EDITORS: Mary Whitton, mcwhitton@gmail.com
Chris Johnson, crj@sci.utah.edu
Dave Kasik, david.kasik@gmail.com

The Design of RenderMan

Pat Hanrahan, *Stanford University, Stanford, CA, 94305, USA*

Edwin Catmull, *Former President Pixar and Disney Animation (Retired), Emeryville, CA, 94608, USA*

FROM THE EDITORS

Bill Lorensen, our late coeditor of the CG&A Department Visual Computing: Origins, first suggested in 2018 that Pat write a piece about the development of RenderMan. RenderMan was prominent among the accomplishments called out when Pat and Ed were jointly honored with the 2019 ACM A.M. Turing Award. In recognition and celebration of that award we asked them to coauthor this story of RenderMan.

The design of RenderMan was driven by the requirements of rendering for the movies. The rendered images could have no digital artifacts and they had to be able to be composited seamlessly with live-action footage. This article recounts the development of RenderMan. It tells the story of the invention of the fundamental architecture and algorithms, the application of good engineering and design principles, RenderMan's use in feature films, the long-term influence of the RenderMan Shading Language on real-time graphics and high-performance computing, and, where the name came from.

Pixar's RenderMan, although designed in 1988, continues to be used today. It is still, in fact, the rendering system most widely used in movie production. Why has the system survived for so long since it was designed? This article is about how RenderMan came to be; it tells not only the story of the people, but also the story of the algorithmic innovations, the disciplined application of good engineering and design principles, RenderMan's successful use in feature films, and its longer term influence on high performance computing.

In 1981, while working as part of the Computer Division of Lucasfilm, Ed Catmull, Loren Carpenter, and Rob Cook were eager to show that computer graphics rendering systems could produce images that look good on the big screen. Much of the work on image synthesis in the late 1980s was targeted at real-time training and simulation, and interactive 3-D graphics workstations, such as those produced by Silicon Graphics Inc., were starting to be available commercially. The focus at that time was on how to create an image in real time given little computing power, even with specialized graphics chips such as Silicon Graphics' Geometry Engine. This meant images had simple shading and low polygon complexity (typically fewer than 40,000 polygons per image).

0272-1716 © 2021 IEEE
Digital Object Identifier 10.1109/MCG.2021.3085374
Date of current version 12 July 2021.

RENDERING FOR THE MOVIES

Ed, Loren, and Rob started by asking the question: what would it take to make an image that would be realistic enough to merge seamlessly into a live action movie? They articulated three goals for the rendering system.

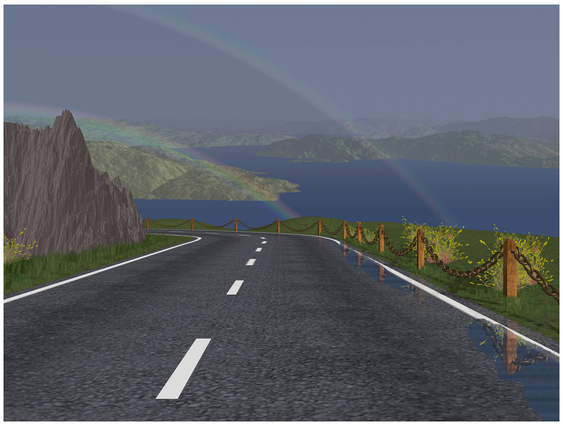

FIGURE 1. *The Road to Point Reyes*. This image is composed of fractal mountains (Loren Carpenter), plants (Alvy Ray Smith), grass (William Reeves), puddles (David Salesin), road, fence, and rainbow (Rob Cook). The picture was rendered in 1983 on a Digital Equipment Corporation VAX and displayed on an Ikonas Graphics Systems full-color (24 bpp) framebuffer. (Source: Pixar Animation Studios; used with permission.)

