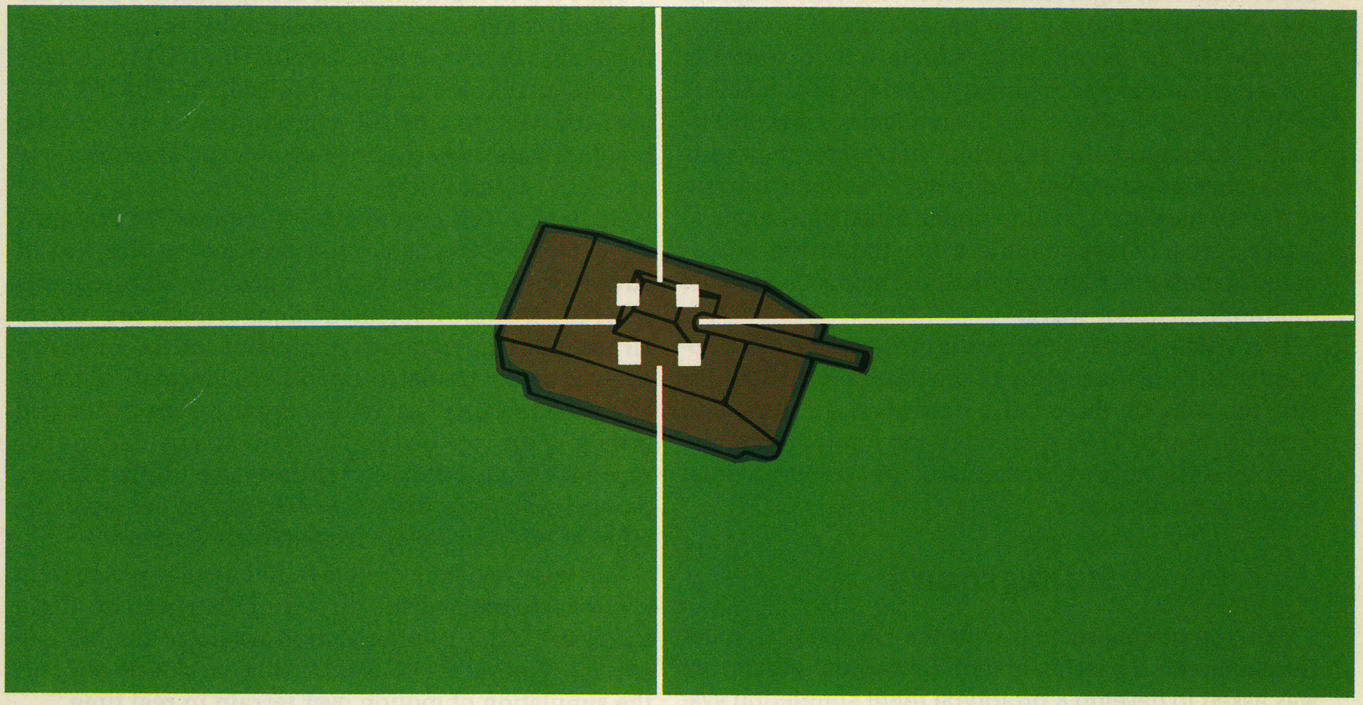
Flight Simulators for Under $100,000

Michael J. Zyda, Robert B. McGhee, Ron S. Ross, Douglas B. Smith, and Dale G. Streyle

Naval Postgraduate School

Monterey, California



Flight simulators have played an important role in the training of both military and commercial pilots during the last two decades. Although flight simulation systems in general have proved cost effective, the large capital investment required for initial development and implementation has restricted their use to the military and other large organizations. To demonstrate the feasibility and practicability of designing and building "low-cost" flight simulators, a prototype system was developed to model the performance of a new US Army remotely piloted missile system. The flight simulator displays a dynamic, three-dimensional, out-the-window view of the terrain in real time while responding to operator input from the command and control system. The total development cost of the flight simulator is an order of magnitude less than that of the sophisticated systems currently in use.

Recent technological advances in computer image generation have opened up exciting new horizons for real-time computer graphics applications. Perhaps the single most important application using real-time computer image generation is visual training simulation. In the most general sense, visual training simulation is the presentation of 3D scenes to an observer to facilitate repetitive practice of various tasks in a particular application area. The classic example is illustrated by the development of computer-generated visual systems for flight simulation and pilot training. A detailed and comprehensive review of flight simulation devices, as well as the related benefits of those devices, is provided by Schachter.[1]

The decision to design and build a computer-based flight simulator is driven by pure economics. The cost of providing extensive, long-term flight training to both military and commercial pilots is prohibitive, given the finite supply of natural resources for fuel and the shrinking budgets of most organizations. In fact, as illustrated during the 1973 oil embargo, the shortage of aircraft fuel and the resulting high cost threatened the readiness of US military forces by imposing drastic reductions in flight training time. Therefore, it is no surprise that alternative methods, such as sophisticated flight simulation systems, emerged to supplement the actual "in-flight" training time of pilots. While the overall cost effectiveness of flight simulators has been demonstrated, the extremely high cost of development has limited their use to the military and other large organizations.[2]

Given the history of computer image generation and the evolution of flight simulation devices, the Graphics and Video Laboratory of the Department of Computer Science at the Naval Postgraduate School recently embarked on a research project to test the practicability of developing "low-cost" flight simulators using off-the-shelf, commercially available hardware. For this research, a low-cost flight simulator is defined as a system with hardware costing under $100,000. This is at least an order of magnitude less than the current cost of many operational flight simulation systems that top the multimillion-dollar mark.

