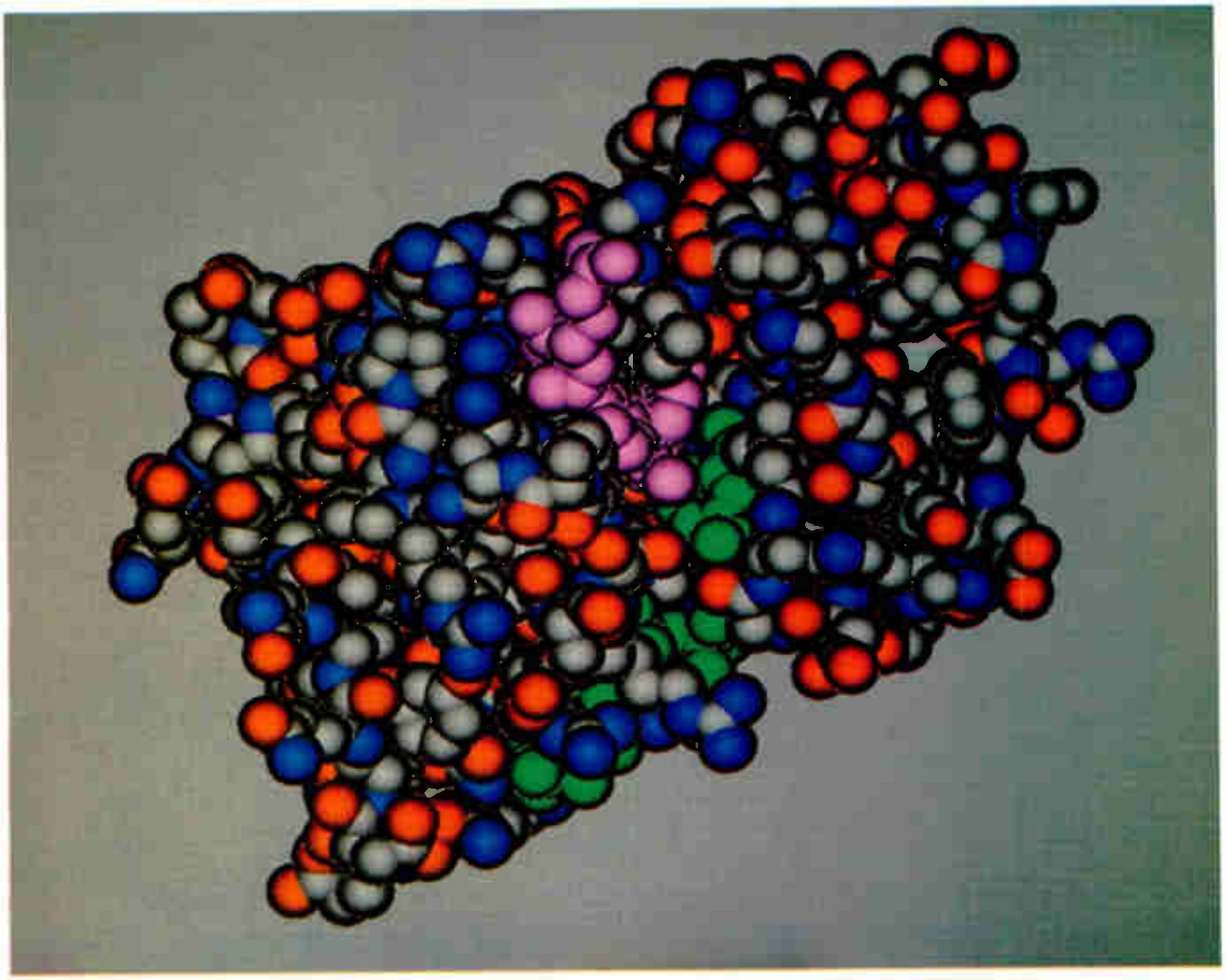


Work station unites real-time graphics with Unix, Ethernet

Dedicated chips manipulate images; hierarchical graphics structures lighten the programmer's burden

by James H. Clark* and Tom Davis
Silicon Graphics Inc., Mountain View, Calif.