First, the images had to be high quality and indistinguishable from images captured by cinematic cameras, and, particularly, they could have no digital artifacts such as polygonization, jaggies, or other forms of aliasing. The bar had already been set high: while making *Star Wars*, George Lucas assembled the best special effects team ever. One of their breakthroughs was to photograph physical models while the models or camera were moving under computer control, thus, allowing them to capture the blur of moving objects. This was essential for creating imagery that could be composited with live-action photography. Computer generated imagery would not be acceptable unless the blur could be simulated in the renderer.

Second, the images must reflect the visual complexity of the world around us: The environment would need to contain complex curved and rough surfaces, not just large flat polygons. The objects should also appear to be made of real materials and the lighting should be natural. These two goals were codified in the slogan Render Everything You Ever See, and REYES, the initial name for the new rendering system, was born. These rendering goals were symbolized by the picture, *The Road to Point Reyes*, a national seashore in Marin County where Lucasfilm was located (see Figure 1).

The third goal was that the new rendering system should be fast and efficient enough so that making a movie would be economical. Though Lucasfilm was not yet developing hardware, Ed, Loren, and Rob approached inventing algorithms with the thought that they were likely to eventually run on parallel vector supercomputers: adequate performance would only be achieved by carefully thinking out the interaction between software and hardware. At that time, it was known that curved surfaces or fractal surfaces could be rendered by converting them into fine meshes of very small polygons called micropolygons. However, the state of the art in rendering polygons was around 40K total. The Lucasfilm developers understood that the rendering system would need to handle several orders of magnitude more micropolygons. In order to estimate the computing resources required for movie quality rendering, the team produced the analysis in Table 1. They came to the 80,000,000 number by making assumptions about the complexity of scenes, the average number of objects in front of the background, the fineness of the mesh of micropolgyons, the sampling requirements, and the calculations for lighting and other effects.

TABLE 1. REYES Machine goals.

Micropolygons	80,000,000
Micropolygons per pixel	4
Pixels	5,000,000 (3000 × 1667)
Depth complexity	4
Samples per pixel	16
Geometric primitives	150,000
FLOPS per micropolygon	300

The research group was well acquainted with ideas from signal processing and sampling theory. Applying the Sampling Theorem, which says that a signal should not have frequencies greater than one half the sampling rate, they knew that they would need to sample surfaces at least twice per pixel in each direction. This led to the conclusion that four micropolygons were required per pixel, and, at that time, the team assumed they would be creating 5 megapixel images, approximately equal to the resolution of a frame of film. The final assumption was that the depth complexity was four; that is, the center of each pixel would contain on average four micropolygons at different depths, only one of which was visible. This was a very low depth complexity, but it was considered reasonable because it would be too expensive for an artist to create scenes with geometry that could not be seen. The result was the goal of handling 80,000,000 micropolygons per image, an insanely ambitious number at the time. The team welcomed the insane number because it would force them to rethink everything about rendering since none of the known techniques were going to get to those performance numbers through incremental advances. They then allocated a bare-bones computer budget of 300 floating point operations per micropolygon, very little by today's standards. Given these requirements, and the state of the art in digital flight simulators, they estimated that they could compute a frame in 2 min. However, there was a big if. They would need to develop a new rendering architecture—both software and hardware.

ARCHITECTURAL INSPIRATION AND INNOVATION

The realization that complex surfaces could be diced into micropolygons led to a Eureka moment for Loren. The pipeline came together in his mind while standing waist deep in the surf in the afternoon sun at Point Reyes National Seashore. Loren had previously developed methods to render curved surfaces[1] and fractal terrains[2] using a two-stage process of recursively splitting and then finally dicing. Applied in the new REYES system, a front-end process split the geometric primitives into new, smaller geometric primitives. The primitives could then be split further. For example, a sphere could be split into bicubic patches, and the bicubic patches could be split into smaller patches. The primitives were sorted into rectangular "buckets," which were processed top to bottom, left to right. When the primitives became small enough, they were tessellated into grids of micropolygons. Next, the grids were shaded, and the resulting shaded micropolygons were then passed to a hider, which was responsible for finding the visible surfaces and performing antialiasing calculations. Loren had developed an antialiased hidden-surface removal algorithm called the A-buffer.[3] A major advantage of this architecture was that shading a grid of micropolygons could be vectorized on a virtual array processor. This approach efficiently used memory and exploited image coherence. The pipeline of bucket, split, dice, shade, hide (sample and visibility), filter, and display was extremely friendly to hardware implementation, and became known as the Reyes Architecture.[4]

There were two other major innovations in REYES: stochastic sampling and the shading language.

