Hyper-Rendering

Jürgen Emhardt
Thomas Strothotte

Interactive Systems Lab
Department of Computer Science
Free University of Berlin
Nestorstraße 8-9, D-1000 Berlin 31, Germany
{emhardt, strothotte} @inf.fu-berlin.de

Abstract

Current systems for the automatic generation of information presentation or the automatic illustration of objects are mainly generation-oriented, i.e. they generate solutions to a user who in turn has only few possibilities to lead a dialogue with the generation system. By contrast, systems which emphasize dialogues tend to be weak in graphical interaction.
In order to build systems which are both generation- and dialogue-oriented, we present a new software architecture for computer graphics applications. We develop the concept of *hyper-rendering* which produces a formal description of the scene as viewed by the user and can either be carried out within a renderer or in a separate program. The output of the hyper-rendering program is an *image description* which consists of information which conventional renderers usually compute but typically throw away. We describe two broad categories of applications of hyper-rendering and a prototypical implementation. Examples of the pictures produced as well as a sample session with an application are included.

Keywords: rendering, image description, photorealistic images, virtual reality, multi-media, interactive systems.

1. Introduction

Today's graphics systems for producing photorealistic images consist of two main software components: a modeler with which a user can define a scene (model objects to be rendered, choose the camera position and perspective, etc.) and which produces as output a scene description; and a renderer, which accepts as input a scene description and produces as output the image. It is a fundamental characteristic of such graphics systems that the output of the renderer is designed exclusively for viewing by a (human) user. While the machine produces the image based on a scene description, no information is stored in general as to what a user can actually see. This lack of information makes it impossible to establish a link between the renderer and an application program, although a main purpose of producing photorealistic images is *communication*.

Indeed, rendering algorithms "waste" a great deal of information which could be gathered and represented explicitly. For example, hidden surface removal algorithms--as their name implies--throw away information about surfaces which are not visible to a user: But if a user wants to know which objects of the scene she or he *cannot* see? Such information pertaining to objective and subjective perception of the picture by the user and its relationship to the modeled scene is not made available systematically.

In this paper we propose that the usability of graphics systems can be enhanced greatly by making information about the rendering process explicitly available to other programs. We refer to this process of deriving such information as *hyper-rendering*. Hyper-rendering produces a formal description of the scene as viewed by the user and can either be carried out within a renderer or, as in our prototypical implementation, in a separate program. The latter approach has the advantage that separate algorithms can be used for rendering and hyper-rendering of the same image, especially when hardware-rendering is used. Furthermore, when an image is to be produced with ray-tracing it may suffice to carry out the hyper-rendering with a fast z-buffer algorithm. Indeed, in certain applications it may even suffice initially just to carry out hyper-rendering and only later to perform the more time-consuming rendering.

This paper is organized as follows: Previous research is surveyed in Chapter 2. In Chapter 3, we present a new software architecture for graphics systems with integrated hyper-rendering. Next, we point out two broad categories of applications of hyper-rendering and discuss the benefits to end-users. In Chapter 4, we survey the implementation techniques of our prototypical hyper-renderer. Chapter 5 gives an example of the capabilities of an application which uses this hyper-renderer and shows a sample dialogue session. Further work is discussed in Chapter 6.

2. Background

Our work on hyper-rendering is intended as a bridge between the generation-oriented systems which present "canonical" solutions to a mainly *passive* user, and the dialogue-oriented systems which rely on an *active* user and which are able to answer questions. A representative of the first category is APT (A Presentation Tool, [Mackinlay, 1986]). The system generates presentations of two dimensional relational information, where the two main criteria for the generation are *expressiveness* and *effectiveness*. Expressiveness criteria determine whether a graphical language can express the desired information, and effectiveness criteria determine whether a graphical language exploits the capabilities of the output medium and the human visual system. Thus, users get a large variety of "canonical" presentations fulfilling these two criteria. However, the techniques developed by Mackinlay are very difficult to extend to user-interaction, for example, for cases in which users wish to modify the presentation. The information about such a modification of the layout is difficult to report back to the generation system, which is in addition not able to judge the quality of the modification.

The WIP system (Knowledge-based Presentation of Information, [Wahlster et al., 1991]) generates a variety of multimodal documents from an input consisting of a formal description of the communicative intent of a planned presentation. The focal point of WIP is the *generation* of illustrated texts that are customized for the intended audience and situation, but interaction with the illustrations is not yet possible.

