KNOWLEDGE-BASED AUGMENTED REALITY

Steven Feiner

Blair MacIntyre

Dorée Seligmann

The term *virtual reality* is often used to describe systems that attempt to replace much or all of the user's experience of the physical world with synthesized 3D material such as graphics and sound. Current systems typically use head-mounted displays that track the user's head position and orientation, and instrumented gloves that allow the user to manipulate virtual objects manually, together providing a sense of immersion in a surrounding virtual world. A well-designed virtual reality can make it possible for one or more suitably outfitted users to experience a world that does not exist or one that exists at another time or place.

There are many situations, however, in which we would like to interact with the surrounding real world. An *augmented reality* can make this possible by presenting a virtual world that enriches, rather than replaces, the real world. Instead of blocking out the real world, this approach annotates reality to provide valuable information, such as descriptions of important features or instructions for performing physical tasks. A complementary ap-

proach, known as *ubiquitous computing* [25], embeds large numbers of computers and displays in the world so they become an inextricable and socially invisible part of our surroundings. Similarly, information may also be projected on and read from the environment using projection displays and cameras [26].

By extending this vision to include small, lightweight, comfortable "see-through" (and "hear-through") wearable displays, we can craft a personalized (and, where appropriate, private) augmented reality whose user interface is not restricted to the displays and interaction devices embedded in the surrounding world or held in the user's hands. Ivan Sutherland, in his pioneering research on head-mounted displays that inspired current virtual reality systems, developed the first binocular see-through system [24]. Each eye viewed a miniature vector CRT, whose synthesized graphics were merged with the user's view of the real world by means of an optical beam splitter. Since then, other graphics researchers have explored the use of see-through displays, both as part of head-mounted displays (e.g., [1, 5, 13]), and desk-mounted displays (e.g., [17, 22]).

One of the most promising potential applications of augmented reality is to provide explanations of, and assistance with, complex 3D tasks. This could be accomplished by highlighting objects to call attention to them, using callouts and leader lines to name objects and arrows to guide the user in manipulating them, rendering otherwise invisible structures so they can be seen through objects that block them, and presenting abstract information near the physical objects it describes. Designing such an information presentation takes significant skill and time. As the richness and variety of the information a system can present to a user increases, so does the difficulty of presenting it well. For example, desktop publishing systems require more skill and time to master than do simple word processors, while multimedia presentation editors demand yet more design expertise to use effectively. Perhaps the ultimate design demands are posed by augmented realities, which can require the coordinated design of material that affects all sensory modalities, and that must respond continuously to the user's interactions. Furthermore, if this supplementary material were generated entirely in advance, it could not take into account information about the specific user and task, such as where the user is looking or the present state of the objects in the world.

We believe these issues can be addressed by developing knowledge-based systems that automate the design of presentations that explain how to perform 3D tasks. An automated presentation design system could create a presentation that was customized for an individual user and situation, providing the exact information that was needed. Here, we discuss KARMA—Knowledge-based Augmented Reality for Maintenance Assistance—a testbed system for exploring the automated design of augmented realities that explain maintenance and repair tasks. This research concentrates on the knowledge-based generation of virtual worlds that overlay and complement the user's view of the real world, and that dynamically take into account information about the user, task, and changes in the real world. It builds on our previous work on generating static and animated graphics [7, 16, 23], and multimedia presentations [9] that satisfy a high-level statement of the information to be communicated.

Knowledge-Based Graphics Component

The knowledge-based graphics component we use is based on IBIS (Intent-Based Illustration System) [23]. IBIS is a rule-based system that designs *illustrations*, a term we use to refer to pictures that are designed to satisfy an input communicative intent. The communicative intent is specified by a prioritized list of *communicative goals*. Each communicative goal specifies something the picture is to accomplish; for example, to show a property of an object, such as its location or shape, or to show a change in a property. IBIS is initially provided with its communicative goals along with representations of the physical objects that it will depict.

IBIS distinguishes between design and style. The *design* of an illustration corresponds to its high-level structure, specifying those visual effects that must be accomplished together to satisfy the illustration's communicative goals; the *styles* used in an illustration represent the different ways each visual effect may be accomplished.

IBIS uses two kinds of rules: methods and evaluators. A *method* specifies how to accomplish a particular style or design; an *evaluator* determines whether a particular style or design has been accomplished. IBIS's rules allow it to examine partial solutions and to backtrack when conflicts occur. The rule base is organized so that communicative goals are achieved by design rules that decompose communicative goals into a set of lower-level goals called *style strategies*. Design rules consist of design methods, each of which attempts to accomplish a communicative goal by invoking a set of style strategies, and design evaluators, each of which attempts to evaluate the success of a communicative goal by evaluating the success of a set of style strategies. Style strategies are achieved by style rules consisting of style methods and style evaluators, each of which respectively attempts to accomplish or evaluate a style strategy by invoking procedures that directly access the illustration data structure.

The illustrations that IBIS generates are dynamic rather than static: IBIS can continuously receive and handle changing constraints and goals, while at the same time realizing a particular intent. For example, IBIS normally determines a viewing specification of its own when designing an illustration. If the viewing specification is provided externally, however, then IBIS attempts to use it in generating a picture that satisfies the goals, and will incrementally redesign the picture to maintain the goals if the viewing specification changes. This allows IBIS's user to navigate within the illustrated world by changing the viewing specification.

IBIS's navigation facility is different from that used in traditional "synthetic camera" graphics in that the binding between the input intent

IBIS determines the presence and appearance of the objects in the display list—all information for which it has not explicitly relinquished control to the head and object servers.

and the output illustration is preserved. IBIS continuously examines the illustration as it is altered to attempt to satisfy the illustration's communicative goals. As the user changes the viewing specification, conflicts may occur causing IBIS to adopt a new design or set of stylistic choices. Thus, the illustrated world itself may change as the user explores it.

