Attachment A

Computers

Fast graphics use parallel techniques

Designers of computer graphics systems exploit parallel processing to provide the speed needed for interactive performance

The main strength of computer graphics is in its ability to exploit the massive parallel processing capacity of human vision—the capacity to perceive almost instantly complex visual patterns. However, until recently, the graphics displays have generally been prepared serially by the computer hardware. Because of the large volume of information to be computed in three-dimensional graphics, this has meant long processing times, a problem not critical for scientific and entertainment applications, but serious for the interactive, real-time systems used in computer-assisted design and in real-time simulations. In these systems, the main trend now is toward the development of parallel processing hardware that can dramatically decrease processing times.

Some large computer graphics systems using parallel processors are already operating, and many others are being developed. While parallel processing systems are currently limited to highly expensive simulators, VLSI designs under development could conceivably bring the cost of sophisticated computer graphics down to the price range of personal computers. Given the rapidly growing demand for interactive graphics and the suitability of VLSI circuits for parallel processing of increasing sophistication, it seems likely that this field will become one of the first to use VLSI technology in a big way. (See "The technologist's own 'super computer'," *Spectrum*, September 1980, pp. 48–51.)

At the same time, new methods are being developed to enter rapidly the large amounts of data in interactive graphics, and these methods, combined with faster hardware, will enlarge the already widespread applications of computer-assisted design.

Complex scenes broken into sections

When used in computer graphics, most parallel processing systems handle complex scenes or images by breaking them into sections. Each section, with its many similar calculations, can be generated in isolation from every other section. In this way special-purpose parallel processors, each handling only a part of the final image, can work far faster than serial processors.

The greatest premium on speed of image generation is in real-time simulations, where extremely complex images must be updated 30 times a second in response to the actions of the user. Such systems are used for the training of jet pilots and other military personnel. It is not surprising therefore that such simulation systems have been among the first computer graphics systems to use parallel processing.

An example is the Computrol system developed by Advanced Technology Systems of Fairlawn, N.J. The system can produce 30 full-color images a second, each using up to 30 000 edges or 10 000 light points. Many moving objects can be displayed, while fog, clouds, textures, and transparent objects can be modeled realistically (Fig. 1). The system permits fairly rapid generation of new data bases—a new airport can be programmed in a few man-days, for example. Computrol is to be used in the F-18 Weapons Tactics Trainer, being built by Hughes Aircraft for the Navy for operation in 1982. It will simulate maneuvering aircraft, terrain, gunfire, and missiles and will be equipped to train two pilots simultaneously. The pilots will be able to maneuver against each other in simulated missions. Similar systems are also used for training the crews of tanks, ships, and commercial aircraft.

The detailed architecture of the Computrol system is treated as confidential by Advanced Technology Systems—in fact, most current simulator designs are kept confidential, a practice that has hindered progress in this field. However, the general design of the system has been published, and it gives a good idea of the concepts applied.

The Computrol hardware consists of eight subsystems or blocks. An off-line terminal is used for creating the "world" within which the simulation operates and is connected with a minicomputer that controls the modeling process. A conventional CPU and its associated main memory contain the data base for the simulated world and control the movement of objects through it in three dimensions. Three specialized units are concerned with converting the three-dimensional world into a two-dimensional graphic representation on a CRT screen. The frame processor projects the three-dimensional world into the appropriate two-dimensional field of view and simultaneously converts the objects into edge-based descriptions—that is, the edges define the borders between differently colored patches. The raster processor calculates the intercepts between these edges and each scanline on the CRT. Finally, the pixel processor takes the intercepts of the visible edges, together with shading data, and generates the color and intensity of each pixel on the scanline.

The main parallel processing features are in specialized processors. The frame processor uses parallel arithmetic units to perform the calculations that transform world coordinates in the data base to eye coordinates centered on the apparent viewpoint of the trainee. Similarly, the raster processor has parallel circuits to calculate the intercepts, and each subsection of the pixel processor does identical calculations for a CRT subsection.

Applications to CAD

While the current application of parallel processing to computer graphics is mainly limited to simulations, the same techniques would be of great use in computer-assisted design if the hardware could be made sufficiently cheap. This has become an increasingly urgent necessity, since CAD systems are now using three-dimensional techniques that, with existing hardware, tend to slow processing radically and prevent easy interaction with the user.

