The Origin of Computer Graphics within General Motors

FRED N. KRULL

This article traces the history of the development of computer graphics technology at the General Motors Research Laboratories during the period from 1958 to 1967. A concept demonstration was formulated in the late 1950s to show the feasibility of applying computer technology to the problem of vehicle body design. The narration then traces the history of a joint project between GM and IBM for development of new and unique computer graphics hardware. The salient features of the Design Augmented by Computer (DAC-1) system are summarized in terms of nine separate technologies that were brought together for the first time to form a complete computer-based design environment.

It is always instructive to rediscover an idea that still has some merit today and is relevant to current technology and problems. In writing this article, I have, across the distance of time, gained a new perspective of some things that were done right and some that were done wrong during the course of a rather large and involved technical project.

As early as 1952, the General Motors Research Laboratories (GMR) were using a card-programmed digital computer for engineering and scientific analyses. However, notably absent from the applications were problems related to graphical design. To gain an insight into the automotive design process, research personnel began discussions with General Motors (GM) engineers and designers. It soon became obvious to the researchers that drawings, pictures, and models were the principal media for communication and documentation of design ideas.

As a result of these discussions, four distinct types of man-machine communication were identified:

1. *Existing engineering drawings.* The research project was realistic enough to realize that computers could not replace *all* the drawings used in the design process. Such a claim would have been akin to the office automation claims of the paperless office, which still shows no prospect of being achieved. Therefore, it was concluded that a computer system must provide means for reading existing engineering drawings and for creating these drawings. Fortunately, because of the nature of automotive design, body drawings are primarily drawn to scale on a background of grid lines with no dimensions. Digitizing this line information seemed to be a feasible task.
2. *Interactive manipulation of graphic information.* The design process frequently involved one person indicating a problem on a drawing to another person and then their joint exploration of potential changes to the drawing. It had to be possible to "point to" or indicate the location of the problem, and to make an immediate change so that the implications of the change could be evaluated.
3. *Comparison.* Often there was a need to compare last year's design with the latest design efforts. In most cases, what was desired was the ability to overlay or superimpose one drawing on another so that the differences between two graphical images could be seen. Obviously, if both designs were represented by mathematical models, they could be displayed in different colors on a screen. However, color displays were not yet commercially available. In addition, it was assumed that much of the archival design information would be available only in the form of drawings.
4. *Nongraphical information.* At that time, the commonly used method for communicating numbers and text to computers was via punched cards, which were read in batch mode. Interactive terminals that would provide immediate access to computer processing power were still years in the future. It was deemed that interactive or "immediate" input of moderate amounts of alphanumeric information would be an important part of an effective design environment.

As a result, GMR began a study of the potential role of digital computers in the graphical aspects of design. The question that was being asked was, "How could computational techniques significantly impact the design process?" A series of feasibility experiments led to the decision to establish a more comprehensive laboratory for studies in man-machine communication. The facilities were to permit the computational power of a large-scale digital computer to be brought to bear on the problems of graphical design, while recognizing the essential part played by the human in the design process.

The initial goal of the project was the establishment of a laboratory that would permit "conversational" communication between individual and computer and provide a plat-

1058-6180/94/$4.00 © 1994 IEEE

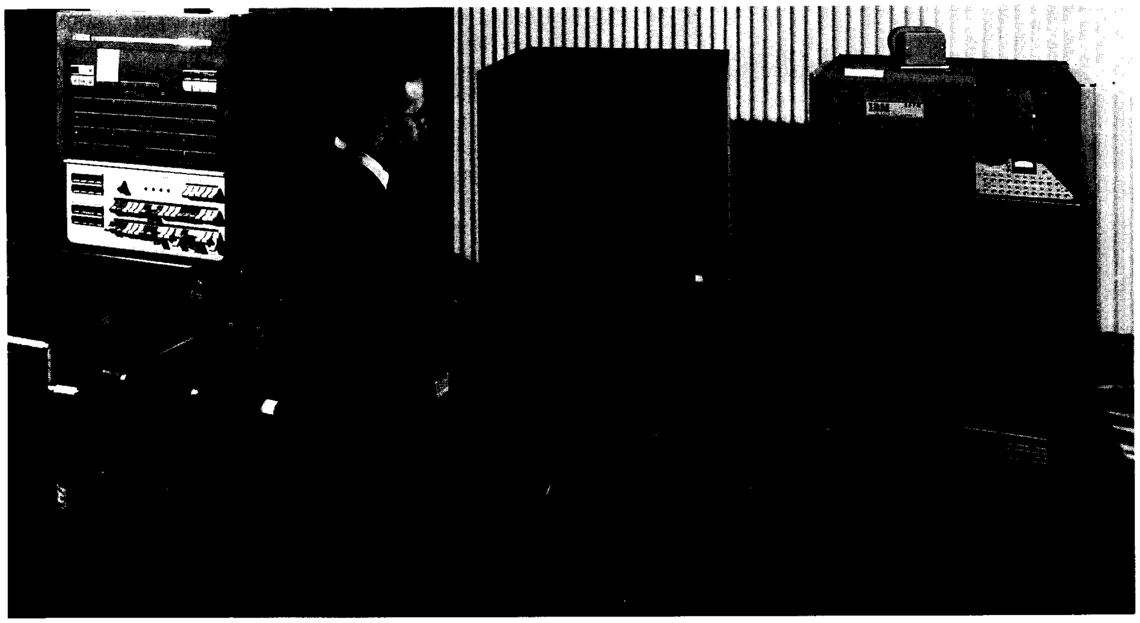


Figure 1. IBM 704 computer system with display unit and film recorder.

form for experimentation in the design process. This goal was achieved in 1963. This article traces the events that led to the installation of the first computer graphics system within GM and discusses the implications of this new technology in terms of both hardware and software.

Introduction

In 1955, the computer scientists at the General Motors Research Laboratories became recognized as a separate group within the Special Problems Department. Donald Hart was appointed assistant department head with George Ryckman and Edwin Jacks as supervisors. The group was named Data Processing since computer science was not yet recognized as a separate discipline by most universities. The group was later made into a separate department in 1961.

Under the leadership of Donald Hart, two major departmental activities were undertaken. George Ryckman's group was responsible for operation of a computing center that, at that time, included an IBM 704 computer, keypunch stations, and off-line card-to-tape and tape-to-printer equipment, together with the system programs to support their operation. The remainder of the department, under the supervision of Edwin Jacks, consisted of a programming staff devoted to the development of software applications. Both systems and applications were very fertile areas for work at that time. The systems group was just beginning to gain recognition in the IBM user community as a leader in the development of computer batch operating systems. Early experiments among GMR, North American Aviation, and IBM led to the development of a batch monitor* program for an IBM digital computer.[1] Most of the programming was still being performed in IBM assembly language, but by 1958 a new Fortran compiler from IBM was being tested at selected customer sites, including GMR.

During this same period IBM marketed a film recorder for the IBM 704 computer that could be used to record "point plots" on 8-mm film. This facility provided engineers with their first opportunity to view computer-generated graphs and computer-animated movies. Computer-generated traffic simulations were recorded on film using this equipment.[2] For demonstration purposes, IBM also provided a display unit that operated as a slave to the film recorder so that the plotting could be seen by the machine operator. The film recorder and display unit (Figure 1) became the basis for the initial GMR experiments in interactive computer graphics.

June 1958 saw the start of discussions with various GM divisions, such as Styling, Fisher Body, and Chevrolet, to gain an appreciation for the many problems of vehicle design and engineering. From these discussions, it became apparent that the time-consuming problems were in the areas of drafting and the translation of drawings into models, templates, production tools, and fixtures. It was felt that, if a computer could read sketches and drawings, then it could be programmed to produce further drawings, engineering data, and control tapes for numerically controlled machine tools.

*At the time of the events described in this article, the term "monitor" was used for what would now be termed an "operating system."

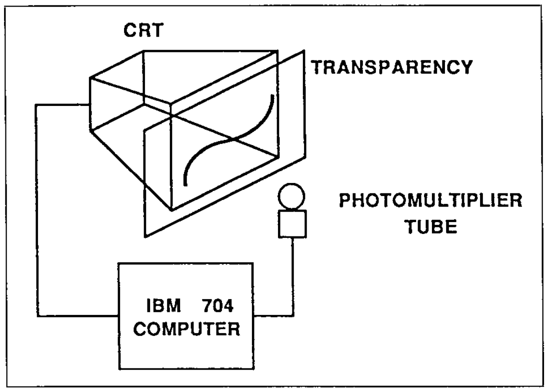


Figure 2. Functional process of digitizing a drawing.

By 1960, the process for programming machine tools was already under development at MIT, resulting in the Automatic Programmed Tool part programming language.[3] Numerically controlled machines were just beginning to be installed within GM. The Manufacturing Development group within GM was already at work on numerical control machining software. The MIT and the Manufacturing Development efforts were both based on the premise that drawings were the finished documentation of a design. Languages were defined for preparing numerical control part programs. In contrast, GMR personnel decided to concentrate their efforts on the product design aspect of the process that would result in the creation of drawings. It was believed that this approach would open the door to the creation of numerical control information without the need to write extensive numerical control part programs.

In November 1956, George R. Price, a research associate at the University of Minnesota, had published an article about engineering design titled "How To Speed Up Invention."[4] On the basis of some of Price's thoughts about design and the rapidly evolving technologies of digital computers and numerical control machine tools, there developed the concept of a computer system for vehicle design. The problem to be addressed was the design of a vehicle's exterior surface geometry together with all the interior components that make up the chassis and body. These objectives did not include the design of engines, transmissions, or other mechanical subassemblies. In 1959, Donald Hart encouraged the applications group to begin the implementation of such a system, and a five-person project team was formed. The project was named "Digital Design," and it went full steam ahead to develop a feasibility demonstration. The name of the project was later changed to DAC-1, Design Augmented by Computers, since GMR management did not want to give the impression that digits were being designed!