This article presents the results of the design, development, and implementation of the flight simulation system, focusing on the relative hardware, software, and database issues. The capabilities and limitations of the prototype system are also discussed, as are the potential uses of such devices.

Background

Working in conjunction with the US Army Combat Developments Experimentation Center, our laboratory was asked to develop a prototype flight simulation system to model the performance of a new Army weapon system, the Fiber-Optically Guided Missile, or FOG-M. The FOG-M is a remotely piloted missile system with an on-board television camera capable of transmitting live battlefield pictures to an operator's console on the ground. The operator can view the terrain as if in flight using the pan, tilt, and zoom mechanisms of the camera and thereby locate possible targets for the missile. The flight dynamics of the missile can also be controlled by the operator through a joystick and other controls. After a target is located by the operator on the ground, it is "locked in," with subsequent control passing to the automatic tracking system. The operator can visually follow the flight of the missile until impact.

The first release of the prototype flight simulator displays a dynamic, 3D, out-the-window view (as seen from the operator's console) of the terrain in real time. An interactive user interface and two-dimensional contour-map display allow the operator to input initial launch position and estimated target location during the prelaunch phase of the flight simulation. After the final missile-launch countdown, a 3D view of the selected terrain is displayed. The terrain data is obtained from a special digital database product prepared by the Defense Mapping Agency, and the realism of the terrain is enhanced through the use of an illumination model. The operator can interactively control the FOG-M television camera by using the mouse for pan, zoom, and tilt adjustments within specified parameters. The operator also has interactive missile control for in-flight changes in direction, speed, and elevation.

Hardware implementation

The FOG-M flight simulator is implemented using off-the-shelf, commercially available hardware from Silicon Graphics. The system is built around the Iris 3120, a high-performance color graphics workstation. The Iris 3120 uses the Motorola 68020 microprocessor and special graphics hardware. Figure 1 shows a general overview of the Iris architecture.

On the Iris, the graphics speed required for real-time flight simulation displays is realized by the use of advanced VLSI technology. Proprietary VLSI circuits provide hardware implementations of many fu lamental graphics operations. A pipeline of 12 ɛ eometry engines performs matrix operations for rotati ɔn, translation, and scaling—as well as clipping, perspective, and orthographic viewing—at 80,000 matrix operations per second. The customized chip design results in a highly parallel architecture that is well suited for the rapid creation of terrain displays for flight simulation.[3]

The Iris provides a double-buffer display system that is crucial to the realistic display of motion in the flight simulator.[3] The z-buffer option available for hidden-surface removal is not used, as it is too slow to support the simulation of motion over terrain in real time.

The total list price for the hardware used to support the project is under $75,000.

Database implementation

The source of data used for the representation of terrain in the FOG-M simulation is a digital database provided by the Defense Mapping Agency (DMA). The digital terrain database is an array of data points for Fort Hunter Liggett, California, and vicinity. Each data point consists of 16 bits of information. The first 13 bits specify one of 8192 possible elevation values. The last three bits designate one of eight possible surface-covering categories for the height of the vegetation at that particular point, that is, less than 1 meter, 1 to 4 meters, 4 to 8 meters, etc. The data points are sampled at 12.5-meter intervals, producing a terrain database of much higher resolution than the standard Level I or Level II Digital Terrain Elevation Databases provided by the DMA. The