Eric J. Lerner Contributing Editor

Until recently computer-assisted design has been applied mostly in the two-dimensional world of electronic design, but now it is expanding rapidly into three-dimensional applications in mechanical engineering and architecture. A typical commercial package, Synthavision, developed by the Mathematical Applications Group Inc. of Elmsford, N.Y., gives the user the ability to create any arbitrary solid, to manipulate it and view it from any angle, and to obtain its volume, weight, center of gravity, moments of inertia, and other geometrical characteristics.

General Electric has begun using Synthavision in the design of mechanical components, such as gear trains. Similar systems are being used by the Oak Ridge National Laboratory in Tennessee and the Lawrence Livermore National Laboratory in California in the design of complex magnets for controlled fusion experiments. Synthavision is also used in computer animation applications. In some cases, the computer-assisted design of components has been supplemented by computer movie simulation of what happens to the components under stress, thus automating both the design and test-evaluation procedures.

Synthavision and most similar systems use built-up complex solids by using a set of primitive shapes, such as cylinders, spheres, cubes, and wedges, as well as arbitrary shapes that can be parametrically specified. The solids appear on the CRT screen as if illuminated from a specified angle, and the user can adjust the reflective characteristic of the object's surface to simulate diffuse or specular reflectance.

Such techniques are also coming into use in construction engineering and landscaping. A group at the University of Massachusetts has developed a program called Ecosite that enables the user to construct land forms, to be used to reform surfaces that have been disrupted by strip coal mining. The system allows a designer to create landfill shapes that will blend naturally into the surrounding topography.

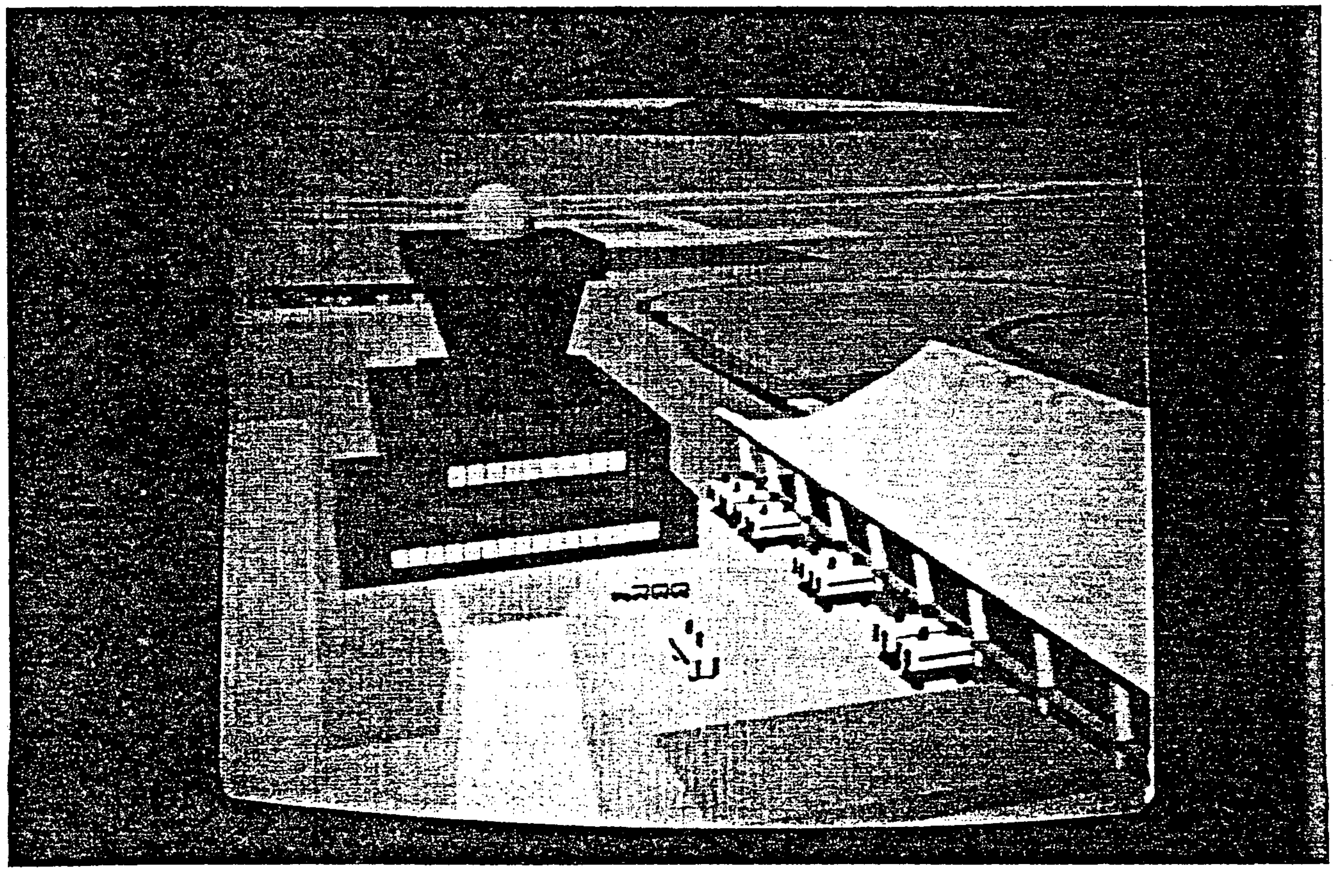
These existing systems, now implemented on conventional computers, would benefit enormously from the higher speed and interactive modes that would become available with the perfection of parallel processing systems now being designed.

