Existing 3D graphics standards do not address rendering of realistic images or encompass advanced graphics functionality. The conceptual model of these standards is often rooted in 1970s display architecture. Attempts to update the conceptual model have resulted in a complex hardware-specific graphics library for each new hardware generation. Applications demand insulation from rapidly changing hardware. Standardization must occur with software that does not make the peculiarities of today's hardware visible to the application. The RenderMan Interface was designed to promote and further the realistic rendering of 3D scenes and data in a hardware-independent manner. Software developers are developing applications using RenderMan, assured that they can take advantage of advances in computer technology and graphics hardware without impacting their applications.

In 1987 Pixar examined the positive impact on the graphics community that Adobe Systems had generated by formalizing and publishing Postscript as a 2D-imaging model and page-description language. Postscript only describes the appearance of the page. It does not require that the vagaries of the printing and display hardware be exposed to its users. Postscript provides a truly device-independent mechanism that...
gave page-layout applications both freedom from hardware changes and, more importantly, transparent simultaneous access to both low-resolution and very high-resolution printing devices. By giving the application users the power to easily access typeset quality pages that were previously considered unattainable. Postscript created the desktop publishing market.

Perceiving a similar need for realistic 3D rendering, Pixar set out to draw on the extensive background and talent in image synthesis of its technical staff (who pioneered many of the advances in realistic rendering techniques) to develop a powerful, consistent, yet simple interface that could adequately describe the most common rendering techniques and provide extensibility for future growth. Working with other leading 3D-graphics companies, Pixar developed the elements of a 3D-scene-description interface for realistic rendering that came to be known as RenderMan. RenderMan brings to realistic 3D graphics the hardware independence that Postscript brought to 2D page layout. More than 5,000 copies of The RenderMan Interface Specification\textsuperscript{1,2} have been distributed, and numerous companies have embarked on the task of creating software and hardware products that use it.

Modeling versus rendering

As shown in Figure 1, the RenderMan Interface was designed to partition the generation of realistic images into two distinct areas, modeling and rendering.

Modeling is a very interactive process that requires the involvement of a trained user capable of designing 3D objects and scenes. On the other hand, realistic rendering has traditionally been a time-consuming noninteractive process which users have been content to treat as a “black box.” The difference of interactivity requirements is one of the keys that allow the partitioning of modeling and rendering.

Modeling

The interactive nature of the modeling environment, and the need to simplify the user interface as much as possible, place many constraints on a modeling package. Some modeling systems have been designed to use only powerful, 3D-graphics hardware, thereby allowing more interactive and powerful user interfaces to be implemented. But, by constraining the modeling system to platforms with specific 3D hardware, the modeler will have fewer potential users. Other modeling systems choose to limit their use of 3D-graphics hardware and instead focus on making their modeling software available to a larger number of potential users.

The RenderMan Interface allows developers of modeling systems to concentrate on designing and implementing modeling systems by clearly specifying the requirements for the rendering system, by providing a wide variety of geometric primitives, and by ensuring there is no need to create a sophisticated rendering program. Developers need only choose a renderer that adheres to the RenderMan Interface to perform realistic rendering.

Rendering

The process of rendering an image requires that a complete 3D scene be specified, including the point of view of the observer, camera, parameters, placement of the 3D objects in the scene, the objects’ color and other visual attributes, and parameters of lights used to illuminate the scene. All these elements of a 3D scene are then used to create a 2D image of the view from the observer’s camera position with hidden surfaces removed and visible surfaces shaded appropriately for the lighting conditions, surface color, and shading attributes associated with the visible surfaces.

The RenderMan Interface provides a comprehensive definition for all the data that can be output as part of a 3D scene description. It also provides a simple, yet extremely powerful way to allow user-extensible control over the shading process. Figures 2 through 5 show the same scene with increasingly re-
alistic features applied to the images. The RenderMan Interface is the first standardized graphics interface able to generate images like the one shown in Figure 6.

**RenderMan features**

The RenderMan Interface boasts the most comprehensive scene-description feature set of any 3D graphics interface: Convex and concave polygons (with and without holes) and polyhedral surfaces, as well as quadric surfaces, including spheres, cylinders, cones, hyperboloids, paraboloids, disks and tori, and the partially swept versions of each, are available. Bicubic patch and patch mesh surfaces are supported with not simply the normal Bezier, Hermite, or B-spline bases, but with arbitrary basis in each parametric direction. Nonuniform rational B-spline surfaces with trim curves are also available. Constructive solid geometry (CSG) is supported, with solid primitives made out of any combination of the standard geometric primitives.

