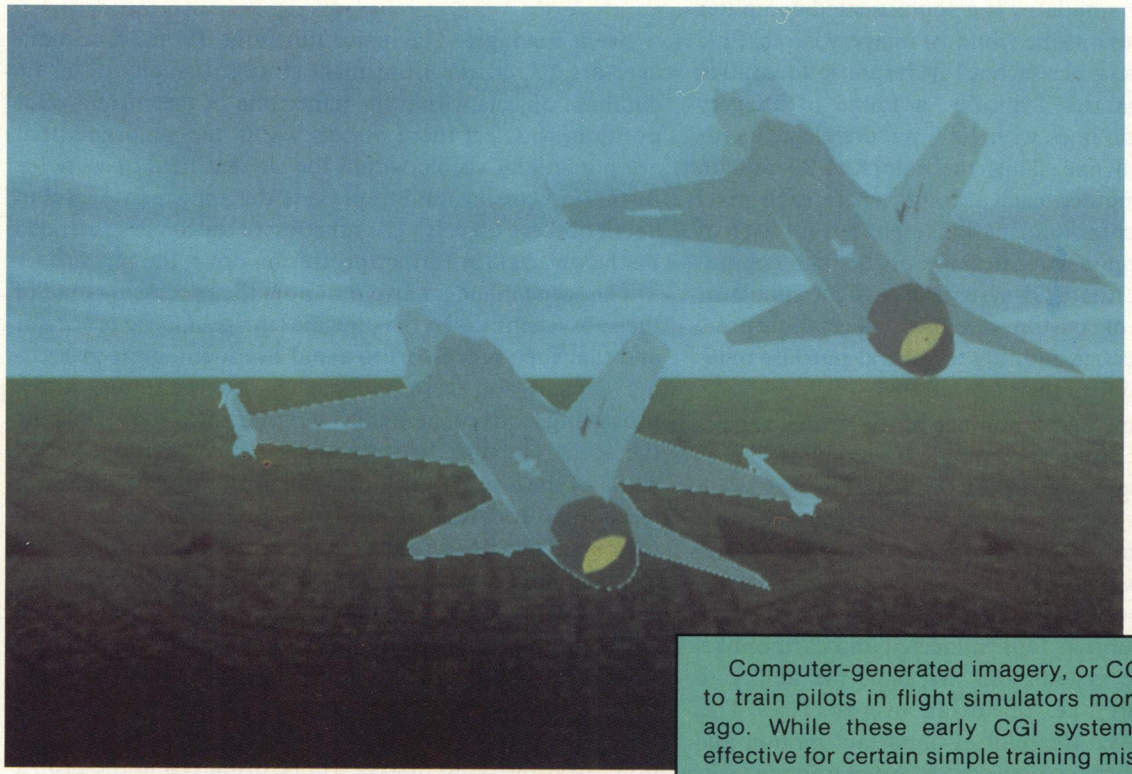
*The world of aviation flight simulation is an old one as computer graphics applications go. How did it progress to its current state, and where is it going next?*

# Advances in Computer-Generated Imagery for Flight Simulation

Johnson K. Yan
The Singer Company



Computer-generated imagery, or CGI, was first used to train pilots in flight simulators more than a decade ago. While these early CGI systems proved to be effective for certain simple training missions, they were incapable of such sophisticated training tasks as terrain following and other tactical maneuvers. Early CGI systems could depict only objects consisting of planar polygons. Furthermore, these CGI systems did not have the ability to superimpose "texture," i.e., fine detail, over the surfaces. Another deficiency was the inability of these systems to transition from one level of detail to the next one in a smooth manner. Finally, early CGI systems had no provision for suppressing such aliasing symptoms as stairstepping, line breakup, crawling, and scintillation. Advances in electronics and algorithms over the past decade have made it feasible to cure most of the deficiencies of early CGI systems. This article discusses these advances.


0272-1716/85/0800/0037$01.00

The flight simulator was invented more than fifty years ago by Edwin Link. A flight simulator is a device in which air crews can be trained without the use of the actual aircraft. There are two major reasons for using flight simulators to train pilots. First, the cost of operating a flight simulator is usually lower than using actual aircraft. Second, pilots can be trained safely on the ground in a flight simulator without risking their lives. For example, flight simulators can be used to train pilots to react to emergency situations (such as an engine out), which are too dangerous to carry out in an actual aircraft. Obviously a flight simulator is a valuable training substitute for an actual aircraft.

Early flight simulators simulated only the motion and instruments of the simulated aircraft. Later, out-the-window imagery was simulated by movies or by photographing model boards with servo-driven TV cameras. Computer-generated imagery (CGI) was first used to train pilots in flight simulators more than a decade ago. Simply put, CGI in a flight simulator is a sophisticated computer system that generates out-the-window imagery in real time (30 frames of imagery per second) in response to control inputs from the trainee. Typically, a frame of imagery consists of 1000 scan lines with 1000 picture elements, or pixels, in each scan line. Thus the colors of 30 million pixels must be computed each second, where each pixel contains 24 bits of color information (eight bits for each of the three color components). Because of the high computation requirement, the CGI system in a flight simulator typically consists of custom-designed, special-purpose, high-speed computers, in addition to general-purpose minicomputers which control their operations.

Early CGI systems had a number of major deficiencies that restricted their use to such simple training missions as take-off and landing. First, the electronic technology available for early CGI systems led to CGI architectures that severely limited their performance. Also, they could depict only objects approximated by a mosaic of planar polygons (henceforth these polygons are referred to as faces). Thus, man-made curved objects looked faceted and cartoonlike. Furthermore, these systems did not have the ability to superimpose "texture"; i.e., fine detail, over the faces. This lack of detail deprived the pilot of important motion and attitude cues necessary for such sophisticated training tasks as terrain following and other tactical missions. Another deficiency was the inability of these systems to transition from one level of detail to the next one in a smooth manner. (Objects are usually represented in several levels of detail, with the higher levels shown when the pilot is close to the objects.) This results in a sudden change ("popping") that tends to distract the pilot. Finally, early CGI systems had no provisions for suppressing such aliasing symptoms as stairstepping, line breakup, crawling, and scintillation.

With advances in electronics and algorithms, it is now feasible to cure most of the deficiencies of early CGI systems. The following sections describe progress made over the past decade. A number of advances will be discussed: first, the evolution of CGI architectures; second, the generation of curved surfaces using primitives other than planar polygons; third, the antialiasing capability, which eliminates most of the distracting aliasing symptoms plaguing early CGI systems; fourth, the translucency capability, which allows objects to be switched gradually and inconspicuously from one level of detail to another; and fifth, texture mapping, which provides spatially fixed and perspectively correct textural detail on surfaces.

## The evolution of CGI architectures

A CGI system usually consists of four major subsystems arranged in a pipeline as depicted in Figure 1.[1] The subsystems are the scene manager, the geometric processor, the video processor, and the display. The following subsections discuss the evolution of the architectures of these subsystems.

**Scene manager.** The major functions of the scene manager are to retrieve from mass storage (usually disc) the database objects within the panorama of the current pilot position, to select those objects within the panorama that are potentially visible within the current field of view for further processing, and to provide the appropriate level of detail of these objects (as a function of the angle subtended by the objects) for further processing down the pipeline.

The scene manager also monitors the processing load of the subsystems down the pipeline (the geometric processor and the video processor) to avoid overloading the processing capacity of these subsystems. For example, if the processing loads of these subsystems are near their capacity, the scene manager will gradually reduce the amount of detail of the objects sent down the pipeline (by using lower level detail models), or it could delete from further processing those scene elements that are deemed less important for the particular training mission. This dynamic scene content-control mechanism was discussed by Newhard and Nicol at an Industry Training Equipment Conference.[2]

In most high-performance CGI systems used for flight simulation, the scene manager also sorts those potentially visible objects within the current field of view in visual order of priority, so objects will be displayed in the proper order with hidden surfaces removed. This ordering by priorities is one solution to the hidden-surface removal

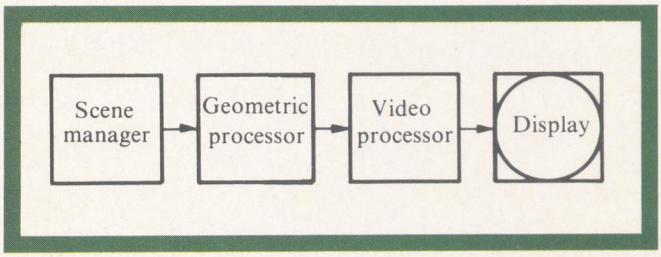



**Figure 1. CGI system pipeline.**

problem. Several solutions to this problem were discussed in the classical survey paper on this subject by Sutherland, Sproull, and Schumacker.[3] Most modern CGI systems employ a variation or an extension of one of these classical solutions to the problem of ordering.