A representative of generation-oriented dialogue systems is the IBIS system of [Seligmann and Feiner, 1991] (see also [Feiner, 1985]). It is intended for the automatic generation of intent-based illustrations and shares several research interests with the WIP system, but differs in the system's architecture, for example. An *illustration* is a picture that is designed to fulfill a communicative intent such as showing the location of an object or showing how an object is manipulated. The design of an illustration is treated as a goal-driven process within a system of constraints. The system uses a generate-and-test approach which relies on a rule-based system of methods and evaluators. Methods are rules that specify how to accomplish a visual effect, while evaluators are rules that specify how to determine how well a visual effect is attained in an illustration. As users are able to manipulate illustrations interactively, IBIS maintains the methods of *visibility* and *recognizability* during an interactive session. The current implementation supports user-controlled view specification, that is, the user can zoom objects or specify a new camera location. However, there is no possibility for a user to *ask* anything, for example about the interior of a particular object or to find out through a dialogue where shadows are located. However, the visibility methods of the IBIS system do provide very simple information for further processing through application modules. For example, the evaluator which calculates whether an object which must be visible is partially obscured returns a binary value.

More flexibility is desirable for human-computer dialogues concerning the graphics, particularly in user interfaces using graphics for teaching purposes. Flexibility here refers not only to the viewing specification but also to supplementary information about the objects. To this end, [Strothotte, 1989] developed a prototypical chemistry explanation system which generates pictorial explanations automatically and is capable of leading a dialogue with the user. For example, as an answer to the question "How is N_2 produced?", several pictures showing the steps of the chemical production are presented. In this system, the user can manipulate the labels on the diagrams to obtain more information. However, the flexibility of the dialogues is attained at the expense of the quality of the graphics: Strothotte's system relies on handmade bitmapped images, that is, users are not able to modify the pictures presented.

Besides the research on the generation of information presentation, much work is carried out on algorithmic methods for computing the images to be presented. Improving the performance of image rendering can be done by using visibility precomputations, since by performing work off-line they reduce the effort involved in solving the hidden-surface problem. In particular, many spatial subdivision techniques have been proposed for speeding up ray tracing ([Weghorst et al., 1984], [Glassner, 1989]) as well as for preparing interactive walkthroughs through complex environments, for example ([Teller et al., 1991]). Another method to improve the performance of image rendering is to convey most of the information to the user as early as possible, with image quality constantly improving with time, that is, to render images by adaptive refinement ([Bergman et al., 1986]).

3. Working with a Hyper-Renderer

3.1 The Process of Hyper-Rendering

A software architecture for graphics systems using hyper-rendering is illustrated in Figure 1. The non-shaded boxes with a scene description, a renderer and the resulting picture are as used in conventional graphics systems. Conceptually, hyper-rendering software is built around the renderer. It computes various pieces of information about both the rendering process and the rendered picture, which conventional renderers either throw away or don't bother computing in the first place. This information is stored in an "image description file".

The primary purpose of hyper-rendering is to allow users to do more with the rendered pictures than just look at them; this is facilitated by various kinds of applications which take as input the image description. Depending on the application, the scene description may also be used or modified. An application thus has information about what the user can see in the picture and is able to handle the dialogue with the user about the picture, evaluate the picture under certain criteria or even modify it so as to change the visibility of the objects.

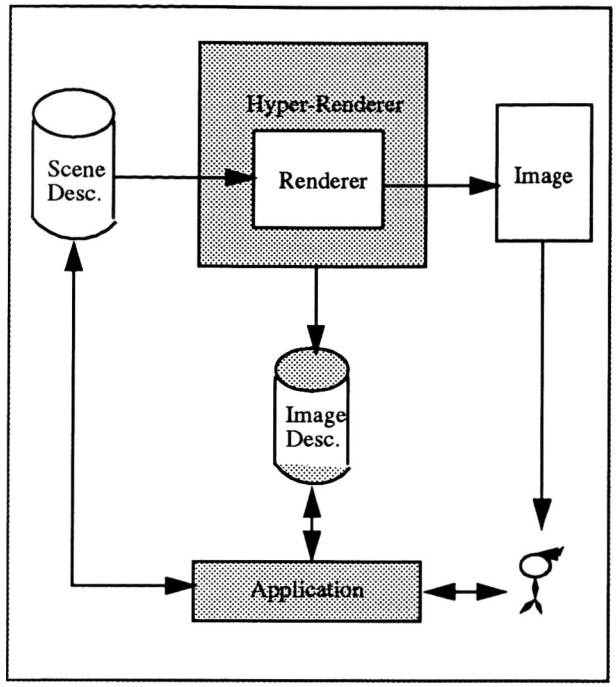

Figure 1: *A new software architecture for graphics systems with integrated hyper-rendering.* The hyper-rendering software is built around the renderer and computes various pieces of information about both the rendering process and the rendered picture, which conventional renderers do not provide for further processing.