What resources were needed to carry out a feasibility demonstration? Members of the department had already programmed the IBM 704 computer to produce a point plot on the IBM 780 display unit. To maintain an image on the display only required putting the program into a loop. Interactive control of the computer was solved in a unique fashion. The normal control of the system was via sense lights and sense switches on the faceplate of the central processing unit. Programs were started by keying in bootstrap commands through these sense switches. Clearly, this was inadequate for any type of interactive design activity. Attached to the central processing unit was a line printer that was normally used for operator messages. An arrangement of five 10-position dials was wired into the echo checking logic of the on-line printer. A monitor program was then written that would poll the printer and read the positions of the five dials. These five digits were then used to control program execution, thus providing the project with an interactive input device.

With this arrangement, it was now possible to control the software and to direct program execution of the IBM 704 computer interactively. The display unit and numerical control machines provided good methods for the creation of graphic output and physical models. The same boundary curve and section line drawings that were viewed on the display screen could also be redirected to a GMR-designed drawing table that was mounted on a three-axis milling machine located at Bendix Controls, Inc.[5] A Bendix-supplied postprocessor was used to create machine tool commands. Mylar machine tool control tapes were generated by creating a tape image on IBM cards, followed by converting the cards to punched mylar with an off-line IBM 047 card-to-tape machine.

The spindle of the milling machine was simply used to hold a ballpoint pen. This idea had already been used some time earlier at Boeing Aircraft to produce full-size drawings or "lofts" of airfoils. The only added flourish was to divide the full-size drawing into 32 × 32-inch segments that could be drawn one segment at a time. The drawing table was then equipped with an indexing head and set of rollers that increased the effective drawing area to 96 inches by the length of the roll. After a few false attempts to coordinate the table indexing with the drawing commands, a program was written to generate a printed setup procedure that told the machine tool operator what table setup operations to perform and when. A computer-generated process plan became standard operating procedure for all numerical control drawings.

There still remained the problem of providing graphical input to a computer and, with the help of the GMR Instrumentation Department, this aspect of the problem was addressed at the same time. A digital circuit was designed to use the signal from a photomultiplier tube mounted in a hood covering the display unit to detect the presence or absence of light and set one of the sense lights on the IBM 704 computer accordingly. Drawings and sketches were traced onto a clear plastic overlay mounted on the face of the IBM 780 display unit. The lines on the overlay were then digitized by finding those points where the light, from a spot of light plotted on the display unit, was occluded by the line on the overlay (see Figure 2).

Since a drawing consisted mainly of "white space" and the actual lines themselves covered only a minute percentage of the total area, digitizing was not done by a raster scan. This would have been much too slow using existing scan rates. Instead, program logic was developed to "lock onto" a line and digitize the path to each terminal point in much the same manner as an ant might crawl along a wire. This facility provided the last ingredient from which a feasibility demonstration was created.

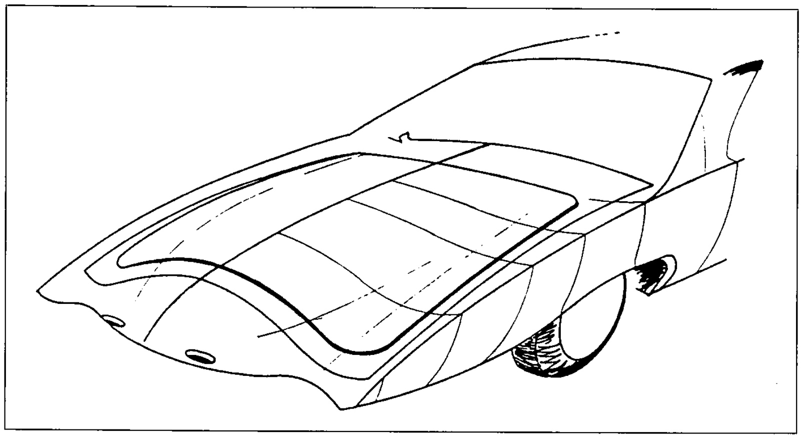


Figure 3. Designer sketch of a styling concept.

The Digital Design feasibility demonstration

The Digital Design feasibility demonstration had to show how computers could be applied to the vehicle design process. Also, in August 1959, preliminary discussions were started with IBM for the acquisition of custom-designed graphics hardware and an IBM 7090 computer. Therefore, the demonstration had to do more than demonstrate feasibility. It had to justify the investment in a multimillion-dollar joint project with IBM for the development of a laboratory for the study of graphical man-machine communication.[6]

The feasibility demonstration was designed to show how a mathematical model of a vehicle might be created starting from a series of designer sketches. After the three-dimensional mathematical model is created, it must be capable of being modified quickly and then the resulting mathematical model used to create physical models and full-size drawings. All the mathematical algorithms were to be programmed for the IBM 704 computer using the new Fortran language and compiler. System routines or I/O drivers were coded in assembly language.

During discussions with Styling, Fisher Body, and other divisional personnel, it became apparent that they were very concerned about the smoothness of the initial line drawings used to show a new styling concept. Line drawings must represent aesthetically pleasing curves from which all irregularities, flat spots, or hollows have been removed. The conventional process was to use the skills of a trained clay modeler to translate from a rough design sketch to a full-size clay model of the vehicle. It was felt that a design machine could address some of these same problems if it were introduced early enough in the design process. Therefore, it was decided that early styling concept sketches would be the starting point. Stylists were requested to sketch the concept in perspective on paper (Figure 3), and the computer would then be used to construct a three-dimensional mathematical model that not only reflected the design intent but also represented the object to the necessary degree of detail, smoothness, and accuracy.

A transparent overlay was used to trace the location of four boundary curves of the hood surface. The two-dimensional shape of these boundary curves, along with the orientation and position of the model in Cartesian coordinate space, provided enough information from which to compute the spatial location of the hood boundary curves.

The transparent overlay containing the four boundary curves was mounted on the display unit of the IBM 704 computer, and these lines were digitized by plotting points on the screen and testing to see if the light triggered a sense light using the light-sensitive photomultiplier. Points were plotted from left to right and from top to bottom to locate each boundary curve. The center of each line was estimated from the two locations where the point plots first detected an edge and subsequently reappeared on the opposite edge. Successive locations along each boundary curve were then digitized using a mathematical compass to plot points of light according to the desired sampling interval (S), as shown in Figure 4. It was possible to plot at a rate of 6,000 points per second, where the delay and rise times of the photomultiplier were the limiting factors.

Smoothing the digitized data and estimating the direction of the boundary curve were done dynamically by fitting the data with cubic polynomials. A simple least-squares fit was used to compute the cubic coefficients, and the polynomial was used to extrapolate for an estimated location of the next

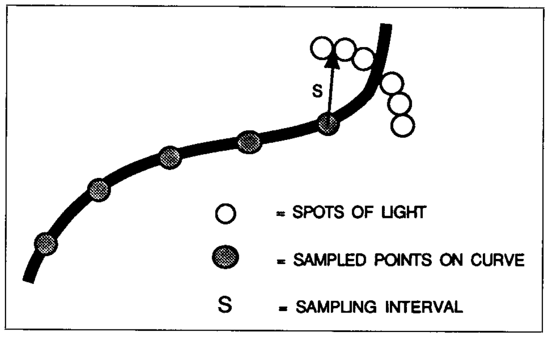


Figure 4. Line digitizing procedure.

sample point. If subsequent sampled points were within a specified tolerance of the estimate, then the sampling continued. If sampled data were outside this tolerance, the slope and ordinate of the polynomial at the last good point were used as initial conditions for constructing a new estimating function. Thus, the digitizing automatically produced for each boundary curve a set of cubic polynomials whose first derivative was continuous — that is, they were C^1 continuous.

The interior surface of the hood was represented by an interpolation formula, as shown in the sidebar on the facing page. It would appear that this method of interpolation anticipated, by several years or more, the basic (nonparametric) Coons surface patch.[7-9] Limited to just the blending of four boundary curves without cross-boundary derivative input, the representation lacked the flexibility of higher order Coons-Gordon patches to interpolate a network of curves that had C^1 continuity. Joining of adjacent patches with C^1 continuity was possible, and the developers did not overlook this fact. In the simplest case, surface blending formulas were expressed as $\phi_x = (1 - x^2)^k$, with the value of k left at an arbitrary value. The only method for the stylist to change the surface interpolation, other than modifying the boundary curves, was to change the value of k.

The final decision as to what was an acceptable interpolation was left up to the original designer. The design system provided a variety of tools by which the designer might evaluate the quality of the surface interpolation. The shortest and most immediate means for evaluation was to display the surface and original boundary lines on the IBM 704 display unit. Either perspective or orthographic projections of the boundary lines and section cuts through the surface were available upon demand. However, it was extremely difficult to make decisions relative to three-dimensional surface quality from a presentation on a 17-inch display screen. This statement is still true today.

To demonstrate that the computer provided a feasible solution to the design problem, it was necessary to show how numerically controlled machine tool technology could be used to make accurate drawings or to machine models. The GMR drawing table, located at Bendix Controls, was used to produce full-size body drafts — 6×20 feet. Subsequently, the same three-axis milling machine was also used to produce three-dimensional models of the styling surfaces, although for purposes of the demonstration, all that was ever machined were templates representing section cuts through the surface.

Finally, the feasibility demonstration had to address some aspect of surface geometry that was more engineering oriented. A body designer was asked to sketch the plan view of a desired hood opening line. This curve was digitized in the same manner as the surface boundary curves and the opening line projected onto the interpolated surface. The body designer was then asked to draw the cross section of the flange that must be attached to the curve formed by projecting the hood opening line onto the hood surface. Flange surfaces that met engineering criteria were created automatically from this data.

