Department: Visual Computing: Origins
Editors: Chris Johnson, crj@sci.utah.edu
Dave Kasik, dave.kasik@gmail.com
Mary C. Whitton, mcwhitton@gmail.com

Interactive Graphics in Industry: The Early Days

David J. Kasik
The Boeing Company-Emeritus,
Battelle-Columbus Laboratories (1972-1977)

John C. Dill
Simon Fraser University-Emeritus,
GM Research Laboratories (1969-1980)

***Abstract*—Computer graphics has a long history. Industrial organizations and laboratories drove significant improvements as they adapted and assembled basic capabilities into complex interactive applications. Of particular concern in the early days was providing interactive 3-D applications for computer-aided design and engineering. This article describes the experience of two early industry practitioners who built successful 1970s interactive 3-D systems.**

Computer graphics was christened in 1960. Most early graphics devices produced paper, microfilm, and microfiche output. Early interactive graphics (1960–late 1980s) focused on computer-aided drafting (CAD). Even though both industry and academia contributed, CAD was driven significantly by product manufacturing companies, especially automotive and aerospace.

Product manufacturing companies like Lockheed and Renault created thousands of 2-D engineering drawings. Creating and correcting ink-on-mylar drawings was labor-intensive and drawing configuration management error prone. A few far-sighted companies saw the value interactive computing and computer graphics could bring and began working on 2-D CAD applications.

Others [e.g., General Motors Research (GMR) Laboratories] saw the value of interactive 3-D CAD: GMR started work on its design augmentation by computer (DAC-1) system[1] even before Ivan Sutherland developed the seminal Sketchpad system as his MIT Ph.D. thesis.[2]

Companies would have far preferred buying turnkey applications. Unfortunately, commercially viable solutions just did not exist. Computer vendors (e.g., Control Data, IBM, and DEC) supplied

Digital Object Identifier 10.1109/MCG.2020.3012228
Date of current version 24 August 2020.

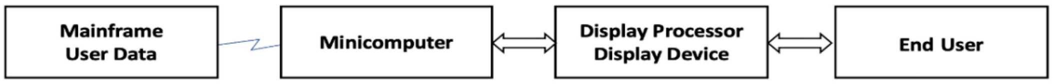

Figure 1. Generic Interactive Components

only hardware and some basic operational software. Buyers had to develop their own sophisticated interactive applications. In addition, many of the fundamental components needed for highly interactive applications didn't exist. Minicomputers did not have sufficient computing power or storage capacity. Mainframe operating systems (OSs) were aimed at batch jobs and could not handle the interactivity demands needed to support hundreds of users. Remote users were connected with slow telecommunications lines.

Drafting, fundamentally a 2-D task, drove much early CAD development. However, organizations like GM Research Labs and laboratories like Battelle realized that the real value lay in working in 3-D. GM and other automotive companies wanted to do 3-D CAD as computer-aided *design* (not drafting) to augment the full-sized clay models they constructed to capture surface shape. Automobile bodies contain smooth, complex geometric surfaces that define the products' aesthetics and aerodynamics. Other organizations needed 3-D computer-aided engineering applications. Battelle and others were starting to use 3-D finite element methods to supplement and even replace classic stress analysis methods.

GM and Battelle wanted to move to interactive 3-D systems. By the early '60s, GM had demonstrated that interactive computer graphics could make designers more effective.[3] Battelle staff was using punch cards to define 3-D models. Therefore, almost any interactive system would increase model understanding. Both organizations faced similar problems implementing interactive 3-D systems.

Our job seemed like a *Mission: Impossible* task: Conceive a system architecture based on commercially available gear, integrate the latest research from industry and academia, and do original research to fill in gaping holes.

We had to work across all aspects of a rapidly evolving hardware landscape: mainframes, communications networks, smart peripherals (including the then new minicomputers), graphics hardware, and interaction devices. Interestingly, each of our quite dissimilar organizations chose to move toward a distributed model based on an intelligent, terminal-based graphics system (DEC GT40/GT48 at GMR, CDC 777 at Battelle). Each intelligent terminal relied on a general-purpose minicomputer (DEC PDP11/05, CDC SC17).

Each of us also worked across the complete software stack: geometry algorithms, the user interface, data bases and data management, graphics packages, and operating systems. At a high level, we both succeeded in building useful and usable interactive 3-D systems although the computing environments we worked with and detailed architectural decisions we made were quite different.

In this article, we share a number of issues and problems we faced in those early days. We observe that today's systems have similar architectural challenges. The details may differ, yet the decisions are just as critical and just as difficult.

BACKGROUND

We briefly examine 1970s hardware, what was available, and enough of how it works to explain the problems we faced building interactive 3-D systems.

System Components

In the 1970s, many graphics systems contained the components shown in Figure 1. Current systems are similar, though sometimes with different component names (e.g., PCs rather than minicomputers).