3.2 Applications

We consider there to be two broad categories of applications of hyper-rendering. The first category pertains to applications with direct end-user involvement. If a user is to engage in a dialogue about the contents of the graphics with the machine, it is an absolute prerequisite that the application has at its disposal detailed information about what the user can see. This is particularly important in virtual reality or cyberspace applications, when the user will ask for information and explanations about certain objects in the scene. It should be particularly clear here that the scene description itself is not sufficient for the machine for the purposes of leading such a dialogue, since a user's input must be interpreted with respect to what he can see (or in some cases more importantly what he cannot see) rather than only with respect to the model of the scene. The hyper-renderer can also be used in teaching material, when a pupil will ask for information and explanations about certain parts of the graphics.

The second broad application of hyper-rendering is in situations in which a graphic shown to the user (or intended for display) must be evaluated with respect to its appropriateness by the machine. Linking an application program to a renderer without modifying the latter is not possible without storing the information the renderer calculates. With our architecture, it is in particular possible to build knowledge based systems which analyse the modeling of a scene with respect to given constraints, generate improvements, and prepare explanations for the user. For example, [Fischer et al., 1990] discusses the critiquing approach to building knowledge based interactive systems and describes a critiquing system for the 2D design of kitchens. A problem in 3D modeling, in particular when dealing with a virtual environment, is to avoid penetrating rigid objects or having them collide. As standard methods for computing collisions have high computational complexity, [Pentland, 1990] describes more efficient methods for the calculation of dynamics, collision detection, and constraint satisfaction. Another deficiency of scene modeling is when parts of the scene where the observer should focus on lie in shadow. In this case, an improvement generator can suggest better locations and parameters for the light sources.

It is important to note that hyper-rendering is *independent* of the modeling software employed and can be used to improve the modeling process indirectly in contrast to other approaches, where new modelers are developed (for example [Hall, 1991]). This is particularly important, as many deficiencies in scene modeling can only be determined with significant effort by modelers. For example, only a renderer knows where shadows are located and how they influence the perception of the image.

The hyper-renderer can, however, hardly be used to design good (or appropriate) presentations automatically. It can be used in the evaluation of a particular graphic with respect to communicative goals. If the evaluation is negative, other means must be found to correct the situation.

In both these categories of applications, the facilities of a hyper-renderer provide vital input to programs using photorealistic graphics as a communicative tool. The success of such applications depends on the sophistication with which the hyper-renderer works to obtain information on the graphics and its perceptions by the user.

4. A Prototypical Hyper-Renderer

Rather than to integrate a hyper-rendering facility into an existing renderer, we chose to work with a commercially available renderer (in the present case Pixar's RenderMan) and build a hyper-rendering facility as an extra program. While this has the disadvantage that certain code must be duplicated, it allows us in fact to use different algorithms for rendering and hyper-rendering.

Our hyper-renderer is written in about 5k lines of C code on an IRIS 4D35. Input is the scene description file in the format used by the RenderMan. To support maintainability of the scene description file and to facilitate dialogues about the

graphics produced by the renderer, we extended its format to include symbolic names of the objects as well as grouping of objects into compound objects.

The hyper-renderer contains typical rendering algorithms which have been enhanced to record information symbolically about the graphics. In particular, an extended z-buffer-algorithm was implemented for hidden-surface removal; however, as opposed to conventional z-buffers, information about hidden surfaces is stored, not thrown away. Our implementation is related to Atherton's implementation of an object-buffer ([Atherton, 1981]). However, while Atherton's three-dimensional display buffer was implemented in the form of a solid object description, we approximated quadric surfaces through polygons. Although the basic algorithmic method used is not new, the *output* produced by the hyper-renderer, namely the image description file, is a significant enhancement. Finally, we used our z-buffer to generate shadows which are caused by opaque objects, as proposed by Atherton as well (see also [Appel, 1967] and [Bouknight et al., 1970]).