A variety of methods of using parallel processors are under development to speed various aspects of computer graphics generation. Much of the work is focused on the hidden surface problem: the elimination of those portions of objects that are obscured by other objects. One particular software approach to this problem, the Z-buffer method, is especially suited to implementation by parallel processors, since the determination of what surface is visible is done independently for each pixel. The Z-buffer (described in "The computer graphics revolution," *Spectrum*, February 1981, pp. 35–39) is a buffer for each pixel of the image that allows only the nearest-object pixel to be entered.

One architecture proposed by Frederic Parke of Case Western Reserve University implements the Z-buffer method by splitting the image into regions and feeding the calculated surfaces in each region to separate parallel processors. Each parallel processor then determines the appropriate intensity and color values for each pixel in its area, loading them into the appropriate Z-buffers as it does so. Since only the closest pixels in each raster point will be allowed into the Z-buffer, the resulting image will automatically show only the surfaces that should be visible.

This system has a few limitations. Like all Z-buffer systems, it has difficulty in dealing with "aliasing"—the tendency of computer graphics systems to turn diagonal lines into staircases

[1] Complex scenes like this are generated by a flier training simulation system called Computrol, developed by Advanced Technology Systems. Computrol uses custom-designed parallel-processing hardware to produce 30 frames of three-dimensional simulation a second. The system costs approximately $2.5 million.



because of the finite dimensions of pixels. Also, the system becomes inefficient if all the objects are concentrated in a few regions, because most of the processors will then be idle.

A second architecture, developed by Henry Fuchs of the University of North Carolina, uses a central broadcast controller to distribute the data on each object to each processor. The data is broken up not by contiguous regions, but according to an interlace pattern, so that each processor handles pixels scattered over the whole of the image. This eliminates the problem of having some processors idle if the objects are concentrated in a certain area. However, this architecture turns out to be considerably slower, in general, than the split regions approach.

Mr. Parke at Case Western Reserve has proposed a hybrid architecture in which the input data is first split into a small number of regions and then distributed within each region, as in the broadcast approach. In this way the disadvantages of both approaches are minimized.

VLSI designs sought to cut costs

The best way to decrease the cost of parallel processing hardware is through VLSI. This approach is being pursued by James Clark and associates at Stanford University. He has designed, and is in the process of fabricating, a highly parallel VLSI computer graphics system consisting of a "geometry engine" and smart image memory.

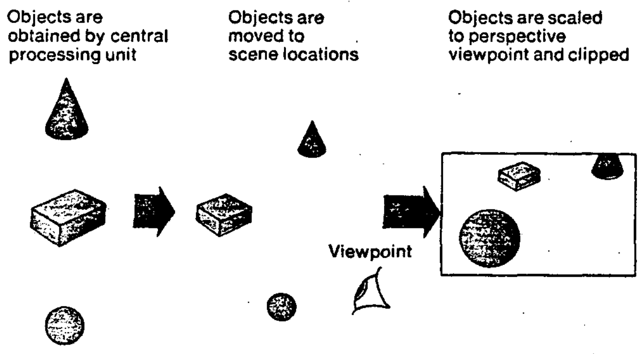
The geometry section of the processor consists of 12 identical geometry engine chips, each containing about 55 000 transistors. The processor performs the basic operations common to practically all computer graphics operations—transformations, clipping, and scaling of two- and three-dimensional polygons. It can perform about four million arithmetical operations a second, processing 900 polygons or 3500 edges every 1/30th of a second.