The feasibility demonstration along with a proposal to enter into a joint development project with IBM was finally shown to the GMR policy group in July 1960. In attendance were Lawrence Hafstad, vice president of research; John Gordon, president of GM; and Fred Donner, the board chairman. The presentation was well received. At the conclusion of the demonstration, all of the presenters except for the assistant department head (Donald Hart) were asked to leave the room for a few minutes. Since a quorum was present, a meeting of the policy group was convened. Within two minutes the proposal to initiate a joint development contract with IBM was approved in total. Needless to say, all members of the team were elated, but a lot of hard work lay ahead.

Without any real planning, a number of other spin-off R&D activities resulted from the concept demonstration. Significant was the resulting work in line and surface approximation. During this same period, the GMR Mathematics Department, under Henry Garabedian, became interested in the more mathematical aspects of the vehicle design problem. Some of the earliest work in spline-based curve and surface approximation resulted from these efforts.[10,11] Later, the DAC-1 concepts for scanning a drawing became the basis for research in image processing.

The joint development contract with IBM

The approval by GM management of the DAC-1 project triggered a serious effort to draw up a legal agreement between GMR and IBM. In July 1960, IBM presented a formal proposal to GMR for IBM to design and build a Graphic Expression Machine (Project GEM). IBM proposed to design and construct five hardware components:

1. a special data channel,
2. a display adapter,
3. a display unit,
4. a photo-recorder-projector, and
5. a photo scanner.

The characteristics of these hardware components are described in detail in a joint GMR/IBM paper titled "Image Processing Hardware for Graphical Man-Machine Communication."[12]

The agreement was based on GMR taking responsibility for all software development and IBM to be the supplier of computer hardware and associated special graphics equipment. A common assumption was that GMR would install an IBM 7090 processing system to provide the basic computing platform for the joint project, and support high-end scientific batch processing for GM users. The IBM 7090 was a normally configured commercial machine except for the addition of two special data channels: one for the yet to be announced 1301 disk file, and the second to serve as a communication path between the host computer and the GEM special graphics devices. In addition to the special data channels, IBM provided a memory protect mode instruction, which prevented programs in high (low) memory, by error, from referencing low (high) memory. The memory protection instruction was the same or very similar to the instruction provided to MIT at the same time to support development of the Compatible Time-Sharing System (CTSS).[13]

It was assumed that the principal design inputs would come from two sources: freehand sketches and drawings or records of previous designs. There was a widespread mistaken impression that the medium for previous design data would be drawings rather than archival copies of computer models. While drawings continue to be a major medium of communication, few, if any, computer systems use archived drawings as their input medium.

Interpolation formula

The surface interpolation is expressed in terms of the function $y = F(x, z)$ over the domain $R = [x_0, x_1] \otimes [z_0, z_1]$ defined by the containing box. A definition of the function $F(x, z)$ is as follows:

$$\tilde{F}(x, z) = \sum_{i=0}^{1} F(x_i, z)\phi_i(x) + \sum_{j=0}^{1} F(x, z_j)\varphi_j(z) - \sum_{i=0}^{1}\sum_{j=0}^{1} F(x_i, z_j)\phi_i(x)\varphi_j(z)$$

where ϕ_0, ϕ_1, φ_0, and φ_1 were chosen as arbitrary blending functions but with the constraints:

$$\phi_0(x_0) = \phi_1(x_1) = \varphi_0(z_0) = \varphi_1(z_1) = 1$$

$$\phi_0(x_1) = \phi_1(x_0) = \varphi_0(z_1) = \varphi_1(z_0) = 0$$

Thus, $\tilde{F}(x, z)$ was certain to coincide with $F(x, z)$ along the boundaries of the domain R. If the constraints $\phi_0'(x_0) = \phi_1'(x_0) = 0$ are imposed, then the derivative of $\tilde{F}$ across the boundary $x = x_0$ becomes:

$$\tilde{F}^{(1,0)}(x_0, z) = F^{(1,0)}(x_0, z_0)\varphi_0(z) + F^{(1,0)}(x_0, z_1)\varphi_1(z)$$

An adjacent surface patch sharing the $x = x_0$ boundary will join with C^1 continuity if it also shares $F^{(1,0)}(x_0, z_0)$, $F^{(1,0)}(x_0, z_1)$, $\varphi_0(z)$, and $\varphi_1(z)$.

A display unit was configured with an electronic pencil to be used for "sketching," and a photo scanner/recorder was designed for "reading" existing engineering drawings. Graphics output could be displayed dynamically on a screen or recorded on 35-mm film for rapid processing and subsequent projection or printing. Delivery of the system was estimated at 18 months after the contract was signed in November 1960. Needless to say, it took a bit longer (approximately 30 months).

IBM was particularly anxious to obtain the hardware contract as it represented a commercial project for IBM personnel who were just phasing out of the SAGE early warning air defense project that featured both graphic input and output capabilities. The new IBM special data channel and the graphic console represented straightforward implementations of prior technology. However, the photo scanner/recorder unit represented much more of a technical challenge and later would present obstacles to both partners. A technical liaison activity was organized between GMR and IBM to monitor the progress of hardware development.

IBM planned a series of commercial products to evolve from the joint endeavor. There were no restrictions on IBM selling similar or identical hardware to other customers. IBM certainly knew the goals and technical objectives behind the project, but GMR systems and applications software were regarded as proprietary information. After delivery of the DAC-1 system in April 1963, IBM undertook a follow-on project named "Alpine," which resulted in three new commercial graphics products: the IBM 2250 graphics console, the IBM 2280 film recorder, and the IBM 2281 film scanner.

All three devices were intended for use with the just announced IBM System/360 series of computers. The graphics console received a warm reception from the just emerging CAD/CAM industry. The film recorder and film scanner received a lukewarm reception and were later withdrawn as supported products.

As the hardware development progressed during 1961 and 1962, two problem areas were identified. First and foremost was the time GMR needed to develop and test software. Long before the DAC-1 hardware was delivered to Warren, Michigan, an IBM 7090 had been installed at GMR. The central issue became how to test graphics software prior to delivery of the graphics hardware. IBM agreed to allow GMR to have access to an IBM 7090 in Kingston, New York, where the IBM graphics development work was being done. The arrangement called for IBM personnel to use the system first and second shifts for their purposes. GMR personnel were allocated third shift for software testing.

Many long, trying weeks were spent in Kingston testing the software. The scenario was to go to work Monday morning in Warren, Michigan, travel to Kingston, New York, and work there five consecutive nights. The weekly return to Warren was scheduled for Saturday morning. This was the early 1960s. IBM was very hesitant to allow GMR female employees to work third shift because they could not be escorted to the ladies' room. GMR female employees were both annoyed and amused by IBM's attitudes, but it did not stop them from fully participating in all activities of the test team — both working and social. A male keypunch operator was specially recruited by IBM for third-shift coverage.

As the deadline for delivery of the system to GMR grew closer, a larger and larger contingent of GMR employees traveled to Kingston each week. The cost of travel became so acute that management secured use of a GM executive

Convair aircraft to ferry out project personnel on Monday and pick up the team on Saturday. Competition for third-shift machine time was very keen. Programmers competed with one another for one or at the most two "shots" at the hardware each night. In addition, IBM customers engineers, who normally performed preventive maintenance on the IBM 7090 during third shift, now found the machine room full of GMR programmers. However, the spirit of cooperation remained high, and a lot of good work was accomplished.

The second major problem to arise was the performance of the photo scanner when used to read drawings. Partway through the project in 1961, IBM held an internal technical audit and changed project management. The new manager, Dr. Ernest Newman, met with GMR liaison personnel and presented revised specifications that he felt IBM realistically could achieve. The hardware delivery acceptance tests for optical resolution and analog linearity and stability were revised based on internal IBM tests.

There was some consternation on both sides of the table during the meeting but, in the final analysis, GMR had to accept a less ambitious performance level lest it be forced to settle for no equipment at all. Later on in actual use, it became clear that photo scanning would be a less significant aspect of the project. The major technical hardware achievement of DAC-1 turned out to be the delivery of an interactive graphic console attached to an IBM computer intended for commercial rather than military applications.

Finally, a demonstration of the system hardware and software in operation was given to IBM and GM upper management in December 1962 at the IBM laboratory in Kingston, New York. Everything that didn't move was painted either gray or blue in preparation for the arrival of the executives. The demonstrations were repeated at least four times with bleachers set up so that everyone could see the single small terminal in operation. During the next-to-last step in the final demonstration, a fatal system error message appeared on the console screen. The IBM 1301 disk had crashed. However, all acceptance tests were subsequently completed to the satisfaction of both corporations, and the graphics hardware was delivered to the GM Technical Center in Warren, Michigan, during April 1963.

The DAC-1 system

What was the DAC-1 system? Was it merely a collection of software that manipulated graphic images on a display, or was there much more to it that was not intuitively obvious to the casual observer? Certainly, the twenty plus professionals on the software development team knew that there were many subtle aspects to the system. One way to describe DAC-1 would be to examine the technical achievements within the following nine categories: operating systems, languages and compilers, databases, the graphic console, the photo scanner/recorder, mathematics, graphics, numerical control machining, and DAC-1 design procedures.

Operating systems. In 1961, an IBM 7090 computer system was installed at GMR to support the DAC-1 project and to provide batch processing for GM scientific applications. This machine was a standard configuration in almost every respect — with 32 Kbytes of memory, two channels, and an assortment of peripheral equipment. At this time, the only concession to the DAC-1 project was to add an IBM 1401 processor to channel B so that the project might gain some experience with a disk-based operating system.

What type of computer operating system would be needed in support of the DAC-1 project? Clearly, IBM standard batch systems were inadequate. There had to be immediate access to a large volume of data and programs. This implied that the system must be disk-based. Furthermore, there had to be immediate (millisecond) response to graphic interactive operations while, at the same time, a smooth flow of batch processing had to be maintained, but at a lower priority. All these requirements led the team to conclude that a multiprogramming operating system must be an essential part of the graphics system.