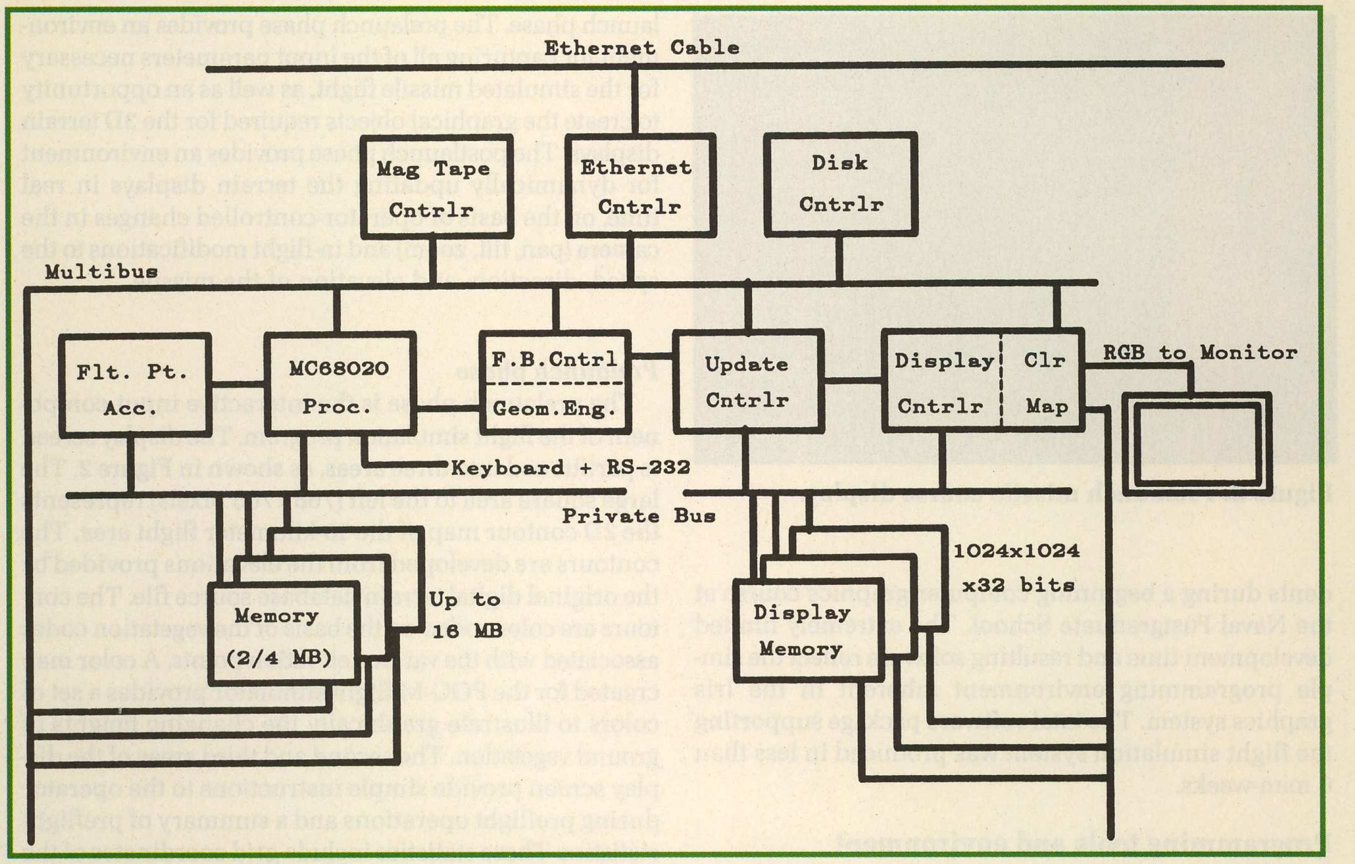


Figure 1. Block diagram of the Silicon Graphics Iris workstation.

exact geographic location of each sample point is determined implicitly by the sequential positions of the data points in the terrain file.

The terrain database covers a region 35×36 kilometers. Each 1-kilometer grid square contains 80 data points along the latitudinal axis and 80 along the longitudinal axis, for a total of 6400 sample elevations/vegetations per grid square. The data points are arranged sequentially by column within each 1-kilometer grid square, starting from the lower southwest corner. The patterns are repeated throughout each grid square in the database.[4]

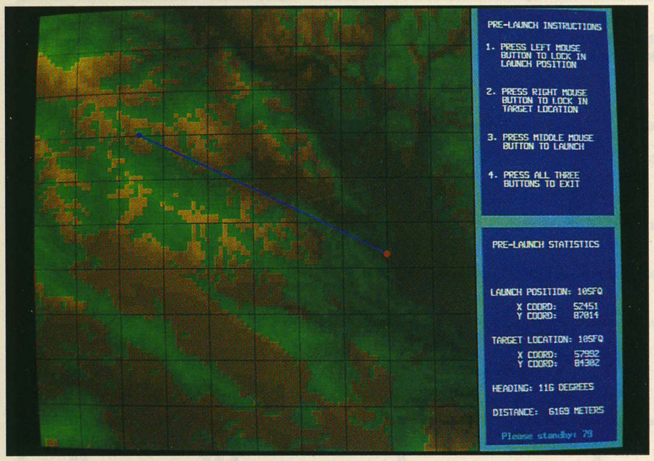
The entire 16M-byte digital terrain database resides on a DEC VAX 11/785. An interactive database program written in C allows the user to create a subset of the master terrain database by specifying a region where the FOG-M is designated to fly for a particular mission. The user-supplied input parameters consist of (1) the size of the test area (in square kilometers), (2) the Universal Transverse Mercator (UTM) grid coordinates of the lower southwest corner of the test area, and (3) the sample frequency to extract from the database, that is, 12.5- or 100-meter grid spacing.

The current implementation of the FOG-M flight simulator assumes a 10-square-kilometer potential flight area and uses 100-meter samples for the digital data points. The 10-kilometer assumption is based on the average distance projected for FOG-M missions, given the missile's minimum and maximum ranges. The 100-meter sampling is sufficient to capture the realistic features of the terrain without sacrificing the speed required for real-time display. Upon completion of the database program, a binary file is produced that serves as the input terrain source file for the flight simulation system. The binary file is subsequently transferred to the Iris disk storage area in preparation for running the FOG-M simulation.

Currently, the input terrain source file is created off line. The operator cannot change the flight area of the missile directly from the Iris workstation. However, a recently completed data communications interface using Ethernet allows direct user access to the master digital terrain database from the Iris workstation.[5] This feature promotes maximum flexibility in allowing rapid changes to flight areas when multiple runs of the simulator using different test areas are required.