Each chip has a basically simple architecture, consisting of an arithmetic logic unit, three registers, and a stack, all working together to form four identical functional units. The 12-chip system consists of 1344 copies of a single bit-slice layout.

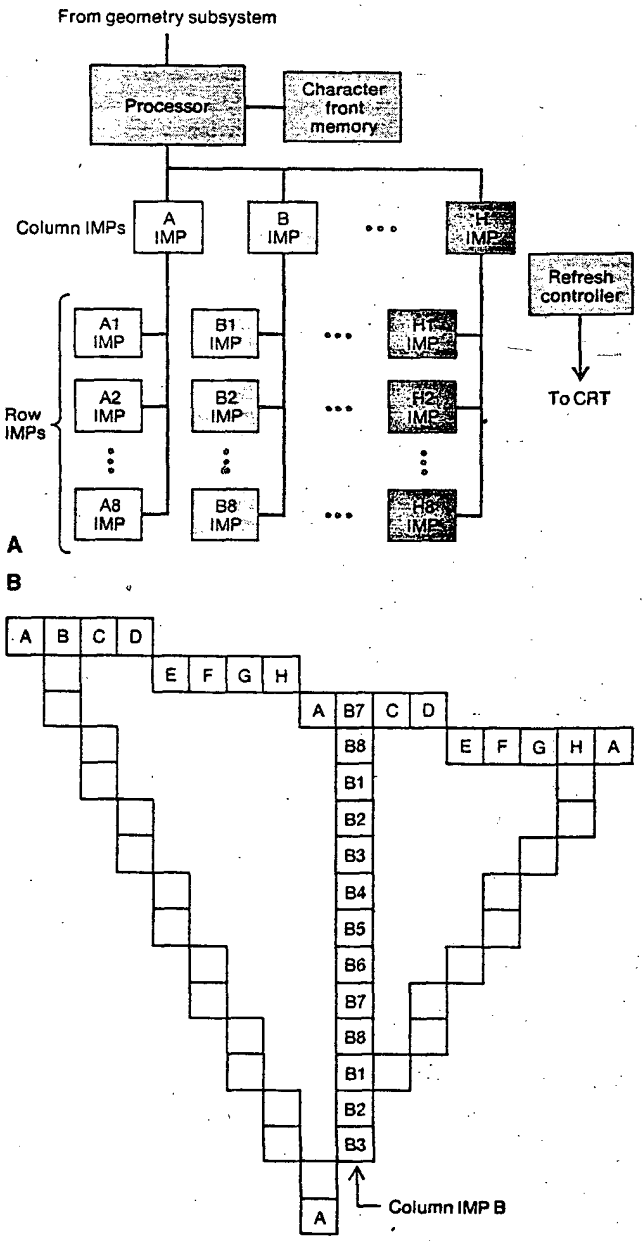
In operation, the geometry unit first receives the coordinates of polygons from a central processor and transforms them into the coordinates centered on the viewer. Four of the chips perform this operation by a combination of 4×4 matrix multiplications and vector dot products that accomplish the necessary rotations, translations, and projections to place the polygons in their locations in image space (Fig. 2). Since each chip has four identical subunits, 16 multiplications are being performed simultaneously in parallel.

The transformed polygons are then passed to the clipping subsystem, which determines what part of each polygon is within the field of view. For the three-dimensional case, the field of view is defined by five or six planes that bound the volume visible through the viewport of the image space. Each chip clips the polygons for one of the bounding planes and then passes it on to the next chip. Each geometry engine compares the coordinates of

[2] The geometry system developed by James Clark and colleagues at Stanford University uses parallel processing in a VLSI architecture to carry out procedures common to nearly all computer graphics. A scene consisting of lines, points, and polygons is first rotated and translated to correspond to the viewing position of the user. The scene is then "clipped" to eliminate those portions outside the viewer's field of view. Then the scene is scaled to fit within the viewing area of a CRT screen. The resulting polygonal coordinates are then passed to a smart image memory.