An operating system was designed based on the computer configuration shown in Figure 5. However, in 1961, many of the components did not yet exist. The DAC-1 graphics hardware on channel C would not be delivered to GMR for another 18 months. The IBM 1301 disk did not yet exist and was temporarily replaced by an IBM 1401 and 1405 disk attached as a tape unit. The initial IBM 7090 was equipped with only a single 32-Kbyte memory. The final DAC-1 configuration was an IBM 7094-II with two 32-Kbyte core memory boxes.

The multiprogramming system was managed by a trap control system (TCS), which resided on the batch side and acted as a traffic cop between the interactive and batch computing demands. All interrupts were fielded by TCS and control was passed to the appropriate control program: BATCH or the DAC-1 monitor, depending on the type of activity.

If an interrupt originated from the graphics channel, TCS switched memory protection to the DAC-1 side and transferred control to a graphics monitor. When the graphics monitor had no more work to perform, it relinquished control to TCS, which in turn transferred control to the IBM batch monitor. These two modes of operation became known internally as D^2 and E^2, where of course D^2 meant "digital design" and E^2 was "everything else."

The foreground D^2 task was assigned the higher priority and was allowed the privilege of interrupting the background E^2 task whenever necessary. Since the graphics monitor received immediate access to the main processor, system users became accustomed to immediate response times. Immediate response time became an important factor for user acceptance of the system. The normal system delays inherent in conventional time-sharing would later become a significant distraction for the interactive-graphics user community.

The problem of operating with a single 32-Kbyte memory box was solved by logically splitting it in half, with 16 Kbytes assigned to support the interactive graphics operations and 16 Kbytes allocated for batch processing. The memory protection feature was effectively used to isolate high memory from low memory and vice versa. While it was not always convenient to fit software into 16 Kbytes, the job was possible, except for one problem. IBM's new Fortran IV language was receiving increased use within GM as a scientific batch programming language; however, the IBM compiler was written to run in a full 32 Kbytes of memory. The predecessor Fortran II compiler ran nicely in 16 Kbytes, but it was being replaced by a richer, more powerful language. Therefore, GMR requested that IBM provide access to the Fortran compiler source code, and it was modified to run in 16 Kbytes.

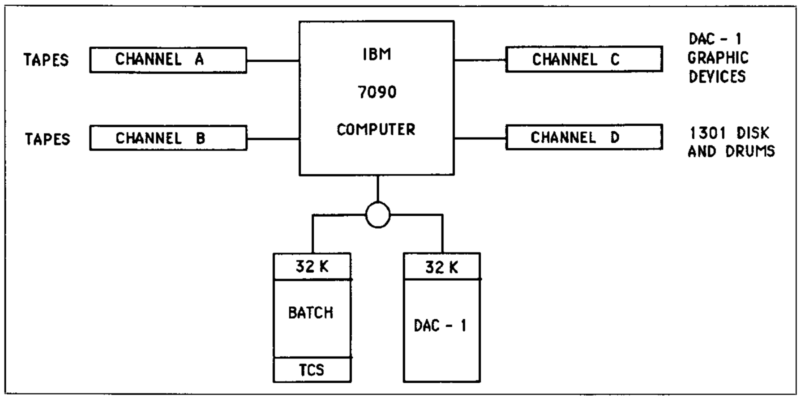


Figure 5. System configuration.

Fortunately, shortly after delivery of the graphics hardware to GMR in April 1963, the initial 32-Kbyte machine was upgraded to 64 Kbytes by the addition of a second memory box. There were very few IBM 7090 computers with two 32-Kbyte core boxes, and no two were programmed in the same way. A full 32 Kbytes of memory was provided for the interactive programs, while batch operations had their own 32 Kbytes, which permitted the execution of any industry standard application that was not time-dependent. Other batch operating systems that obeyed the rules for interrupt handling could have been supported in a like manner.

The use of two separate core boxes also provided a rudimentary form of memory protection since one 32-Kbyte memory was not addressable from programs resident in the other 32-Kbyte memory. This use of a standard IBM processor in a multiprogramming mode with priority interrupts was considered a first by many in the industry. These ideas were adopted by IBM and others in the implementation of operating systems for subsequent generations of multiprogrammed or time-shared computer systems.

The graphics monitor program was the kernel for a complete disk-based interactive operating system. A more detailed description of the disk subsystem is contained in a paper entitled "Operational Software in a Disk Oriented System."[14] The disk subsystem provided access and storage for four types of information:

1. system processors, such as memory managers or a relocatable program loader;
2. application subroutines including math, graphics, or photo scanning modules;
3. a numerical representation of graphic design data; and
4. scratch storage.

It did not take long for the project to recognize the value of maintaining all programs on line for immediate access. Most batch programs were executed from a batch stream contained on a sequential tape device. If the batch program needed intermediate programs or data, it usually was necessary to issue a message to the operator for additional tapes to be hung on scratch tape drives. When the scientific and engineering computing community learned that all graphics programs and data were maintained on line using disk storage, some jealousy and envy arose.

However, there was a rather unexpected performance benefit of multiprogramming that seemed to pacify the batch programming community. Since the multiprogramming overlapped I/O, it became less expensive to run a batch program on a multiprogrammed computer than on a dedicated batch system. It was hoped that the advantages of overlapping I/O would extend to later time-sharing systems, but the inherent nonrepeatability of system response time when nearing system saturation became a large distraction.

The next most important contribution of the operating system was in the area of memory management and dynamic storage allocation. It was recognized from the inception of the project that applications would easily exceed the size of the available physical memory of the computer. Initially, graphics applications were allocated 16 Kbytes, of which 14 Kbytes were dedicated to resident system code and disk I/O routines. Therefore, some type of memory management scheme was essential. The program overlay solutions invented for batch systems were inadequate because the sequences of module execution in an interactive environment were unpredictable.

The solution that was adopted relied on dynamic memory allocation and the use of relocatable object software that could be loaded, dynamically linked, and executed at any memory location. Calls to system routines from an application were provided to allocate or free memory and to load or unload programs at the subroutine level. A subroutine "combine" function was invented to load groups of object programs as one physical "load" or "clump." The implementation did provide a solution to the memory management problem, although it was not practical until a second 32-Kbyte memory box was added to the system, since clumps easily exceeded the size of available memory.

The software developed in support of a disk-based system was an essential aspect of the system. The IBM 1301 was a fixed-block-size device with preformatted tracks, each containing space for 465 36-bit words of data. System processors, subroutines, or data requests were then assigned cylinder-track addresses according to the size and number of blocks requested. Disk memory management was based on high-water-mark logic with garbage collection and cleanup of unused space accomplished by running disk utilities during off-shift hours as a batch program. An applications programmer requesting disk space for small elements of geometric data asked for space from the small-block-size pool (25 words per block, 15 records per track). Conversely, if a large scratch storage area was needed, the largest block size (465 words per block, 1 record per track) was requested.

Languages and compilers. The experience in implementing the feasibility demonstration showed the value of high-level languages both as an implementation tool and as an aid to documentation. When the DAC-1 operating system and applications software implementation began, it became clear that some language and compiler changes in support of graphics programming would be necessary regardless of which high-level language was selected for use.

One alternative would have been to create a preprocessor for Fortran, but this would have increased the already high compilation times of IBM's Fortran compiler. An available alternative was the work at the University of Michigan on the MAD (Michigan Algorithm Decoder) language.[15] The MAD compiler was extremely fast. MAD, an Algol-58 derivative, had firmed up in 1961 and was widely used at the entire University of Michigan as well as several other universities. An important advantage was that the authors of MAD were close at hand and able to effect the necessary language and compiler extensions.

Significant additions to the MAD language included 13 new operators, new variable types (notably the "global variable"), a new relocation scheme, and real-time statements. The new operators and data types were designed to allow systems programs or graphics programming (bit manipulation) to be expressed in a high-level language. The rapid compilation boosted the productivity of the software developers. A project was begun in which the authors of MAD made the necessary changes to the compiler and worked with GMR personnel to produce the resulting "NOMAD" compiler. During its development, the name NOMAD stood for "Non-Operational" MAD. Later, when NOMAD reached production status, the name meant "Newly Operational" MAD.

The selection of NOMAD as the name also became appropriate because the compiler was designed to produce location-independent code — that is, code that could execute at any load address or wander from place to place. Compiled object modules were stored on disk in libraries in a nonlinkage edited format and loaded into memory for execution upon demand.

NOMAD became the primary language for DAC-1 operating systems development and for supporting applications development (math, graphics, and so on). Over 90 percent of DAC-1 code was written in NOMAD, the most notable exceptions being TCS and the NOMAD compiler itself, which were written using assembly language.

A second rather unusual software development led to another systems language named Maybe, since it was not clear that it would ever work or ever be used. The IBM 7909 channel could interpret a number of special commands for error checking, retry when necessary, and off-line searching and staging of data. While the numerical commands were quite primitive, involving not much more than incrementing or decrementing a register and comparing a register to zero for a conditional branch, there were enough commands to suggest that a high-level language could be designed to support the development of channel software. Consistent use of the channel I/O capabilities by the applications programmers was essential to the integrity of the system. During 1962, the development of Maybe became a joint effort of the University of Michigan and the DAC-1 systems group. This effort may have resulted in possibly the only attempt to develop a high-level language and a compiler for a data channel.

While project management was adamant that Maybe be used to write all channel programs, there was much skepticism among the programming team. In point of fact, Maybe did work and was used to synchronize the operations between the graphical devices and the host 7090 computer. Maybe provided the linkages to keep track of the interrupts and to transfer control to the appropriate NOMAD subroutine. As the process was a back-and-forth transfer of control across the data channel, it was feasible to nest NOMAD and Maybe subroutines to an unlimited depth without the system losing track of where it was or from where it had been interrupted.