It is important to note that the resolution of the hyper-rendering algorithms need not be the same as the resolution of the actual rendered picture; the resolution is determined dynamically by the application program. A coarser hyper-renderer suffices for many applications and means that the results of hyper-rendering can be made available significantly before the rendered picture is in fact available.

By default, our hyper-renderer carries out a fast z-buffer-algorithm in a first pass. By recording which objects are in the line of sight of each pixel, and with information about which surfaces cause, for example, specular reflection or refraction, those parts of the scene which require ray tracing are determined. This way, our ray tracer, which is currently being implemented, will be "selective", that is, restricted to such parts of the picture as are deemed necessary. Furthermore, the visibility information supplied by the z-buffer algorithm can be used for hidden surface removal which makes the ray tracer itself much more efficient (this concept is related to that of an "item-buffer" as described by [Weghorst et al., 1984]). The calling application program can, however, override these "fancy" features and force the use of particular hyper-rendering algorithms.

The output of our hyper-renderer is an image description. It is implemented as a file organized in an object-oriented manner. The names of objects of the scene description file are associated with various pieces of information, including low-level data as to pixels affected by the object and the colors as well as high-level information pertaining to the visibility (and invisibility) of objects, their interior, the intersection of objects, certain prepositional attributes (in front of, behind of, on,...) as well as a description of which objects lie in shadows. Figure 2 summarizes the data produced by our hyper-renderer.

Note that our prototypical hyper-renderer is by no means complete in the sense that is conceivable that new applications will require more or different information. Indeed, the modular implementation of the hyper-renderer supports extensions by a systems programmer. In particular, the complex information alluded to in Chapter 3.2 needs further study before its extraction can be implemented.

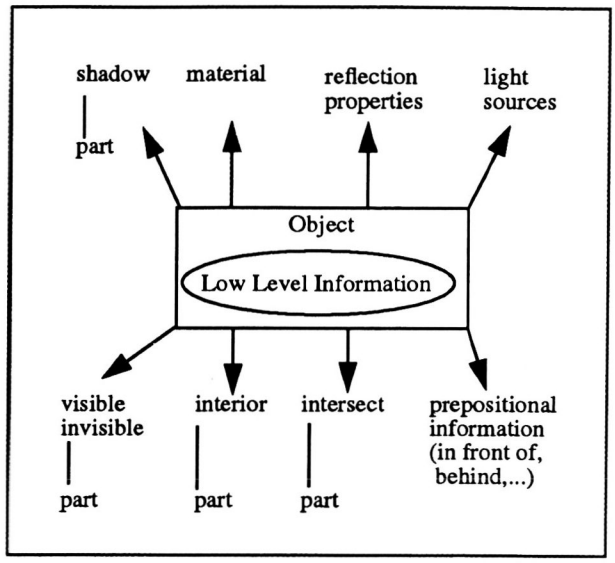

Figure 2: *Information contained in the image description file.* The hyper-renderer produces as output for each object and its parts a semantic net which contains information about visibility and invisibility, shadows, the interior of the object, other objects intersecting it, prepositional information, information about the position of light sources relative to the object, its reflection properties, and its material properties.

5. An Example

5.1 A Prototypical Application

In order to demonstrate the capabilities of hyper-rendering, we developed a simple dialogue system for navigating in a scene. The dialogue is conducted in a restricted natural language which allows a user to specify the kind of graphics to be displayed (wire-frame or full-surface-rendering). The most significant feature of the application is that it allows the user to formulate a constraint on the view, upon which the application computes a change in the scene description file and initiates re-rendering and re-hyper-rendering of the graphics.

5.2 The Dialogue and a Sample Session

We will explore the interior of an IKEA cupboard. A description of the cupboard and its contents was designed with a modeler and stored in a RenderMan "rib"-file. The end-user now uses the application to draw the picture by typing an appropriate command:

Application: Please enter command.
User: Draw cupboard.

The RenderMan full-surface renderer is invoked by the application and produces Figure 3.

The user now wishes to see more of the object in the picture and decides to switch to a wire-frame image.

Application: Please enter command.
User: Draw wires.

The RenderMan is invoked again and produces Figure 4.
The user realizes that the wire-frame image does not provide enough information to find out what is in the cupboard. Hence he asks for information on what he cannot see:

Application: Please enter command.
User: What can I not see?
Application: Ball, box, safe.

In this latter response, the application has made use of the hyper-renderer tool. The image description contains information about the visibility of objects and from this it is easy to compute which objects are *not* visible.