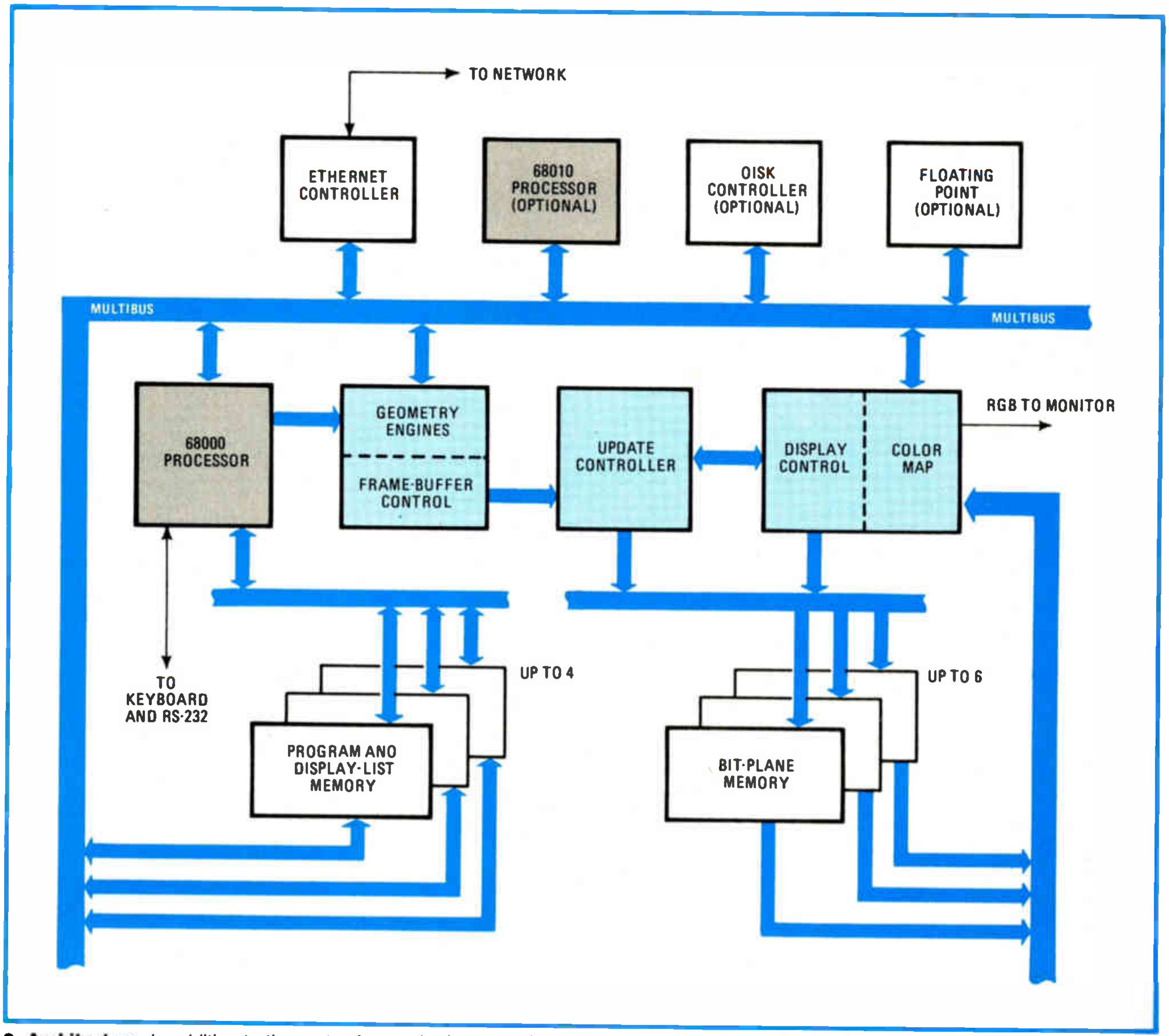
□ Productivity in engineering design requires powerful computing tools that reduce the time needed to do the design. Because virtually all forms of engineering require manipulation of geometrical and graphical constructs as well as general-purpose computing, a work station with real-time graphics decreases the time the engineer must wait to see the results of design changes and hence increases productivity.

Still, for general scientific and engineering use, computer work stations require more than graphics. A work station's computation should be supported by a general-purpose operating system and by appropriate languages and tools for developing application programs. Besides allowing for the local development and execution of programs, the work station must also be able to serve other computers as a graphics terminal.

Thus, as the engineer's principal connection to the outside world, the work station needs to function within a communications network, allowing separate work stations to share data and resources such as disk drives,

1. Molecular synthesis. Shaded solids and vectors with motion, color, and 3-d perspective aid scientists in composing new molecules and determining their properties. Since the IRIS work station can draw such screens in real time, it raises productivity.

*On leave of absence from the Computer Systems Laboratory, Stanford University, Palo Alto, Calif.



2. Architecture. In addition to the system's standard 68000 microprocessor, which runs a real-time kernel, an optional 68010 runs virtual-memory Unix 4.2BSD. Proprietary geometry chips perform the matrix manipulations that are needed for real-time graphics.

hardcopy devices, and other computers. Simultaneously, as the engineer's principal connection to the computer, the work station must be capable of high-speed graphics to simplify the manipulation and analysis of data but must not expose the user to excessive delays.