DAC-1 also provided a descriptive geometry language (DGL) as a means for expressing the procedural decision process that would go into a design sequence. In contrast to the other languages mentioned in this section, DGL was a programming language intended for the users of DAC-1 rather than the builders of the system. The language was actually developed as a test bed for the mathematical operations prior to delivery of the DAC-1 hardware.

Consider the following DGL language statements:

```
LN001 = INTERSECT(SU025, SU043);
LN006 = SMOOTH(LN003, TOLERANCE = .005);
        DISPLAY(LN001, LN006);
```

In the first statement, SU025 and SU043 are the two elementary geometric surfaces to be intersected, INTERSECT is the name of the mathematical operator to perform the intersection, and LN001 is the result. In the second statement, LN003 is a line to be smoothed to a tolerance value of .005 and LN006 is to be the result. Finally, the DISPLAY statement creates an image of the lines LN001 and LN006 on the graphic console screen.

DGL was a programming language with variables, constants, statements, branching, looping, subroutines, and parameterization in which INTERSECT, SMOOTH, and DISPLAY were just three of a large number of operational statements. It was assumed that designers would "program" and desk check these procedures before sitting down at the DAC-1 terminal.

DGL procedures were then entered into the system via decks of punched cards. Execution of a parameterized procedure then meant supplying the parameter values from the keyboard, photo scanner, or electronic pencil.

Databases. The disk database associated with DAC-1 consisted of programs, descriptive text, and geometric quantities such as points, lines, and surfaces that represented components of a vehicle design. To the extent that DAC-1 was a computer-based geometric modeling system, the geometric elements of the database represented the model. Many nongeometric elements also became part of this database, such as subassemblies: collections of other elements, or drawings, which were a definition of the data needed to create a physical layout (scale, orientation, content, notes, dimensions, and so on). Each data element was named by a two-character prefix that identified the type of data (LN for lines), followed by a three-digit item number, followed by a file number (LN015-28). The data name, when passed through a catalog of assigned disk space, was then mapped into cylinder, track, and record address. While this type of file organization was not naturally hierarchical in structure, it became popular to save the names of related data within a data element, thus providing the beginnings of a default hierarchical file organization.

Of course, the usual functions were provided for dynamic retrieval or storing of data, but the allocation of space for a file was done as a system function at the time a file was created. Disk contents were backed up onto tape for protection against the inevitable disk failures. However, no provision was made for recording a log or an audit trail of database activity. Data security, back-out, recovery, and restart were not done initially. However, before termination of the project in 1967, the need for some of these facilities became obvious.

Graphic console. During the development of DAC-1 there was so much preoccupation with reading and writing drawings that the graphic console (or operator's console) almost became an afterthought (Figure 6). Little did the development team realize that interactive graphics consoles would become the dominating method of design and that the need to "read" drawings was of little consequence. In fact, it was probably the case that IBM supplied the console more for operator use than as a design terminal. To be sure, the graphic console did come equipped with both input and output graphics capabilities. The graphic console could display wireframe drawings or system messages. However, the complexity of the image was very limited by two factors:

1. the display buffer for the vector stroke image was held in main memory (which already was very limited), and
2. the vector drawing rate was such that the display would flicker for images with more than a few thousand vectors.

Frequently, the screen turned into a wheat field. The operator immediately knew the display buffer had been corrupted,

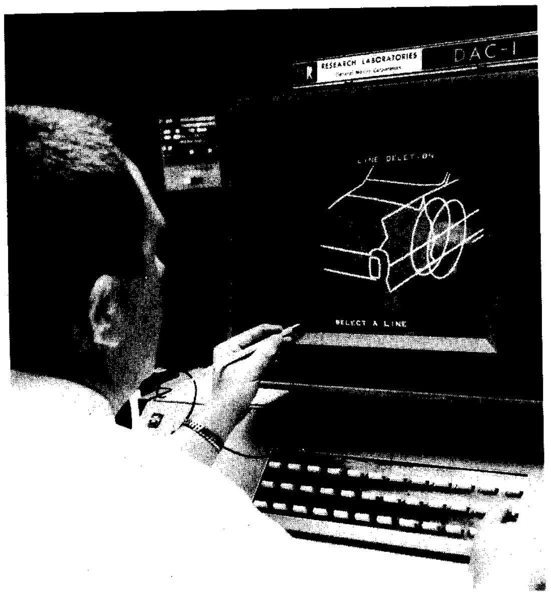


Figure 6. DAC-1 graphic console.

since the vector display of computer instructions looked like random vectors.

The graphic input medium was an electronic pencil whose position was sensed by a conductive coating applied to the screen of the display unit. However, by 1965 it became apparent that designers could not easily sketch on a vertical display screen, and the mode of sketching a design was quickly abandoned. The human factors of raising an arm to point or sketch on a vertical screen were very objectionable. There also was an abortive attempt to use the electronic pencil to draw or print characters on the screen with automatic character recognition, but this technology was also abandoned because of human factors and problems with the recognition algorithms. In 1968 the development team had an opportunity to visit Douglas Engelbart at Stanford Research Institute, and everyone was very impressed with the graphic input possibilities of a mouse.

Since graphic input at a console was not productive, how was the graphic console used? First, it served as an operator's console for controlling the image-processing functions of the photo scanner/recorder. Second, the development team quickly realized that computer-aided design was really a series of geometric constructions that could be initiated and controlled from a graphics console. The console was used to input commands via the alphanumeric keyboard, and, of course, the console could be used to input parameters or select alternative operations via function button selection.

Utility routines were developed to provide multiple choice, text entry, and query capabilities.[16] In fact, the mode of operation was to program specific application-defined functions for each button on the keyboard. The keys themselves were fitted with a transparent overlay such that the

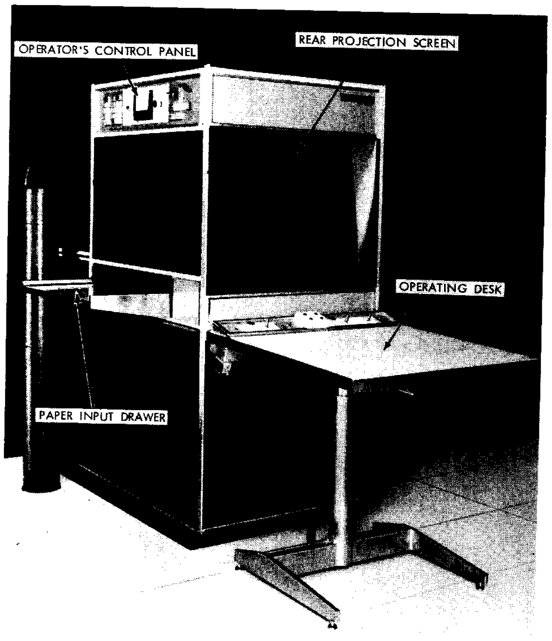


Figure 7. DAC-1 scanner/recorder.

labels could be changed for each new application by replacement of the overlay. One button might be assigned the function of computing the intersection of two lines. Depressing this button would cause a program to prompt the user to "pick" the two lines from the display from which to compute the intersection. Other buttons were assigned more generic functions such as Left, Right, Up, Down, or even Yes and No. Thus, computer-aided design became a process of selecting function button operations in a desired sequence and supplying parameter values for each function. Within a very short period of time, the development team realized that the buttons (today we call them icons) could just as easily be programmed to appear on the screen and be selected via the electronic pencil. Specific regions of the screen were dedicated to specific functions, as in a menu area or a graphics display area. So began the development of a menu- and window-based design environment (see the later section "DAC-1 design procedures").

Photo scanner/recorder. The photo scanner/recorder was combined into a single subsystem and packaged in a single enclosure, as shown in Figure 7. The purposes for which the photo scanner was intended were perhaps the most ambitious portion of the joint project with IBM. The entire system was extremely complicated and, at one time, the IBM engineers estimated space utilization inside the scanner/recorder enclosure at over 300 percent. (Light paths for the scanner and recorder were sometimes reused three and four times by bouncing images off multiple mirrors.) The digital electronics represented state-of-the-art technology, and the performance specifications for the analog elements of the system pushed the very limits of available technology.

Very complicated lens optics were used to focus the light onto a 35-mm film image and subsequently onto light-detection circuitry. Special distortion-correction circuitry was added to the system to compensate for nonlinearity in the positional accuracy of the CRT spot. Software was developed by GMR personnel to calibrate and dynamically compensate for both optical and electronic distortions within the field of the image.

Mechanical failures would continually dog the hardware, which contributed to a high failure rate and abnormally low availability time. For example, the 35-mm film mechanisms frequently scratched the film and, instead of analyzing data, the program logic became confused by distortions in the field of view. Shortly after the hardware was moved to the GM Technical Center in Warren, Michigan, a supply line for moist air was left on all weekend. The temperature dropped, and on Monday morning the development team stood and watched IBM bail water out of the electronic enclosures.

The purpose for which the photo scanner had been justified was to "read" engineering drawings.[17] As the project evolved, there was less and less need for this technology simply because engineering drawings were not needed as an input medium to the system. However, even without the need to use drawings as input, the analysis and understanding of the information content of an engineering drawing are complicated problems. Even if one makes a lot of assumptions about standards such as labels, dimensions, title blocks, and so on, the topology of a drawing can be ambiguous and extremely difficult to understand. Fortunately, the CAD/CAM applications for which the DAC-1 system was intended found other methods for entering digital information into the system (IBM cards or mylar punch tape).