Objects are obtained by central processing unit
Objects are moved to scene locations
Objects are scaled to perspective viewpoint and clipped
Viewpoint

[3] The memory system for a 1024-by-1024-element display being developed at Stanford University contains eight column image memory processors (IMPs) and 64 row IMPs that convert sections of incoming polygons into alterations of specific pixels in the scan lines of the output display (A). Each row IMP is linked to one or more 16-kb memories, and the system has an output of 160 million bits per second. In operation (B), the edges of the sample triangle shown are scan-converted by the column IMPs and the interior pixels, by the row IMPs. The letters indicate which column IMP has converted each pixel, and the numbers indicate the row IMPs.



the end points of the polygon edges with the plane equation of the boundary surface. If both coordinates are outside the boundary, the edge is rejected; if both are inside, the edge is passed on to the next chip. If only one coordinate is inside—that is, the edge intersects the boundary—the chip finds the point of intersection. It does this by logarithmic search for the intersection point. Each of the four subunits of the chip computes one coordinate of the midpoint of the edge and determines if that midpoint falls outside or inside the boundary plane. If it falls outside, then the midpoint of the line connecting the inside end point with the original midpoint is then calculated and tested and the cycle repeated until the desired precision of the intersection point is achieved.

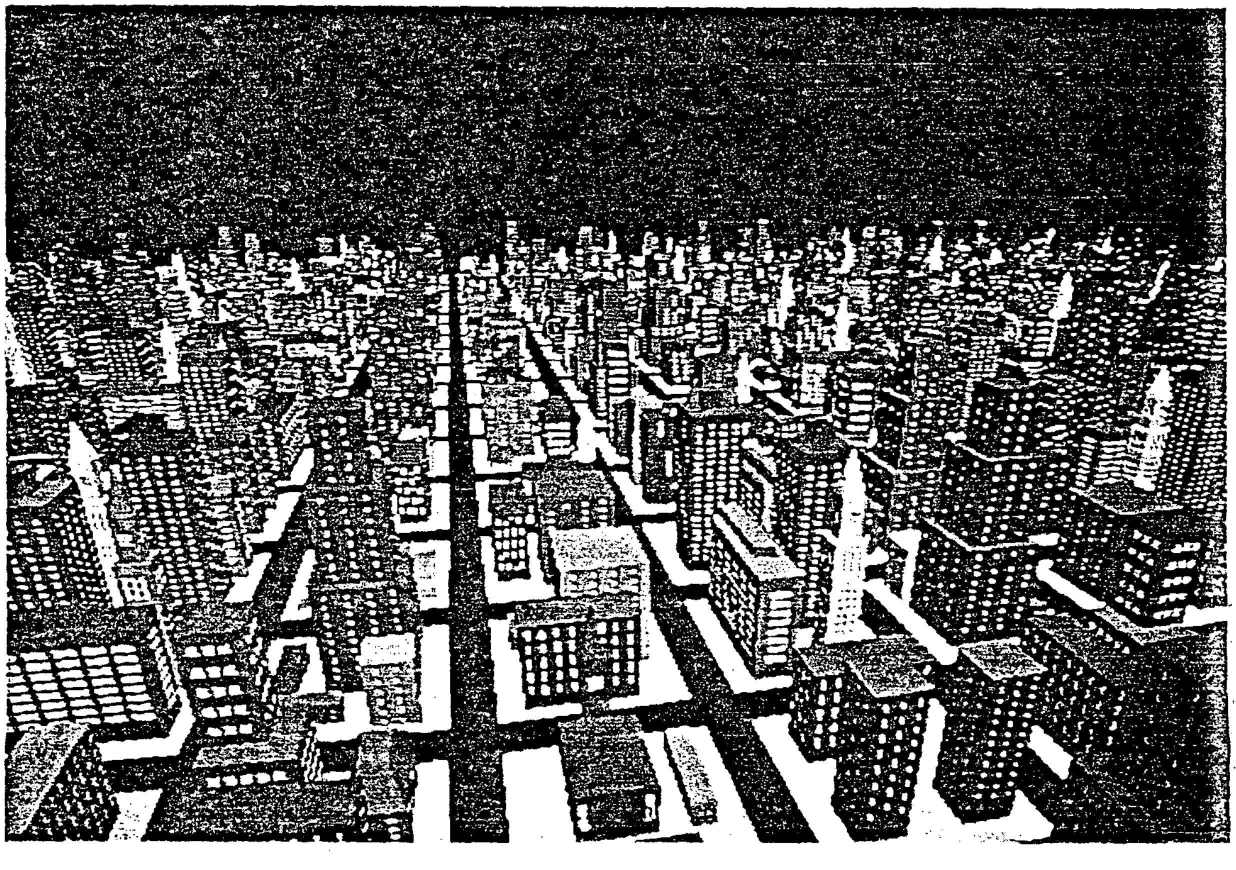
Finally, once the clipping operation is completed, the dimensions of the polygons are scaled to the size of the image viewport —that is, the farther away the objects are, the more the boundary area must be scaled down to the dimensions of the viewing screen for correct perspective. Two chips—one for the *x*, *y* scaling and the other for the *z*, or depth, scaling—perform the division of the coordinates simultaneously and feed the finished results to the smart image memory.