Early CGI systems were used mainly in applications where, in the pilot's panorama, the database objects and their visual priority order change slowly. Therefore, they did not have to be updated every frame. The processing requirement of the scene manager for these early CGI applications is about 0.5 to 1.0 MIPS (Million Instructions Per Second). This processing requirement is usually satisfied with a general purpose minicomputer. With the rapid advance of the processing capability of modern microprocessors, this processing requirement is beginning to be satisfied with microcomputer systems.

As flight simulators are becoming more sophisticated, more sophisticated training applications are found for them. For example, low-light-level TV systems on modern aircraft are equipped with telescopes which allow the pilot to zoom onto distant areas instantly. Pilots can be trained to use this sophisticated equipment in a flight simulator, which also provides a display for low-light-level TV. These sophisticated training tasks require that objects in the pilot's panorama and their visual order of priority be updated every frame. The scene-manager processing requirement for these sophisticated training applications is about 40 MIPS. Because of the greater computational requirement, the trend in most modern CGI systems is to implement the scene management functions in special-purpose programmable computers. These computers could be implemented with off-the-shelf high-performance microprogrammable multiprocessors, with off-the-shelf high-performance microprogrammable array processors, or with custom-designed microprogrammable multiprocessors. The design philosophy in most modern CGI systems tends to emphasize modularity. Modular design allows the same design to be employed in a wide variety of training applications with varying processing requirements; applications requiring more processing power simply use more parallel identical processors than applications requiring less processing power.

**Geometric Processor.** The major functions of the geometric processor are to project potentially visible objects onto the pilot's screen, to clip (and remove from further processing down the pipeline) portions of the objects that are outside the view window, and to compute the geometric and color gradient parameters required by the video processor subsystem down the pipeline to compute the color for each pixel occupied by the visible objects. The techniques for performing these geometric-processing functions are discussed in books by Foley and Van Dam and by Newman and Sproull.[4,5]

One of the key performance measures of a CGI system is the number of faces (the average number of edges defining a face is around three) that the geometric processor can process within the time allowed for each field of the frame. This is an important performance measure because the number of faces that can be displayed determines, to some extent, the complexity of the displayed scene. Early CGI systems could only process several hundred faces. Modern CGI systems can usually process several thousand faces. The processing requirement of geometric processors in modern CGI systems is on the order of 60 MIPS.

Because of the limitations of computer and electronic technologies about a decade ago, early CGI systems implemented the geometric processing function with a hardwired, pipelined architecture. This approach made changes to the design difficult. Also, the processing time in this approach is not used efficiently, because in a hardwired, pipelined architecture, the pipeline is designed to handle the worst-case geometric-processing requirement. Consequently, objects that do not require worst-case geometric processing take the same amount of processing time as the worst-case objects. In modern CGI systems the trend is toward implementing the geometric-processing function in special-purpose programmable computers. As in the scene-management function, the special-purpose programmable computers can take the form of off-the-shelf microprogrammable multiprocessors, off-the-shelf microprogrammable array processors, or custom-designed microprogrammable multiprocessors. In fact, the trend is toward using the same processor design (with different microprograms) for both the geometric processor and the scene manager.

***Early CGI systems could only process several hundred faces; modern systems can usually process several thousand faces.***

**Video processor.** The major function of the video processor is to take the projected face geometric and color gradient information output from the geometric processor and compute the color for each pixel occupied by visible portions of each face. The resulting color of each face on a pixel takes atmospheric attenuation into account by blending the color of the face on each pixel toward the haze color as a function of the distance of the face *on each pixel* from the pilot position. It also takes into account illumination due to the landing light of an aircraft for landing and takeoff training missions at night. The video processor also computes the color contribution of each face on each pixel

resulting from such advanced CGI features as antialiasing, translucency, and texture. These advanced CGI features will be discussed later.

The color computation is performed for each pixel of the entire screen. With a million pixels on the screen, the processing requirement of a typical video processor is on the order of 10,000 MIPS. Moreover, this video-processing requirement is increasing rapidly as more sophisticated video processing algorithms are employed to provide more advanced CGI features. Unlike scene management and geometric processing, the high processing rate required for video processing will keep the video processor in the domain of special-purpose hardwired computers for the foreseeable future. Like the scene manager and the geometric processor, the design philosophy in most modern CGI systems tends to emphasize modularity. With the rapid advance in VLSI, or Very Large Scale Integration, technology, future video processors are expected to take advantage of advanced VLSI technology for efficient design of custom video-processing circuitry.

Video processor architectures fall into two basic categories: scan-line-based and frame-buffer-based architectures.[1] Video processors of early CGI systems were mainly scan-line based. Scan-line-based video processors perform video processing one scan line at a time in synchronism with the raster display of the scan lines. Thus, no matter how complicated the scene is on a given scan line, each scan line is allowed equal time to complete its video. Within the allotted time for each scan line, intersections of edges of all objects active on that scan line are sorted, and the visual priorities of overlapping faces are resolved to give visible segments on the scan line. While the visible segments are being computed for the next scan line, the visible segments of the current scan line are expanded (using color gradient parameters computed in the geometric processor) into a stream of colors, one color for each pixel of the scan line, and displayed. Since all these calculations must be completed within the allotted time, scan-line-based video processors can only handle a limited number of intersections, typically from 256 to 1000. If the number of intersections on a scan line exceeds the limit, a bad scan line results. While some scan lines (around the horizon) may be overloaded, others (those depicting the sky) contain very few intersections. Thus, in a typical image the number of intersections are not evenly distributed among the scan lines; consequently, scan-line-based video processors do not use their processing time very efficiently.

With advances in memory technology, the trend in modern CGI systems is to employ frame-buffer-based video processors. A frame buffer, which buffers the red, green, and blue components of the color on each pixel of the whole frame, allows each face of all objects to be processed independently by "painting" the visible portion of each face into the frame buffer. This strategy allows the sharing of processing capacity over the entire frame. Compared with scan-line-based video processors, the complexity of images that can be displayed without anomalies by frame-buffer-based video processors is now limited by the busiest image, as opposed to the busiest single scan line. A further advantage of frame-buffer-based video processors is the intensive use of VLSI memories, which have undergone a tremendous increase in speed and density and a decrease in price over the last few years. This trend in memory technology is expected to continue.

Given a frame-buffer-based video processor, faces can be painted into the frame buffer in bottom-up order (farthest faces first), top-down order (closest faces first), or random order. In bottom-up writing, occultation (i.e., hidden-surface removal) is accomplished automatically when nearer faces overwrite more distant faces. In top-down writing, occultation is accomplished by detecting portions of a face that are already written by nearer faces and suppressing their writing into the frame buffer. The detection can be accomplished with the bed-of-nails concept, in which a subpixel bit mask is stored for each pixel on the screen for face occultation down to the subpixel level. This bed-of-nails concept will be discussed further in connection with antialiasing. In random order writing, occultation of faces is resolved during video processing by distance comparison on each pixel (or subpixel). This random face-writing order is known as the Z-buffering approach,[4] which does not require the visual priority-ordering step in the scene manager discussed above. The advantages and disadvantages of these different face-writing orders with regard to implementation of advanced video-processing features were discussed by Latham at the 1983 Industry Training Equipment Conference.[6]

***With advances in memory technology, modern CGI systems tend to employ frame-buffer-based video processors.***

Most modern high-performance frame-buffer-based video processors employ the top-down writing order for several reasons. The first is that with top-down writing, a mechanism can be provided to allow skipping over the video processing of portions of faces that are hidden behind closer faces. On the other hand, a bottom-up writing-order video processor will have to process every pixel of every face regardless of whether it is hidden or not. The second is that antialiasing implementation is easier (as will be explained later on) when faces are written into the frame buffer in top-down order.

**Display.** The display unit used in CGI systems for flight simulation usually consists of high-resolution 1000-line color monitors. A typical flight simulator consists of three display windows (one front window and two side windows) using one 1000-line color monitor for each window. The three windows typically cover a field of view about 120 degrees horizontal by 40 degrees vertical. This results in an angular resolution of about 4.8 arc minutes per line pair.