It was also the ambition of the DAC-1 team to input three-dimensional measurements taken from models of vehicles. Structured lighting patterns were projected onto full-size clay models of concept vehicles. Alternatively, designers were requested to mark the clay models with tape or to scribe lines on the model to highlight the important "features" of the design. Two stereo photographs of the model were then transferred to 35-mm film, and continuous-tone images were digitized using the same photo scanner. Close-range photogrammetric measurement techniques were then used to correlate the two images to compute the three-dimensional coordinates of the feature lines on the surface of the model (see Figure 8). Computationally, the techniques worked well, but the nonrepeatability, the unreliability of the scanner, and the distortions in the optics made the system unusable. Years later, other systems were built to perform the same function using glass plates and lasers to obtain much higher digitizing accuracy.

Mathematics. As might be expected from the nature of the applications, the DAC-1 system made a substantial commitment to the development of new mathematical modeling techniques. Creation of a three-dimensional mathematical model of vehicle geometry was in the forefront of many minds. Mathematical operations to create and manipulate points, lines, and surfaces were a major part of the system. However, the two topics that received the most attention from the standpoint of mathematics were free-formed line and surface approximation.

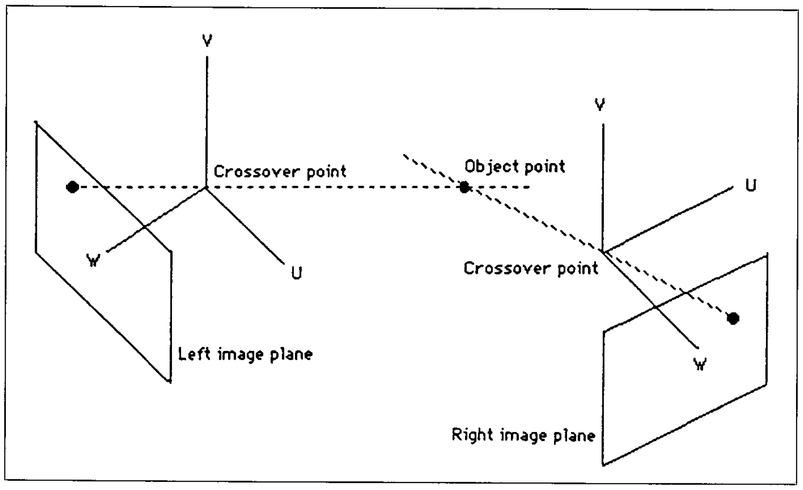


Figure 8. Photogrammetric stereo images.

Early on, it had been assumed that most of the input to the system would come from drawings and from designer sketches. When both of these modes of input became less viable, conventional coordinate measurements taken from physical models became the starting point for the design activity. These coordinate measurements were defined in a Cartesian coordinate system and related to vehicle body position by an arbitrary origin and orientation. Physical models were digitized as a network of intersecting point-set space curves lying on the surface. The network of surface curves was chosen very carefully to capture surface "flow" lines or any apparent feature lines that represented surface discontinuities. The process of converting the network of point measurements to a single mathematical representation was then formulated in three phases:

1. translate the CMM (coordinate measuring machine) data to an internal format, remove inconsistencies, and make sure that all crossing lines met at common points;
2. approximate the intersecting surface network of lines using polynomial equations to guarantee a smooth representation, while maintaining constraints such as the coordinates of the intersections; and
3. interpolate the smooth network of polynomial space curves with a formula that matched each boundary and provided a faithful representation of design intent between boundaries.

The following discussion will attempt to provide a brief outline of the development efforts in each of these three phases without getting into the details of an implementation. This work did represent some of the earliest CAD/CAM efforts in polynomial-based line and surface approximation.

Removal of inconsistencies was conceived as an automatic process.[18] The ground rules were that the network of surface lines would be composed from collections of coordinate points organized as sets of U lines that did not intersect one another and sets of V lines that did not intersect one another. The two sets of lines were assumed to intersect each other at one point and one point only, as shown in Figure 9.

The first objective was to resolve any inconsistencies in the CMM data and to guarantee that all intersections were defined by a single common point. An optimal search strategy was devised for comparing each coordinate point on every U line with all points on every V line to locate approximate intersections. Parabolic interpolation of each U and V line along with iteration was then used to locate optimal intersection points between pairs of lines. Linear, cubic, or quadratic blending formulas were then used to adjust spans of point sets lying between pairs of adjusted intersection points.

In practice the problem was straightforward, but errors in the input data model called for more interactive solutions. This was the first instance where interactive computer graphics was used effectively to locate a problem, display the situation, and prompt the user to resolve the error. Issues such as double data, stray points, or missing information were readily discovered by displaying the network on the graphic console and asking the user to indicate corrective action. The turnaround time to solve these problems was absolutely astounding compared with normal batch processing techniques. This turned out to be the real value of interactive computing — to keep the user in the loop for decision-making purposes.

After interactive changes were made to the data, the

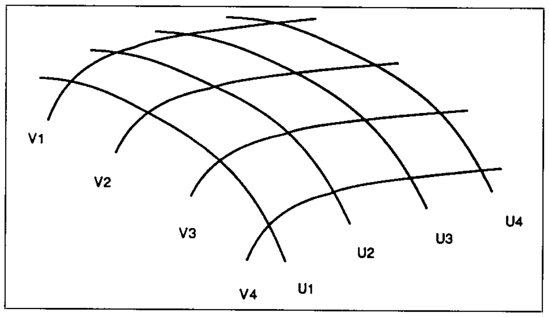


Figure 9. Network of surface feature lines.

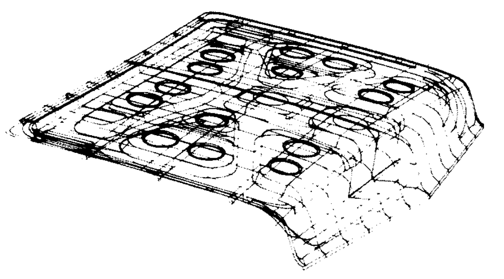


Figure 10. DAC-1 graphic image.

automatic method for adjusting a grid of lines was invoked again. This process continued until no errors were detected or until the user was satisfied with the quality of the adjusted line-grid network. The coordinate point data were then ready for approximation as a series of polynomial equations.

Piecewise cubic or quintic polynomials were used to approximate spans of coordinate points between successive intersections. Usually, polynomial curves of degree greater than two that are generated from a least-squares fit to noisy data exhibit unwanted inflections; to avoid this, a form of quadratic programming was applied to the fitting process. The method could be described as constrained least squares with an infinite number of linear inequality constraints imposed, such that the fitted curve's second derivative was nonnegative (or nonpositive) at every point of the domain. The accumulated chordal length l between successive points normalized to the range $(0 < l < 1)$ was used as the independent parameter, and this seemed to produce good results.

To establish a starting point for each piecewise polynomial and to join them with at least C^1 continuity proved to be a more difficult task. A marching procedure was used, where each successive segment was extended to take in as much of the data as it could and yet provide a close fit. This sometimes led to a bind, with no way of completing the fit within the specified tolerance. As was the case for adjusting a grid of lines, the solution to a fitting problem was to display the segment in question and prompt the user for corrective action, which might be the relaxation of the fitting tolerance or elimination of extraneous data points. Today, of course, infinitely constrained least-squares fitting of polynomial splines with variable knots is, if not commonplace, at least well within the capability of most CAD/CAM systems.

Surface interpolation in DAC-1 was of two kinds: rail and key blending,[19] and interpolation of patches through a network of space curves.[20] The terminology of the former approach was taken from the body designer's lexicon. A "key" is a line that is imagined to stretch between two opposing "rail" curves and to deform continuously as it moves along these rails, sweeping out a surface. The surface boundary, consisting of initial and final keys and the two rails, and the surface normals along the boundary, are given as input to this interpolation process. In an emulation of traditional drafting technique, the traces of the interpolated surface could be controlled in two different projections, such as side view and plan view. The viewing directions did not have to be at right angles, however; nor were they necessarily invariant from station to station along the rails. The overall method was particularly well suited to the interpolation of developable surfaces between prescribed boundaries such as were required for the design of windshields.

The second interpolation method was based on the definition of mathematical patches between pairs of opposing curves in space. Initially, the methods described below were limited to the four-sided boundary patch. However, the degenerate case of three boundary curves or the cases of five and six boundaries were treated as extensions to the four-boundary case.

Surface patches were defined parametrically in their analytic representation. The composite surface consisted of an image in (x, y, z) space of the range of values $(0 < u < 1, 0 < v < 1)$ in the u, v domain under the mapping:

$$P(x, y, z) = Q(u, v)$$

The Cartesian coordinates of all points interior to the area enclosed by the four boundary curves were then expressed as a linear combination of the corner points, points on the boundaries, and mixed partial derivatives at the corners. The blending function method of S.A. Coons was used for patchwise interpolation of the boundary values, which provided C^1 continuity overall. By using the same basis functions (namely, cubic splines) for blending in the interior of a patch as were used on the boundaries to "blend" sampled data points, a certain economy of representation was achieved. Point values on the boundaries and cross partial derivatives at the corners were the only input data needed to define the surface. Local interpolation, involving a boundary point and its nearest neighbors, was the means for determining suitable cross derivatives (or "twist") vectors. While the twist assignments and the choice of blending functions alone determined the distribution of normals along a boundary, the interpolation scheme could be enlarged to accept an arbitrary prescribed distribution — again emulating Coons. To ensure C^1 differentiability, a problem of reparameterization had to be addressed. Beach[9] describes the problem and its solution, and also gives an exposition of Coons surfaces.