STOCHASTIC SAMPLING

The REYES team had to match or exceed the image quality produced by Lucasfilm's visual effects house, Industrial Light and Magic; thus, the question of the day in the rendering group at Lucasfilm was how to do both spatial and temporal antialiasing on sequences of images. Rodney Stock suggested to Rob Cook that he explore the use of stochastic dithering patterns like those used in printing. Rob implemented this method and found it was promising. The objectionable regular patterns such as jaggies due to aliasing were replaced with unsightly noise. Eventually, Rob came up with the idea of converting the unsightly noise to blue noise (shifting the frequencies in the noise from low to high) using a jittered sampling pattern.[11] Alvy Ray Smith showed Rob a *Science* article by Jack Yellott that showed that the nonfoveal photoreceptors in the monkey eye were formed using a Poisson Disk Sampling pattern: each photoreceptor was a minimal distance from another.[5] Poisson Disk sampling has a blue noise spectrum and the fact that our eyes sample with a blue noise distribution of sensors convinced the team that stochastic sampling was the best approach.

As noted above, optical motion blur was one of the major innovations made at Lucasfilm. Physical models of spacecraft and scenery would be photographed with a camera; either the model or the camera could move via robotic arms. Rather than stop the camera and take each picture, the camera or model would continuously move while the shutter was open. The motion blurred frames were carefully processed and overlaid on top of each other using optical compositing to create the final shot. Motion blur is an example of temporal antialiasing, i.e., antialiasing in the time domain, and so it was natural to ask the question whether stochastic sampling could be extended to motion blur. Tom Porter suggested that Rob's stochastic sampling strategy in x and y be extended to time, t, with each sample point representing a different moment in the interval of time that the shutter was open. In order to demonstrate his idea, Tom modified an in-house ray tracer written by Tom Duff and, using this technique produced the famous image *1984* shown in Figure 2. Temporal sampling solved the motion blur problem.

Then Rob realized that the technique could be extended to integrate over the lens to simulate depth of field and so it was now possible to perform spatial antialiasing, motion blur, and depth of field using stochastic sampling. Now the images were starting to look very good! In their paper on distributed ray tracing, which was originally called probabilistic ray tracing, Rob, Loren, and Tom showed that random sampling could also be used to simulate penumbras from area light sources, blurry reflections, and many other effects.[6] Later James Kajiya introduced the general light transport equation, or the *rendering equation* into computer graphics.[7] This laid down the foundation for Monte Carlo Ray Tracing as a solution to the problem of global illumination, and is the basis of modern path tracing, bidirectional ray tracing, and