RenderMan is the first graphics interface to support procedural primitives. A modeling application linked to a RenderMan renderer can supply an arbitrary refinement subroutine as the description of a procedural primitive, such as a fractal, fractal, or particle system. This subroutine will be called by the renderer as necessary during the rendering process to adaptively refine the primitives until they are small enough to be approximated by standard primitives.

The RenderMan Interface also supports all the basic 3D graphics functions that users of any graphics package require. A hierarchical transformation stack with a full set of transformation operations, orthographic and perspective viewing transformations, and device-independent image-size control are available. In addition, RenderMan supports a full set of antialiasing, filtering, and image quantization quality controls. This gives the user the power to generate true high-quality images without annoying artifacts.

RenderMan includes standard support for such high-end features as transformation and deformation, motion-blur, and camera depth-of-field. It also includes a very general interface for specification of the data that is present at the vertices of a geometric primitive. Primitives can specify not only the position, but also surface color, surface normals, and texture-map coordinates on a per-vertex basis. In addition, the vertex structure can be extended at run-time to include arbitrary information, such as temperature, density, or any other values that might be of interest to an application.

Readers interested in the details of RenderMan Interface are encouraged to refer to *The RenderMan Interface V3.1* and *The RenderMan Companion: A Programmer’s Guide to Realistic Computer Graphics*.

**RenderMan Shading Language**

The most powerful and most distinctive tool in RenderMan’s repertoire is the RenderMan Shading Language. Most renderers and 3D graphics packages have a subroutine which determines the color of an object. Typically, this subroutine implements a single mathematical equation that uses a simple model of the reflection of light to calculate the contributions of the light sources and texture maps to the surface color. The equation often has many parameters (5 to 20, depending on the implementation), which can be ad-

---

*Due to a lack of contrast between text and background, this page did not reproduce well*
justed to control the appearance of different kinds of materials. Very often, however, the image needs a surface that has characteristics that can’t be achieved with the fixed equation. When this occurs, there is no recourse.

RenderMan changes this situation. It provides the Shading Language, a C-like programming language that has functions and data types designed to calculate colors based on geometric information. The Shading Language programs (shaders) are user-defined material characteristics that are part of the RenderMan scene description. They are loaded into the renderer at run-time, and they replace the built-in shading equations. This provides the power to model the appearance of objects as carefully as shapes are modeled.

In fact, the RenderMan Interface allows the user to specify four types of shaders, which provide different material characteristic information about each object: a surface shader, which determines the color seen when light reflects off the surface; a displacement shader, which can move the surface by small amounts to add dents or fillets that are too small or too complex to model geometrically; a light shader, which describes how luminous objects emit light; and a volume shader, which describes how light is attenuated as it passes through the interior of a translucent object.

Because the Shading Language is a full-featured programming language, the shaders need not encode just a simple mathematical formula for surface reflectance. They can be arbitrarily complex, detailed descriptions, with conditionals, procedural textures, and special cases. Moreover, the shaders have full access to the user-extended vertex variables described above, which gives them the ability to texture and color surfaces based on the application-specific data provided. In this way, RenderMan encompasses the full power of many special-purpose application-specific pseudocoloring renderers, freeing busy application programmers from the burden of low-level graphics programming.

**RenderMan Interface**

**Bytestream**

The RenderMan Interface Bytestream protocol (RIB) is an archive format that can contain a complete RenderMan scene description. It is not a predigested or partially rendered intermediate form, but rather a complete transcription of the model as specified to the RenderMan Interface. This protocol is suitable for
network database transmission as well as file storage. It also supports both binary and ASCII encoding of requests and is independent of machine architecture, floating-point formats, and byte ordering.

Current generations of computers generate photorealistic images slowly enough that such rendering is universally considered a batch process, and as a result, almost all modeling applications prefer to use this RenderMan binding rather than link directly to a renderer. This decision immediately gives such modelers the power to support rendering as a network service and to offload the jobs to remote rendering servers while the local modeling workstation continues unencumbered. This adds yet another facet to the platform independence enjoyed by RenderMan users.