Powerful real-time graphics is fundamental to an engineering work station and is a fundamental part of the IRIS work station from Silicon Graphics Inc. The IRIS combines Bell Laboratories' Unix operating system and the Ethernet communications network with a general-purpose, real-time, three-dimensional color-raster graphics system. Custom very large-scale integrated circuits in the IRIS, such as Silicon Graphics' trademarked Geometry Engine (see p. 117), reduce its cost and power requirements, increase its reliability, and provide real-time speed and high graphics functionality suitable for a broad spectrum of two- and three-dimensional applications.

In addition to general-purpose computing, the IRIS supports real-time color display and manipulation of bit-mapped or stroked characters, 2-d or 3-d vectors, parametric curves and surfaces, 2-d areas, and 3-d solids with shading and hidden surfaces removed. These graphic objects are defined in the user's coordinate system, in either 32-bit floating-point or 24-bit integer values. All the geometric transformations, rotations, translations, scaling, clipping, multiple windows and viewports, perspective projections, and so forth are done by the engine at rates approaching 10 million floating-point operations a second. In addition, the geometry chip is provided as a general-purpose geometric computing subsystem suitable for calculating the intersection of solids and for matrix arithmetic at these floating-point rates.

Instantaneous drawing rates decrease engineering design time. Engineering work requires the manipulation of complex 2-d and 3-d geometric models that frequently are hard to visualize or contain large amounts of information. If the display of information is instantaneous, the engineer spends less time in analysis of the information and hence is more productive. The list of applications needing this high level of performance spans engineering disciplines from mechanical to electrical engineering (Figs. 1, 3, 6, and 7). Even fields such as physical chemistry—where shaded models of molecules provide hints of chemical properties (Fig. 1)—require high-performance real-time graphics. Circuit designers need to be able to view multiple windows of graphics and textual data (Fig. 3 and cover). For architectural engineers, the ability to view 3-d architectural layouts speeds the design process (see cover). Similarly, viewing trajectories of robot manipulators and checking for collisions allows for faster programming of movements (Fig. 6). Aircraft, automobile, and ship designers continually need to model exterior surfaces based on parametric curves (Fig. 7) and even to simulate their motion (again, see the cover illustration).

The IRIS is a real-time graphics computing node in an Ethernet communications environment. Three main configurations are possible: work stations, terminals, and file servers. The IRIS terminal runs a small real-time operating system called the V Kernel—developed by David Cheriton and co-workers at Stanford University—and provides a multiwindow environment for access to one or more computers on the network. The work station runs Bell Laboratories' Unix operating system and, with a disk, is capable of operating independently. A file server supplies data files to other computing nodes over the network. Both terminals and work stations provide the graphics capability.

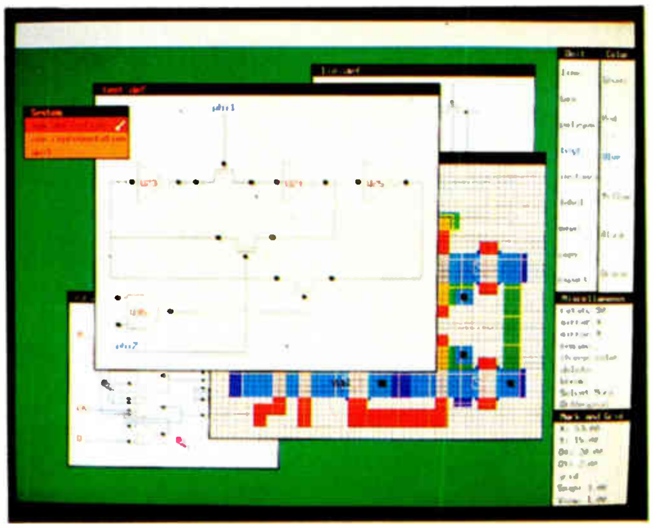
Graphics node

When the IRIS is used as a terminal, the graphics application runs on a remote host connected to the terminal by a network. (Serial and parallel connections to the host are also possible.) Graphics commands issued on the host are sent to the IRIS to be executed, where it either draws them immediately or stores them in display lists for local rapid drawing, depending upon the mode. Once a display list is defined, the application program typically sends only editing and redraw commands, so that instantaneous response is achieved over the network.

The terminal uses the Motorola 68000 and a geometry-chip pipeline with output to a high-resolution color raster-scan display. All graphics pipeline data is transmitted on a private data bus, and all the circuit boards use the standard Institute of Electrical and Electronics Engineers (or Intel) Multibus for general communications.

The work station may have either one or two processors. The dual-processor form just adds a second processor to the backplane, which serves as the host to the terminal processor. The second processor runs a 68010 with Virtual Memory Unix 4.2BSD, while the terminal processor continues to run the V Kernel. Here, the "terminal" has a dedicated host, and communication is through shared memory, rather than with the network.

The single-processor work station runs Unix 4.2BSD on a 68010 processor and memory subsystem, which serves as the real-time graphics processor as well. The Unix kernel is slightly modified so as to provide real-time graphics service. For most real-time applications, performance can be improved by a factor of almost two with the assistance of a dual-processor work station.