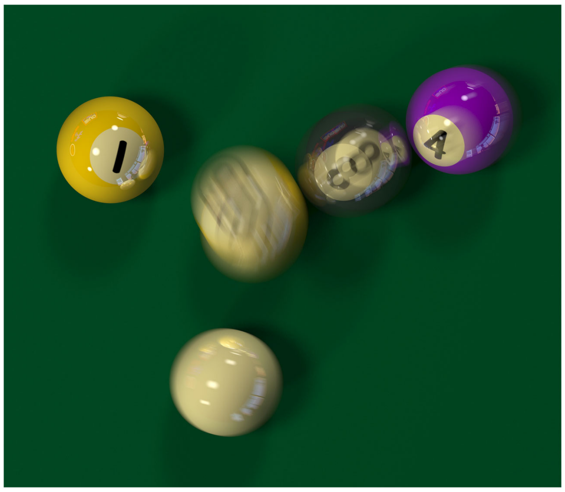

FIGURE 2. *1984*. The first demonstration of motion blur using stochastic sampling. The pool balls were modeled by Tom Porter and numbered to mark the year of creation. The felt texture map was created by Loren Carpenter. John Lasseter suggested the reflection map which was acquired in a pool hall in Novato, CA, and painted some additional graphics to create the ambiance of the pool hall. This image was created using a ray tracing algorithm by Tom Porter based on code by Tom Duff. The image took 3 weeks to render on two VAX 11/750s, more than 290 hours of CPU time. It first appeared on the cover of *Science* magazine in 1984 with the title "Is this Picture Fake?" (© 1984 Thomas Porter, Pixar Animation Studios.)

metropolis ray tracing. The use of stochastic sampling for antialiasing and Monte Carlo algorithms for global illumination was one of the major advances in the development of computer graphics.[8]

In 1981, Loren implemented a version of REYES based on the A-buffer; this version was used to render the Genesis Sequence in *Star Trek II*. In 1983, Rob implemented a software rendering system that incorporated all the key algorithmic innovations. Having worked at DEC before joining Pixar, Rob was a very experienced software engineer and his well-written code formed the basis of the software product. Cook's 1983 version of REYES was used to produce the short *The Adventures of André & Wally B.* (1984) (see Figure 3), and also to render the famous stained glass knight in *Young Sherlock Holmes* (1985).

CONSIDERING HARDWARE IMPLEMENTATION

The early focus on hardware had a big impact on the design and scalability of the REYES architecture. First, all algorithms had to run in linear time or sublinear time. An $O(n^2)$ algorithm could not handle 80 million micropolygons. Second, the system had to use memory efficiently: each stage in the rendering process needed small working sets and needed to exploit and use caches to exploit locality. The bucketing approach conserved memory by creating and then immediately sampling only micropolygons in a small region of space. Another example of exploiting coherence was the texture subsystem. Textures were sampled at grid locations. This made texture sampling more efficient and allowed the use of coherent access textures (CAT). Textures were large and could not be preloaded and had to be cached. The size of the memory footprint used by textures could be reduced dramatically by loading texture data at the required level of detail only when needed; this technique became known as the principle of texture thrift.[9] This hardware project again taught the lesson that it is much more challenging to design an implementation of an algorithm that can be implemented efficiently on hardware than to design a software proof of concept.

Once the REYES algorithm had been demonstrated successfully in software, a project began to build hardware to execute it. The REYES machine was originally conceived of as having multiple boards: The floating point array processor (FLAP) for geometry processing and shading, a hider for stochastic sampling, an intelligent sample memory (ISM) to store and process samples, a filter board, and display subsystem. Building a complete machine was too ambitious for a small company like Pixar, so the hardware was simplified to consist of the FLAP and the ISM. Then, the FLAP was dropped and the entire algorithm ported to the ISM. The ISM, based on transputer chips, eventually turned into the RM-1, a Pixar hardware product.

Pat Hanrahan joined the project in 1986. He inherited the REYES software and became responsible for the architecture of Pixar's rendering products. As the Reyes Machine began to take shape, Pat and Bill Reeves began to work on a new Rendering Interface. This interface would be used in Pixar's hardware and software products as well as the in-house developed modeling and animation tools. A critical aspect of the environment at Pixar was that the modeling and animation group (Bill Reeves, Eben Ostby, and John Lasseter) were eager, sophisticated, and demanding users—an invaluable combination for identifying requirements and stress-testing code. The rendering interface evolved into the RenderMan Interface Specification. As the interface developed, the goal expanded to support any photorealistic rendering system, not just REYES.

The name RenderMan came up during a conversation between Jaron Lanier and Pat Hanrahan. Jaron Lainer had founded VPL Research, a company that had developed a VR headset and a position sensing glove. Jaron demonstrated the system to Pat. This was in the era of the SONY Walkman and Diskman, so Pat and Jaron mused, wouldn't it be nice to have a portable RenderMan to make sound and imagery in real time for a head-mounted display system. Once Pixar had developed the first rendering hardware prototype, the Pixar team quickly started calling it a RenderMan. The name of the software system was also changed from REYES to RenderMan.