As a simple example of how IBIS works, consider the need to maintain object visibility. In the course of satisfying the input communicative goals, IBIS will determine that certain objects must be visible, and therefore *unoccludable*. This typically occurs if the objects participate directly in a communicative goal, such as objects whose location is to be shown. Unoccludable objects must not be obscured by others in the world. IBIS can maintain the visibility of unoccludable objects by selecting an appropriate viewing specification for the illustration, by generating an inset subillustration, or by altering the appearance of the obscuring objects. For example, IBIS can decide not to render the obscuring objects at all, to render them as partially transparent, or to use a cutaway view [10].

Extending IBIS for Augmented Reality

In extending IBIS to support augmented reality, we have to take into account a number of important differences:

- The original IBIS assumes that it generates all of what the user sees. In an augmented reality, however, IBIS must instead enrich the user's view of the real world with additional information.
- The user could modify the viewing specification after the original IBIS chose an initial viewing specification. In contrast, when designing an overlaid virtual world, our extended IBIS must from the beginning relinquish to the user all control of the viewing specification, since only the user can determine where to look.
- The world was assumed to change only after an illustration was completed in the original IBIS. This was easy to enforce since IBIS maintained control over what was visible, and based its illustrations on a world model that was frozen throughout the illustration's life. In contrast, our extended IBIS must take into account ongoing changes in the real world as it designs the virtual world.
- The original IBIS was responsible for achieving all communicative goals by itself. Because the extended IBIS has only partial control of what the user sees, the user becomes an active participant in achieving the communicative goals. For example, if a goal is to specify an object's position, and the object does not lie within the view volume, IBIS must indicate to the user where to look.

Based on these differences, we have developed a new set of rules to handle the input communicative goals. As in the original IBIS, low-level procedures add representations of objects to the illustrated virtual world, including physical objects and metaobjects (such as arrows, textual callouts, and leader lines), and set drawing styles. Other procedures evaluate the relationship of an object to the view volume, and determine whether an object is obscured from the user by other objects.

Testbed Application: End-User Laser Printer Maintenance

To test our ideas, we have been experimenting with a simple end-user maintenance application for a laser printer. This entails relatively straightforward operations, such as refilling the paper tray and replacing the toner cartridge.

Our system is shown in Figure 1. The user is wearing an experimental see-through head-mounted display that we built using a Reflection Technology Private Eye [21]. The Private Eye is a small display that generates a 720-×280 resolution, red, bilevel virtual image that can be focused to appear to be at a distance from 10″ to infinity. We directed the display downward and mounted a mirror beam splitter at a 45° angle relative to the display, so that when the user looks through the beam splitter, the user's view of the world is combined with the display's image. The large white triangles toward the top and bottom of the figure are transmitters for a Logitech 3D position and orientation-tracking system [19]. Each transmitter has three small ultrasonic sources near its corners. The small triangle on the head-mounted display and the two small triangles on the laser printer are the tracker receivers, each of which contains three microphones. The top transmitter is for the head; the bottom transmitter is for the sensors that track the printer's paper tray and lid. Our head-mounted display is strictly a research prototype: it produces a relatively dim image, and overlays graphics on a narrow portion of the user's visual field, providing an approximately 22° horizontal field of view. Nevertheless, it has proved to be a useful research tool.

Figure 2 illustrates augmented reality designed by KARMA and photographed through the head-mounted display. It fulfills three communicative goals: a *show* goal for the toner cartridge, and *location* and *identify* goals for the paper tray. To achieve the show goal, IBIS's design rules invoke the *visible* style strategy for the toner cartridge. If the cartridge were to lie within the view volume and were not occluded by other objects, then this strategy would succeed without doing anything: the cartridge would not be rendered, since it is already visible in the real world. (In contrast, in the original IBIS, the cartridge would have to be