Flight simulators used for training commercial aircraft pilots emphasize the image quality of light points, because most of the training for these pilots is for maneuvers at night or dusk, when the visibility of the environment is low. For these applications the light points are usually displayed calligraphically, because a calligraphic display can position the light points more accurately than a raster display. In fact, most early flight simulators for training commercial pilots used calligraphic systems for display. In those early systems, the faces defining the runway were usually painted calligraphically with closely spaced straight lines.[7] However, the quality of faces painted calligraphically is usually poor. Faces can be painted more efficiently and with better quality by a raster display system. To maintain the high quality of light points while improving the quality of faces, most modern CGI systems used for training commercial pilots employ a dual-mode display system in which part (usually half) of the frame-time the display system operates in the calligraphic mode for light points and part of the frame-time in the raster mode for faces.

Flight simulators used for training military pilots usually require a much wider field of view than those used for training commercial pilots. A field-of-view requirement of 360 degrees horizontal by 120 degrees vertical is not uncommon for military applications. Early flight simulators satisfied this wide field-of-view requirement by a mosaic of monitors covering the entire field of view (see Figure 2). This arrangement is unwieldy, especially because the entire mock-up cockpit and the display monitors are mounted on a motion base. The trend in modern flight simulators for wide field-of-view applications is toward projecting imagery onto a dome with high-resolution TV projectors (see Figure 3).

As more and more military training missions that can be carried out in a flight simulator are identified (such as target identification), higher and higher resolution is required of the display systems. The display resolution requirement for these demanding training missions is on the order of one arc minute per line pair, the limit of human visual acuity.[8] Providing such high resolution over the entire wide field of view of 360 degrees horizontal by 120 degrees vertical would require over a hundred 1000-line CGI systems. The cost of such a flight simulator would be prohibitive.

Advanced display systems are now under development to provide such high resolution over wide fields of view at a reasonable cost. Such advanced display systems attempt to provide high-resolution imagery within a small field of view around the line of sight and low-resolution imagery in

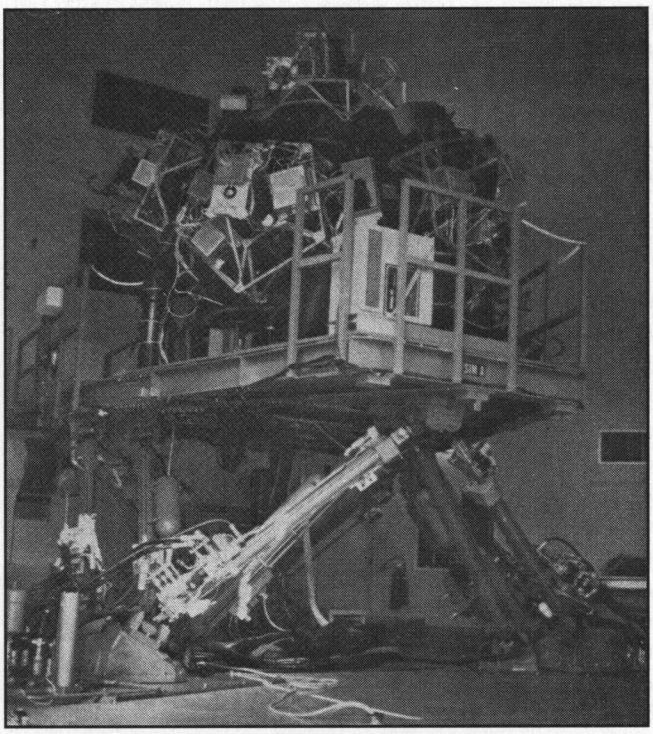


**Figure 2. Mosaic of monitors covering wide field of view.**

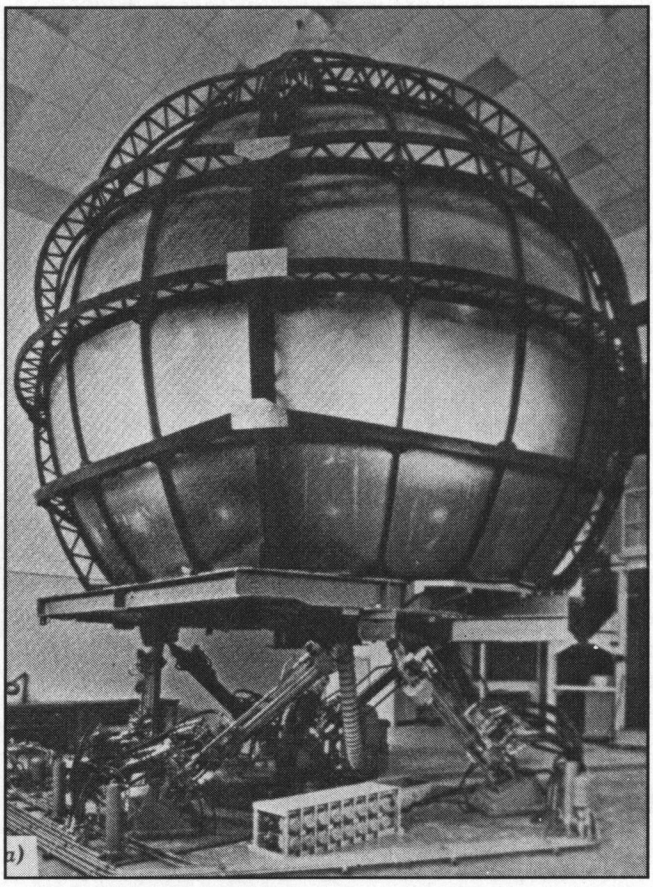


**Figure 3. Dome display system.**

the periphery of the line of sight. These advanced display systems are referred to as area-of-interest, or AOI, display systems. AOI display systems take advantage of the fact that human visual acuity is highest near the line of sight and poor in the periphery. Therefore, AOI display systems offer the potential of significant reduction in the number of 1000-line CGI systems (channels) required for wide fields of view and high-resolution applications.

Two such advanced AOI display system are in their final stages of development. One of these is the Eye-Slaved Projected Raster InseT (ESPRIT) system developed by Link.[9,10] In the ESPRIT system, the pilot's line of sight is tracked using an eye-tracking device. An eye-slaved high-resolution imagery covering about a 20-degree field of view around the line of sight is inset onto a low-resolution imagery covering the entire field of view. Both the high-resolution and the low-resolution imagery are projected onto a dome. The line of sight of the eye-slaved high-resolution imagery is slewed with servo and optical mechanisms to match the pilot's line of sight. (In a variation of the ESPRIT approach, a low-resolution peripheral imagery covering a smaller field of view [180 degrees horizontal by 80 degrees] than the entire field of view is eye-slaved along with the high-resolution imagery.[11])

The other AOI system is the eye-slaved variable-acuity lens display system under development at McDonnell Douglas.[12] The variable-acuity display system has a focal length that varies with the field angle in the same manner as human visual acuity. The lens projects a uniform pixel-density imagery in its focal plane onto a screen (in a dome) with variable angular increments (smaller near the center) exactly as required by the human visual system. The variable-acuity lens display system therefore allows more pixels to be computed near the line of sight where they count most, thus significantly reducing the number of pixel computations for wide field-of-view applications.

With the advanced AOI display systems discussed above, only about one to three 1000-line CGI channels are required to provide the one arc minute per line-pair resolution required near the line of sight. Both AOI display systems require that the projector's line of sight be slewed with a sophisticated servo and optical mechanism to match exactly the pilot's line of sight. A fiber-optic helmet-mounted display system (see Figure 4) is now under development by CAE to eliminate the requirements for such servo and optical mechanisms and domes (which are typically twenty-four feet in diameter and occupy a large room).[13] In this display system, imagery is transmitted with fiber-optic bundles to a miniature projection and display system mounted on the pilot's helmet.

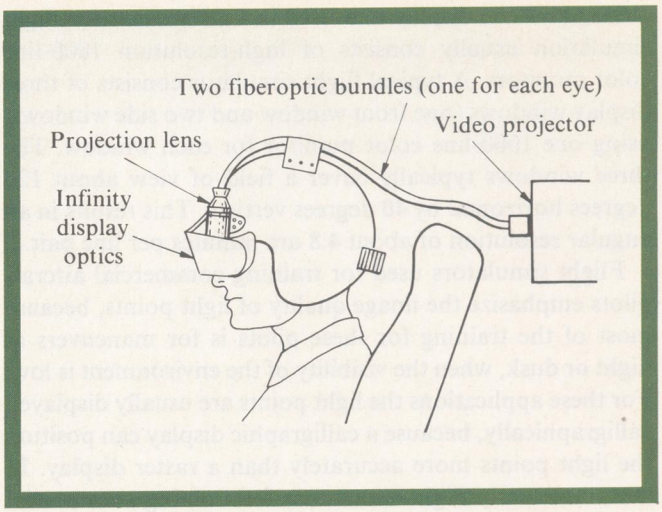



**Figure 4. Helmet-mounted display system.**

## Database primitives

In flight simulation it is sometimes necessary to portray such man-made curved objects as water towers, oil storage tanks, silos, and other aircraft. Early CGI systems approximated curved surfaces with a number of faces (planar polygons). The shade across each of these faces was constant and was computed by calculating the dot product of the sun's illumination direction with the normalized surface normal. As a result, curved objects looked faceted. To eliminate this faceted appearance, Gouraud introduced an algorithm for continuous shading across boundaries of faces.[14] In Figure 5, Gouraud's continuous smooth shading is applied to the aircraft, approximated with about 1000 faces. This algorithm improves the appearance of curved objects, but the silhouette of an object is still composed of straight edges.

