before writing to the screen. The shifter is critical to the implementation of animations since pixels are packed four to a byte (2 bits/pixel) and laying a pattern down on non-byte boundaries would otherwise require pitifully slow shifting, masking and writing of each pixel in Z-8Ø machine instructions. The shifter does read-modify-writes and all the masking to make transfers to memory with shifts as fast as the logical function writes. Straight access to screen memory can be done by addressing 16-32K. Of course, reads from Ø-16K yeild instructions and data from the system ROM.

In the middle of 1981, memory prices dropped on 64K bit RAMs. Since we had 32 sockets for screen memory anyway, it was simple to modify the system to accept 32 64K bit RAMs to give 256K bytes of screen RAM. The difference in cost is about $200.00. Changing the amount of memory by 16 times in any system profoundly affects its performance. The subject of this paper is describing just how profound the change is.

(Just for completeness, the second 16K of the 32K system ROM also overlaps the addresses 16-32K. The code in that 16K is constrained to never read or write screen RAM. All the graphics code is in the lower 16K or loaded into the RAM located at 32-64K.)

Keep It Obvious and Friendly

One of Zgrass's main tenets is "keep it obvious." Computer hardware tends to be devious and clever so keeping it obvious requires an astonishing amount of creative thought, testing and reworking. Yet, flexibility is also a primary goal, one more important to the user with a task at hand than anything else. Before explaining how the extra memory is used in detail, let us state it briefly:

The 256K of screen RAM can be used for:

a. sixteen screens, switchable instantaneously
b. four screens plus 192K disk cache
c. four screens plus panorama screen(s)

"Keeping it obvious" allows one to provide flexibility even if it is forced to be idiosyncratic by the hardware. Zgrass implements special device variables which are always set to default values on startup but may be altered by the user when the feature the variable specifies is needed. Specific to

this discussion, $MW indicates on which screen of 16 the writes should be (e.g. $MW=5 means write on screen 5). $TV is set to the screen you wish to view on the television monitor. Clearly, you can be modifying one screen while viewing another, easily implementing double or even 16-buffered schemes. Thus, with a procedure not much different from changing the channel on a television set, users can have easy access to 256K of screen RAM. $MW and $TV are taken modulo 16 normally. However, when disk cache or panorama modes (b. and c. above) are enabled, $MW is taken modulo 4 to prevent accidental destruction of data.

The Disk Cache

The disk cache is setup by the DLOAD command which reads a whole floppy disk side into 12 of the 16 screens. Both floppies and winchester-type disks are formatted into 192K byte logical units comprised of 384 512-byte sectors. After DLOAD'ing, all subsequent reads and writes to that logical disk actually go to the cached memory. DLOAD.ZAP writes the disk back out again. The disk cache eliminates all seek time and transfers programs, arrays, SNAPs, and so on at memory to memory speeds (80K bytes/second at 1.7mhz clock rates). Of course, a disk file structure is not the optimal way to use RAM, but it has the distinct advantage that the modifications to the system to increase the user's RAM by sixfold only required a small amount of code to be added. More important though, the user has no problem understanding the disk cache, can move easily from a 16-screen system to a 4-screen system with disk cache. Furthermore, user backup is easy to do. If the 256K memory had been available when the system was designed in 1977, a different scheme might have been used, although given that the screen RAM is constrained by hardware to contain only data (the Z-80 cannot execute code out of it), the 256K memory would need special treatment anyway.

With the disk cache, a rather elaborate paint program was written in Zgrass by Copper Giloth to enable artists to draw and animate. The numerous modules are loaded and executed without perceptible delay based on menu choices of the user. Animations occupying about 1/4 the screen can be easily animated at 20 times a second.

Panoramas

The third option for using the screen memory is by building panoramas. This option was designed and implemented by Stephen Joyce, the author of most of the graphics code in Zgrass. The BUILD command allocates some or all of the last 12 screens as one or more "super screen." One can specify a single 3x4, 4x3, 1x12, 12x1, 6x2 or 2x6 super screen. A 3x4 super screen, for example, has dimensions 960x804, given that each screen is 320x201. Or, you can have several smaller super screens like two 2x3's or six 1x2's for example. The

DISPLAY command which ordinarily places SNAPs anywhere on any screen has an option to use a super screen instead of a SNAP as the source. The data is clipped to either the whole current screen (as specified by $MW) or a subset of the current screen set by the WINDOW command. Thus, a large image like a map may be viewed through a 320x201 or smaller window and you can roam around quite easily and quickly. SCALE is a command that works like DISPLAY except that it allows shrinking or expanding of the data while writing to the current screen. The PLACE command stores rectangular areas of the current screen on a super screen.