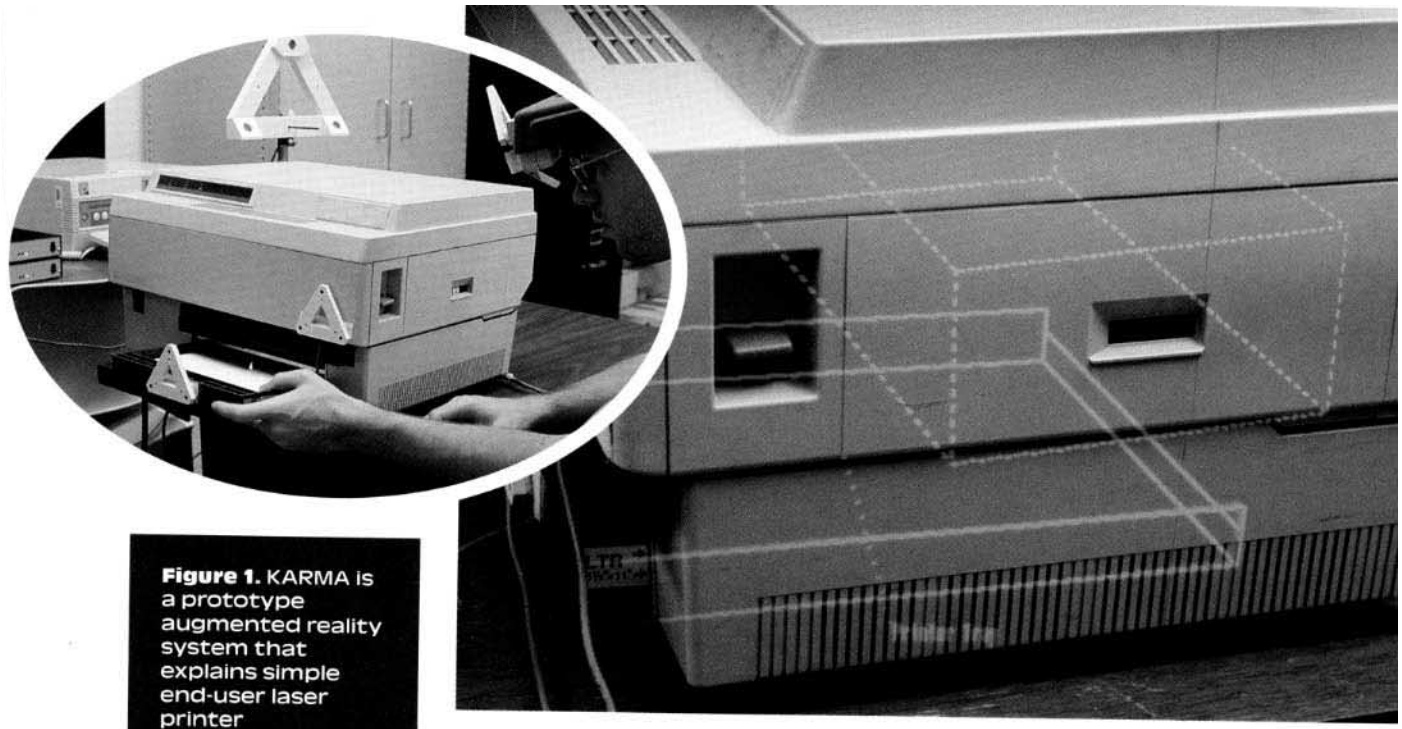


Figure 1. KARMA is a prototype augmented reality system that explains simple end-user laser printer maintenance using a see-through head-mounted display.

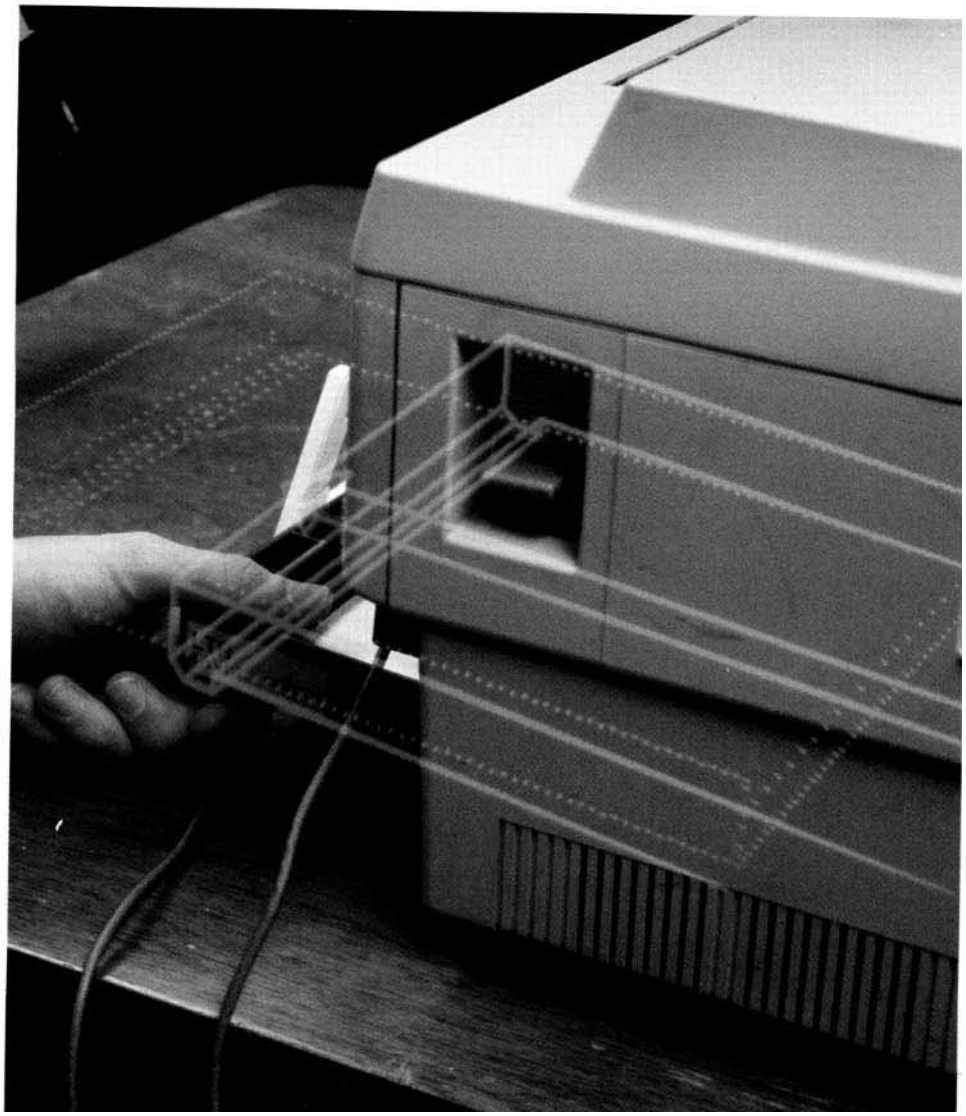
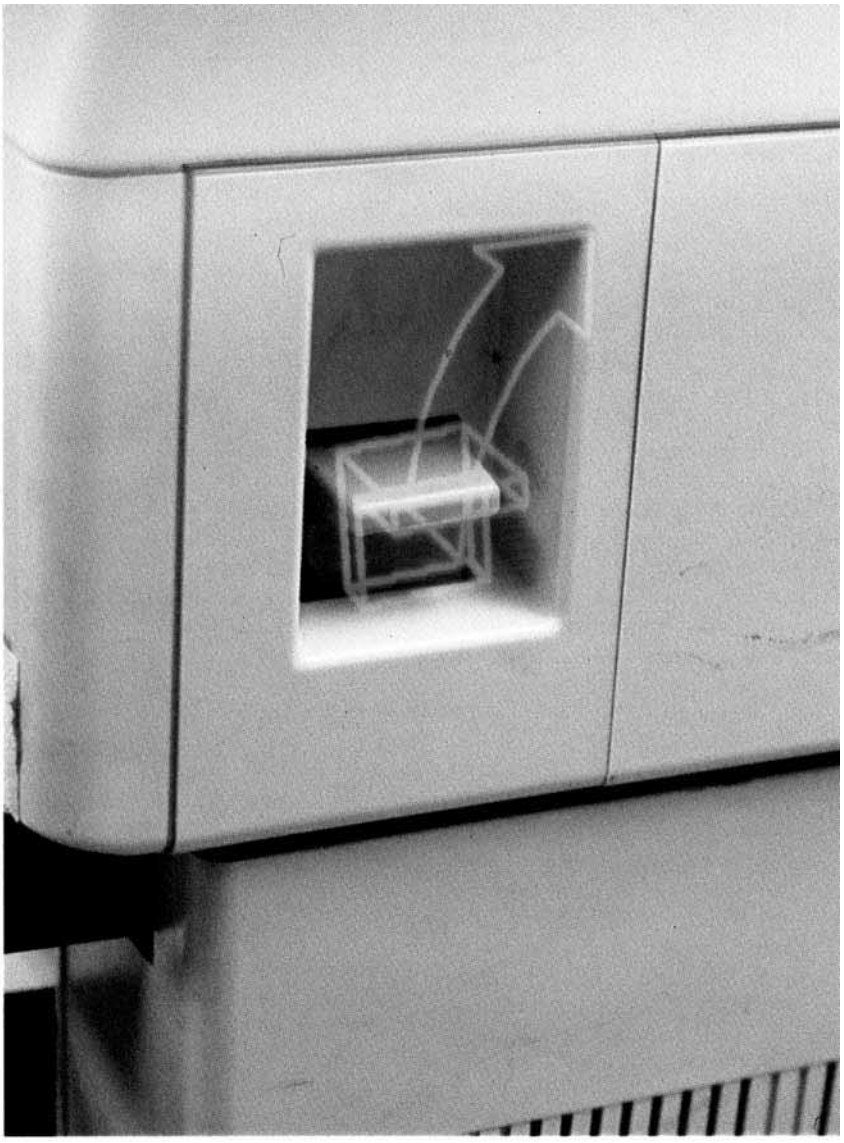