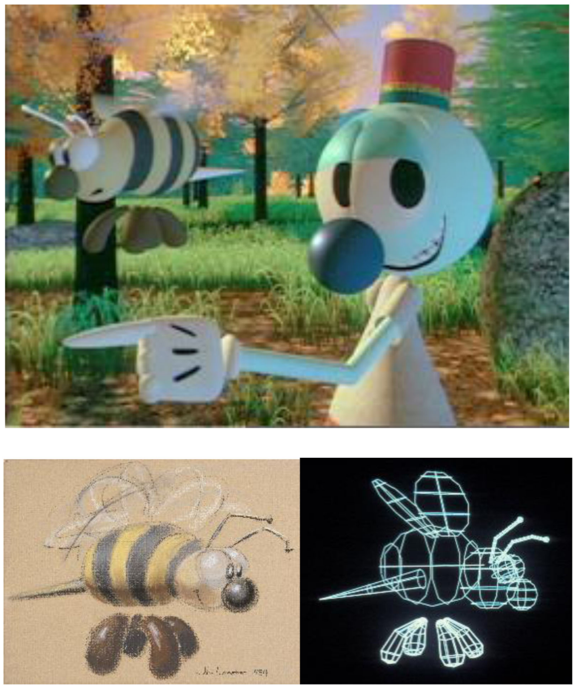
FIGURE 3. *The Adventures of André & Wally B.* was conceived by Alvy Ray Smith as a short film to challenge and demonstrate the breakthroughs of REYES, including new approaches for particle systems, modeling, and motion blur. John Lasseter designed and animated the characters, adding his remarkable story-telling skills. The top image is a frame from the short film. The drawing on the lower left is John's illustration of what he thought Wally B. should look like. (Source: Pixar Animation Studies; used with permission.)

GEOMETRIC PRIMITIVES

One of the big issues in the design of the new interface was deciding what set of geometric primitives to support. The split-and-dice interface that Loren had developed was written in an object-oriented style (still new at the time), and it was easy to add a new primitive if the required methods were implemented. However, as a result, there were dozens of clever implementations of different geometric primitives, even though they were rarely used. Most famous at Pixar was the deformable teardrop primitive originally developed by Ed and extended by Eben Ostby for the short *The Adventures of André & Wally B.* (See Andre's body and Wally B. legs in Figure 3.)

REYES was designed to render parametric surfaces, so polygons were not widely used as a modeling primitive at Pixar (quadrilaterals were considered bilinear patches). Consequently, the RenderMan interface had to be updated to support polygons and triangle meshes, primitives that were standard in the animation industry. Similarly, although Cubic B-spline patches had been implemented, the trimmed NURBS required by advanced CAD tools also had to be added. One of the most novel parts of the new interface was a user-extensible procedural primitive, where callbacks could be used to lazily convert higher level primitives to lower level primitives and eventually to micropolygons.

At the time the RenderMan Interface was designed, the theory of subdivision surfaces was still being developed. Ed and James Clark invented a generalization of B-spline surface patches and developed a recursive algorithm.[10] When Tony DeRose, an expert on geometric modeling and subdivision surfaces joined Pixar, subdivision surfaces were added to RenderMan and used in the short film *Geri's Game*.[11] Subdivision surfaces were so versatile that they were immediately drafted for use in Pixar's second film, *A Bug's Life*. They have since become the main primitive for curved surfaces in the film industry.

THE RENDERMAN SHADING LANGUAGE

Arguably the most novel aspect of RenderMan was the shading language. The idea of programmable shading had already been added to REYES by Rob Cook. His shade tree system made it possible to write shaders for light sources and surfaces.[12] Pixar used UNIX internally and the idea of domain-specific little languages was a big part of the UNIX culture.[13] Most of the feature requests from the animation group had to do with shading and texturing and could have been implemented by many one-off additions to the interface. This approach would have been both a significant implementation effort and would have made the system much more complicated and difficult to maintain.