Comparing graphics device characteristics, network, memory, graphics acceleration, and CPU performance between the 1970s and current systems shows the limited compute power available to us in the early days (see Table 1).

There were numerous options for the generic system components. Larger organizations like Battelle and GM needed more computing power than the 1970s era minicomputers could provide and relied on mainframes from IBM, Control Data (CDC), Burroughs, etc. The mainframes of yesteryear provided centralized timeshare and batch services. Some companies (CompuServe, Boeing Computer Services) sold mainframe time the same way Amazon, Google, and Microsoft sell cloud computing services today. The mainframes connected secondary storage for user data devices (e.g., disks, drums, tapes), batch input devices (e.g., card readers), and output devices (e.g., printers).

Smaller organizations and research groups with limited budgets often did all their application processing and data storage on minicomputers, devices that generally supported only one user at a time. The minicomputers had limited programming tools and often ran only special-purpose code.

Mainframe manufacturers sold their own graphics devices. The devices were connected to the mainframe over short distances (hundreds of feet). Connections were often based on proprietary communications protocols designed to maximize data transfer rates (e.g., IBM 2250[4] and its channel connection*).

Not all graphics devices relied on a minicomputer. Many devices acted as conventional terminals that had an added drawing capability. They were attached to either a mainframe or minicomputer via conventional telecommunications lines. The Tektronix direct view storage tube (DVST) is a good example.

System components came in many different flavors. As system developers, we had to choose components that would optimize the whole. Fundamentally, our graphics applications had to be interactive for as much 3-D data as the system could handle. Being interactive and 3-D required optimization of the system architecture, algorithms, and individual components. This forced us to make tradeoffs that led to complex decisions.

Acceptable Interactive Performance

For our interactive applications to feel natural, we had to process input quickly enough to prevent discontinuities so that the user/designer's train of thought stayed intact. Instantaneous feedback, where the computer provides feedback for what the user perceives to be a trivial action (e.g., echoing a keyboard entry) means a response of 0.1 second or less. The complete loop (user input -> computer processing -> feedback to user) needs to finish in under 1 second to allow a person to keep his/her train of thought.[5] Users will tolerate longer response times for complex processing if the computer provides a "Working" indicator.

It was essential for both of us to understand how display processors and devices worked so we could provide sufficient interactive performance for applications to feel natural.

*A channel is a high-performance input/output (I/O) architecture implemented on a number of computer architectures, especially on mainframes. It is a special-purpose peripheral processor dedicated to I/O operations between mainframe memory and peripheral devices (e.g., disk, graphics terminal).

Table 1. Components: Then and Now.

Component	<1977	2020
Graphics Subsystem	Random scan, monochrome	Raster, full color
	Lightpen, keyboard, dials, function key set	Mouse, keyboard with function buttons
	Cost ~$120K	Cost <$1000
Network	56 kB was fast	>500 MB
CPU Memory	Mainframe: 1 MB was big; Mini <64 kB	128 GB is small for a PC
Graphics acceleration	Dedicated board(s) for 2-D	Plug-in board for 2-D, 3-D
	Minicomputer for local processing	Multi-processors with >1.5 GHz clock.
	Display memory <65 kB	On-board memory >8 GB
Main CPU performance	Mainframe Clock speed <100 MHz	Clock speed >4 GHz in the cloud

Graphics Devices

Random scan (also called calligraphic, stroke, or vector) devices dominated interactive graphics in the 1960s and 1970s. The devices created images by moving an electron beam across the inner surface of a cathode ray tube (CRT). A special processor executes instructions in a loop. The instructions contain display (x, y coordinates and beam status) or jump commands. When the beam was on, it excited a rapidly decaying phosphor coating and the user saw a line between one screen position and another. When off, the beam moved to a new screen position. In this way, the displayed lines were drawn "randomly" on the screen surface.[†]

On random scan screens (e.g., IBM2250[4]), the phosphorus glow from the beam decayed in milliseconds and the display had to be continuously refreshed. To achieve an acceptable refresh rate, we had to limit the number of lines the display processor drew each cycle. If there were too many, flicker occurred.[‡]

Flicker and Its Causes

Two factors, refresh rate and frame rate, impact random scan graphics device performance.

First, it's possible to show a stable image on a medium that self-erases (e.g., a CRT and a monitor). A person sees a stable image on a screen coated with a naturally decaying phosphor if all the screen contents are redrawn at least 60 times per second, the rate that meets the human flicker fusion threshold. In practice, if an image has more lines than can be drawn in 1/60 second, the image will flicker. This makes the image more difficult to understand.[6]

Second, changing images over time lets a viewer perceive motion. The greater the frame rate (i.e., the number of new/changed frames per second), the smoother the motion can appear. In general, computers attempt to compute images for new frames as fast as the monitor refresh rate to create the illusion of smooth motion.[7] Interactive software must compute a new frame 60 times per second to meet or exceed the flicker fusion threshold.