Graphics. All graphic images in DAC-1 were created from points, lines, surfaces, and text (Figure 10). Since the system could display only a wireframe image, the representation for a surface was a series of section lines through the surface and the boundary lines of the surface. Remember that all of the geometric data were created in three dimensions. Therefore, a format was defined that contained the projection transformation (orthographic or perspective), the window location, and the font or line format information. A drawing was defined in terms of a display list containing names of formats, points, lines, surfaces, or collections of geometric elements. The display software performed the function of replacing points, lines, surfaces, and text by a series of blanked or unblanked vector endpoints that were assembled in an in-memory display buffer. If necessary, clipping computations were performed to limit the image to the defined size of the display window. After the vector endpoint display list was formed, a channel program was started that simply looped on the contents of this display buffer, refreshing the image on the display screen of the graphic console.

No work was done on hidden-line or hidden-surface removal simply because the user community was accustomed to looking at all the hidden construction lines on engineering drawings. Since all of the applications for DAC-1 were with stylists, product engineers, or tool engineers, this never became a problem. Years later, the technical illustration people became interested in this technology, at which time hidden-line and hidden-surface removal became a more important problem.

What was of more importance to the use of graphics could better be described under the category of human factors problems. Project personnel became concerned about this problem and did some early evaluation of user attitudes as a joint study with IBM human factors personnel. Because most images contained too many vectors for the hardware refresh rate, there was a tendency toward flicker in the image. As a result, there were many concerns about the visual perception of the users. This became an important problem as soon as graphics terminal access became more readily available. What types of vision problems would be experienced by a user who sat at a graphics terminal for eight hours? Some of the solutions were obvious: improve the resolution, reduce the flicker, and eliminate background reflections. These still are valid concerns today, although visual images are much cleaner and color seems to have made a big impact on image legibility.

The other human factors issues related to the input devices such as the electronic pencil, the function keyboard, and a card reader. The only use that was ever made of the card reader was to process an accounting record when logging onto DAC-1. The function keyboard was more easily replaced by functional icons displayed on the screen. Alphanumeric input could have been more readily prepared on a normal typewriter keyboard, instead of the graphic console keys, which were awkwardly arranged in 6 × 6 alphabetic order. Finally, as was mentioned earlier, the electronic pencil was used only as a pointer to pick items or icons that were displayed. Sketching was not practical, and even the fact that the operator had to raise his arm to screen level to make an icon selection became objectionable to the users.

Numerical control machining. During the period of DAC-1 development, numerically controlled machine tools also were becoming more and more popular within GM. The APT language for programming a machine tool was under development at MIT.[3] Within GM, a part programming system called INCA was already being used to program machine tools for cutting sheet metal stamping dies.[21] The DAC-1 project did not want to be in direct competition with the INCA system for the creation of part programs. Instead, the DAC-1 project decided to concentrate on the automatic

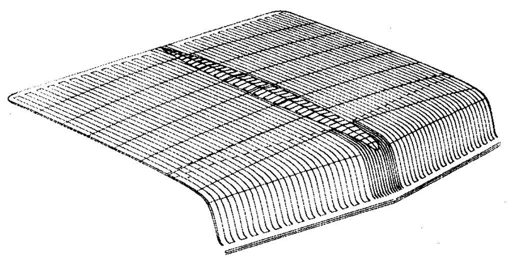


Figure 11. Trunk exterior tool paths created directly from mathematical data.

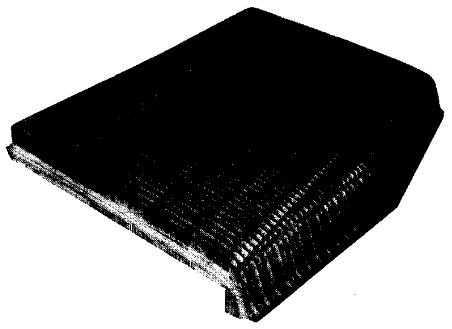


Figure 12. Model of a trunk lid created by numerical control.

creation of numerical control cutter paths directly from a mathematical model. The machining technology that was chosen was five-axis contour machining, and steps were taken to put together a feasibility demonstration. Logic was developed to optimize the machining of sheet metal die surfaces. By using the geometric programming language, a car roof model was machined a year before the graphics hardware arrived. The model demonstrated to GM management (at the executive vice president level) that hard progress (rather than "soft" progress) was being made. The demonstration helped to keep the project alive until the DAC-1 hardware could be installed at a GM site. In November 1963, a full-size rear deck (trunk lid) for a prototype Cadillac design was machined entirely from mathematical data. Three-axis tool paths (Figure 11) were used to machine a wood model (Figure 12) that had all the same characteristics and appearance of a conventional model before final hand finishing.

Conventional three-axis machining was better accepted in the tool room because it resembled the manual pantograph techniques. Even today, five-axis machining has had a small impact because three-axis machining is perfectly adequate, as long as the machining step size is held small.

DAC-1 design procedures. DAC-1 provided I/O facilities to digitize and create drawings. A wide range of mathematical procedures consisting of the basic data manipulation operations available in DAC-1 were evolved for the creation and manipulation of geometry. Disk storage devices made it possible to build up large databases of this design data. However, how did DAC-1 allow the designers to specify the content or essence of their designs?

Many books have been written on the subject of what constitutes design. However, most authors agree that design is a decision process based on rules and experience. The use of a written language, such as DGL, to express a design procedure, was generally not well accepted. Only a few of the end users or designers who were recruited from the Fisher Body and Styling Divisions of the corporation were ever able to master DGL. Instead, the project pioneered the development of design "overlays" or an icon-based menu mode of operation that is seen today in most general-purpose CAD/CAM systems. All line-related operations for creating and manipulating lines were grouped together into a so-called "line overlay." Similarly, all surface operations were grouped together into a "surface overlay." The term "overlay" came from the fact that a transparent overlay was used to relabel the function buttons for the group of operations to be performed.

Of course, the value of preplanning or programming a design procedure should not be discounted. Even today, most commercial systems provide some type of macro language that can be used to describe sequences of operations. The real power of this approach is when an operational language can be used to customize a general-purpose CAD system for a specific class of operations. There has been a very effective use of macro languages to customize generic systems for specific problem domains such as architecture, geology, and electrical design. Perhaps the designers of DAC-1 had the right idea from the beginning, but the technology needed to be packaged in a fashion that would be more amenable to the designer's natural language of communication.

Termination of the DAC-1 project

In 1967 Edward Cole was GM's president. He had observed the beginnings of computer-aided design at GMR. He made the observation that this was becoming an increasingly large undertaking. Therefore, he decreed that DAC-1 was no longer a research project and that the responsibility for further development and application of this technology should be transferred to the Manufacturing Development Staff. Accordingly, the DAC-1 development group at GMR was broken up, and key members were transferred to carry on this work under new management. Other members of the DAC-1 team transferred to other divisions of the corporation, including Marketing Staff and the GMR Mathematics Department. A smaller team of engineers and scientists, reporting to Edwin Jacks, remained without a project.

After the personnel transfers were completed, Edwin Jacks called a meeting of the remaining lead team members. Each member was polled as to the key technical issues that needed to be addressed in the field of computer science. Almost everyone agreed that the computer operating system was the weakest element that limited the future of interactive computer graphics applications. This conclusion was reached because of the universal dissatisfaction with IBM time-sharing systems. A time-shared operating system needed to be tailored to the needs of interactive graphics. Furthermore, the system had to be implemented in a high-level language and support a hierarchical file system. So was born the GMR version of Multics that was called MCTS for Multi-Console Time Sharing.[22] The processor selected for this project was the Control Data Star-100, and GMR began to develop operating system software. However, that is another story and perhaps worthy of another article.

By the late 1960s IBM 7094 computers were being replaced by IBM System/360 computers. The DAC-1 hardware had been moved to the Fisher Body Division but was now obsolete and no longer of use to GM. Therefore, all graphics hardware and software were donated to the University of Michigan for experimental purposes. However, maintenance problems plagued the hardware, and the university never really made any use of the system. The equipment was finally scrapped for parts to populate an electronics laboratory.

How well did the achievements of the DAC-1 project satisfy the original objectives? As a laboratory for experimentation in graphical man-machine communication, the system was a huge success. Certainly, computer graphics and CAD/CAM systems are today a multibillion-dollar business. GM uses the technology in almost every aspect of the business, from engineering to advertising. Nevertheless, there remain two questions: How has this technology impacted the design process, and have we really achieved computer-aided design? Today, most vehicle designs are expressed in the form of three-dimensional mathematical models. Vehicle geometry is expressed as a surface model or, in some cases, a solid model, rather than just a collection of lines and dimensions on a drawing. However, studies of the design process by GM and others[23] have concluded that design is principally a decision-making process. DAC-1 technology was applied mostly as a detailing tool rather than as a decision-making tool.

How has the technology encompassed by the DAC-1 system evolved over the past 30 years? Certainly, computer graphics is now recognized by itself as a separate field of technology. The wide array of graphics hardware and software offerings displayed each year at the ACM Siggraph or Design Automation conferences makes the early DAC-1 efforts look primitive. Raster-display technology has replaced stroke or vector displays, and with raster technology came 3-bit, 8-bit, and 24-bit color systems. DAC-1 display software has been replaced by hardware accelerators that automate many of the time-consuming aspects of image generation.

In spite of multiprogramming, it became much too expensive to dedicate an entire IBM 7094 computer to support a single graphics terminal, as was done for the DAC-1 system. Time-sharing was an attempt to distribute the expense of a large mainframe among a community of users. In most cases, the objectives were achieved, but two fundamental problems became more and more evident:

1. the limited bandwidth between remote graphics terminals and a mainframe computer, and
2. the variations in response time of a time-shared system when operating near saturation.

As a result, more and more graphics-related applications have migrated toward minicomputers and 32-bit microprocessors. The engineering workstation revolution of the late 1980s has made it such that only a workstation can deliver the graphics performance expected of today's systems. Of course, there is also a wide range of smaller two-dimensional drawing applications that execute quite successfully on personal computers. In many respects technology has recycled back to where DAC-1 began — namely, a single computer dedicated to a single graphics display.