Figure 2. Augmented reality intended to *show* toner cartridge and show *location* of and *identify* paper tray. (Designed by KARMA.)

Figure 3. Augmented reality intended to show *action* of pulling out paper tray and resulting *change* in tray's state. (Designed by KARMA.)



rendered.) However, in the situation shown in Figure 2, while the cartridge is within the view volume, 3D geometric processing reveals that it is occluded by other objects (e.g., the printer cover). Therefore, another style rule specifies that the cartridge should be marked for rendering to allow it to be seen through its occluders. (The dashed-line rendering style chosen in this case is determined by additional style rules.) If the cartridge did not intersect the view volume, the *visible* style strategy would fail and other design rules would invoke an alternative style strategy to help the user find the cartridge.

To achieve the *location* goal for the tray, a design rule specifies that a *highlight* strategy should be invoked. The style rule that accomplished this strategy specifies that if the tray is within the view volume, it should be marked for rendering and tagged with a highlighted style (e.g., solid lines). To achieve the *identify* goal, *visible* and *label* style strategies are invoked for the tray. The labeling approach depends on the relationship of the tray to the view volume. If the tray were to reside completely within the view volume, which it does not, a textual label would be placed at the tray's center; otherwise, as is the case here, a label is fixed in the 2D space of the display and a dotted rubber-band leader line is drawn from the label to the tray's center. (If the tray did not intersect the view volume, then the dotted line would extend to the edge of the display in the direction of the tray, and could be followed by the user to find the tray; the callout and its end of the leader line are always in view.)