Input

User input processing was handled more simply than display. Local, hardwired logic or processors echoed keystrokes on the screen. When the user pushed the Return key, the device sent the collected characters to the mainframe as a text field. Since display device screens were flat, users could enter (x,y) positions using a lightpen. Positions could be returned to the mainframe at a program-determined rate. The user could also place the lightpen tip on a visible line and depress the tip to generate an interrupt.

[†]Early raster devices also moved an electron beam across a screen coated with material that decayed rapidly. The beam moved in a fixed pattern of horizontal lines. Each point in the pattern (a pixel) is defined by a set of attributes (e.g., red, green, and blue). Raster devices increased in popularity in the early 1980s with offerings from Raster Technologies, Ramtek, and Ikonas.

[‡]Informal 1970s experiments at GMR showed that experienced designers would put up with some flicker just to see images of their data.

THE GM PROJECT: JOHN'S STORY

Fresh out of grad school at Caltech, I started a job at GMR in Warren MI at the end of 1969. I still believe I got the job in part based on my interview seminar on developing interactive graphics on an IBM 2250 for neurophysiology research.

Computer graphics began at GMR in the late 1950s with a feasibility study to address the question: "Could computer techniques significantly improve the design process?" GM wanted to understand the potential role of computers in the graphical phases of product design. The result was a major effort to build a laboratory to study graphical man–machine communication. This led to the DAC-1 project.[1] Other early GMR efforts included the Gordon Surface, a curved surface generalization of Coons patches.

Because of the nature of automobile design, DAC-1 was required to work with free-form curves. To provide compatibility with existing design procedures, precise input to define the curves was needed. These requirements argued against[1] a "sketchpad" approach, which dealt with 2-D prismatic wireframe objects.[2]

This early history ultimately resulted in an "existence proof" system capable of supporting production design, and in 1967, Ed Cole, then GM President, decreed that "DAC-1 was no longer a research project and that the responsibility for further development and application of this technology should be [moved to a production/maintenance group]".[1] Accordingly, the DAC-1 development group was divided into three groups. The first managed the existing DAC-1. The second looked at CAD across other GM Divisions. Ed Jacks led the third group, the one I was in. Our job was to identify and address the key computing tasks needed to develop a production quality, interactive 3-D CAD system. Many areas needed work: OSs needed the most. Time-shared OSs needed a complete rethink to deal with interactive graphics and a large number of simultaneous users. File systems needed software enhancements to support CAD data. This work became the GMR multiconsole time sharing system (MCTS) development project. The target platform was the yet-to-be-built CDC Star-100 with CDC GPGT graphics stations (a special version of the CDC 777 Dave discusses later).

When CDC ultimately couldn't deliver Star-100 on time, GMR reverted to IBM hardware. Our group had to find a new way to deliver interactive 3-D graphics in a cost-effective manner over distances measured in miles, not feet. IBM's solution, the 2250, was under-powered, expensive, and, most importantly, not able to be connected remotely. GMR selected the DEC GT40 random scan terminal with DEC's promise of a more powerful version in the "near future."

Goals

The MCTS project had to address a number of specific requirements. (1) Support a large number of design consoles: ~100 or more, an enormous number in the 1970s. (2) Provide a time-shared mainframe. Minicomputers were just appearing and had neither adequate compute power nor data storage for CAD. (3) Interactive graphics was mandatory. (4) Design consoles had to be effective for remote design groups and motor divisions (Design Staff, Fisher Body, Chevrolet Engineering at the Tech Center, Oldsmobile in Lansing, Buick in Flint).

Because the computing industry was still in its early years, high-speed data communication was still new and very expensive. Vendor-supplied, complete CAD "systems" were still in the future. Intelligent terminals and intelligent workstations were still research. Even defining "intelligence" was a hot topic.[8]

My graphics group's task was to figure out which components to buy from which vendors, how to modify vendor software to meet our much more stringent needs, and to develop the basic graphics package support to meet these needs. The basic 3-D algorithms and high-level design algorithms had already been developed. Our challenge was the graphics package. In today's practice, the graphics package is the "driver" bundled with a card and it is a total black box for application developers. In those days, it was the developer's responsibility to figure out how to adapt the graphics package to application requirements.

CAD at GM had evolved to the computer aided design and numerical control environment (CADANCE) system. CADANCE ran on mainframes and IBM 2250/IIIs. The 2250s were expensive and had to be connected by channel to the mainframe over less than a few hundred feet. GMR had to develop a (relatively) low-cost terminal that could be connected remotely, achieve interactive 3-D performance, and be useful for applications in addition to CADANCE.