The second part of the computer graphics design is the image memory, a high-performance system for scan conversion. This is the process of determining which pixels on the screen correspond to the calculated images. The image memory is composed of a parent processor, an array of image memory processors, or IMPs, and a refresh controller. The IMP array consists of eight column IMPs and 64 row IMPs, each of the latter being responsible for 16 000 pixels of a 1024 × 1024 array (Fig. 3). The IMPs are connected with the CRT pixels through a two-level hierarchical busing structure, with interleaved processors along each bus. The interleaving is such that for any 8 × 8 array of pixels on the screen, each pixel is controlled by one processor. Thus, as in the Fuch's broadcast scheme at the University of North Carolina, each processor controls pixels scattered across the screen rather than concentrated in a single contiguous area. Each of the IMPs is a single LSI chip that contains two main functions, a linear difference engine and a memory interface processor.

In operation, the geometry engine passes the characteristics and locations of the elementary polygons to the parent processor, a standard microprogrammed chip. The parent processor prepares the polygons for scan conversion and broadcasts the resulting data to all of the column IMPs. The polygons are represented by the coordinates of their vertices. Each column IMP (C-IMP) uses its linear difference engine function to calculate what part of the line falls within the column controlled by its row IMPs and sends this information to the R-IMPs that cover that part of the column. The R-IMPs, in turn, use their linear difference engines to calculate which individual pixels should be altered and send this information to the memory interface processor for storing. At regular intervals, the refresh controller sends a signal to each of the memory interface processors to obtain updates of the new pixel values and uses them to form the new image on the CRT.

The edges of the polygon are thus converted by the C-IMPs to their new values, while the interior is converted by the R-IMPs. This architecture is being modified to implement the shading and coloring algorithms of most polygon systems. The modified architecture will obtain the shading values for the interior of the polygon by interpolating between the values for the edges. In ad-

[4] Procedural modeling employs a set of rules to generate more complex objects from simpler ones and combines both specific data entry and computer-generated repetitions. An example graphic, developed by Charles Csuri and associates at Ohio State University, illustrates how entire city blocks might appear.



dition, the system can be extended to the sort of broadcast *Z*-buffer hidden surface remover described by Dr. Fuchs.

The Clark architecture shows a conceptual similarity to that of the system developed by Advanced Technology Systems, with the column IMPs performing similar functions to the raster processors and the row IMPs analogous to the pixel processors.

Alternative ideas explored

A number of alternative ideas being developed use parallel processing to speed certain special functions useful in computer graphics. One example is a two-dimensional shift register designed by George Chaikin of the Goddard Space Flight Center office in New York City and Carl Weiman of General Electric.

This device is intended to speed the calculations involved in two common graphic transformations: scale changes, or "zooming," and rotation. In a conventional system, these transformations require many individual calculations to change the coordinates of each pixel in the image. Instead, Mr. Weiman and Mr. Chaikin have proposed using a hard-wired polar logarithmic transformation data channel to convert rotation and scale transformations into translation motions on a shift register. The data channel would connect pixels in the image plane arranged in a polar logarithmic pattern to those in the shift register arranged in a rectangular pattern.

In other words, a circle in the image plane is always mapped into a vertical line in the register, and a radius in the image plane is mapped into a horizontal line in the register. Rotation of the image is achieved when each pixel register is commanded to shift its content to its neighbor above or below. Scale conversion is achieved when each register is commanded to shift its content to its neighbor on the left or right. A single command can thus perform the work of many coordinate calculations.