The user now wishes to look at a particular object, the safe. By looking at the wire-frame image, he recognizes that there are *two* candidate objects which can be a safe (one is on the bottom and another is below the top on the left side). Hence, he enters the following command:

Application: Please enter command.
User: Show safe through glass.

The application responds with Figure 5.

The application used the information in the image description file to determine that it was the cupboard itself which blocked the user's view of the safe. It then changed the material of which the upper left part of the cupboard is made to "glass" in the scene description and drew the resulting picture.

5.3 Résumé

The dialogue illustrated above is a simple example of the use of hyper-rendering. While it is in principle possible to extract such information as "What can I not see?" directly from the scene description with a significant amount of work, we believe our hyper-rendering to be the first general purpose *tool* for interactive graphics which allows an application to determine this kind of information in a simple manner. Furthermore, implementing a command such as "Show safe through glass", specifying a constraint on the visibility of a scene is easily accomplished when a hyper-renderer is available but would be tedious to program without such a tool.

6. Concluding Remarks

In this paper we introduced the concept of hyper-rendering which makes information about the rendered image available to an application. We argued that such information is useful for providing information about the graphics to end-users as well as to evaluate the graphics displayed. We demonstrated the important facilities of the hyper-renderer with a sample dialogue application.

The concepts we introduced open up a range of new problems for further study. One area is to use hyper-rendering to improve the rendering process itself. As the performance of the rendering process can be improved by refinement, that is, by subdividing it into phases where the results of each phase are carried over to the next phase ([Bergman et al., 1986]), hyper-rendering could be used to supply additional formalized information which in turn speeds up the performance of a succeeding phase.

In addition to this, hyper-rendering provides a tool for developing more effective multi-media dialogue systems. Since more information about the graphics is available as a result of hyper-rendering, speech or text interaction concerning or integrated with the graphics has a better chance of being appropriate. In particular, hyper-rendering can offer additional interaction facilities and information for users who are directly placed in a virtual environment or cyberspace. For example, users could ask questions about their location and the objects they are looking at. Therefore, hyper-rendering could prevent users from being "lost in cyberspace". Furthermore, if users have difficulties in interpreting the graphics output by the machine, it is in turn possible to *train* them to see and to guide them where to look. For example, during an interactive presentation it is often difficult to look at the "right" objects displayed, as humans are, in general, not trained in this capability. Using our hyper-renderer, it is possible to develop interactive multi-media systems which allow users to lead dialogues about the graphics with the goal of discovering what is important.

7. Acknowledgments

We would like to thank Walther Beck for various helpful discussions on aspects addressed in this paper and Petter Ranefall for implementing the prototypical hyper-renderer and application described. We would also like to thank an anonymous referee for some very detailed comments on the initial draft of the paper.

8. References

1. A. Appel, The Notion of Quantitative Invisibility and the Machine Rendering of Solids, in *Proc. ACM, Vol. 14,* pp. 387-393, 1968.

2. P.R. Atherton, A Method of Interactive Visualization of CAD Surface Models on a Color Video Display, in *Computer Graphics, Vol. 15, No. 3,* pp. 279-287, August, 1981.

3. L. Bergman, H. Fuchs, E. Grant, S. Spach, Image Rendering by Adaptive Refinement, in *Computer Graphics, Vol. 20, No. 4 (Proc. SIGGRAPH'86),* pp. 29-34, August, 1986.

4. W.J. Bouknight, K.C. Kelley, An Algorithm for Producing Half-Tone Computer Graphics Presentations with Shadows and Movable Light Sources, in *SJCC, AFIPS, Vol. 36,* pp. 1-10, 1970.

5. S. Feiner, APEX: An Experiment in the Automated Creation of Pictorial Explanations, in *IEEE Computer Graphics and Applications, Vol. 5, No. 11,* pp. 29-38, November, 1985.

6. G. Fischer, A.C. Lemke, Th. Mastaglio, Using Critics to Empower Users, in *Proceedings of the CHI '90 Conference on Human Factor in Computing Systems,* pp. 337-347, April, 1990.

7. J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice, 2nd Edition.* Addison-Wesley, Reading, MA, 1990.

8. A.S. Glassner, *An Introduction to Ray Tracing.* Academic Press, San Diego, CA, 1989.

9. R. Hall, M. Bussan, P. Georgiades, D.P. Greenberg, A Testbed for Architectural Modeling, in *Proc. EUROGRAPHICS'91,* Vienna, pp. 47-58, September, 1991.