Because one of the motivations for the intelligent terminal approach was cost, we could substitute mainframe processing power for costly graphics hardware and invest in developing remote communication software. Architecturally, we could rely on mainframe power to compute 3-D-to-2-D projections and transformations, fit curves and surfaces, and do database processing. The configuration we ended with was a DEC GT40[§] connected over an asynchronous, half-duplex voice grade telephone line to an IBM 370/168 mainframe running IBM's TSS OS. This was the development configuration; we would have to move to higher speed lines for production work.

General Approach

The system objectives and constraints we faced were daunting. We had limited memory (8K 16-bit words) on the PDP-11/05. Network communications were so nascent that we had to develop modem interface code and a communication protocol that handled both graphic data and text to and from the mainframe. The new system had to support IBM 2250 specific features that CADANCE and other legacy GM applications used. One consequence was that a considerable chunk of 11/05 memory had to be reserved for display instruction storage. This further limited the amount of memory for our own code. The new display device had to support multiple 2250-style input devices, including lightpen, keyboard, and a GM special function keypad. User feedback features such as highlighting (intensifying) a display entity "seen" by the lightpen, lightpen tracking, text input, etc., were critical.

Life became somewhat easier in the mid '70s when we moved to the higher performance GT48 (see Figure 2) (a GMR special version of DEC's

[§] The GT40 consisted of a PDP-11/05 minicomputer with 8K core and no disk. The graphics display processor drew lines on a 14" screen and had a lightpen for graphic interaction (ref Wikipedia https://en.wikipedia.org/wiki/DEC_GT40).

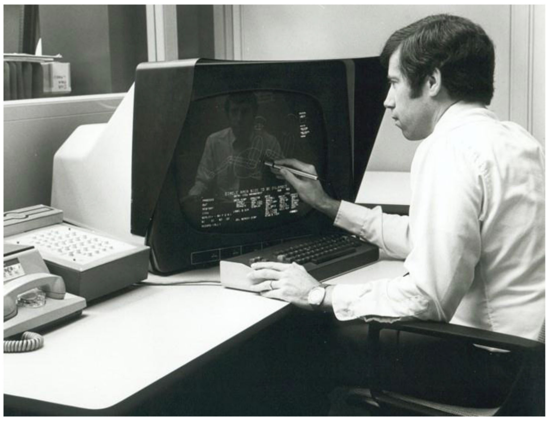

Figure 2. John at a DEC GT48. Interaction devices included lightpen, keyboard, and function buttons.

GT62) and a DEC PDP -11/34 processor with 16K memory.

Programming the PDP-11/05 itself was challenging. There was limited software support (an assembler, the RT-11 OS, which we ignored—we needed interactivity, not real-time). We had to program advanced lightpen support, support a function-key set like that of the 2250, and provide reasonable communications speeds even though DEC delivered no communications software. We also needed to decide what additional functions, if any, could be put in the little PDP-11/05. In the end, we discovered little else could be added.

The overall configurations are shown in Figure 3. For details see the article by Dill and Thomas.[9]

Challenges

There was no lack! The first was developing an approach to the division of labor between the intelligent terminal and mainframe; the mini was only a PDP-11/05 with limited memory, and limited speed. Of course, we wanted to move as much function into the terminal as possible, but the limits just outlined argued for an intelligent "terminal" rather than intelligent "satellite."[8]

Code reliability was also an issue: software had to be as close to rock-solid as we could make it. We had to satisfy our large user group doing production design. Downtime had to be no worse than the 2250 even though we were dealing with a much more complex configuration and increased functionality.

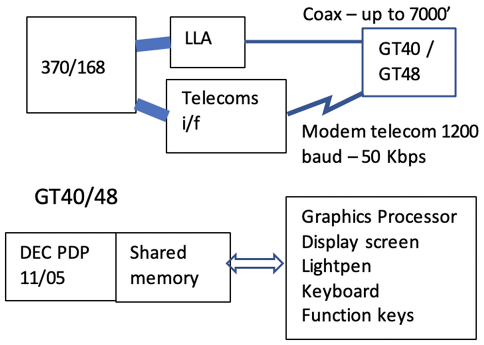

Figure 3. IBM and GT40/48 configurations. LLA: Long Line Adaptor – interface to drive long (up to 8000') coax cables.

Shoehorning the code into the 11/05 for display file management, graphics display, interaction, and mainframe communication was more challenging than anticipated.

Adventures

Although a 1200 bps communication rate was acceptable for a limited number of applications, especially for new users, it was clearly insufficient for experienced GM CAD users. After the GT40 was sufficiently developed, we experimented with various communications speeds to see what users would accept: 9600 bps, 19.2 kbps, and 56 kbps (synchronous). We determined 19.2 kbps was satisfactory while 9600 would be (barely) tolerated. We also determined that 1200 bps generated enough unhappiness that, should we try to deploy it, we would need to enter a witness-protection program equivalent.