3. VLSI design. Two-dimensional area-fill functions—along with pan, zoom, rotation, and scale operations—help circuit design all the way from VLSI layout to schematic entry. Engineers can use the system's multiple windows to view sections of circuits.

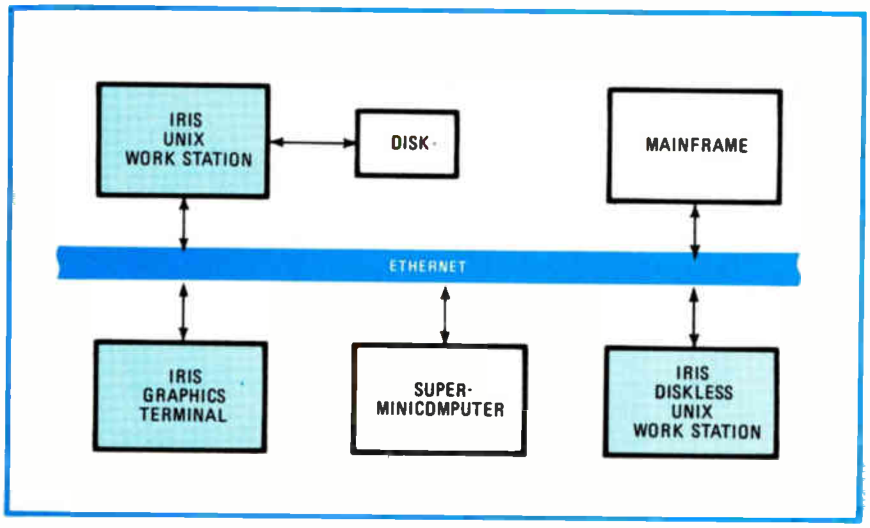
The electronics consists of one or more general-purpose processors and a geometry pipeline made up of the engine subsystem, the frame-buffer controller, the update controller, and the display controller. Also, standard network and disk controllers may be present (Fig. 2).

The 68010 processor and memory subsystem generates 24-bit virtual addresses that are applied to the memory map, on-board read-only memory, and on-board input/output interfaces. The page map uses a fast, 1.5-level structure and 4-K-byte pages to provide up to 16-megabytes of virtual address space for up to 256 process contexts in a demand-paged Unix environment. Physical addresses from the page map contend with accesses from the Multibus and the hardware memory-refresh circuit through a triple-ported arbiter. Space is provided for up to 128-K bytes of erasable programmable ROM and four RS-232-C serial lines. Using 64-K random-access memory, the 0.5 megabyte of memory on the processor board can be extended in 1-megabyte increments. With 256-K parts, these numbers increase to 2 and 4 megabytes, respectively. Memory references from the Multibus are mapped to provide scatter-gather direct memory access, as well as control, when there is more than one central processing unit on the Multibus.

In the terminal form, the 68000 acts as the display processor, executing all graphics drawing instructions and directing data into the graphics pipeline. It manages display-list memory, input devices such as the keyboard, and communications with the host processor. In the single-processor work station, the 68010 does these functions as well as running Unix 4.2BSD.

Graphics pipeline

Graphical output is initiated by the CPU when it sends commands and data to the graphics pipeline. In normal mode, the engines do matrix transformations, clip to normalized coordinates, and finally scale the transformed, clipped points to screen coordinates. The frame-



4. Multiple modes. Real-time graphics is not enough by itself to enhance productivity. The IRIS work station meets the challenge by fitting into an Ethernet network as a stand-alone work station, graphics terminal, and server for other network nodes.

buffer controller (a 16-bit 2903 bit-slice processor) does such things as interpret characters, control the font memory, and compute coefficients for rendering lines and polygons. The update controller does scan conversion, including filling polygons and lines, clipping characters, and placing the results in the frame buffer. Finally, the display controller fetches picture-element values from the frame buffer and draws them on the face of the color cathode-ray tube.

The Multibus is normally used only for I/O communication with the disk or the Ethernet. The graphics pipeline is a separate data path, not on the Multibus. Likewise, the memory on the processor is dual-ported, and accesses to it need not use the Multibus.

The geometry subsystem consists of a pipeline of up to 12 identical engine chips (see "Gearing up for real-time graphics," p. 117). Each chip can be configured in software to do dot products, clipping, scaling, or nothing (a null device). A 12-chip pipeline is typically configured with the first four as dot product chips (forming a 4-by-4 matrix multiplier), the next six as clippers (to clip against the left, right, top, bottom, near, and far planes), and the last two as scalers to convert the normalized coordinates into physical screen coordinates.