Software implementation

The FOG-M flight simulator represents the design, analysis, and programming efforts of three graduate stu-



```
PRE-LAUNCH INSTRUCTIONS

1. PRESS LEFT MOUSE
   BUTTON TO LOCK IN
   LAUNCH POSITION

2. PRESS RIGHT MOUSE
   BUTTON TO LOCK IN
   TARGET LOCATION

3. PRESS MIDDLE MOUSE
   BUTTON TO LAUNCH

4. PRESS ALL THREE
   BUTTONS TO EXIT

PRE-LAUNCH STATISTICS

LAUNCH POSITION: 10SFQ
      X COORD:   52451
      Y COORD:   87014

TARGET LOCATION: 10SFQ
      X COORD:   57992
      Y COORD:   84302

HEADING: 116 DEGREES

DISTANCE:  6169 METERS

Please standby: 79
```

Figure 2. Prelaunch missile course display.

dents during a beginning computer graphics course at the Naval Postgraduate School. The extremely limited development time and resulting software reflect the simple programming environment inherent in the Iris graphics system. The total software package supporting the flight simulation system was produced in less than 6 man-weeks.

Programming tools and environment

The flight simulator is implemented using a simple yet powerful program development environment. This environment, which includes an optimized C-language compiler and a complete graphics library, is available on the Iris workstation through use of the AT&T Unix System V operating system. The graphics library provides a full set of high- and low-level graphics routines for system development in the areas of fast polygon fill, hidden-surface removal, and fast pixel access.[3]

Implementation strategy

The flight simulator was developed in a top-down, modular fashion, taking full advantage of an object-based segmentation technique available on the Iris. This methodology employs the concept of "graphical objects," or sequences of graphics commands, that are used repeatedly within the applications program. The commands are placed into an object and compiled into a display list as defined. Calls to the previously defined objects are interpreted directly without further compilation. This technique greatly reduces the overhead associated with continuous translation of graphics commands.[6] Coupled with the hardware mechanism of double buffering, this modular approach results in a marked increase in the speed of display for real-time flight simulation. A description of the graphical objects and the methodology used to achieve 3D motion through the terrain is provided below.

The software system supporting the FOG-M flight simulator is divided into a prelaunch phase and a postlaunch phase. The prelaunch phase provides an environment for capturing all of the input parameters necessary for the simulated missile flight, as well as an opportunity to create the graphical objects required for the 3D terrain displays. The postlaunch phase provides an environment for dynamically updating the terrain displays in real time, on the basis of operator-controlled changes in the camera (pan, tilt, zoom) and in-flight modifications to the speed, direction, and elevation of the missile.

Prelaunch phase

The prelaunch phase is the interactive input component of the flight simulation program. The display screen is partitioned into three areas, as shown in Figure 2. The large square area to the left (768×768 pixels) represents the 2D contour map of the 10-kilometer flight area. The contours are developed from the elevations provided by the original digital terrain database source file. The contours are color coded on the basis of the vegetation codes associated with the various elevation points. A color map created for the FOG-M flight simulator provides a set of colors to illustrate graphically the changing heights of ground vegetation. The second and third areas of the display screen provide simple instructions to the operator during preflight operations and a summary of preflight statistics. These statistics include grid coordinates of the launch position and target location, direction of flight, and distance to target.

During the prelaunch phase, the operator can move the cursor freely around the contour map using the mouse. The current UTM grid coordinates are displayed continuously in the appropriate area of the statistics box. When a launch site is selected, the coordinates are "locked in" by pressing one of the three available mouse buttons. Next, the operator moves the cursor to the location of a possible target. A rubberband line shows the potential flight paths of the missile from the launch site. The final destination is selected as above, with the respective coordinates appearing in the target location section of the statistics box. From this initial information, the missile heading and distance to target are computed and displayed for the operator. Changes to either the launch site or target location are possible before launching the missile and entering the postlaunch phase. Once the operator selects the launch and target positions and all preflight statistics are computed, a mouse button is pressed to initiate the countdown. Then, during the countdown, the graphical terrain objects are created.

To create individual graphical objects for the terrain model, the 10-kilometer flight area is partitioned into 100 1-kilometer grid squares. A typical graphical object (1 kilometer square) consists of triangular polygons constructed by connecting every set of three data points within the given array of elevations from the subset of the digital terrain database. The object plane is bounded by the x- and z-axes, with the *y*-axis representing eleva-

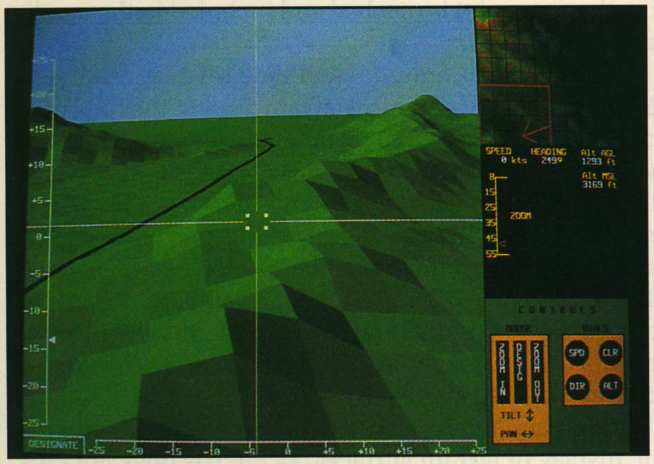


Figure 3. In-flight view of ridgeline and road.