Another communications adventure arose from the need to communicate at high speed with users 50 or 100 miles away. The technology to do this was still new and expensive. This necessitated meeting with Bell representatives ranging from local people to "Ma Bell" senior managers. After explaining our needs, it was fascinating to listen to Bell marketing people say "yes, of course, we can do that" while the engineers turned pale and shook their heads.

We needed much higher speed communications to support users at the Tech Center where consoles could be two miles from the mainframe. Fortunately, a local startup had just developed a "Long Line Adaptor" (LLA) and claimed they could transmit data at channel speeds at distances up to 8000' via a simple coax cable. While the LLA worked well, issues arose both with the connection to the mainframe and with laying more than a mile of coax across the 2-mile square Tech Center.

In retrospect, the problem of connecting the LLA to the mainframe is amusing—at the time it certainly wasn't. The issue was who would physically connect the LLA to the IBM Channel processor. IBM refused to connect anything "foreign" (i.e., hardware they hadn't designed); the startup didn't want the responsibility for any problem the LLA might create. It was a stand-off until one of the GMR support people from IBM "just did it." We also ran into a coax cable problem: we needed to lay cable in ditches that sometimes filled with water. We discovered that our coax connectors were not waterproof and that water inside coax does nasty things to data transmission!

Dealing with development time pressures was the final adventure. For the most part, things were going well and on schedule. But the user community wanted to get started earlier and put pressure on management, who responded by asking us: "If we give you more budget and two more people, can you cut the delivery time from 18 months to under 10 months?" Fortunately, I had a wise mentor who strongly advised me to "just say no." And I did.[$]

Development time pressure increased on the group because we had to develop all the 11/05 code from scratch with few programming tools. Nothing vendor-provided would support acceptable interactivity. This included the graphics package, the 11/05 OS, the communications software, and device handlers. The key was always achieving interactivity.

GM Assessment

The system did work: the GT48 (the GM special version of the DEC GT62) was distributed to major GM Tech Center users (Design Staff, Fisher Body, Chevrolet Engineering), to Lansing (Oldsmobile), and to Flint (Buick).

Overall, it was a balancing act between the division of labor, communications speed, and time-shared mainframes. In the end, we did well, but it was only temporary as the field shifted to raster and full-capability workstations in the early 1980s. CAD companies gradually developed the skills to catch up to GMR and GM began to buy vendor systems. Fortunately for GM and the group, graphics and geometry research continued at GMR, and we continued on to make major contributions in solid modeling, realistic image synthesis, and geometry analysis.

THE BATTELLE PROJECT: DAVE'S STORY

I took a job with Battelle Laboratories in Columbus OH with a freshly earned Master's degree in computer science from the University of Colorado. The year was 1972; the starting salary was a phenomenal $1000/month. I think I got the job because I had experience with Control Data (CDC) computers in graduate school, the same machines Battelle used. And I could spell "graphics" from experience making computer-animated films as a Johns Hopkins undergrad. Goodbye mountains; hello flatland.

In early 1972, Battelle formed a two-person computer graphics group consisting of Ed Edwards and me. The Lab was a fascinating place for a techno-junkie. I still am. As a contract research organization, Battelle conducted projects that ranged from nuclear engineering to early laser development.

Goals

Interactive 3-D graphics was not part of Battelle's 1972–1973 portfolio. Its graphics devices included a Calcomp plotter, a Stromberg-Carlson 4060 microfilm recorder, and a Computek DVST terminal (based on Tektronix DVST technology). The storage tube could be used as an interactive terminal that was able to draw a single image. Seeing the next picture required a bright green flash to erase the screen before the new image could be drawn.

Battelle's Computek and plotter were used successfully before I arrived. The Calcomp produced images on paper from batch programs. The Computek was relatively cheap (~$10 000 for the terminal alone) and could run interactively. One Ph.D. statistician used a Computek application to compile the annual national salary survey for

[$]This event predated the 1975 publication of Fred Brooks' *The Mythical Man-Month*.

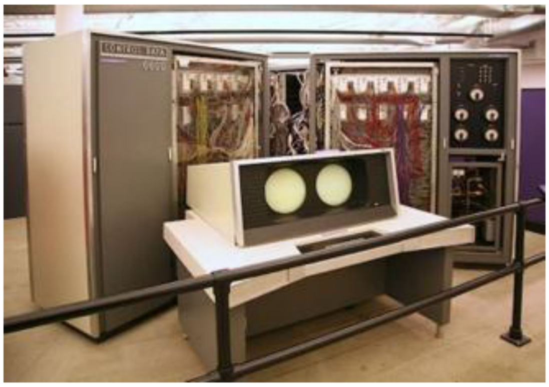

Figure 4. CDC 6600 Mainframe showing typical "spaghetti wiring" (Used under CC by 2.0 license).[10]

science and engineering. He often didn't like the way a curve looked and tweaked the input data until he was happy. A theoretically rigorous and objective process was actually highly subjective.