The matrix multiplier transforms points from their original coordinate system into a normalized eye-coordinate system. The clippers clip those lines and polygons that would have extended outside the viewing area (clipping is different for lines and polygons). The clippers may need to add or delete points in the process. Finally, the scalers convert the transformed and clipped points into physical screen coordinates.

The data that comes out of the geometry pipeline is primarily a set of commands in absolute screen coordinates. The raster subsystem's main jobs are to fill in the pixels between the endpoints of the lines, fill the interiors of polygons, and convert character codes into bit-mapped characters. Each line that is drawn has certain attributes, among them a width (of 1 or 2 pixels), a stipple pattern, and a mode.

An IRIS system may have from 4 to 24 bit-planes in increments of 4. Any graphical object that is drawn—whether a line, a polygon, or text—is drawn in the current color with the current write mask. Both the write mask and the color have 1 bit per bit-plane. Only if the bit in the write mask is on is the color bit written into the corresponding bit-plane.

The IRIS system handles three display modes: single-buffer, double-buffer, and red-green-blue. With 24 bit-planes, the system can be configured in the RGB mode, where 8 bits of the color specify the intensity for red, for green, and for blue. Both the other modes work through a color map that has 4,096 entries of 24 bits each. The 24 bits contain the red, green, and blue intensities for each of the two entries. One access mode uses the values in the bit-planes as the entry into the map. Another method (especially useful with 8 or fewer bit-planes) is to use the bit-planes for the 8 low-order bits in the map and a software-controlled map register for the top 4 bits. Thus the map may be configured either as one 12-bit map or as 16 8-bit maps.

In single-buffer mode, all the bit-planes are visible, and changes appear on the screen as soon as they are made in the bit-planes. This is unsuitable for fast, animated graphics, since the image must immediately be erased before the next frame can begin to be drawn, and the viewer will see, on the average, a half-drawn image.

Real-time graphics is therefore usually done in double-buffer mode—one image is viewed while the next one is being drawn. In double-buffer mode, the bit-planes are divided into halves. One half is viewed while the other half is modified. When the modified frame is complete, the halves are exchanged and the new frame appears all at once. There are routines to synchronize buffer swapping with a real clock to get uniform motion. All the color-mapping features described in the last section work the same way in single- or double-buffer mode.

The raster subsystem has hardware support for the cursor. The cursor is any 16-by-16 pattern that always appears at the current cursor position drawn in the cursor color with the cursor's write mask. Commands to set the cursor's current position are set automatically by the software, so that the application program need not do it. The cursor color and write mask are independent of the geometry color and write mask. In double-buffer mode, the cursor is drawn on the front buffer, while the geometry is written on the back one.

The raster subsystem supports variable-pitch raster fonts and stores them in a 64-K-byte font RAM. The contents of the font RAM are controlled by software on the 68000, which allocates the memory for fonts with different numbers and sizes of characters. As characters from the raster font are drawn, the current character position is automatically updated by the width of the