Following the next section on EPROM/ROM disks, an attempt will be made to justify which of these memory structures make sense in a system with lots of memory (like 68000's, PDP-11's and Z-8000's with memory management). Zgrass, of course, is an experiment in inexpensive graphics technology for personal access by artists and educators and, in such, provides many lessons to the designer of a new system.

EPROM/ROM Disks

The Zgrass system is ideal for the cable-tv operator who desires graphics better than those offered by teletext systems. Zgrass in this mode acts like a remote character generator with animation capability. Several problems had to be solved for this application, however. First, a suitable way to send commands and data had to be designed so that human operators would not be needed. This was not very difficult and was quickly done. Second, rotating memories like disks are simply not rugged enough for the environment of a cable head end block house. Cable TV equipment is designed for negligible downtime so a disk without moving parts had to be designed. Clearly, mass chip memory was the only answer.

The EPROM/ROM disk is configured as a board with 24 8K byte EPROMs (one 192K disk image) or 24 16K byte ROMs (two 192K disk images). For hardware simplicity, the maximum number of boards is three so a total of 576K EPROM or 1152K ROM may be installed. Picking 192K as the logical size once again allows the user to fully debug the package on a floppy, winchester-type or cache disk before committing to EPROM or ROM. Once the application is ready, it is a simple matter to transfer the whole disk to EPROM using a conventional EPROM programmer. ROM's, of course, have a much more involved manufacturing process. Once again, adding support for the EPROM/ROM disk required only a tiny increment in code given that this memory also resides in pages at the 16-32K address space.

Thus, it may be observed that this modest Z-80 system may be configured to have up to 1472K bytes of memory, all but 48K of it mapped into 16K pages at 16-32K.

Applicability to 16-Bit Systems

The Zgrass community eagerly awaits a higher-resolution system. A 640x480 screen requires 38,400 bytes/bit plane. To maintain the animation speed, a much faster processor is needed. Fortunately, Z-8000's, 68000's and PDP-11 chips are fast enough and also allow development of the software in the C language. Current work is proceeding on a VAX in simulation mode for several types of graphic display units. Faced at this point with a total re-design, what is worth keeping?

Without a doubt, the EPROM/ROM disk is a good idea. Rotating memories are simply unacceptable in poor environments where low-cost graphics may be needed. The EPROM disk is also quite a bit cheaper and much faster than disk drives. Its maintenance-free, operator-less operation is very desirable. It also fits right into a card rack using available power. These benefits, of course, are recognized by home video game manufacturers who supply software on ROM cartridges.

The disk cache is also a transferable concept. Creative users of a programmable system have no trouble dealing with disk files, if only to facilitate creation of libraries of software tools and images. Having disk images execute out of memory saves time and considerable wear on the mechanics of the disk drives.

The panorama idea also has validity in higher-resolution systems. Hardware support for choosing the window would be desirable so roaming around a large database could be done in real-time. Hardware scaling would also be quite useful.

User main memory, limited to 32K in the current Zgrass, should be much larger, in fact, expansion to any affordable size should be automatically supported. The current memory allocation and reclamation schemes are quite usable, however, and work well enough to be modified for much larger memory spaces.

It is also clear that multiple screens are important. Two screens allow double buffering; more allow animation. Although 16 have been very effective in Zgrass, a new system should provide for as many as the user can afford to buy. Five seconds of full animation at 12 screens per second (the speed of conventional animation on two's) requires 60 screens. Of course, at a resolution of 640x480x8 bits/pixel, 60 screens require 18 megabytes of memory. High resolution has its price, although, at current costs, 18 megabytes is not out of the range of studio broadcast television equipment.

Conclusions

This paper has narrowed its focus to memory paging techniques found useful in extending a Z-80-based graphics system to fully utilize a large memory space in a user-friendly way. Many of the techniques are directly applicable and desirable in

systems having much greater memory addressing capability. Working on a small, low-resolution animation system with extensive memory has given insight into how to design higher-resolution workstations for artists, and much practice with delivery systems in situations applicable to videogames, interactive movies, education, public information displays and conventional television.

Reference:

[1] DeFanti, Thomas A., "Language Control Structures for Easy Electronic Visualization," Byte, Vol. 5, No. 11, November 1980, pp. 90-104.