Figures 3 through 5 show a series of simple tasks prescribed by a con-

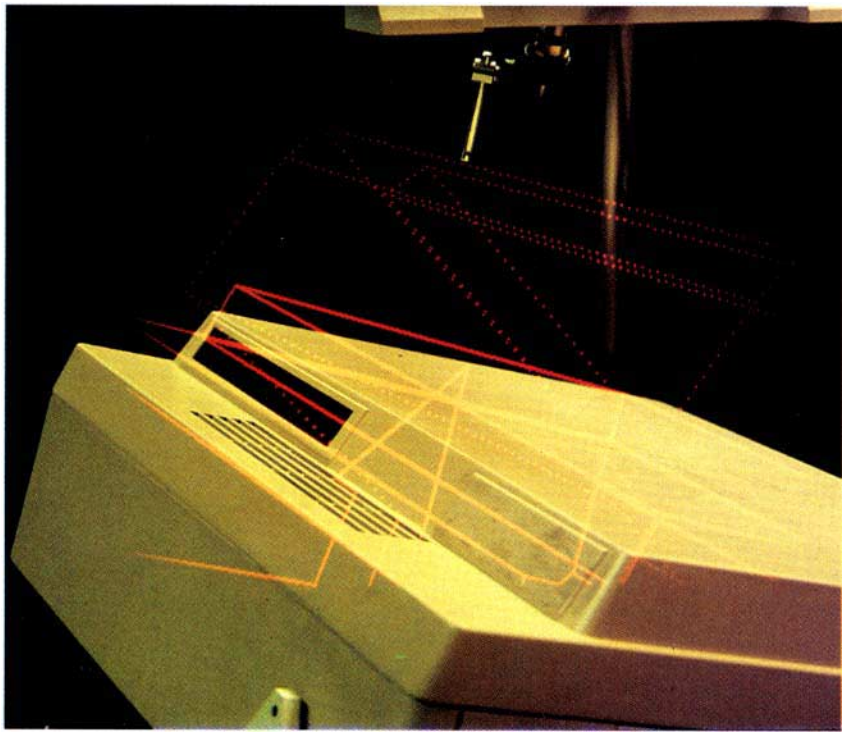


Figure 4. Augmented reality intended to show *action* of pulling up lid's lever. (Designed by KARMA.)

Figure 5. Augmented reality intended to show *action* of lid and resulting *change* in lid's state. (Designed by KARMA.)

KARMA represents our first steps in designing a testbed for the knowledge-based generation of maintenance and repair instructions using a head-mounted, see-through display.

tent planner that presents IBIS with communicative goals to satisfy. Figure 3 is designed to show the *action* of pulling out the paper tray and the resulting *change* in its state. The *action* goal is realized by activating *highlight* and *move* style strategies for the tray. Satisfying the *highlight* strategy causes the tray to be rendered with solid lines. The *move* strategy causes an arrow metaobject to be added to the illustrated world to depict the tray's trajectory. The *change* goal is realized by activating the *ghost* style strategy, which depicts the tray's desired state after the action has been completed. The *ghost* style strategy is realized by style rules that cause a model of the tray in the destination state to be marked for rendering in a "ghosted" style (in this case, using dotted lines).

The user is next instructed to pull up the lid's lever in Figure 4 (only an *action* goal is assigned here, not a *change* goal, so a ghosted lever is not shown). When the lid itself starts to move up, the user is asked to open the lid fully (by showing the *action* of opening the lid and the resulting *change* in its state), as shown in Figure 5.

System Architecture

KARMA's graphics must change in response to head motion and must be registered with objects in the real world. In addition, we also need to process data from a number of motion trackers. Based on our own experience [8] and that of other researchers [2, 14, 18], it was clear to us that it would be necessary to distribute processing over multiple processes and processors.

We created a small 3D structured display-list–based display server. The server supports line, polygon, and text primitives, linestyle, fillstyle, and font attributes, and a set of structure management operators that allow hierarchical objects to be created, edited, and deleted. The server's move and draw commands allow 2D device coordinates and 3D world coordinates to be intermixed in a single primitive, making it possible to create lines that are anchored to the screen on one end, and to a point in the 3D world on the other. As shown in Figure 6, when the system is initialized, IBIS creates the initial display list by sending a set of object models to the display server.

Each tracker is handled by a low-level tracker process. These processes in turn interact with a set of object servers and a head server. Each object server represents a tracker that may be associated with an object in the real world. At initialization, IBIS provides each object server with the identifier of the display server structure containing the object's vector representation. It also tells the server the position and orientation of the object's tracker relative to the object's coordinate system. Similarly, IBIS provides the head server with the identifier associated with the scene in the display server. (The head and object servers actually use the same executable, so each must also be told whether it is tracking the head or an object.)

Figure 7 shows the system's operation after initialization. The head and object servers are responsible for maintaining the integrity of the object and head motion information that they represent. They edit the display list stored in the display server directly. Each object server regularly edits the display list position and orientation information associated with its object. The head server updates the display list viewing specifications for the scene. Both head and object servers also report their information to IBIS. This avoids the delay that would result if IBIS were to serve as a go-between, making possible relatively smooth visual response to head and object motion, while assuring that IBIS always has the latest information on which to base its illustration design.

IBIS determines the presence and appearance of the objects in the display list—all information for which it has not explicitly relinquished control to the head and object servers. This includes the specification of metaobjects, such as arrows and text. (IBIS can also dynamically reassign tasks to the head and object servers; for example, to make it possible to detach and reattach trackers to change the objects being tracked.) In work on the automated design of multimedia presentations, IBIS is presented with a set of communicative goals to satisfy, which are output by other components that determine what to say and which media should be used to say it [9]. KARMA currently uses a much simpler content planner to generate the set of communicative goals that IBIS receives.

IBIS first designs an illustrated world that satisfies the initial set of goals set by the content planner, obeying the constraints imposed by the head and object trackers. Then it loops, using its evaluators to determine whether the current illustration design is satisfactory in the face of changes in the goal set, the user, and the world. Whenever the illustrated world is determined to be unsatisfactory, IBIS's rules result in modifications that are communicated to the display server. For example, if a labeled object that KARMA determines must be identified no longer lies entirely within the view volume, then IBIS will generate a new 2D screen label and 3D leader line that points to the object.