Instead, Hanrahan proposed to enhance Rob's shade tree system. Gradually, what began as the shade tree system developed into a complete language, with built-in types designed for shading calculations, a more complete set of functions and libraries, new control structures (`illuminate` and `illuminance`), and different kinds of shaders (deformations, light sources, surfaces, volumes, etc.).[14] The original shade tree system had many built-in functions for lighting calculations such as `diffuse` and `specular`. New constructs were added to cast light from within a light shader (`illuminate`) and perform reflection calculations from within a surface (`illuminance`). As a placeholder for global illumination calculations, the system included a `trace` function which is, to this day, the key method for implementing path tracing. The `trace` function is an example of how the system was designed for photorealistic rendering systems in general.

One big feature/limitation of the shading language was that the shaders were not guaranteed to obey the laws of physics (conserve energy, obey the reciprocity principle, etc.). The technical directors loved the fact that shaders were not physically correct since this gave them complete freedom when coding a desired look. However, the lack of physical correctness became a big drawback when using RenderMan to build physically correct ray tracers. Experience making movies led to the need for cinematic lighting, which could be hacked with thousands of fake light sources, but was eventually replaced with physically correct illumination.

STANDARDIZATION

As the RenderMan Interface began to take shape, we started exploring how to productize the system. Ed and James Clark, founder of Silicon Graphics, met to discuss proposing a new standard for 3-D graphics. Silicon Graphics was redesigning and modernizing their graphics API in what was to become OpenGL.[15] Ed and Jim wrote a whitepaper which was released in 1987 announcing this new joint effort between Pixar and SGI.[16] We were all inspired by the role Postscript played in the desktop publishing industry: Postscript, the standard interface between desktop publishing programs and the laser printer, revolutionized printing and publishing. We hoped that a standard interface for photorealistic graphics might encourage the adoption of these rendering technologies and have a similar effect on the CAD and movie industries.

Pixar and SGI put a lot of effort into working together to specify such a 3-D graphics interface although eventually the two companies went their separate ways due to fundamental differences in their primary use cases. In the same way that Postscript was designed to specify what was on a printed page and not how to draw the page, the RenderMan Interface was designed to describe a scene, not specify how the scene is rendered. Rendering high quality pictures often took hours and so the system was very batch oriented.

Interactive graphics systems such as those from SGI had very different requirements. They needed to run in real time and to support the graphical user interface and 2-D applications. A lower level procedural interface that implemented drawing commands and exposed other mechanisms (like z-buffers and accumulation buffers) was more appropriate in that setting. The SGI/Pixar collaboration was spearheaded by Kurt Akeley and Pat Hanrahan. Although eventually deciding not to continue working on the joint interface, Kurt and Pat became close colleagues and eventually Kurt completed his Ph.D. under Pat's supervision.[17]

DESIGN PROCESS AND PRINCIPLES

One reason that the RenderMan Interface has stood the test of time is that the interface was very carefully designed. Before joining Pixar Hanrahan had a brief stint at Digital Equipment Corporation's Systems Research Center and worked on the specification of their workstation graphics system. SRC employed a rigorous approach to building systems. This is reflected in Butler Lampson's famous paper "Hints on System Design."[18] Fred Brooks, in his book *The Mythical Man-Month*[19] and article "No Silver Bullet,"[20] also advocated for high quality systems design and the importance of architecture. The RenderMan Interface Specification went out to 17 computer graphics companies including Sun, Stardent, Wavefront, Alias, and others for review and comment. Although many groups participated, the process was intentionally not design-by-committee, but was driven by a single Chief Architect, in this case, Hanrahan. New feature requests were critically examined and functions were added only if they did not compromise the conceptual integrity of the system. A guiding principle was that broadly applicable mechanisms that added composable features were strongly preferred to adding isolated features that did not fit naturally into the system. Features that removed complexity were strongly preferred over features that added complexity (less is more). It was an iterative process: consensus was reached on the goals (what the system should do), different approaches for adding those capabilities would be proposed, issues in the different approaches were identified both pro and con, and then a decision was made and a particular design was adopted. This resulted in a lot of functionality from a relatively simple interface (about 100 API calls).