While parallel processing will markedly speed computer graphics processing, the most efficient use of such savings in time will necessitate faster methods of data entry. Complex design problems in three dimensions often produce difficult problems simply in getting the design concepts into the computer in the first place. A number of techniques for data input and structuring are both easing data entry and simultaneously making some processing tasks more efficient.

Two of the most important techniques are the related approaches of procedural and hierarchical modeling. In procedural modeling, a set of laws is used to generate more complex objects from simpler ones. The hierarchical approach breaks down complex objects into simpler components or simpler representations with less detail.

Using procedural techniques, which combine both specific data entry and computer-generated repetitions where necessary, Charles Csuri and co-workers at Ohio State University have developed a system for the design of buildings by use of standardized components. Computer graphics can then be used to "construct" entire city blocks of a variety of such standardized buildings (Fig. 4). The results give one a realistic view of how the buildings would look in a city. An entire downtown area of two thousand buildings was designed with this system in less than two weeks. Such modularized techniques may have important applications in West Europe, where modularized building is far more common than in the United States. It was, in fact, such techniques that made possible construction of the elaborate data bases in systems such as Computrol.

Hierarchical techniques take into account the fact that as an object becomes more distant, less detail appears, and thus it is a waste of computing power to calculate distant objects to the same precision as nearby ones. In a hierarchical data base, a single object may be represented by a number of representatives, each having greater detail than the previous ones and the more detailed ones being used for when the object is closer to the viewer. Hierarchical representations also speed such processes as hidden surface elimination, since if it is found that an entire object will be obscured in a given image, elaboration of that object to finer degrees of detail will be obviated. Thus, once it is found that one building lies entirely behind another, the exterior windows, doors, and so on in the building will not be calculated at all.

Degree of detail required can be varied according to circumstances. Thus, moving objects can be calculated in less detail than motionless ones, or whole scenes can be less rigorously imaged if the field of view is moving rapidly. Another advantage of hierarchical data sets is in the reduction of memory storage requirements for graphics processors. A working set of images can be formed, consisting only of the images that have, in recent frames of the sequence, been resolvable. This working set can be kept in a fast access memory and only slowly changed or replenished as the field of view changes.

A third technique of importance in efficient data entry is the use of piecewise continuous surfaces, or splines, for defining continuously curved surfaces. In many CAD applications, the manipulation of sculptured surfaces, such as ship hulls, is extremely important, yet with even the fastest processing capabilities, point-by-point entry of such curves is very time-consuming. Spline surfaces simplify the creation of such sculptured surfaces on a computer graphics system.

A B-spline, a widely used type, consists of a network of points, each having associated with it a set of vectors that define the directions of curvature of the surface at the point. The combination of points and vectors can be used to produce smoothly curving surfaces that can have almost arbitrary characteristics. One can modify the surfaces by selecting a given point and either moving it or changing the associated vectors.

The combination of faster input algorithms and the increased speed and decreased cost of parallel-processing hardware will rapidly make very powerful CAD graphics systems widely available. In the next few years, such systems will be becoming a standard tool in engineering.

For further reading

A description of the Computrol system is given by Sam Ranjbaran and Ron Swallow in "Graphics of Complex Images in Training," *Second IEEE Workshop in Picture Data Description and Management,* 1980.

Frederic Parke describes two approaches to parallel processing in "Simulation and Expected Performance Analysis of Multiple Processor Z-Buffer Systems," *SIGGRAPH 1980*, pp. 48–53.

James Clark's designs for VLSI computer graphics systems are outlined in "A VLSI Geometry Processor for Graphics," *Computer*, July 1980, pp. 59–69, and also in "Distributed Processing in a High Performance Smart Image Memory," *Lambda*, Fourth Quarter 1980, pp. 40–45.

Hierarchical data organization is discussed by Steven Rubin and Turner Whitted in "A 3-Dimensional Representation for Fast Rendering of Complex Scenes," *SIGGRAPH 1980*, pp. 110–117.

Robert Marshall *et al*, outline a system for procedural data generation in "Procedure Models for Generating Three-Dimensional Terrain," *SIGGRAPH 1980*, pp. 154–159. The use of B-splines is examined by David Rogers and Steven Satterfield in "B-Spline Surfaces for Ship Hill Design," *SIGGRAPH 1980*, pp. 211–217. ◆