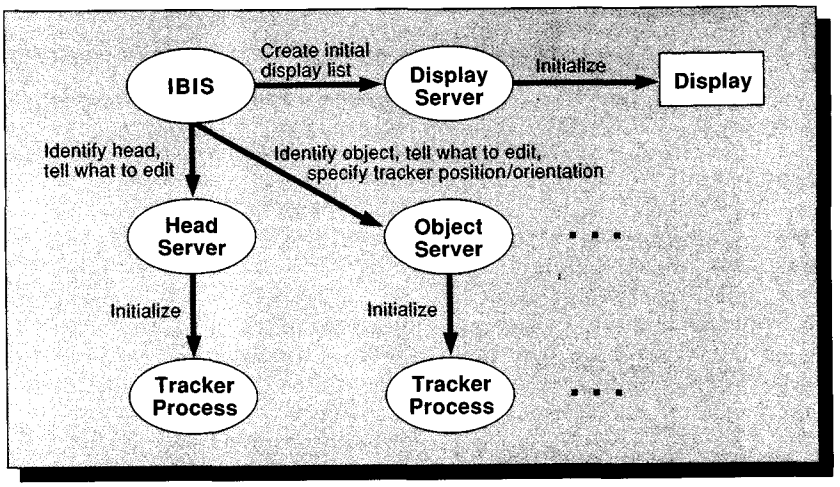
Because IBIS has relinquished to the head and object servers the straightforward matters of display-list traversal and updates of the viewing specification and monitored object transformations, user interaction with the current illustrated world stored in the server is fully interactive, and occurs while incremental

redesign takes place on a separate processor.

The head server is responsible for instructing the display server to render the image. Since polling the head's motion tracker requires much less time than rendering the image, we need to avoid building up a backlog for the display server. We accomplish this by having the head server send its commands for the current frame to the display server and then wait until it receives the display server's acknowledgment that the previous frame was rendered. This allows rendering and polling to proceed in parallel. All of the actions requested of the display server (e.g., modifying the contents of an object) are processed atomically to ensure that the scene always appears in a valid, drawable state.

Implementation

KARMA's components run on several different machines under different flavors of Unix, and communicate through sockets. IBIS is implemented in C++ and the CLIPS production system language [6], and runs under HPUX on an HP 9000 380 TurboSRX graphics workstation, which provides a fast hardware z-buffer–based graphics accelerator that IBIS uses in its illustration design process. The display server is written in C and runs under Mach on a 50MHz Intel 486DX-based PC that supports the Private Eye display entirely in software. We currently achieve about 15 frames per second, double-buffered, for a scene containing well over 100 displayed 3D vectors. A significant portion of the display server's time is spent implementing double-buffering, copying the graphics frame buffer to the Private Eye's frame buffer and clearing it for the next frame. To avoid the substantial overhead of writing or reading the memory-mapped device frame buffer, we scan convert into two buffers in main memory that contain the current and previous frames, compare the two buffers, and copy to the device frame buffer only those bytes that have changed from the previous frame. The head and object server and the lower-level tracker processes are written in C and C++. Since the tracker hardware requires only a serial interface, and the servers can impose a large load on the machine on which they execute, we run them on other workstations.



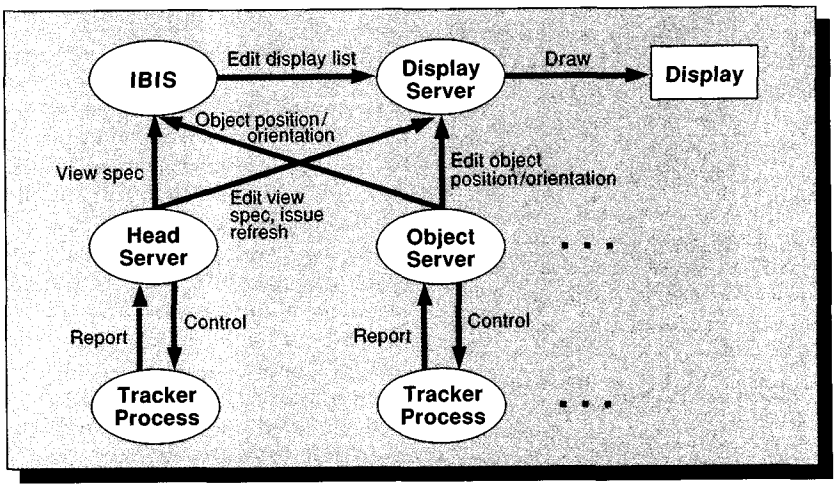


Figure 6. System architecture: Initialization

Figure 7. System architecture: Steady state

Our current trackers include three Logitech ultrasonic sensors and four Ascension Technology magnetic sensors. Our software allows all trackers to be used interchangeably, for example to trade off an ultrasonic tracker's freedom from magnetic interference against a magnetic tracker's ability to work without a direct line of sight to its source.

One important point, that others have previously noted [3], is that experimental interfaces that are coupled tightly to the user often require a fair amount of calibration. Our head-mounted display is no exception. Since the generated image must be registered with that of the real world, each user must perform three kinds of calibration:

One of the most promising potential applications of augmented reality is to provide explanations of, and assistance with, complex 3D tasks.

• *Focus.* The Private Eye is focusable from 10" to infinity, with each user requiring separate focus adjustments based on their eyesight. After putting on the display, the user must position a slider until a calibration image is comfortably in focus.

• *Visible area.* Due to the physical relationship between the Private Eye, the beam splitter, and the current focus setting, a small portion of the display may not be visible to a viewer. Therefore, we request that the user determine the viewable area by adjusting the size and position of a visible rectangle until it is as large as possible. This establishes a "safe" area whose dimensions are communicated to IBIS, in which IBIS can assume that anything drawn will be visible.

• *Viewing specification.* To register the image with the world, the user must first physically adjust the display on his or her head. Next, the user must register a virtual object with its corresponding physical object, which in turn must be viewed in a known position relative to the user. The software takes into account the difference in position and orientation of the user's eye and the measured position and orientation of the sensor.

As Figures 2 through 5 indicate, registration is a serious issue. Our current motion trackers have neither the spatial nor angular resolution needed to register graphics precisely with the surrounding world. However, we have yet to try mapping sensor inaccuracies to correct for nonlinearities [22], and believe we can also improve accuracy with a better calibration strategy.