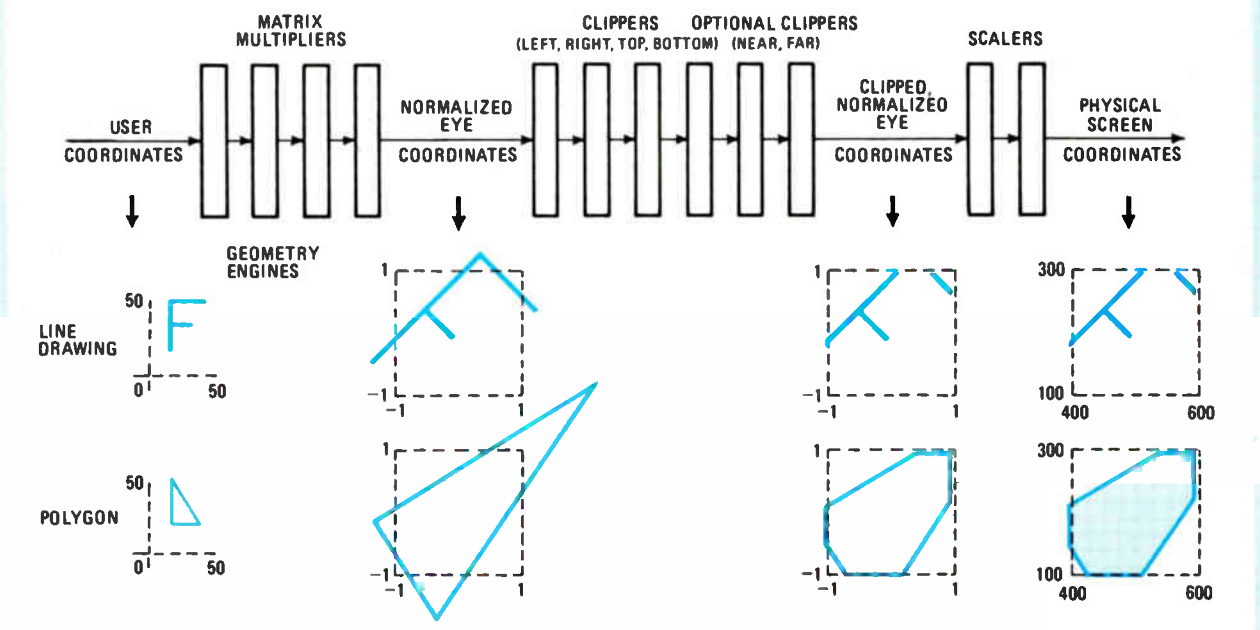
Gearing up for real-time graphics

The Geometry Engine, trademarked by Silicon Graphics, is a very large-scale integrated circuit with about 75,000 transistors that can be configured to do one of three basic operations: matrix transformation, geometric clipping, and mapping to output device coordinates. Working internally with four-dimensional vectors of floating-point numbers, the hardware directly supports all the commands necessary to save the internal state of the system, manipulate matrixes and viewports, draw lines, curves, surfaces, polygons, and characters, and change colors and other attributes of the system.

Besides four floating-point arithmetic and logic units, the engine contains a control store for the microcode to implement floating-point operations, line and polygon clipping, perspective division, stack management, and curve generation. For example, in the engine's curve command, microcode generates points along any three-dimensional rational cubic spline, employing user-supplied coefficients and endpoints, which it treats as if they had been sent with separate move and draw commands.

In another mode of operation, called hit-testing, a pair of commands causes the engine to indicate against which planes an item is clipped, without actually drawing the item. For example, a polygon surrounding the viewport would record four clips, even though none of its lines passes through a visible region. The figure shows how a line drawing and polygon would appear at various stages in the pipeline.

Besides the configuration illustrated, others are sometimes useful. For example, if special commands are employed to make the clippers and scalers merely pass data without operating on it, the pipeline can be used as a hardware 4-by-4 matrix multiplier. Furthermore, the pipeline can be configured to clip polygons against an arbitrary plane—ideal for solid-modeling applications. The first four geometry chips can transform the coordinate system to make that plane a standard clipping plane, the fifth is a clipper, and the next four invert the transformation. All polygons passed through this pipeline will be clipped against the given plane.


MATRIX MULTIPLIERS
CLIPPERS (LEFT, RIGHT, TOP, BOTTOM)
OPTIONAL CLIPPERS (NEAR, FAR)
SCALERS
USER COORDINATES
NORMALIZED EYE COORDINATES
CLIPPED NORMALIZED EYE COORDINATES
PHYSICAL SCREEN COORDINATES
GEOMETRY ENGINES
LINE DRAWING
POLYGON
50
0
50
1
-1
-1
1
300
100
400
600

characters drawn. Characters are clipped by a screen window on bit boundaries, so a character does not suddenly disappear as soon as a part of it moves off the screen. It is possible to read and write individual pixels on the screen—a useful feature in such applications as interactive "video painting" as used in the graphic arts.

System software

The IRIS work station is a 68010-based Unix 4.2BSD machine with network support and real-time graphics hardware supported by a graphics library. All standard Unix utilities are available, so the system can be used as a stand-alone work station. The IRIS system software is written in C, but the graphics library is callable from other languages (currently Pascal and Fortran).

The IRIS terminal is attached to its remote host or hosts over a network or through a serial connection (Fig. 4). The graphics software is independent of the type of connection used, but greater network bandwidth gives better performance. IRIS systems use network protocols that are compatible with both the Government-sponsored IP/TCP and Xerox's XNS. A VAX machine running the Unix 4.2BSD operating system uses IP/TCP, for example, but a VMS VAX communicates with IRIS terminals with XNS software provided by Silicon Graphics Inc.

The IRIS graphics software provides a convenient, high-level interface with the hardware. It also provides low-level access for applications demanding it. Its main features are:

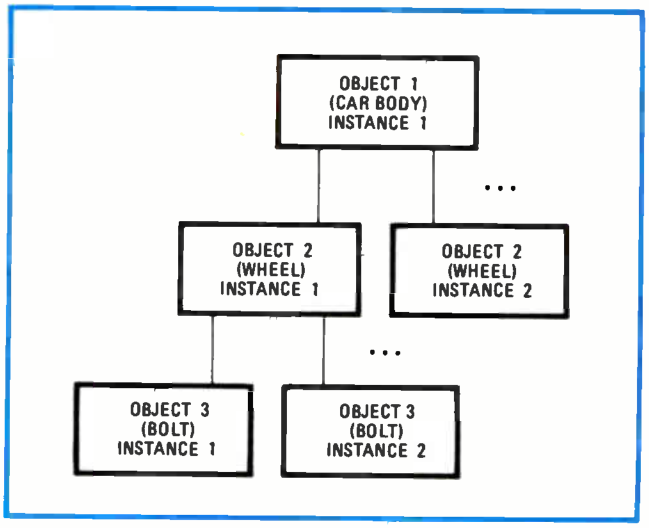
- Multiple windows and multiple viewports for mixed text and graphics.
- Hierarchical display-list definition and maintenance.
- User-space object transformations at the command level.
- Immediate and compiled display-list modes.
- Display-list editing.
- Object-selection mechanism, including picking and collision detection.
- Geometric computation.
- Queued I/O.