**Driving hardware directions**

One of the main strengths of current graphics standards is that they are designed with hardware in mind. This permits users to take full advantage of the machines they paid dearly for. But it is precisely this approach that undermines such standards. They are tied to specific hardware architectures and graphics algorithms, and future innovations or advances in graphics technology are not accounted for or supported.

RenderMan, on the other hand, was designed to be independent of hardware platforms. Current implementations of RenderMan run on general-purpose computers and are enjoying the advances in speed and memory that have brought sophisticated graphics to the desktop. Graphics is no longer the sole venue of special-purpose graphics processors. Yet RenderMan has not abandoned graphics accelerators. Its basic data model is compatible with both advanced graphics pipelines and parallel multiprocessors. But the capabilities provided by RenderMan are not limited to the capabilities of the latest hardware. Rather, fact adding new features for realistic image generation. This year, hardware vendors have introduced a new generation of graphics hardware which provides not only real-time texture mapping, but near real-time motion-blur. Figure 7, the “Geometry Sophistication Meter,” demonstrates the speed at which hardware platforms are moving toward photorealism. These types of improvements are not useful to an application that is stuck with the graphics hardware architecture of the 1980s (or 1970s!). They can only be tracked by a graphics package that has future hardware as one of its fundamental design goals: RenderMan is one such package.

**Acceptance**

The success of any interface or standard is ultimately measured *only* by the number of people who use it. It is in this sense that Postscript has achieved universally acclaimed success. The RenderMan Interface is beginning to garner similar acceptance. Many significant CAD and product-design software suppliers are using or have announced their intention to use the RenderMan interface as their interface to realistic rendering. Renderers that adhere to the RenderMan Interface V3.1 are now available on a variety of hardware systems/platforms. Pixar distributes its RenderMan-compatible renderer for the Sun 3/Sun 4/SPARCstation. Silicon Graphics Iris 4D computers, 386/MS-DOS personal computers, and Macintosh II computers. Other RenderMan renderers are available3, 5 that provide alternate performance characteristics and run on other platforms.
Why are so many software developers interested in using the RenderMan interface? Three factors seem clear:

1. The RenderMan Interface is a stable 3D-scene description that provides a clear specification for what kind of 3D data must be handled now and in the future as their modeling interface matures and can make use of some of RenderMan's more advanced features.

2. The effort required to make a photorealistic renderer, especially one for PC/MS-DOS or Macintosh platforms, is considerable. The ability for a modeling system to take advantage of this effort with only the small investment necessary to output RenderMan Interface Bytestream provides considerable leverage.

3. The availability of RenderMan renderers on a range of hardware platforms provides flexibility and ensures a guaranteed growth path from modest personal computers to superpowerful graphics workstations in the future. RenderMan has been shown to work on all of these platforms and as performance on all these machines increases, RenderMan renderers only get better. This works to the advantage of developers who are creating modeling systems and rendering systems.

**Conclusion**

The RenderMan Interface was designed to promote and further the realistic rendering of 3D scenes and data. It is the first multivendor 3D graphics interface that is designed with the requirements and potential of future graphics hardware in mind. Standard support for advanced-image-synthesis features provides both a roadmap and a benchmark for graphics hardware development. This ensures that applications written using RenderMan will not merely survive, but will thrive as hardware functionality and performance get better, faster, and less expensive.

**References**


**Anthony A. Apodaca** is a senior member of the technical staff at Pixar, where he serves as the chief architect of the RenderMan Interface and primary technical contact for the RenderMan Advisory Council. He is also one of the unknown implementors of both of Pixar's image-synthesis products. His interests include realistic image synthesis, animation, and parallel processing. Apodaca received his MEng in computer and systems engineering at Rensselaer Polytechnic Institute in 1986. He is a member of ACM and SIGGRAPH.

**M.W. Mantle** is director of 3D graphics at Pixar, where he is responsible for all technical development of Pixar's rendering products and is chair of the RenderMan Advisory Council. His interests include 3D computer graphics methodology and implementations, especially across a variety of computer platforms and hardware accelerators. His background includes directing and implementing numerous real-time graphics systems and rendering algorithms for Evans & Sutherland and Pixar.

Mantle received his BS in computer science at the University of Utah in 1971 and has been working actively with computer graphics since 1970. He is a member of IEEE, ACM, and SIGGRAPH.

The authors can be contacted at Pixar, 3240 Kerner Blvd., San Rafael, CA 94901.

July 1990