We have also found focus to be a particular problem in KARMA. We originally built our head-mounted display for a *hybrid user interface,* which embeds the user's view of a notebook flat-panel display inside a partial spherical information surround presented on the head-mounted display [11]. In that application, the flat-panel display is relatively small and is tangent to the virtual sphere, which is centered about the user's head. Assuming that the user's head is stationary, it is easy to adjust the Private Eye so the material it displays is focused at the same distance from the user as the flat-panel display. In comparison, "heads up" flight displays are focused at infinity [20], since this is effectively where out-of-the-cockpit targets are located.

In contrast to both these applications, KARMA requires that graphics be overlaid on nearby objects that necessitate constant changes in visual accommodation to focus in sequence. Since the Private Eye must be focused manually, the need for readjustment as the viewing distance to the object of interest changes is irritating.[1] There is an interesting benefit, however, to the precise focus control provided by the Private Eye. Even though we are currently using a monocular display, when the display is adjusted so that a small synthesized object is in focus at a particular distance, the illusion of it being at the selected position is quite compelling. (Note that one of the difficulties many users experience in viewing fixed focus stereo displays is developing independence between their ocular convergence and focus accommodation.)

Conclusions and Future Work

KARMA represents our first steps in designing a testbed for the knowledge-based generation of maintenance and repair instructions using a head-mounted, see-through display. We have developed a preliminary set of rules that allow us to augment the user's view of the world with additional information that supports the performance of simple tasks such as finding designated objects and carrying out simple actions on them. Our software architecture enforces a clean distinction between design and rendering to help prevent design decisions from interfering with interactive rendering.

Our experience with KARMA has suggested many research directions that need to be explored. For example, one important problem is the development of a formal model of how a user's performance will be affected by different decisions made in designing 3D illustrations, taking into account the purpose for which the illustration is generated (specified by our communicative goals), as in the 2D design work of Casner [4].

Support for visible-line and visible-surface determination is another issue. IBIS currently bases its illustration design in part on whether selected objects are occluded in the current viewing specification, and computes these relationships itself. Our display server, however, does not support visible-line determination. We are particularly interested in incorporating into the display server what Kamada and Kawai [15] refer to as "picturing functions," which determine how a projected line fragment should be rendered, based on the set of surfaces that obscure it.

For example, while IBIS currently sets the graphical attributes of an entire object based in part on its visibility, we would like more precise control to be accomplished by the display server, based on the visibility relationships of each line fragment. IBIS would then be responsible for determining the high-level policies used by the display server (e.g., making obscured parts dashed). One challenge is to do this while still maintaining real-time performance. We believe this could be possible on our current hardware if IBIS were allowed to select a subset of objects against which the display server would perform visibility tests. An-

[1] We have considered automating the process by using a servomotor to adjust the focus to that of a selected object. Note, however, that focus in the entire overlay would be affected uniformly.

other intriguing research direction is to explore how to design support facilities that would allow IBIS to specify a rich set of additional high-level policies to be enforced by the display server.

As previously mentioned, KARMA must involve the user in achieving those communicative goals that require directing their attention toward parts of the world that may or may not be visible. While this is currently accomplished through graphics alone, we are starting to explore how graphics in combination with speech and nonspeech audio, localized in 3D using a spatial sound processor, can be used to tell the user where to look and what to look for.

Finally, we note that advances in display technology will soon make possible wearable see-through displays that are far smaller and lighter, yet have higher resolution and a wider field of view than our current prototype. Used in conjunction with more accurate tracking technologies, including sensors built into the equipment, we believe the resulting systems will become the method of choice for explaining complex physical tasks.

Acknowledgments

Our work owes much to the portable computing infrastructure software developed for the Columbia Student Electronic Notebook project directed by Daniel Duchamp, Steven Feiner, and Gerald Maguire, and funded by a contract from IBM. Thomas Magdahl built the camera mount used to take pictures through our head-mounted display. We thank Ari Shamash and Sushil Da Silva for their work on the tracker process and bitmap scan-conversion package, Brad Paley of Digital Image Design for sharing his tracker support code, Cliff Beshers for his help in porting his distributed tracker server, and Jim Barnes, Bob Haya, and Rob Romero of Logitech for their assistance.