During the development of DAC-1, no operating system existed that would provide the services needed for interactive graphics applications. Today, most operating systems for computer graphics applications are based on some flavor of Unix. The early experiments with high-level languages in DAC-1, namely NOMAD, have certainly continued. Today most graphics applications are implemented in the C or C++ programming languages, and there seems to be no reason why most system code cannot be implemented in a high-level language. The Unix system has also proved the viability of writing a large operating system using a high-level language.

Whereas the DAC-1 approach to a disk database was very primitive, we now have a wide range of database technologies to choose from, including relational and object-oriented systems. However, the fundamental idea of a mathematical model maintained on a disk file and manipulated by a collection of programs controlled from an interactive terminal, as put forth by DAC-1, remains intact. Perhaps the one new concept widely used today is that the data and programs need not be maintained local to the graphics device itself but instead can be accessed transparently via a network.

While DAC-1 did have some very advanced concepts with respect to systems and programming, it did not provide anywhere near the software tools for testing and debugging that we have today. All DAC-1 programming was done on coding sheets and subsequently keypunched onto IBM cards. Compilation was via submission of cards to a batch stream with a lot of manual checking before execution was ever attempted. Since DAC-1 supported only a single terminal, testing and debugging meant standing in line with 20 other programmers for a five-minute shot at three in the morning. No software tools were made available for interactive debugging of programs. If the programmer happened to forget to initialize a variable and the program blew up, there would not be a chance to make the correction and try it again before the next evening.

Software today is much more complex as compared with DAC-1. The entire DAC-1 system consisted of several hundred NOMAD subroutines. Today, comprehensive modeling systems contain thousands of modules and rely on libraries of utilities for graphics I/O, database access, numerical computation, and operating system services. The DAC-1 displays were limited to a few thousand vectors. Modern graphics devices usually deal with millions of vectors or millions of surface elements.

The DAC-1 system was conceived entirely to address the design of three-dimensional free-formed components, tools, and dies. The successors to the DAC-1 system are being used for that purpose today. During this same period, the Sketchpad system developed at MIT addressed the topic of two-dimensional prismatic wireframe objects.[24] Since that time, we have seen an explosion in the variety of graphics applications. Integrated circuit design would not be where it is today without computer graphics technology. There are also a large number of two-dimensional drawing or graphic illustration applications. However, from day one, DAC-1 always had in mind that the mathematical model was three dimensional and that two-dimensional drawings could be created from a projection of the three-dimensional model.

DAC-1 was both a three-dimensional wireframe and a three-dimensional surfacing system and, as such, the operations on points, lines, and surfaces were geometric in nature (transform, intersect, project, smooth). It was not until years later that solid modeling systems demonstrated the value of Boolean operations on geometry. However, it was recognized early on that the topology of an object or the connectivity of geometry was equally as important as the shape. DAC-1 did provide an operation that made a cutout in a surface. It was recognized that it was important to remember in what surface the cutout had been pierced. Similarly, it was necessary to record which lines were the boundaries of which surfaces and how surfaces were related one to another. These are all elements of modern modeling systems.

As has been mentioned earlier, the importance placed on the photo scanner/recorder for "reading" engineering drawings was greatly diminished by the way the entire CAD/CAM industry evolved. Interactive terminals became the main medium for communication of information between a designer and a computer. However, in many respects the photo-scanning technology became the forerunner of modern image-processing systems. Today, image processing is a field of technology unto itself. The breakthrough was to utilize raster scan devices and use image-processing or filtering techniques to handle the large volume of scan data. ■

Acknowledgments

The DAC-1 project would not have happened without the leadership of the management of the General Motors Research Laboratories. Donald Hart certainly was responsible for providing the supportive environment in which this work could take place. George Ryckman kept the ship of computer operations sailing smoothly while the DAC-1 team tried to break the system. Finally, Edwin Jacks really was totally responsible for the technical leadership and inspiration that caused the entire project to happen.

I am deeply grateful to the following friends and colleagues who have contributed to the manuscript: Edwin Jacks, George Ryckman, George Dodd, Charles Gerrish, Barrett Hargreaves, Charles Metzger, Richard Lewis, Weston Meyer, Hugh Brous, John Bookston, John Joyce, Joseph Olsztyn, Gerald Devere, and Theodore Theodoroff. Special thanks to Philip Dorn and Michael Marcotty, who functioned in a rewrite capacity to make the article much more understandable. Philip was a close friend and did not live to see the final article in print. His views of the field of computer science will be missed.

References

1. R. Patrick, "General Motors/North American Monitor for the IBM 704 Computer," Rand Report P-7316, Jan. 1987, presented at the National Computer Conf., Pioneer Day, June 17, 1987.

2. R. Herman, "Traffic Dynamics: Analysis of Stability in Car Following," *Operations Research*, Vol. 7, No. 1, Jan.–Feb. 1959, pp. 86-106.

3. D.T. Ross, "Origins of the APT Language for Automatically Programmed Tools," in *History of Programming Languages*, R.L. Wexelblat, ed., Academic Press, ACM Monograph Series, New York, pp. 279-336.

4. G.R. Price, "How To Speed Up Invention," *Fortune*, Nov. 1956, pp. 150-228.

5. J.C. Gorham, "Numerical Control in Body Engineering — Fact and Fantasy," *SAE Annual Meeting*, Publication No. SAE-129A, Jan. 1960.

6. E.L. Jacks, "A Laboratory for the Study of Graphical Man-Machine Communication," *AFIPS Conf. Proc.*, Vol. 26, Part 1, AFIPS Press, Arlington, Va., 1964, pp. 343-350.

7. S.A. Coons, "Surfaces for Computer-Aided Design of Space Figures," Design Division, Dept. of Mechanical Eng., MIT, Cambridge, Mass., Jan. 1964.

8. S.A. Coons, "Surfaces for Computer-Aided Design of Space Forms," Project MAC Tech. Report TR-41, MIT, Cambridge, Mass., 1967.

9. R.C. Beach, *An Introduction to the Curves and Surfaces of Computer Aided Design*, Van Nostrand Reinhold, New York, 1991, Chap. 5.

10. C. DeBoor, "Bi-Cubic Spline Interpolation," *J. Mathematics and Physics*, Vol. 41, 1962, pp. 212-218.

11. W. Gordon, "Spline-Blended Surface Interpolation Through Curve Networks," *J. Mathematics and Mechanics*, Vol. 18, 1969, pp. 931-952.

12. B. Hargreaves et al., "Image Processing Hardware for Graphical Man-Machine Communication," *AFIPS Conf. Proc.*, Vol. 26, Part 1, AFIPS Press, Arlington, Va., 1964, pp. 363-386.

13. P.A. Crisman, *The Compatible Time-Sharing System*, MIT Press, Cambridge, Mass., 1965.

14. P. Cole, P. Dorn, and R. Lewis, "Operational Software in a Disk Oriented System," *AFIPS Conf. Proc.*, Vol. 26, Part 1, AFIPS Press, Arlington, Va., 1964, pp. 351-362.

15. B.W. Arden, B.A. Galler, and R.M. Graham, "MAD at Michigan," *Datamation*, Vol. 7, No. 12, Dec. 1961, pp. 27-28.

16. T. Allen and J.E. Foote, "Input/Output Software Capability for a Man-Machine Communication and Image Processing System," *AFIPS Conf. Proc.*, Vol. 26, Part 1, AFIPS Press, Arlington, Va., 1964, pp. 387-396.

17. F.N. Krull and J.E. Foote, "A Line Scanning System Controlled from an On-Line Console," *AFIPS Conf. Proc.*, Vol. 26, Part 1, AFIPS Press, Arlington, Va., 1964, pp. 397-410.

18. G. Cole, "Line Grid Adjustment Procedure for the Removal of Inconsistencies from a Grid of Lines Intended for Body Surfacing," Research Report CT-50, General Motors R&D Center, Warren, Mich., 1968.

19. R. Lewis and J. Horner, "Automobile Body Surface Interpolation in Parametric Space Utilizing Analytic Surface Keys," Research Report CT-31, General Motors R&D Center, Warren, Mich., 1966.

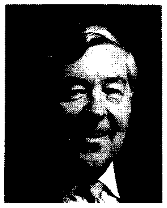
20. J.M. Bookston, "The DAC-1 Procedure for Interpolating Surfaces Through a Network of Intersecting Space Curves," Research Report CT-48, General Motors R&D Center, Warren, Mich., 1968.

21. E. Toton, "Applications of CAD/CAM to Sheet Metal and Die Cast Die Construction," *Soc. Manufacturing Engineers CAD/CAM 2 Conf.*, MS73-964, 1973.

22. R. Brown, J. Elshoff, and M. Ward, "The GM Multiple Console Time Sharing System," *ACM Operating Systems Rev.*, Vol. 9, No. 4, Oct. 1975, pp. 7-41.

23. D. Whitney and E. Nevin, *Concurrent Design of Products and Processes: A Strategy for the Next Generation in Manufacturing*, McGraw-Hill, New York, 1989.

24. I.E. Sutherland, "Computer Reads Design Sketches," *Iron Age*, Vol. 192, No. 9, Aug. 1963, pp. 79-81.



Fred N. Krull is a consulting engineer with Trellis Software and Controls in Rochester Hills, Michigan. During the time this article was prepared, he was a section manager in the Analytic Process Department of the General Motors Research and Development Center. He was one of the original team members of the DAC-1 project. During his more than 30 years at the GM Technical Center, he was involved in a variety of computer-related activities. His major interests have been in computer-aided design of both mechanical and electrical automotive systems.

Krull has a BS in applied mathematics from the University of Wisconsin and an MS in theoretical and applied mechanics from the University of Illinois.

Readers can contact Krull at Trellis Software and Controls, 2619 Product Drive, Suite 106, Rochester Hills, MI 48309; e-mail: fredkrull@aol.com.