The straight-edge appearance of the silhouettes of curved objects can be eliminated if enough faces are used to model them in the database. But this increases the size of the database storage and the time required for geometric processing. For man-made and a limited class of natural curved objects, quadric surfaces were introduced, because they are more economical in terms of database storage (as well as geometric processing) than planar polygon modeling of these objects to the same image fidelity.[15–17] Quadric surfaces are surfaces defined by a second-degree equation in three-dimensional space. They included spheres, ellipsoids, hyperboloids, cones, and cylinders. With the exception of spheres and ellipsoids, all other quadric surfaces have infinite extent; they must therefore be bounded by planes to form bounded cylinders, cones, frustums, and other solids. The bounding of quadric surfaces by planes results in planar conic sections. Thus, quadric surface generation is the generation of bounded quadric surfaces and planar conic sections.

These bounded quadric surfaces, planar conic sections, and planar polygons can be combined to model more complicated curvilinear objects. In Figure 6 the scene elements are modeled with quadric surfaces. The silhouettes of these quadric surfaces are composed of either straight lines or conic sections. Efficient algorithms exist[16,17] for

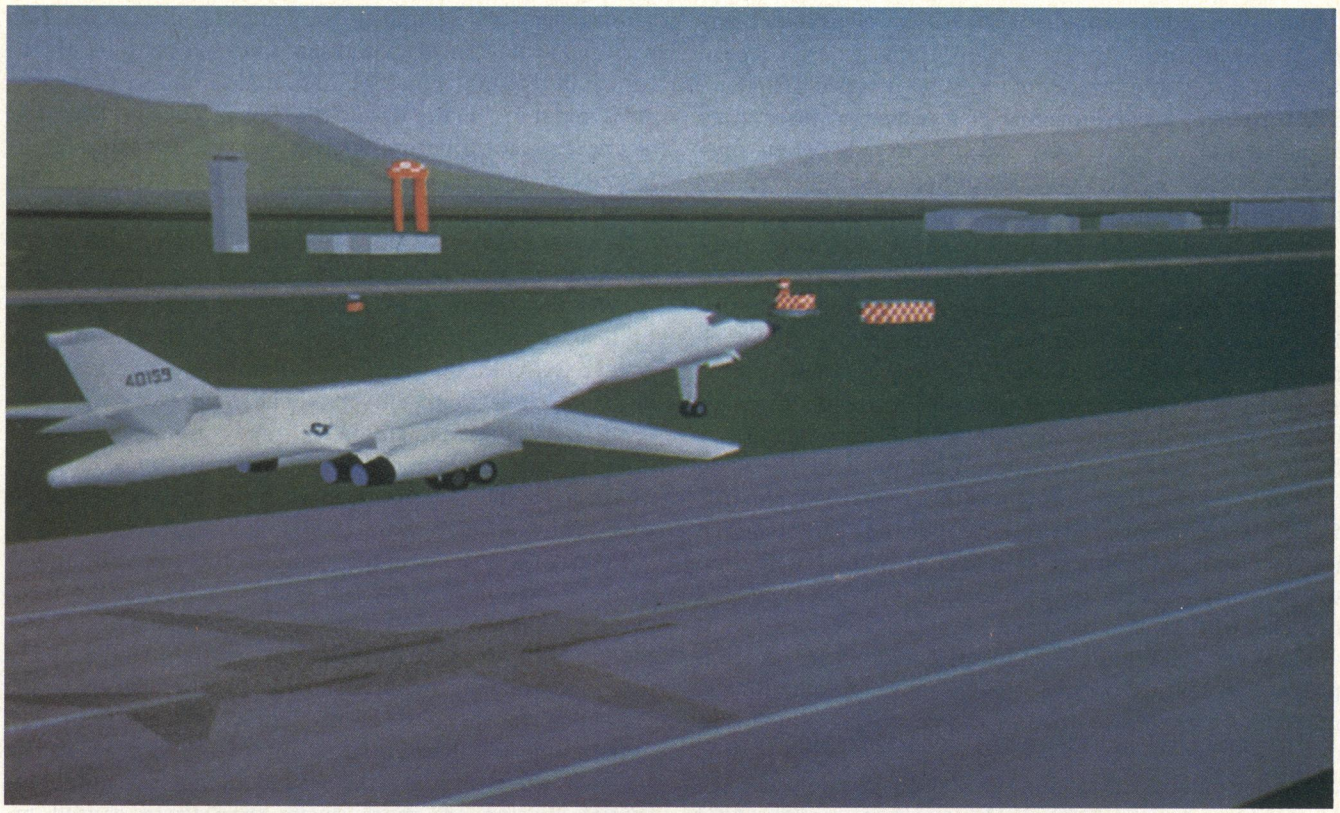


**Figure 5. Illustration of Gouraud smooth shading.**


**Figure 6. Scene modeled with quadric surfaces (courtesy of Grumman Aerospace).**

finding silhouettes of quadric surfaces as well as their intersections with the scan lines. Algorithms have also been developed to compute the intensity on quadric surfaces.[16] Since man-made curvilinear objects are not required in all applications, modern CGI systems usually offer only quadric surface generation as an option to augment planar polygon generation.

While quadric surfaces are useful for modeling man-made and a limited class of natural curvilinear objects, they are unsuited for modeling free-form curvilinear objects. Free-form curvilinear objects are best modeled with bicubic patches.[18] A bicubic patch is a surface in three-dimensional space defined parametrically with three cubic polynomials of two independent variables, hence the term *bicubic.* A free-form, natural, curvilinear object, such as a rolling terrain, is modeled by piecing together many such patches. Bicubic patches have the property that they can be pieced together with gradient continuity across patch boundaries; this property gives rise to the smooth appearance of objects modeled with bicubic patches.

However, rendition of bicubic patches and the superimposition of texture on these patches are computationally much more expensive than is the case with planar and quadric surfaces.[19] Therefore, cost-effective implementation of bicubic patches in real-time CGI systems is not expected in the near future. Instead, the trend in modern CGI systems used for flight simulation is to stay with planar polygons as the major database modeling primitive, with quadric surface modeling as an option for applications where many man-made curved objects are required.

## Antialiasing

Early CGI systems produced images that had many such visual anomalies as stairstepping, breakup of narrow lines, scintillation, etc. These visual anomalies are collectively referred to as aliasing. A discussion of these anomalies can be found in a paper by Szabo.[20] This section discusses advances made during the last decade to suppress aliasing.

Moving image sequences are three-dimensional signals, two spatial dimensions, and one temporal (time) dimension. In real-time CGI systems, these three-dimensional continuous signals are converted to discrete signals by "sampling." A frame of image in a typical CGI system consists of 1000 scan lines with 1000 pixels on each scan line, resulting in an image resolution of about one million pixels per frame. Thirty frames of imagery are computed and displayed per second in a typical CGI system. Images of CGI systems are usually displayed on 2:1 interlaced displays. This means that a frame of image is displayed as two interlaced fields (the odd and even fields). The odd field consists of the odd scan lines of the frame, and the even field consists of the even scan lines of the frame. For a typical CGI system, the sampling frequency for each field is, therefore, one sample per pixel in the horizontal spatial dimension, one sample per two scan lines in the vertical spatial dimension, and one sample per 1/60 second in the temporal dimension. For each field the vertical spatial sampling frequency is, therefore, half that of the horizontal spatial sampling frequency. As is apparent in later discussion, this fact accounts for the asymmetric convolution function required for spatial antialiasing in dynamic movie sequences displayed on 2:1 interlaced displays.

The sampling theorem states that aliasing will occur when the frequency components of the image signal exceed the Nyquist frequencies (half the sampling frequencies, both spatial and temporal). Therefore, the cure for aliasing is to band-limit the frequency components (both spatial and temporal) of the image signal *before the sampling process.* Aliasing problems in real-time computer-generated imagery are the result of not properly band-limiting imagery to the Nyquist frequency before the sampling process.