References

1. Bajura, M., Fuchs, H. and Ohbuchi, R. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. *Comput. Graph.* In *Proceedings of SIGGRAPH '92 26,* 2 (July, 1992), 203–210.
2. Blanchard, C., Burgess, S., Harvill, Y., Lanier, J., Lasko, A., Oberman, M. and Teitel, M. Reality built for two: A virtual reality tool. In *Proceedings of the 1990 Symp. on Interactive 3D Graphics, Comput. Graph. 24,* 2 (Snowbird, Utah, March 25–28, 1990), pages 35–36.
3. Brooks Jr., F. Grasping reality through illusion—Interactive graphics serving science. In *Proceedings CHI '88* (Washington, DC, May 15–19, 1988), pp. 1–10.
4. Casner, S. A task-analytic approach to the automated design of graphic presentations. *ACM Trans. Graph. 10,* 2 (Apr., 1991), 111–151.
5. Chung, J., Harris, M., Brooks, F., Fuchs, H., Kelley, M., Hughes, J., Ouh-young, M., Cheung, C., Holloway, R. and Pique, M. Exploring virtual worlds with head-mounted displays. In *Proceedings SPIE Non-Holographic True 3-Dimensional Display Technologies, Vol. 1083.* (Los Angeles, Jan. 15–20, 1989).
6. Culbert, C. *CLIPS Reference Manual (Version 5.0).* NASA Johnson Space Center, Information Systems Directorate, Software Technology Branch, Houston, Tex., 1991.
7. Feiner, S. APEX: An experiment in the automated creation of pictorial explanations. *IEEE Comput. Graph. Applic. 5,* 11 (Nov. 1985), 29–37.
8. Feiner, S. and Beshers, C. Worlds within worlds: Metaphors for exploring *n*-dimensional virtual worlds. In *Proceedings UIST '90, ACM Symposium on User Interface Software* (Snowbird, Utah, Oct. 3–5, 1990), pp. 76–83.
9. Feiner, S. and McKeown, K. Automating the generation of coordinated multimedia explanations. *IEEE Comput. 24,* 10 Oct. 1991, 33–41.
10. Feiner, S. and Seligmann, D. Cutaways and ghosting: Satisfying visibility constraints in dynamic 3D illustrations. *Visual Comput. 8,* 5–6 (June 1992), 292–302.
11. Feiner, S. and Shamash, A. Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers. In *Proceedings UIST '91, ACM Symposium on User Interface Software and Technology* (Hilton Head, S.C., Nov. 11–13, 1991), pp. 9–17.
12. Feiner, S., MacIntyre, B. and Seligmann, D. Annotating the real world with knowledge-based graphics on a see-through head-mounted display. In *Proceedings Graphics Interface '92* (Vancouver, Canada, May 11–15, 1992), pp. 78–85.
13. Fisher, S., McGreevy, M., Humphries, J. and Robinett, W. Virtual environment display system. In *Proceedings 1986 Workshop on Interactive 3D Graphics* (Chapel Hill, N.C., Oct. 23–24, 1986), pp. 77–87.
14. Green, M. and Shaw, C. The DataPaper: Living in the virtual world. In *Proceedings Graphics Interface '90* (Halifax, Nova Scotia, May 14–18, 1990), pp. 123–130.
15. Kamada, T. and Kawai, S. An enhanced treatment of hidden lines. *ACM Trans. Graph. 6,* 4 (Oct. 1987), 308–323.
16. Karp, P. and Feiner, S. Issues in the automated generation of animated presentations. In *Proceedings Graphics Interface '90* (Halifax, Canada, May 14–18, 1990), pp. 39–48.
17. Knowlton, K. Computer displays optically superimposed on input devices. *The Bell Syst. Tech. J. 56,* 3 (March 1977).
18. Lewis, B., Koved, L. and Ling, D. Dialogue structures for virtual worlds. In *Proceedings CHI '91* (New Orleans, La, April 27–May 2, 1991), ACM Press, pp. 131–136.
19. Logitech, Inc. Logitech 2D/6D mouse technical reference manual (Preliminary). Fremont, Calif., 1991.
20. Norman, J. and Ehrlich, S. Visual accommodation and virtual image displays: Target detection and recognition. *Human Factors 28,* 2 (1986), 135–151.
21. Reflection Technology. Private Eye Product Literature. Waltham, Mass., 1990.
22. Schmandt, C. Spatial input/display correspondence in a stereoscopic computer graphic work station. *Comput. Graph. 17,* 3 (*Proceedings SIGGRAPH '83* July 1983), 253–261.
23. Seligmann, D. and Feiner, S. Automated generation of intent-based 3D illustrations. In *Proceedings ACM SIGGRAPH '91, Computer Graphics, 25,* 4 (Las Vegas, Nev., July 28–Aug. 2, 1991), pp. 123–132.
24. Sutherland, I. A head-mounted three dimensional display. In *Proceedings FJCC 1968,* Thompson Books, Washington, DC, 1968, pp. 757–764.
25. Weiser, M. The Computer for the 21st Century. *Sci. Am. 265,* 3 (Sept. 1991), 94–104.
26. Wellner, P. The DigitalDesk calculator: Tangible manipulation on a desk top display. In *Proceedings UIST '91, ACM Symposium on User Interface Software and Technology* (Hilton Head, S.C., Nov. 11–13, 1991), pp. 27–33.

CR Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Tools and Techniques—*User interfaces*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial realities*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems; I.3.2 [**Computer Graphics**]: Graphics Systems—*Distributed/network graphics*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Virtual reality*

General Terms:

Additional Keywords and Phrases: augmented reality, head-mounted displays, heads-up displays, knowledge-based graphics, portable computers, virtual reality, virtual worlds

About the Authors:

STEVEN FEINER is associate professor of computer science at Columbia University, where he directs the Computer Graphics and User Interfaces laboratory. His research interests include computer graphics and user interfaces, knowledge-based graphics, virtual worlds, visualization, animation, visual languages, and hypermedia.

BLAIR MACINTYRE is is a Ph.D. student in the department of Computer Science at Columbia University, where he works on graphical interfaces for see-through head-mounted displays. His research interests include human-computer interaction, color science, and computer graphics.

Authors' Present Address: Department of Computer Science, Columbia University, New York, NY 10027; email: feiner,bm,doree@cs.columbia.edu

DORÉE SELIGMANN is a member of technical staff, AT&T Bell Laboratories, Holmdel. Her Ph.D. dissertation work at Columbia University was on the IBIS knowledge-based graphics system and her research interests are in computer graphics illustration, visual languages, modeling artistic techniques, user interfaces, and multimedia communication. **Author's Present Address:** AT&T Bell Laboratories, Room 4F-605, Crawfords Corner Road, P.O. Box 3030, Holmdel, NJ 07733-3030. doree@research.att.com

Research on this project is supported in part by the Office of Naval Research under Contract N00014-91-J-1872, the Center for Telecommunications Research under NSF Grant ECD-88-11111, NSF Grant CDS-90-24735, and equipment grants from the Hewlett-Packard Company and Ascension Technology. Work on the original version of IBIS was supported in part by the Defense Advanced Research Projects Agency under Contract N00039-84-C-0165.