10. J. Mackinlay, Automating the Design of Graphical Presentations of Relational Information, in *ACM Transactions on Graphics, Vol. 5, No. 2,* pp. 110-141, April, 1986.

11. A.P. Pentland, Computational Complexity versus Simulated Environments, in *Computer Graphics, Vol. 24, No. 2,* pp. 185-192, March, 1990.

12. D. D. Seligmann, S. Feiner, Automated Generation of Intent-Based 3D Illustrations, in *Computer Graphics, Vol. 25, No. 4 (Proc. SIGGRAPH'91),* pp. 123-132, July, 1991.

13. Th. Strothotte, Pictures in Advice-Giving Dialog Systems: From Knowledge Representation to the User Interface, in *Proc. Graphics Interface'89,* London, Ontario, pp. 94-99, June, 1989.

14. S. J. Teller, C. H. Sequin, Visibility Preprocessing for Interactive Walkthroughs, in *Computer Graphics, Vol. 25, No. 4 (Proc. SIGGRAPH'91),* pp. 61-69, July, 1991.

15. W. Wahlster, E. André, Som Bandhyopadhyay, W. Graf, T. Rist, WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation, in *Computational Theories of Communication and their Applications,* Oliviero Stock, John Slack, Andrew Ortony (eds.). Springer-Verlag, Berlin, 1991.

16. H. Weghorst, G. Hooper, D.P. Greenberg, Improved Computational Methods for Ray Tracing, in *ACM Transactions on Graphics, Vol. 3, No. 1,* pp. 52-69, January, 1984.

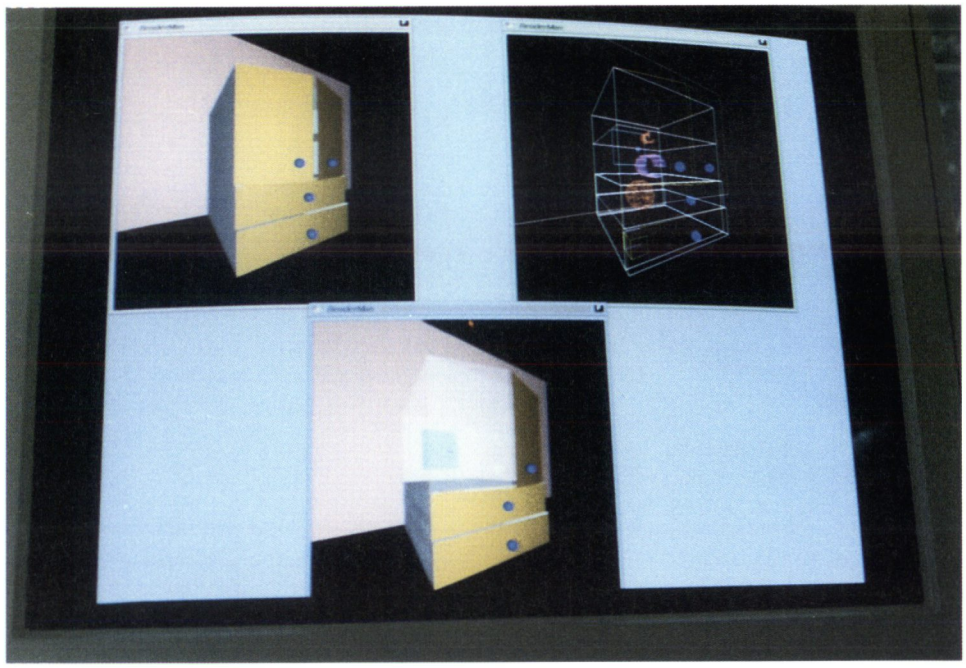

Figure 3 (top left): *The rendered image of the IKEA cupboard.*
The user is unable to see many objects, such as those lying inside or behind the cupboard.
Figure 4 (top right): *The rendered wire-frame image of the cupboard.*
All objects are visible, though largely indiscernible.
Figure 5 (bottom): *The modified image showing the inside of the cupboard.*
The user previously recognized that there were two candidates for the safe in the wire-frame image. He thus entered the command "Show safe through glass". Based on the information placed at the disposal of the application by the hyper-renderer, the application converted the material of which the upper left part of the cupboard is made to glass, thereby offering the user a view of the objects of interest to him.