Early CGI systems simply sampled the un-band-limited image. Consequently, all the symptoms of aliasing were evident in early CGI systems (an example of an aliased picture is shown in Figure 7). Most modern CGI systems now incorporate some sort of spatial antialiasing capability. However, to the best of my knowledge no temporal antialiasing is implemented in any real-time CGI system for two reasons: First, the visual anomalies caused by spatial aliasing are more severe than those caused by temporal aliasing, especially at limited aircraft slew rates; and second, implementation of temporal antialiasing is expensive because it involves either "motion blur" implementations using sophisticated processing,[21] or sampling at a rate higher than 60 fields per second.[1] The following discussion on antialiasing is restricted to spatial antialiasing.

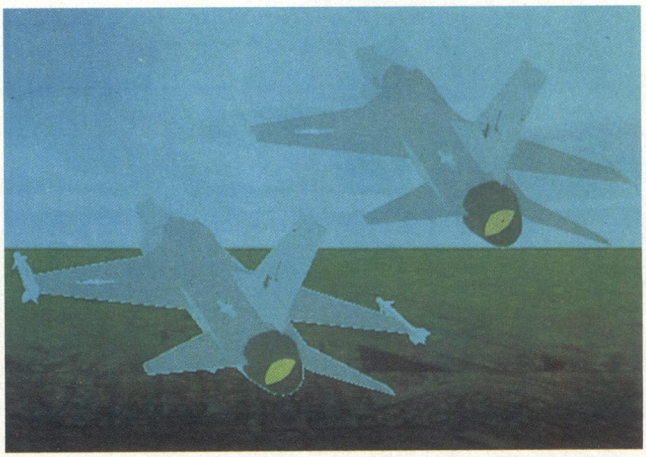


**Figure 7. Aliased scene due to pure sampling.**

As mentioned earlier, the ideal spatial antialiasing approach is to band-limit of the image corresponding to each field to the horizontal and vertical Nyquist frequencies the spatial frequency content of each field of imagery. This can be accomplished by low-pass filtering with a box filter in

the frequency domain, which is equivalent to convolution in the spatial domain by a two-dimensional *sinc* function. A one-dimensional version of this sinc function is illustrated in Figure 8. This convolution function extends to infinity, with less contribution from scene elements farther away from the sample point. Since the wider the convolution area (area over which the convolution function is nonzero), the more computation is involved, implementation of the ideal spatial convolution is impractical. Modern CGI systems approximate this ideal spatial convolution by truncating the convolution area to a few sample periods in the immediate vicinity of the sample point. This is illustrated in Figure 8, where the weight of the convolution function drops off rapidly after one sample period on either side of the sample point. A good approximation of

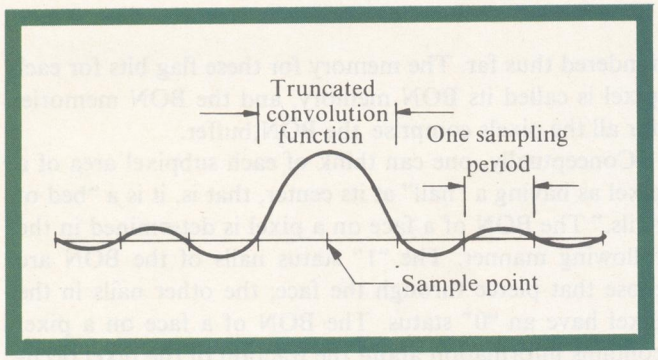



**Figure 8. Ideal 1-D convolution function.**

the ideal convolution function is a bell-shaped, truncated convolution function spanning two sample periods, as indicated in the figure. For 2:1 interlaced fields, the sample period is one pixel in the horizontal direction and two scan lines in the vertical direction. A good approximation of the ideal convolution function for interlaced display is one spanning two pixels horizontally and four scan lines vertically and shaped like a bell curve in both directions as depicted in Figure 9(a). The two-dimensional convolution function within the two-by-four pixel area around the sample point is given by the product of the weights of the horizontal and vertical functions.

**Constant energy criterion.** It has been suggested that the shape of the convolution function should be designed to satisfy the constant energy criterion (CEC) attributed to Erdahl.[22,23] A spatial convolution function so designed will allow such small scene elements as light points and narrow lines to maintain constant total energy contributed to all sample points affected by them (if their sizes on the screen are to remain the same while being moved around on the screen). This is important for minimizing apparent breakup of narrow lines and scintillation of small light points.

The implication of this criterion is that the superimpo-

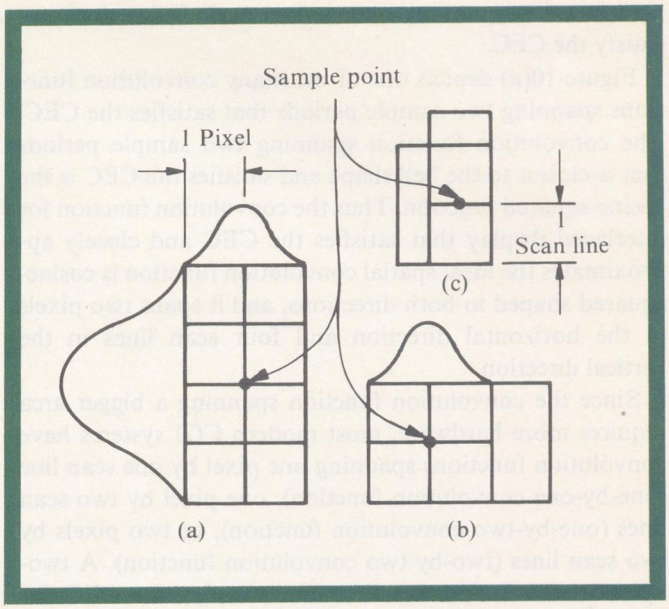



**Figure 9. Convolution functions with different convolution area sizes.**

sition of convolution functions over all the sample points in the field must be *uniform* over the whole image. To illustrate this concept, a one-dimensional triangular convolution function spanning two sample periods that satisfies the CEC is depicted in Figure 10(a). Notice that the superimposition of the convolution functions centered around all sample points are constant. Figure 10(b) depicts a one-dimensional convolution function spanning one sample period that satisfies the CEC. In fact, this uniform convolution function is the only nonzero convolution function spanning one sample period that satisfies the criterion. Figure 10(c) depicts a one-dimensional convolu-

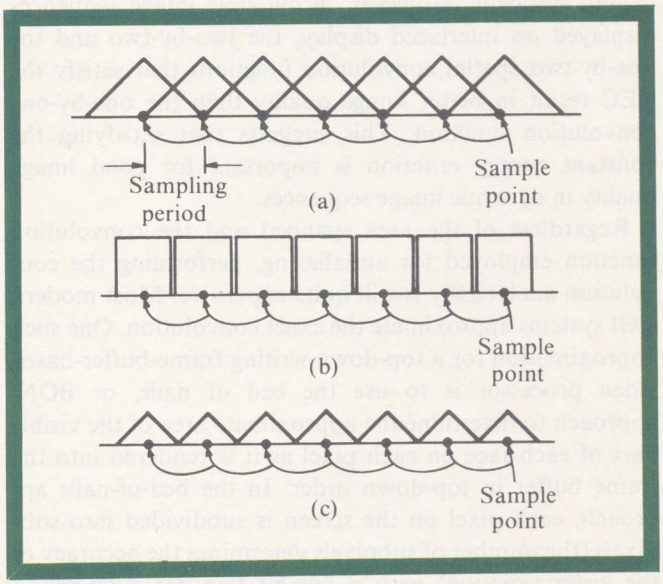



**Figure 10. Illustration of constant energy criterion.**

tion function spanning one sample period that does not satisfy the CEC.

Figure 10(a) depicts one of the many convolution functions spanning two sample periods that satisfies the CEC. The convolution function spanning two sample periods that is closest to the bell shape and satisfies the CEC is the cosine-squared function. Thus the convolution function for interlaced display that satisfies the CEC and closely approximates the ideal spatial convolution function is cosine-squared shaped in both directions, and it spans two pixels in the horizontal direction and four scan lines in the vertical direction.