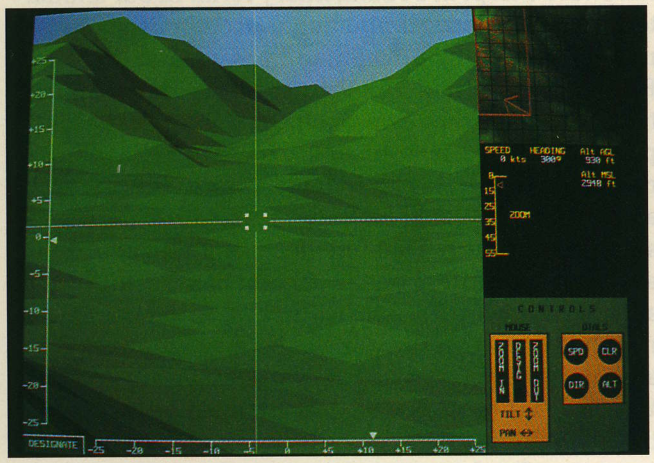


Figure 4. In-flight view of mountain pass.

tions at particular data points. The 1-kilometer graphical object contains a total of 200 polygons, the result of building the triangles throughout the array of data points. Two triangles are created at a time, constituting one cell of the total 100 cells in the object.

A point underneath each triangle is computed and sent to an illumination routine for color selection. The illumination model uses a single point source in computing the respective colors of the polygons. The calculation is based on the application of Lambert's cosine law to compute the intensity of the reflected light from the angle of illumination.[7] The colors returned for the two triangles are averaged, and a polygon-fill routine from the graphics library completes the construction. A checkerboard effect is created for the terrain surface by changing the color ramp for alternating grid squares. This effect creates an artificial texture on the surface to emphasize the variability of the terrain.

Postlaunch phase

The postlaunch phase provides successive real-time terrain displays to the operator as the missile approaches

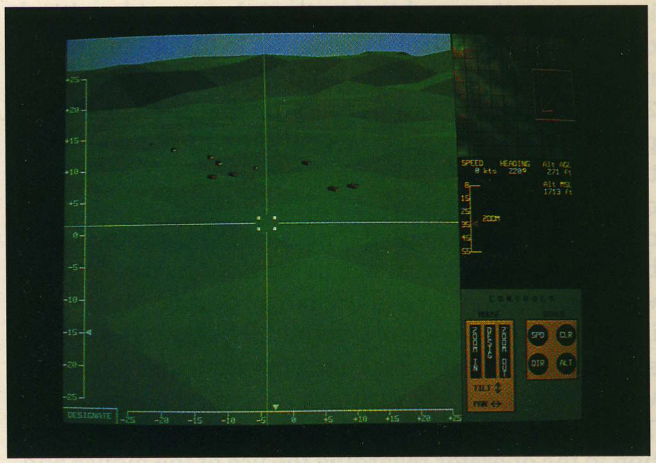


Figure 5. Distant view of vehicle convoy.

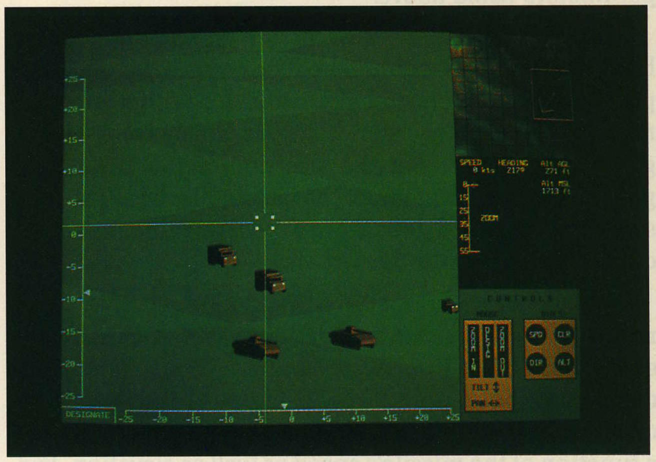


Figure 6. Medium view of part of vehicle convoy.

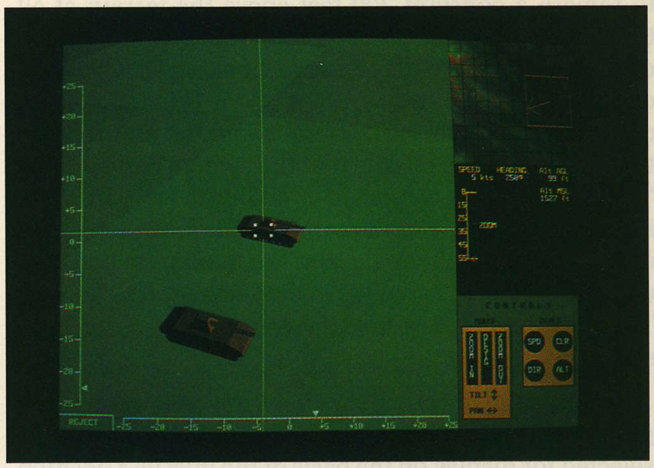


Figure 7. Close view of target vehicle.