Battelle got into the refresh, random scan graphics business in 1974. Control Data's successor to its first interactive graphics device (the 274) was the 777-1 Cyber Graphics Terminal. It featured an upgraded minicomputer controller and a dedicated display processor. Out of the box, both the 274 and 777-1 were 2-D only.

Battelle users were pushing to interactively display and edit 3-D data. The biggest push came from mechanical engineers. Finite element modeling software was in its early days and the engineers were having difficulty visualizing and tweaking 3-D input FEM meshes.

CDC was also interested in getting into the 3-D graphics business and gave Battelle a contract to develop a 3-D extension for its 2-D graphics application program interface.

The 777 and its minicomputer controller, a CDC SC17, and software cost \$130 000+. In 2020 dollars, that's \$640 000+. Further complicating the cost model was Battelle's policy of charging real dollars for every second of mainframe processor time consumed.

CDC focused its mainframe CPUs (see Figure 4) on high-performance computation. Figure 5 shows the system architecture. Battelle's mainframe had 64K 60-bit words. A set of peripheral processors (PP, similar to IBM channel processors) did input/output processing. The PPs enabled access to user data on disk or tape. Network-attached devices like the SC17 minicomputer, attached at 56 kb, used the PPs to transmit and receive data.

Our SC17 had 32K 16-bit words. That memory was used to store both assembly language (there were no compilers) and display instructions. The SC17 CPU executed the assembly language that managed the display list and user input interrupts. The display processor had direct memory access for display instructions. It was responsible for moving the electron beam and generating interrupts from user input devices.

Approach

Working in 3-D was new for our two-person graphics group. The recently published Newman & Sproull *Principles of Interactive Computer Graphics* became indispensable, especially in showing details about transformation matrices and perspective computation.

Our job was to design and build a 3-D extension to the 777-1 2-D graphics package mainframe API. FORTRAN was the only language the API supported. The graphics package itself was also written in FORTRAN.

CDC's 2-D API supported a multilevel, segmented display list. The application could store many or few display instructions in each segment until SC17 memory ran out. The display instructions drew 2-D vectors and had multiple types of branch instructions. Unconditional branches let parts of the display list be skipped; push/pop supported hierarchical display lists. Our 3-D API extension had to be consistent with

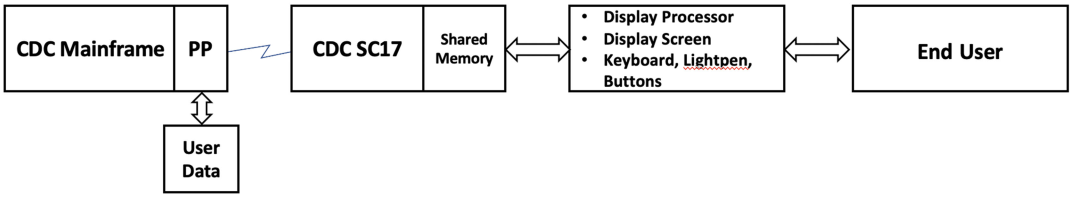

Figure 5. CDC System Architecture.

the legacy 2-D API and retain its ability to create, modify, and delete graphics segments.

We had to create the illusion of 3-D using a vector, wireframe display. There were two basic techniques: transforming object(s) under user control and visual tricks like perspective and depth cueing. We ultimately combined the two and enabled dynamic transformation of data displayed using visual tricks.

Depth cueing proved more effective than perspective. The 777 line draw instruction included *x*,*y*,*intensity*. Since we knew the *z* coordinate and display volume, we could compute *intensity* to make points nearer the user brighter. The scheme was fast and easy to compute. While observing users looking at and working with their 3-D data, Ed and I noted that most had depth cueing enabled.

As at GM, the biggest architectural decision concerned the division of labor between the mainframe and the minicomputer: where should the operation of multiplying each *x*, *y*, *z* by the current transformation matrix and projecting it to the correct to *x*, *y*, *intensity* be performed?

When the 777 arrived at Battelle, we decided to try the mainframe transform/project technique. It was simpler to program but proved problematic. Our first experiments resulted in a number of highly annoyed users because CDC had an OS bug that crashed the mainframe when my program performed dynamic transformations. Doing all transformations on the mainframe cost our users real money. Finally, there were numerous users sharing the computer; it was impossible to guarantee consistent interactive performance on a time-shared computer.

Therefore, we quickly chose to do all 3-D->2-D transform/project operations on the SC17. Our 3-D API packaged the 3-D coordinates and transformation matrices into dedicated 3-D segments. Unfortunately, the 3-D segments consumed memory for both the 3-D and 2-D coordinates. The 32K memory was a challenge.

Doing dynamic rotation required interactive control to change parameters (e.g., rotation axis, scale, and translate). We wrote a minicomputer-based lightpen input handler that let us change the parameters without mainframe intervention.