Since the convolution function spanning a bigger area requires more hardware, most modern CGI systems have convolution functions spanning one pixel by one scan line (one-by-one convolution function), one pixel by two scan lines (one-by-two convolution function), or two pixels by two scan lines (two-by-two convolution function). A two-by-two convolution function that satisfies the CEC for interlaced display is one that is cosine-squared shaped horizontally and uniform vertically, as shown in Figure 9(b). The only one-by-two convolution function that satisfies the CEC for interlaced display is uniform in both the horizontal and the vertical direction, as shown in Figure 9(c). In fact, the smallest convolution function (in terms of the area spanned by the convolution function) that satisfies the CEC for interlaced display is the one-by-two function. There is no one-by-one convolution function that satisfies the CEC for interlaced display. The closest one-by-one convolution function to satisfy the CEC for interlaced display is one that is uniform horizontally and vertically.

Experiments have shown that improvement in image quality with one-by-one, one-by two, or two-by-two spatial convolution was quite dramatic as far as static images are concerned (see Figure 11). Moreover, the difference in image quality among one-by-one, one-by-two, and two-by-two spatial convolution is small when the images are viewed statically. However, in dynamic image sequences displayed on interlaced display, the two-by-two and the one-by-two spatial convolution functions that satisfy the CEC result in better image quality than the one-by-one convolution function. This suggests that satisfying the constant energy criterion is important for good image quality in dynamic image sequences.

Regardless of the area spanned and the convolution function employed for antialiasing, performing the convolution analytically is still quite expensive. Most modern CGI systems approximate the exact convolution. One such approximation for a top-down writing frame-buffer-based video processor is to use the bed of nails, or BON, approach to determine the approximate area of the visible part of each face on each pixel as it is rendered into the frame buffer in top-down order. In the bed-of-nails approach, each pixel on the screen is subdivided into subpixels (the number of subpixels determines the accuracy of the approximation) with a one-bit flag per subpixel to indicate whether or not its center is covered by faces

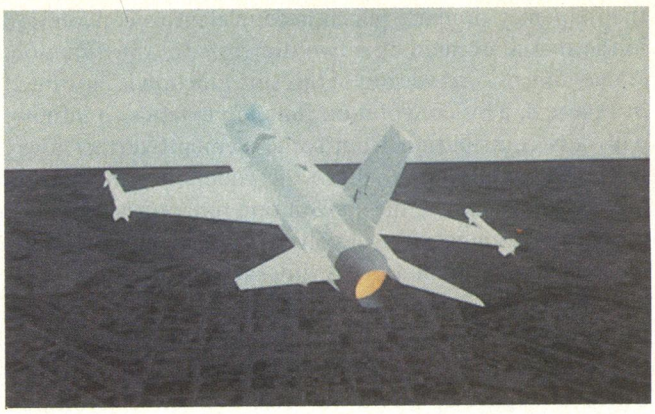


**Figure 11. Antialiased scene.**

rendered thus far. The memory for these flag bits for each pixel is called its BON memory, and the BON memories for all the pixels comprise the BON buffer.

Conceptually, one can think of each subpixel area of a pixel as having a "nail" at its center, that is, it is a "bed of nails." The BON of a face on a pixel is determined in the following manner: The "1" status nails of the BON are those that pierce through the face; the other nails in the pixel have an "0" status. The BON of a face on a pixel contains information about the fraction of the pixel occupied by the face and about which subpixels the face occupies. The latter information is necessary for occulting the faces at the subpixel level. As faces are processed in the top-down order, the BON buffer remembers the BON of the union of the BONs of the top faces; the "visible BON" of a lower priority face on a pixel is obtained by a subtraction (logical difference) between the stored BON and the BON of the lower priority face. The color of all faces, weighted by the contributions of their visible part within the convolution area of a pixel, are summed to give an antialiased color on the pixel.

## Translucency

Early CGI systems could only process opaque objects; consequently, such special effects as translucent clouds, smoke, dust, and shadows could not be simulated. Later, semitranslucent (50 percent translucent) objects were simulated by displaying the semitranslucent object in only one field.[24] This technique was used in early CGI systems to simulate a variety of such semitranslucent effects as suspension bridge wires, rotor blades, and translucent cockpit windows.

The trend in modern CGI systems is to incorporate full translucency capability, which allows objects of any degree of translucency to be depicted. The translucency factor, or T, of a face is defined as follows. A face with a translucency

factor of T (a number between zero and one) will allow T of the light energy of faces behind to be transmitted while allowing 1-T of the light energy of itself to be transmitted. For example, a red face with a translucency factor of 1/4 with an opaque blue face behind will have 1/4 of blue and 3/4 of red visible in the common area of the two faces. With this definition of translucency, the color resulting from the overlap of a multitude of translucent faces with different translucency factors and colors can be computed.

Translucent faces, like opaque faces, have to be antialiased to avoid aliasing problems (because a translucent face could be 99 percent opaque). The "perfect" antialiasing approach for translucent faces together with opaque faces is depicted in Figure 12. In the figure, three translucent faces A, B, and C and one opaque background face D are projected on the convolution area of a pixel. In the example the convolution area can be divided into eight regions. In each region the resulting color due to faces affecting that region can be computed from the translucencies of the faces involved. The "perfect" antialiased color for the pixel is given by the average color of the resulting colors of all the eight regions, weighted by their contribution within the convolution area. This analytic "perfect" solution is prohibitively expensive. Modern CGI systems approximate the "perfect" solution. The strategy used in designing the approximate solution is to ensure that the approximate solution works well for the most common situations encountered in flight simulation, at the expense of rare situations.

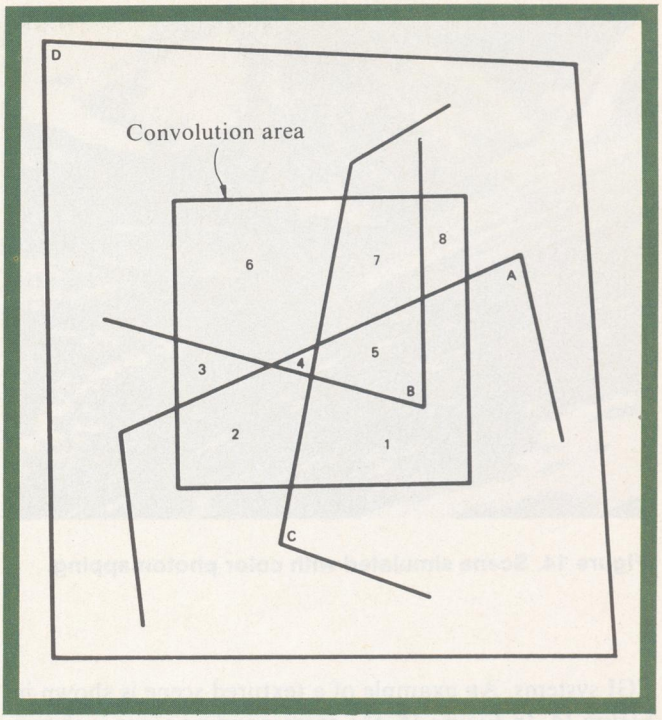



**Figure 12. Illustration of regions created by multiple layers of translucent polygons.**

**Gradual level-of-detail switching.** In addition to allowing simulation of translucent objects, full translucency capability can be used for blending two adjacent levels of details of objects to effect gradual level-of-detail switching in the following manner: Due to the limited face-processing capability of CGI systems, objects cannot be brought into view when they are only about a pixel in size on the screen. Consequently, they are usually brought into view abruptly when they occupy a sizable area on the screen. This causes "popping," the sudden and distracting appearance of an object. To conserve face-processing capability, each object is modeled at several levels of detail; the appropriate model is displayed as a function of the angle subtended. Popping can also occur when an object is switched abruptly from one level of detail to another. Full translucency capability can be used to effect both the gradual introduction of an object and the gradual switching of levels of detail.

An object is brought into view when it occupies a sizable area on the screen, starting with total transparency and gradually reducing the translucency until the object is totally opaque as it comes closer to the observer. The object stays opaque for a certain distance and then gradually fades out, while a higher-level-detail model is faded in by gradually decreasing the translucency of the high-detail model. This gradual level-of-detail switch is depicted in Figure 13 for three levels of detail.