its target. The display screen is partitioned into four areas, as shown in Figures 3-7. In this case the large square area to the left represents the 3D, out-the-window view of the current terrain area visible to the camera in the nose of the missile. The right-hand portion of the display screen consists of three control boxes (navigation, instrumentation, and instruction) that provide critical command and control information to the operator. The navigation control box (top) is a reduced version of the 2D contour map that appears during the prelaunch phase. A red rectangular box outlines the region within the 10-kilometer flight area currently displayed in the out-the-window view. A red arrow within the box indicates the missile heading. The arrow and containment box are continually redrawn as the missile direction and view area (camera angle) change during flight. The instrumentation control box (middle) displays in-flight statistics for missile heading, elevation, and speed, as well as camera-angle readings for pan, tilt, and zoom. The instruction control box (bottom) provides information to the operator regarding the use of the mouse and button box/dials to modify the flight and camera parameters listed above.

The realistic display of 3D terrain surfaces and simulated motion is accomplished through a perspective projection and a special algorithm for computing the number and display order of the previously constructed graphical objects. Several Iris commands from the graphics library implement the terrain displays. The Perspective command defines an aspect ratio, near and far clipping planes, and field of view.

Working in conjunction with the Perspective command, the Lookat command defines a viewpoint, a reference point, and a twist angle.[6] The viewpoint designates the *x*, *y*, and *z* coordinates of the missile or camera position. The coordinates of the viewpoint are a function of the missile's speed as well as elevation. For the FOG-M flight simulation, the missile is programmed to remain a constant 200 feet above the terrain at all times. The reference point specifies the *x*, *y*, and *z* coordinates of the place the camera is looking at. The coordinates of the reference point are a function of the direction in which the camera is looking and the distance ahead of the missile that the camera can see. The "look" direction is computed by adding the missile heading to the pan angle of the camera. The look distance is a function of the tilt angle, and the maximum distance of 10 kilometers is achieved when the camera is level. The twist angle is assumed to be 0 degrees for the flight simulation.

To minimize the number of polygons sent through the graphics pipeline, a candidate set of graphical objects is selected for display on the basis of the missile's position and where the camera is looking. Initially, the grid square containing the look position of the camera, plus two grid squares surrounding the center square in each direction, is included in the candidate set. The result is a minimum candidate set of 25 graphical objects (5×5 out of the 10×10 total). If the current viewpoint (position of the missile) is not in the minimum candidate set, the set is expanded in the direction of the viewpoint. This expansion continues until the graphical object containing the viewpoint is reached and included in the candidate display set.

While the minimum candidate display set is a 5×5-square set of graphical terrain objects, the final candidate set may in fact be rectangular, depending on the viewpoint or position of the missile. In the worst case, the candidate display set contains all 100 graphical terrain objects. In all cases, however, the viewpoint is constrained by the boundaries of the 10-kilometer flight area. The appearance of motion is achieved by successively computing viewpoints and reference points and displaying the perspective projections of the new candidate sets of graphical terrain objects.

The order of polygon display is critical for the realistic display of terrain surfaces, as the Iris does not currently have real-time hidden-surface-elimination hardware except in a rudimentary form. In our simulator, hidden-surface elimination is accomplished in the main processor by a real-time implementation of the Painter's algorithm.[7] The Painter's algorithm amounts to nothing more than drawing the picture in the farthest-polygon-to-closest-polygon order. For the terrain, the correct polygon ordering for hidden-surface elimination is an easily computable function of the missile's line of sight (see Figure 8). That function is nothing more than a scan-line algorithm on the 2D grid of previously constructed triangles. For vehicles driving on the terrain, we merely specify that the vehicles be drawn immediately after the grid cell on which they reside has been drawn. Vehicles that cross boundaries between grid cells are drawn multiple times. The vehicles used in the simulator are stylized in 20 polygons or less. The vehicle hidden-surface-elimination problem is resolved through a combination of backface polygon removal, which the Iris has in hardware, and simple sorting, using the vehicle's direction and the missile's line-of-sight information. Our research group is currently investigating the applicability of Fuchs' binary space partitioning algorithm to our hidden-surface-elimination problem.[8]

System capabilities and limitations

The FOG-M flight simulator, in its current configuration, possesses significant capabilities as well as some noteworthy limitations. On the positive side, the flight simulation system draws between 1500 and 2000 polygons per frame at three to four frames per second. A frame consists of the graphical terrain objects in the candidate display set. The number of polygons created increases in proportion to the number of graphical objects in the candidate set. This capability is due to the geometric transformation and polygon-fill hardware of the Iris as well as the segmentation procedure the system uses for graphical objects. As a result, there is complete freedom of movement in semi-real time within the 3D display.