The final architectural decision was how to divide the work between Ed and me. Because Ed had other responsibilities, he did all the mainframe development. I had to learn CDC SC17 assembly programming, the mechanics of CDC's SC17 code, how to perform 3-D->2-D transformations, how to dynamically build each 2-D display instruction, and how to debug both assembly language and display instructions.

Adventures

Battelle received CDC's first production 777. The SC17 was a royal pain when trying to load new software. Software was read from a deck of punch cards through a short, manually entered boot program. I learned new swear words when one of my coding errors clobbered the boot program and I had to manually re-enter it. An "Enter" button hardware bug often stored the same instruction in multiple locations. It often took 30+ minutes to re-enter the boot program just to find out I had another bug in my code once I could run it.

Minicomputer and display instruction debugging became an art. The minicomputer allowed single step walks through the assembly code. Specific registers could be examined using the console lights. The display processor had toggle switches to create a breakpoint and single stepping from there. The main problem was that the picture disappeared immediately because display loop stopped. A blank screen contained no useful diagnostic information.

The CDC mainframe had 60-bit words with floating point. 60-bit arithmetic proved adequate for accurate scientific/engineering computation. The minicomputer had 16-bit words and no floating point. I learned the hard way that continuous rotation requires lots of multiplication. Numeric significance deteriorated so quickly that 3-D images became unrecognizable when rotated for more than a few seconds. Tom, a colleague, solved the problem by looking up sines and cosines in a table each time the transformation changed.

60-bit vs. 16-bit also caused problems with alphanumeric characters. Each 60-bit mainframe word contained 10 characters; each character had 6-bits. There were no lowercase characters and few special characters. The 16-bit minicomputer had 2 8-bit ASCII characters per word.

More assembly language software had to be written to display text correctly.

Other Battelle staff members visited me in the graphics lab just to see what I was doing. One of the systems programmers summarized his jealousy: “Gee, Dave. You’re lucky. You actually get to see pictures on a screen.” I am still hooked on seeing computer-generated pictures.

Battelle Assessment and Lessons Learned

Writing software for any new device requires grit and patience. The CDC 3-D project was no different. Those frustrations were more than offset by the fun of learning interactive 3-D graphics and how to make the system work.

The new 3-D software API and minicomputer code became part of Control Data’s Cyber Graphics 777 product offering. The code remained in production as long as CDC sold the product. The Air Force (Wright-Patterson) awarded Battelle two different contracts to develop 3-D applications for the 777. Battelle’s graphics group of two grew to six over the next 3 years to keep up with increased internal demand. When I decided to leave Battelle in 1977, I got a job offer because one of my graduate school professors saw me in a film demonstrating my 3D application work. The professor was on sabbatical at Boeing and succeeded in offering me a job. I accepted and began a 35-year Boeing career.

There were too many technology lessons learned to document many here. Most important was finding a way to prevent flicker caused by frame rate limitations. Careful decisions, e.g., doing 3-D->2-D projections on the minicomputer, minimized timeshared mainframe use. Frames could be computed at acceptable frame rates on the single-user minicomputer. I made a poor decision when I decided to forgo double buffering and circular buffering. The result was flicker during continuous rotation. Double buffering required keeping a second copy of the 2-D instructions, which would have let even fewer 3-D coordinates be stored in the limited minicomputer memory. I ran out of development time before implementing a circular buffer. The user community was still reasonably happy because of the steady rate.

The follow-on Air Force contracts had their own lessons—in business rather than technology. I almost lost the first when I decided to propose the same work but organize it differently from the request-for-proposal (RFP). The RFP author listed ~25 tasks. I believed they could be more logically organized and I wrote the proposal my way. The RFP author was totally confused by the new organization and came close to disqualifying the proposal for being nonresponsive. Cooler heads and hands-on with 3-D led to a Battelle win. During contract negotiations, the Air Force had to cut the cost to meet the budget. I couldn’t figure out how to cut enough until the RFP author went over every estimate. He got me to agree to shaving 4 hours here, 8 hours there until the cost came in line with budget. Amazing how saving enough nickels can lead to saving real dollars. Finally, the first demos to the Air Force RFP author occurred at least 18 months prior. I learned that it takes a long time to cultivate customers and get new business. The same holds true today.

SIMILARITIES AND DIFFERENCES

During the early and mid-1970s, John and Dave worked about 200 miles away from one another but in highly dissimilar environments. GMR then had 1500 employees while parent GM had hundreds of thousands of employees who designed and built cars, trucks, and buses. Battelle is still a leading contract research lab with hundreds of employees who generally write research proposals and reports.

Both organizations were starting to rely on computers heavily. Both user communities designed and engineered complex systems, were 3-D-savvy, and wanted to use interactive computer graphics to understand their 3-D data.