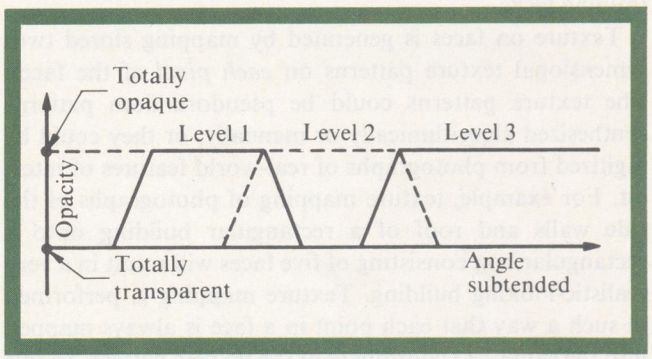



**Figure 13. Gradual level of detail switching.**

## Texture mapping

Texture is the fine detail on a surface, like the grass texture on a patch of lawn. Natural scenes are rich in texture, but early CGI systems were not capable of applying texture to surfaces. Experiments on these systems indicated that untextured surfaces deprived the pilot of motion and attitude cues, which made early CGI systems unsuited for low-altitude flight-training tasks. It is possible to add more fine detail within surfaces by using additional faces; however, this solution is too expensive in terms of the face-processing capacity of CGI systems. With advances in algorithms, it is now possible to add detail by "texture mapping," a technique that does not use and therefore does not consume any of the limited face-processing capacity of

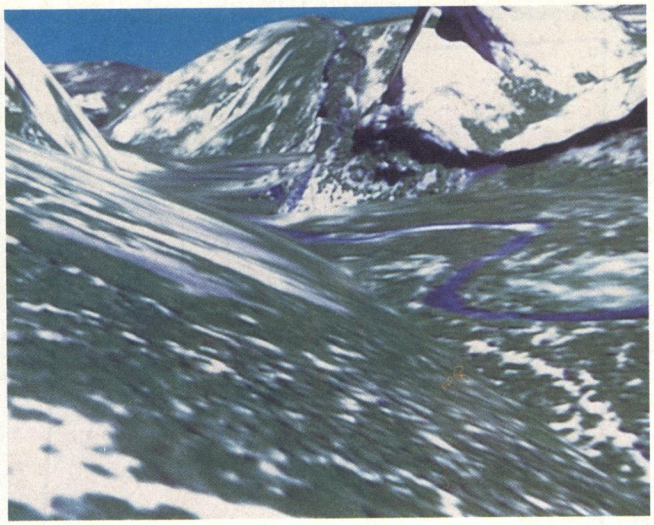


**Figure 14. Scene simulated with color photomapping.**

CGI systems. An example of a textured scene is shown in Figure 14. In Figure 15, the corresponding scene is shown without texture. The amount of scene detail provided by texture mapping is clear from these two pictures. It is anticipated that texture generation will be a standard feature in future CGI systems used for low-altitude flight-training tasks.

Texture on faces is generated by mapping stored two-dimensional texture patterns on *each pixel* of the faces. The texture patterns could be pseudorandom patterns synthesized algorithmically or manually, or they could be digitized from photographs of real-world features of interest. For example, texture mapping of photographs of the side walls and roof of a rectangular building onto a rectangular box consisting of five faces will result in a very realistic-looking building. Texture mapping is performed in such a way that each point in a face is always mapped into the same 2-D coordinate of the texture pattern. In this way, the 2-D texture pattern is spatially fixed to the face and will appear to the pilot in correct perspective as the pilot moves around the face. The mapping is accomplished by computing for each visible pixel of the face the corresponding 2-D coordinates on the texture pattern and using the result to look up a texture modulation value from the texture look-up table stored in memory. The modulation value looked up is used to modulate the intrinsic color of the face.

Texture mapping on faces can therefore be viewed as a 2-D sampling process. This 2-D sampling process is discussed by me in the tutorial notes of SIGGRAPH 85.[25] To avoid aliasing due to the texture on faces, the texture patterns must be band-limited to half the local 2-D sampling frequency (as dictated by the 2-D sampling theorem) on each pixel of the face. To minimize real-time processing, prefiltered texture patterns are usually stored in texture memory for real-time access by the video processor. The appropriate version of the texture pattern is then selected by the computed local 2-D sampling frequency at each pixel of the face. To avoid the sudden transition from one version of the texture to the next on a large face, one version is gradually faded out while the other version is gradually faded in by mixing the two adjacent versions in a complementary manner.

Due to memory storage limitation, texture patterns are stored to a finite resolution, typically one square foot. When the pilot gets very close to a textured face, this one-square-foot-resolution texture modulation could subtend a sizable area (say four pixels by four scan lines) on the screen, and a checkerboard effect will result. To avoid checkerboard effect when the resolution limit of the texture pattern is reached, texture modulation values can be linearly interpolated from the three closest stored texture-modulation values forming a triangle or bilinearly from the four closest stored texture-modulation values forming a square.

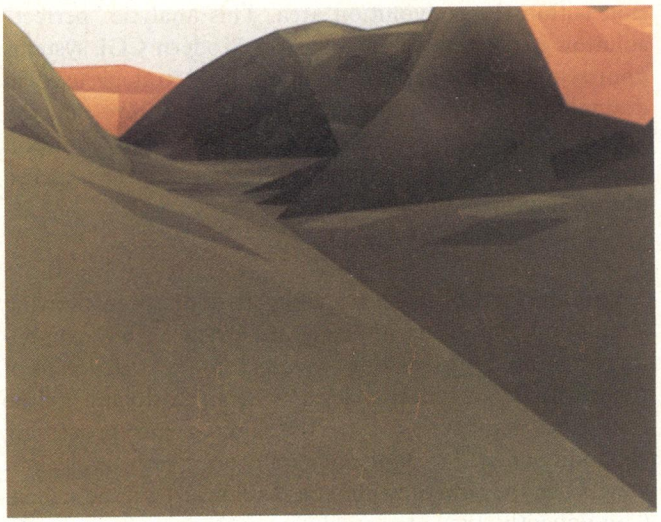


**Figure 15. Untextured scene.**

Texture-mapping capability can be combined with full translucency capability to produce more realistic images, such as the one shown in Figure 16. In the figure, the rugged outline of the tree is produced by a stored texture pattern resembling the outline of the tree; this texture pattern is then used to modulate the translucency of the underlying faces. The color modulation within the tree outline is produced by a stored texture pattern digitized from the photograph of a real tree.

Texture mapping using digitized realistic photographs as texture patterns is called photomapping. This technique has the potential of producing much more realistic computer-generated imagery than early CGI systems. The

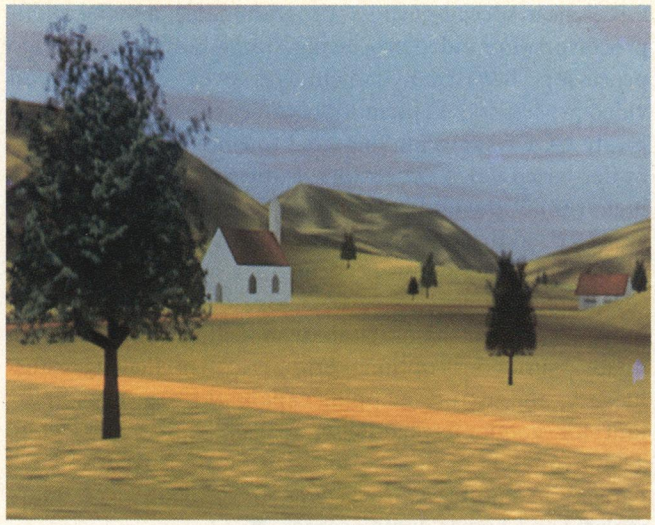


Figure 16. Texture and translucency effects combined.

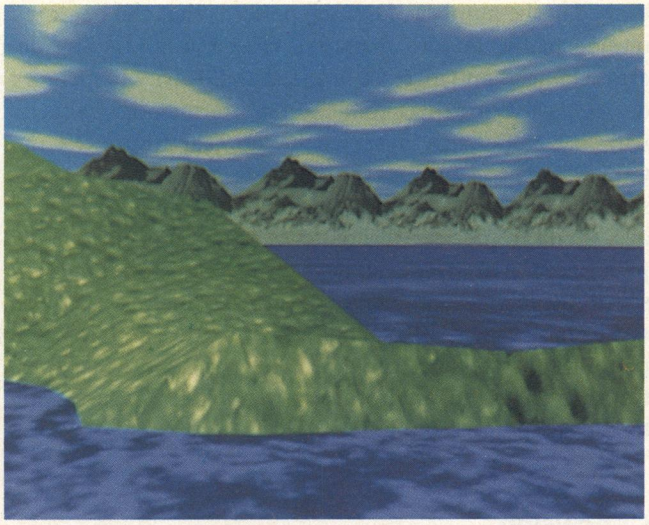


Figure 17. Backdrop mountain simulated with photomapping.