Proposed graphics standards such as CORE and GKS could be implemented using a subset of the IRIS graphics library. Unfortunately, CORE treats transformations as an adjunct to a graphics system, so that the CORE system is not as suitable for real-time applications. GKS, on the other hand, does not provide the 3-d applications, although it does allow transformations to be applied to all objects. The IRIS graphics library is basically a combination of the two systems, but it is specifically tailored to the IRIS, rather than designed to be the sum of GKS and CORE.

Hierarchy and naming

Both the hardware and software support hierarchical graphical objects. A hierarchical description of an automobile, for example, might include one list of commands that draws the body and another that draws a wheel. To draw the car, the software must draw only the body, then draw the wheel four times, each with appropriate translations and rotations. The wheel itself may contain still simpler objects—perhaps five instances of a bolt.

The basic graphical data structure is called an object. It is made up of primitive drawing commands together with calls on other objects that are transformed (rotated, translated, and scaled). In the example above, the bolt object might consist only of move and draw commands. The wheel has commands to draw the rim and tire, plus five calls on the bolt object. The car object consists of the body-drawing commands plus four calls on the wheel object. Only three object descriptions are needed, although there will be 20 bolts in the final drawing (Fig. 5). This hierarchical nesting can be arbitrarily deep, and the transformations for intermediate values are saved in the engine's matrix stack.

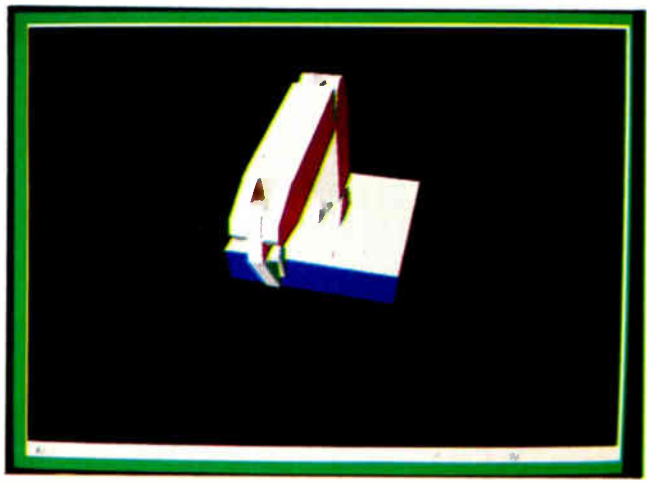


5. Hierarchical nesting. Although the final drawing may be a complex combination of several objects, IRIS software allows designers to deal with multiple objects simply as separate instances of some distinct object, like a car body, a wheel, or a bolt.

Naming is a problem with hierarchical objects. An engineer editing the car drawing needs to know that the bolt he has pointed to is, say, the third bolt on the left front wheel of the car. To specify an object completely, a path of names is required—instance "three" of object "bolt" in instance "left front" of object "wheel" in instance "one" of object "car." Deeper nesting requires a longer path. The graphics library supports paths of instance and object names for both selection and editing.

Since a transformation can be applied every time an object is called within another object, each object can be described in the most convenient coordinate system. This is even true of the highest level of object, since arbitrary viewing and windowing transformations are applied.

Any objects that are conveniently described in terms of smaller objects are good candidates for a hierarchical description. Mechanical parts, automobiles, VLSI and circuit designs, and documentation (chapters, sections, and paragraphs) all fall into this category.



6. Robot animation. Animated shaded-picture generation in real time allows rapid determination of robot manipulator paths through space. IRIS software can also check for collision detection to ensure that the planned trajectory does not conflict with other objects.

High-level access

Programmers usually deal with the system on a high level. For example, the IRIS internally uses 4-by-4 matrixes to represent all the perspective, windowing, rotation, translation, and scaling transformations, but the user commands look like "rotate (angle, axis)" or "translate (x-dist, y-dist, z-dist)" or "perspective (field-of-view, aspect-ratio, near-clipping-dist, far-clipping-dist)." Curves and surfaces are specified in terms of control points instead of different matrixes. Cursor drawing and undrawing is handled automatically, and the system is automatically initialized in a reasonable way.