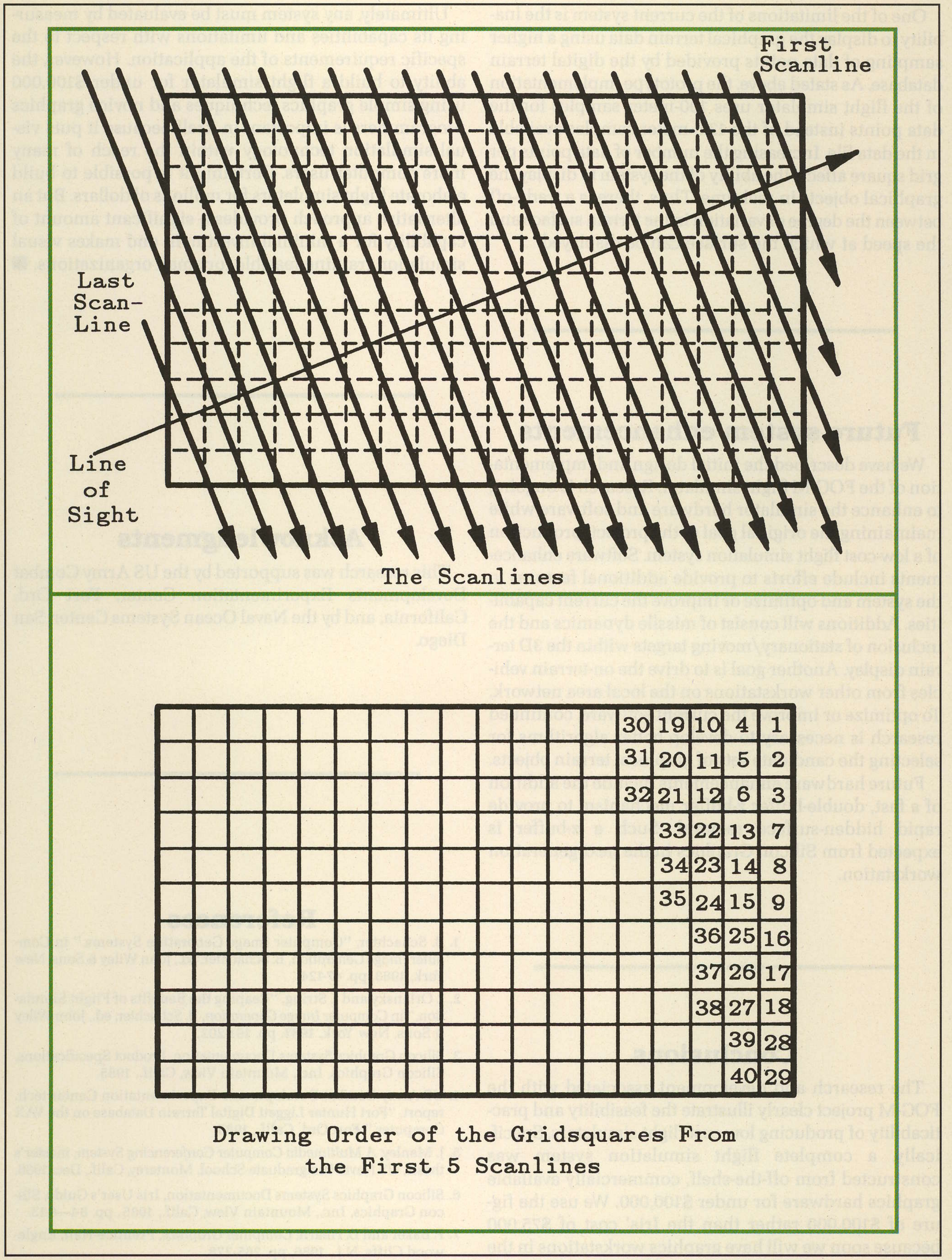


Figure 8. The scan-line hidden-surface algorithm.

One of the limitations of the current system is the inability to display the graphical terrain data using a higher sampling of data points provided by the digital terrain database. As stated above, the prototype implementation of the flight simulator uses 100-meter samples for the data points instead of the 12.5-meter samples available in the data file. Increasing the number of data points per grid square affects the ability of the system to display the graphical objects in real time. Thus, there is a trade-off between the degree of variation in the terrain surface and the speed at which the surface can be displayed.

Future system enhancements

We have described the initial design and implementation of the FOG-M flight simulator. Research is ongoing to enhance the simulator hardware and software while maintaining the original goal of the project: production of a low-cost flight simulation system. Software enhancements include efforts to provide additional features to the system and optimize or improve the current capabilities. Additions will consist of missile dynamics and the inclusion of stationary/moving targets within the 3D terrain display. Another goal is to drive the on-terrain vehicles from other workstations on the local area network. To optimize or improve the current software, continued research is necessary to develop better algorithms for selecting the candidate sets of graphical terrain objects.

Future hardware enhancements include the addition of a fast, double-buffer z-buffer mechanism to provide rapid hidden-surface removal. Such a z-buffer is expected from Silicon Graphics in the next-generation workstation.

Conclusions

The research and development associated with the FOG-M project clearly illustrate the feasibility and practicability of producing low-cost flight simulators. Specifically, a complete flight simulation system was constructed from off-the-shelf, commercially available graphics hardware for under $100,000. We use the figure of $100,000 rather than the Iris' cost of $75,000 because soon we will have graphics workstations in the higher price range with polygon-fill rates some 6 to 35 times faster.