Battelle’s physical environment was simpler. There was a single 777 terminal that had to compute 3-D frames fast enough to satisfy a single user. The minicomputer attached to the graphics terminal was within 100 feet of a single mainframe and was connected by a high-speed, dedicated line. The mainframe itself had to service dozens of time-share users and batch jobs. By contrast, GM had many more users for about 100 DEC GT48s. The users were often remote and connected to the mainframe via slow telecommunications lines. GM’s applications ran on multiple, top-of-the-line mainframes that incurred no usage fees.

Ultimately, both of us loved seeing our work appear as 3-D images. Watching people interact with pictures inspired each to make computer graphics into 50+ year careers.

CONCLUSION

When we started this article, we believed that the primary takeaway lay in convincing readers that system architecture decisions are as important today as they were in the early 1970s. We each had to balance hardware, software, communications, and cost to satisfy interactive performance requirements that let people perceive 3-D on a 2-D screen. The detailed decisions we made were different but resulted in successful systems. John's work led to production use for hundreds in GM. Dave's work led to production use for tens in Battelle and the Air Force and a commercial product. All our success is attributable to making the right architectural decisions.

As we merged our writing, we realized that there is a second, equally important aspect to architectural decisions: scale. 1970s hardware and software could address problems measured in units measured as kilo's- and mega's-. Today's hardware can handle problems measured in giga's- and tera's-. There are now millions of 3-D users. Graphics devices cost hundreds, not hundreds of thousands. Communication that was efficient for tens of miles is now globally efficient. Today's scale forces system architecture decisions that may be different from ours yet no less critical.

Balancing scale and human performance with computing limits and capabilities continues to make interactive graphics challenging. . .and fun.

ACKNOWLEDGMENTS

The authors would like to thank their superb leaders and co-workers. At GM, George Dodd, Fred Krull, Bill Gordon, and Jim Thomas were a constant source of inspiration. Battelle's Bill Evans and Ed Edwards provided guidance. The authors would also like to thank Editor Mary Whitton for her keen eyes, patience, and persistence.

REFERENCES

1. E. Jacks, "A laboratory for the study of graphical man-machine communication," *Proc. FJCC*, vol. 26, pp. 343–350, 1964.
2. I. Sutherland, *Sketchpad: A Man-Machine Graphical Communication System*. New York, NY, USA: Garland Publishers, 1980.
3. F. Krull, "The origin of computer graphics within General Motors," *IEEE Ann. History Comput.*, vol. 16, no. 3, pp. 40–56, Sep./Nov. 1994.
4. Mar. 2020. [Online]. Available: //en.wikipedia.org/wiki/IBM_2250
5. J. Nielsen, "Response times: The three important limits," Jan. 1993. [Online]. Available: //www.nngroup.com/articles/response-times-3-important-limits/
6. G. Erlikhman., S. Sutentag, C. Blair, and G. Caplovitz, "Interactions of flicker and motion," *Vis. Res.*, vol. 155, pp. 24–34, Feb. 2019.
7. R. Silva. "Video frame rate vs. screen refresh rate," 2020, [Online]. Available: //www.lifewire.com/video-frame-vs-screen-refresh-rate-1847855
8. A. van Dam, "Intelligent satellites for interactive graphics," *Proc. IEEE*, vol. 62, no. 4, pp. 483–492, Apr. 1974.
9. J. Dill and J. Thomas, "On the organization of a low cost remote intelligent graphics terminal," *Comput. Graph.*, vol. 9, no. 3, pp. 1–8, 1975.
10. Jul. 2020. [Online]. Available: //en.wikipedia.org/wiki/CDC_6600#/media/File:CDC_6600.jc.jpg, J. Couperous, licensed //creativecommons.org/licenses/by/2.0/deed.en

David J. Kasik is a Boeing Senior Technical Fellow Emeritus. ACM named him a Fellow in 2013 and a Distinguished Speaker in 2018. ACM SIGGRAPH, presented him the Outstanding Service Award in 2012. He is a member of the IEEE CG&A Advisory Board. He holds 10 patents, has published over 35 papers, and coauthored 2 books. He received the Master's degree in computer science from Colorado in 1972 and the Bachelor's degree in quantitative studies from Johns Hopkins, Baltimore, MD, USA, in 1970.

John C. Dill is a Professor Emeritus with the Schools of Engineering Science and Interactive Arts and Technology, Simon Fraser University, Burnaby, BC, Canada. His prior positions include Cornell University and the General Motors Research Laboratories. He has served on IEEE InfoVis and VAST Steering Committees, co-chaired VAST 2007, and was EIC of IEEE's CG&A Editorial Board. He received the IEEE VGTC 2016 Visualization Career Award. He is a Life Member of IEEE and a Member of IEEE CS. He received the Ph.D. degree from Caltech, in 1969.

Contact department editors David Kasik at dave.kasik@gmail.com, Chris Johnson at crj@sci.utah.edu, and Mary Whitton at mcwhitton@gmail.com.