The graphics software can be used in immediate or compiled mode. In immediate mode, the effects of a drawing command appear immediately on the screen. Programming is easy in this mode, but there are two disadvantages: there is no way to save object definitions so they can be called by other objects, and the overhead associated with a subroutine call makes the 68000 the performance bottleneck. Compiled display lists solve these problems. Each graphical object is stored internally as a display list that can be traversed rapidly enough by the 68000 processor to keep up with the geometry pipeline. For example, the car object shown in Fig. 5 might be compiled as:

```
makeobj(car);
[commands for drawing the body]
translate(0.0,7.0,0.0)          /* position first wheel */
callobj(wheel);          /* draw the first wheel */
[commands to draw the other wheels]
closeobj( );
```

Then, all the commands between MAKEOBJ and CLOSEOBJ become part of the object called CAR.

Compiled mode runs fastest, but an existing application can be converted to run on the IRIS very quickly using immediate-mode commands. Later, critical sections of the code can be converted to build display lists for better performance.

Editing objects

Display-list objects can be edited—entries can be added, deleted, or replaced, and an entire object can be rotated in real time by repeatedly editing the rotate command, changing the angle, and redrawing the object. If the IRIS work station is acting as a network terminal, only a few network transactions are necessary—open the object, edit the rotate, close the object, and draw it. Memory management for display-list code space is done automatically.

The IRIS graphics software supports standard graphics primitives—points, lines, polygons, rectangles, circles, arcs, cubic splines, and so on. Most can be filled or unfilled, in two or three dimensions, and in fixed- or floating-point terms.

The graphics pipeline can be put into a feedback mode, where it does not draw anything on the screen but instead sends the data back to the 68000 for further processing. This feature can be exploited in many ways. To select an object, its name must be recovered from the screen coordinates. To accomplish this, the system software modifies the windowing transformation, so that the view is a tiny window around the selection point, and then draws the object again in hit-testing mode. In this mode, an indication of what would have been drawn (including its name path) is fed back to the 68000. Area selection is the same, but with a larger window.

Selection is important for any application where the user needs to point interactively to objects on the screen. Drafting tools, circuit editors, simulators, and document-production tools all fall into this category. In many applications (robot-arm simulation or games, for instance), collision detection is needed. This is similar to selection but uses a 3-d box as a window around the object being tested for collisions.

Real-time support

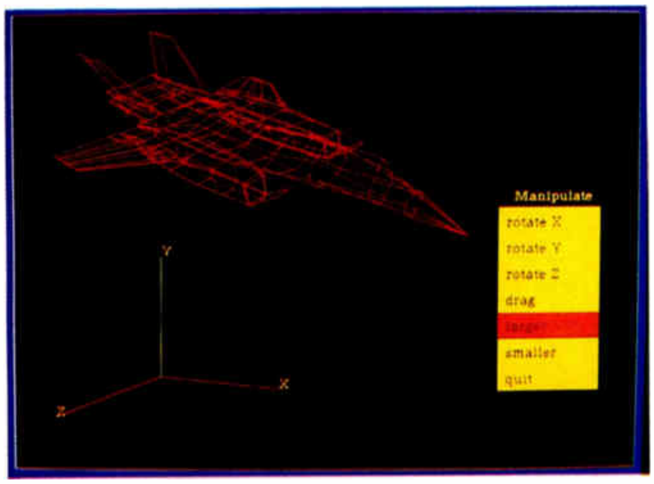
Real-time graphics is usually performed in a double-buffer mode. Here, while one image is being viewed, another is being constructed in a separate memory area. Special routines synchronize swapping between the two buffers to give uniform motion.

Any object transformed by modeling and viewing transformations may have a very small screen area or may even be outside the viewing area. Rather than wasting resources on drawing such objects, the system can automatically prune the display list of them. A bounding box for the object is passed through in feedback mode, and the rest of the object is skipped if the bounding box is either too small or out of range.

Delays associated with network communications make dragging objects on the screen from a remote host difficult. To avoid this problem, the MODIFY command ties a parameter of a drawing command to a linear function of an input-device value. For example, binding a mouse's X and Y positions to the X and Y parameters of a translation command will cause the translated object to be dragged by the movements of the mouse. By confining this type of interaction to the local work station, this technique avoids any network-associated delays.

The software allows either polled or queued input. An application reads the current state of a button (up or down), the current mouse coordinates, or the current setting of a dial. In addition, any combination of input devices may be queued in an event queue. If the state of a queued device changes (a button goes down or up, or the mouse or dial position changes by more than a certain amount), an event is added to the queue. The application then has a time-ordered list of events that can be processed asynchronously.

The keyboard is unencoded, so if desired, the user knows when a key goes down and when it comes up. The usual mode is to have the unencoded keyboard interpreted as if it were standard ASCII, and standard software routines do this. □



7. Aircraft design. Parametric curves and surfaces model exterior surfaces, and vector drawings display results to industrial designers. The ability to examine different portions of the object through rotation and zoom further helps designers to visualize mechanical parts.