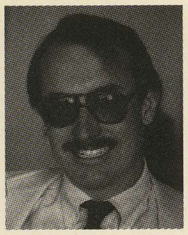
Ultimately, any system must be evaluated by measuring its capabilities and limitations with respect to the specific requirements of the application. However, the ability to build a flight simulator for under $100,000 using simple graphics techniques and novice graphics programmers is important in itself because it puts visual simulation technology within the reach of many more computer users. Certainly it is possible to build elaborate flight simulators for millions of dollars. But an alternative approach provides a significant amount of capability for a minimal investment and makes visual simulation training feasible for small organizations. ■

Acknowledgments

This research was supported by the US Army Combat Developments Experimentation Center, Fort Ord, California, and by the Naval Ocean Systems Center, San Diego.

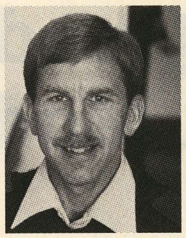
References

1. B. Schachter, "Computer Image Generation Systems," in *Computer Image Generation*, B. Schachter, ed., John Wiley & Sons, New York, 1983, pp. 47-124.
2. J. Orlansky and J. String, "Reaping the Benefits of Flight Simulation," in *Computer Image Generation*, B. Schachter, ed., John Wiley & Sons, New York, 1983, pp. 191-202.
3. Silicon Graphics Systems Documentation, Product Specifications, Silicon Graphics, Inc., Mountain View, Calif., 1985.
4. US Army Combat Developments Experimentation Center tech. report, "Fort Hunter Liggett Digital Terrain Database on the VAX Computer," Fort Ord, Calif., 1985.
5. J. Manley, *A Multimedia Computer Conferencing System*, master's thesis, US Naval Postgraduate School, Monterey, Calif., Dec. 1986.
6. Silicon Graphics Systems Documentation, Iris User's Guide, Silicon Graphics, Inc., Mountain View, Calif., 1985, pp. 8-1—8-13.
7. P. Baker and D. Hearn, *Computer Graphics*, Prentice-Hall, Englewood Cliffs, N.J., 1986, pp. 265-278.
8. H. Fuchs, "Near Real-Time Shaded Display of Rigid Objects," *Computer Graphics* (Proc. SIGGRAPH 83), July 1983, pp. 65-72.

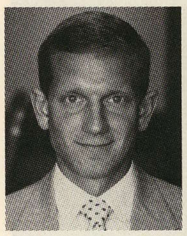


Michael J. Zyda is an associate professor of computer science at the Naval Postgraduate School, Monterey, California. He has been at NPS since February of 1984 and is currently associate chairman of the Department of Computer Science. Zyda has consulted with over 15 companies throughout Japan. He began his career in computer graphics in 1973 as part of an undergraduate research group, the Senses Bureau, at the University of California, San Diego. His research interests focus on producing techniques guides for using the capabilities of high-performance, real-time, interactive graphics workstations.

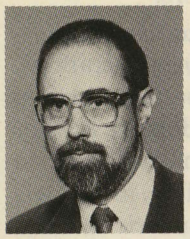
Zyda received a BA in bioengineering from the University of California, San Diego, in 1976; an MS in computer science/neurocybernetics from the University of Massachusetts, Amherst, in 1978; and a DSc in computer science from Washington University, St. Louis, Missouri, in 1984.



Ron Ross, a major on active duty with the US Army, is currently assigned as a PhD student at the Naval Postgraduate School, Monterey, California. His research focuses on developing automatic path-planning algorithms using artificial intelligence techniques and high-resolution digital terrain databases. Ross received his BS from the US Military Academy at West Point and his MS from the Naval Postgraduate School.



Douglas B. Smith, a captain in the US Marine Corps, is the Mission Critical Computer Resource Policy Officer at Headquarters, US Marine Corps, Washington, DC. Smith is responsible for Marine Corps policies on the acquisition and use of hardware and software in embedded computer systems and the resources that support them. He received his BS in computer science from Duke University in 1981 and his MS in computer science from the Naval Postgraduate School in 1987.



Robert B. McGhee is a professor of computer science at the Naval Postgraduate School, Monterey, California. Previously, he held a joint appointment as professor of electrical engineering and professor of computer and information science at Ohio State University. He has also been a member of the technical staff of Hughes Aircraft Company, where he worked on the design of guidance systems for anti-aircraft and antitank missiles. In addition, he has been a member of the electrical engineering faculty at the University of Southern California. His teaching and research interests are computer control of complex mechanical systems in general, and mobile robots in particular. He has also worked extensively in biomechanics and neural control.

McGhee received a BS in engineering physics from the University of Michigan in 1952, and an MS and a PhD from the University of Southern California in 1957 and 1963, both in electrical engineering.

Dale G. Streyle, a lieutenant in the US Coast Guard, is the project manager at the Coast Guard's Electronic Engineering Center, Computer Systems Branch, Wildwood, New Jersey. Streyle is responsible for the management of the Coast Guard's Unit Financial System. He received his BS from the Coast Guard Academy in 1980 and his MS in computer science from the Naval Postgraduate School in 1987.

The authors can be contacted through Michael Zyda at the Naval Postgraduate School, Code 52, Dept. of Computer Science, Monterey, CA 93943-5100, or via electronic mail: zyda@nps-cs.arpa.