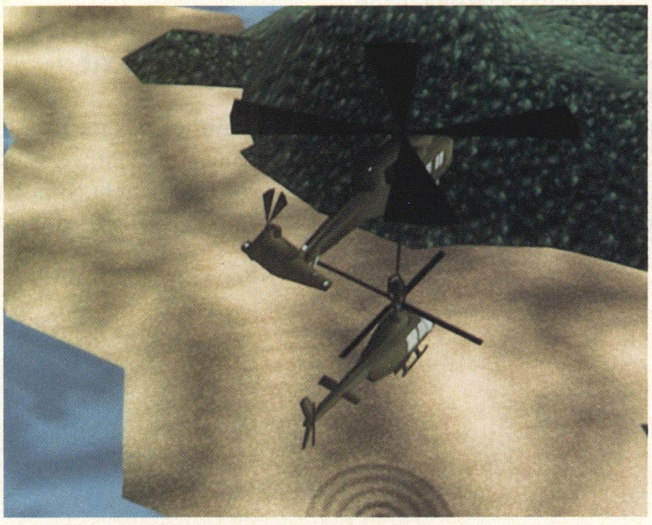


Figure 18. Rotor downwash simulated with dynamic texturing.

textured scene in Figure 14 was produced by mapping the orthographic aerial photograph of Yosemite Valley onto faces approximating the topography of that area. Another example of photomapping is shown in Figure 17, in which the backdrop mountain is simulated by mapping the photograph of a real mountain onto a vertical face. The realistic trees depicted in Figure 16 and the backdrop mountain in Figure 17 are just two examples of the special effects provided by texture and translucency. In fact, the combination of texture mapping and translucency capabilities is so powerful that their effective use is limited only by the imagination of the user. We expect that more special effects and realistic computer-generated imagery will be produced in the near future, using both texture and translucency.

In addition to modulating translucency, texture could also be used to modulate other parameters of faces on each pixel to simulate a variety of special effects. Moreover, texture modulation on faces can be dynamically changed in real time to simulate such dynamic special effects as rotor downwash (see Figure 18), explosion, and drifting smoke. Indeed, the potential of dynamic texture mapping for dynamic special effects is tremendous.

Most modern CGI systems with texture capability are

limited in the number of texture patterns that can be stored on-line in the texture memory. With the continuing decrease in price and increase in speed and density of memory devices, we can expect that the texture memory capacity will be significantly increased in the near future. Future CGI systems are expected to employ video disks as on-line secondary memory for photographic-texture-pattern storage; photographic texture patterns required at the current pilot position are retrieved in real time and stored in high-speed semiconductor memory for real-time access. With more texture memory available, more texture patterns with photographic realism can be stored, and we can expect future CGI systems to produce very realistic imagery using texture-mapping capability.

## Conclusion

Recent advances in computer-generated imagery for flight simulation have been discussed. Tremendous progress has been made since CGI was first used in flight simulation more than a decade ago. The advance in electronic technology and algorithms in recent years has allowed greater realism and scene detail in modern CGI systems than in early CGI systems. Consequently, CGI systems can be used to train pilots to fly more tactical missions than was possible a decade ago.

The trend in modern and future CGI systems is to implement the scene management and geometric processing functions in special-purpose, programmable, parallel multiprocessors. For video processing, frame-buffer-based video processors have many advantages over scan-line-based video processors. As memory becomes cheaper and denser, future CGI systems are expected to employ mostly frame-buffer-based video processors. Future display systems for high resolution and wide field-of-view applications are expected to be area-of-interest based to minimize the number of CGI systems needed.

Future CGI systems are expected to employ planar polygons as their major database modeling primitives, with quadric surfaces as optional primitives. Antialiasing, translucency, and texture-mapping capabilities are very important for good image quality, realism, and scene detail. These capabilities are expected to be improved and expanded in future CGI systems to provide even more realistic and higher quality scenes. ■

## Acknowledgment

The author wishes to thank R. Amador, G. Davis, S. Jiu-Wong, J. Guenther, E. Rosengaus, D. Wallner, and S. Yang of Link Flight Simulation for their contributions to the computer-generated imagery used in this article.

## References

1. B. J. Schacter, *Computer Image Generation*, Wiley Interscience, Wiley Press, New York, 1983, pp. 47-124.
2. J. W. Newhard and M. R. Nicol, "Selective Scene Management in Flight Simulator Visual Systems," Proc. Sixth Interservice/Industry Training Equipment Conf., Nov. 1984, pp. 11-21.
3. I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A Characterization of Ten Hidden Surface Algorithms," ACM Comput. Surv., May 1974, pp. 1-55.
4. J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison Wesley, Reading, Mass., 1982, pp. 268-318.
5. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, 1979, pp. 333-354.
6. R. Latham, "Image Generator Architectures and Features," Proc. Fifth Interservice/Industry Training equipment Conf., Nov. 1983, pp. 19-26.

7. W. L. Chen, "Night Calligraphic Digital Visual System," SPIE, vol. 59: *Simulators and Simulation*, 1975, pp. 40-47.

8. R. Bunker and R. Fisher, "Considerations in an Optimal Variable Acuity Display System," Proc. 1984 IMAGE Conf. III, May 1984, pp. 239-251.

9. H. M. Tong and R. A. Fisher, "Progress Report on an Eye-Slaved Area-of-Interest Visual Display" Proc. 1984 IMAGE Conf. III, May 1984, pp. 279-294.

10. J. D. Basinger, J. M. Wilson, and R. A. Fisher, "The Technical Contribution of the Tactical Combat Trainer Development Program," Proc. Fourth Interservice/Industry Training Equipment Conf., Nov. 1982, pp. 217-230.

11. F. B. Neves, "Design Considerations for an Eye-Tracked AOI Display System," Proc. 1984 IMAGE Conf. III, May 1984, pp. 255-266.

12. R. W. Fisher, et al., *Variable Acuity Display Development*, Basic Program, USAF (AFAL) TR77-156 Vol. I, Aug. 1977, McDonnell Aircraft Co., St. Louis, Mo. 63166.

13. R. Kruk and T. Longridge, "Binocular Overlap in a Fiber-optic Helmut Mounted Display," Proc. 1984 IMAGE Conf. III, May 1984, pp. 363-378.

14. H. Gouraud, *Computer Display of Curved Surfaces*, TR 113, U. of Utah, Comp. Sci. Dept., June 1971.

15. G. Y. Gardner, E. P. Berlin, and R. M. Gelman, "A Real-Time Generation System Using Textured Curved Surfaces," Proc. Image Generation/Display Conf. II, June, 1981, pp. 59-76.

16. J. K. Yan, "Real Time Generation and Smooth Shading of Quadric Surfaces," Proc. First Interservice/Industry Training Equipment Conf., Nov. 1979, pp. 247-260.

17. G. Y. Gardner, "Simulation of Natural Scenes Using Textured Quadric Surfaces," Proc. SIGGRAPH 84, July 1984, pp. 11-20.

18. E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*, U. of Utah, Comp. Sci Dept., UTEC-CSc-74-133, Dec. 1974.

19. J. K. Yan, "Computer Generation of Curvilinear Objects," Proc. Second Interservice/Industry Training Equipment Conf., Nov. 1980, pp. 37-45.

20. N. S. Szabo, "Digital Image Anomalies: Static and Dynamic," SPIE, Vol. 162: *Visual Simulation and Realism*, Aug. 1978, pp. 11-15.

21. E. Catmull, "An Analytic Visible Surface Algorithm for Independent Pixel Processing," Proc. SIGGRAPH 84, July 1984, pp. 109-115.

22. R. A. Schumacker, "A New Visual System Architecture," Proc. Second Interservice/Industry Training Equipment Conf., Nov. 1980, pp. 94-101.

23. W. M. Bunker, "Filtering Simulated Visual Scenes—Spatial and Temporal Effects," Proc. Fourth Interservice/Industry Training Equipment Conf., Nov. 1982, pp. 531-540.

24. F. P. Lewandowski, D. Hinkle, and W. Tucker, "Digital Visual Special Effects," Proc. Second Interservice/Industry Training Equipment Conf., Nov. 1980, pp. 84-91.

25. J. K. Yan, "Texture Generation in Real-Time Computer-Generated Imagery," Course notes for "High Performance Image Generation Systems," SIGGRAPH 85, July 1985.

**Johnson K. Yan** is manager of research and development for Digital Image Generation with the Link Flight Simulation Division of the Singer Company. His research interests include real-time computer-generated imagery, ditigal image processing, pattern recognition, and applications of VLSI technology to computer-generated imagery.

Yan received his BSEE from the University of Texas at Austin and his MSEE from the University of California at Berkeley. He is a member of IEEE and ACM.

Yan can be contacted at Link Flight Simulation Division, Singer Aerospace and Marine Systems, 1077 E. Arques Ave., Sunnyvale, CA 94088.