RENDERMAN IN THE MOVIES

The RenderMan Interface was released in 1988. Subsequently, Pixar adapted REYES to conform to the standard and released that implementation as the RenderMan product. An important early adopter was Lucasfilm Ltd., which produced a series of movies that used RenderMan. The famous pseudopod from *The Abyss* (1989) was rendered using RenderMan and demonstrated the achievement of the goal of seamlessly integrating a computer-generated character into a live-action movie. The success of the technologies, including RenderMan, that were used in *The Abyss* gave Jim Cameron the confidence to allow ILM to use CGI to create the T-1000 in *Terminator 2: Judgment Day* (1991). In the same year Disney released *Beauty and the Beast* (1991). The Ballroom waltz scene was a combination of 2-D animation created using Disney's CAPS system (Pixar had written the graphics for CAPS) for Belle and the Beast, and a 3-D environment rendered using RenderMan. This was also the same year that Disney and Pixar signed an agreement to produce *Toy Story*. Meanwhile, ILM used RenderMan for the dinosaurs in Steven Spielberg's 1993 movie *Jurassic Park*. That film was a milestone: It convinced the movie industry that it was technically possible to create realistic characters using CGI. The success of the film also made the business case for CGI compelling. Then, in 1995, Pixar produced *Toy Story* which was the first full length computer generated animated film. A 20-year journey finally had its "overnight success"—photorealistic rendering software had changed forever the way movies are made.

RENDERMAN AND GPUs

One unexpected side effect of the RenderMan Shading Language was its effect on high performance computing. Rendering at the time was constrained by the available computing resources, in particular memory bandwidth and number of FLOPs. As real-time graphics became affordable on personal and mobile computing devices, a large demand for high quality games drove AMD, Intel, and NVIDIA to develop very powerful programmable graphics processing units (GPUs). Hanrahan, now at Stanford University, led a project that developed an early real-time shading language (RTSL).[21] RTSL strongly influenced the design of NVIDIA's Cg, Microsoft's high-level shading language (HLSL), and the OpenGL Shading Language (GLSL). Today the main processor in your laptop or phone contains both a CPU and a GPU, with the majority of the die area devoted to the GPU. Hanrahan and his students (and others) started to investigate using GPUs for simulation and other compute tasks.[22] Ian Buck, a graduate student in Hanrahan's group, developed Brook which evolved into CUDA. As more and more features were added to GPUs they started to become the most powerful computers in the world. They did this by building highly parallel machines with tens of thousands of floating point cores. Now, the fastest supercomputers in the world contain GPU accelerators. The latest NVIDIA GA100 GPU contains 54 billion transistors, 8192 floating point, and 512 tensor cores with a peak performance of 600 Teraflops.[23] The GPU architecture is well suited to execute modern machine learning algorithms, and convolutional neural networks and tensor computations. GPUs have become central to the artificial intelligence revolution going on in computing today.

Returning to our opening question, why is RenderMan still used in movie production after so many years? Articulating the lofty goals of high quality imagery and rich visual complexity provided a vision for CGI that was both challenging and still guides researchers today. The key algorithms and technologies were also good choices. Using stochastic sampling and Monte Carlo algorithms were the right fundamental choices for antialiasing and integration, and were the key to incorporating advanced lighting simulation. The fact that RenderMan was developed in a time of scarce computational resources led to a design that was scalable. Figure 4 shows two representative frames, one from *Toy Story* (1995) and another from *Toy Story 4* (2019). The increase in visual complexity is remarkable. The statistics from *Monsters University* (2013) illustrate this increase in complexity. The scene has now grown to billions of micropolygons per frame and a frame takes 29 hours to compute (approximately 1 petaflop per frame or 1 gigaflop per pixel). The fact that the system has been able to scale to modern scene sizes is a testament to the wisdom of the early design decisions and the commitment to maintain the integrity of the system design.

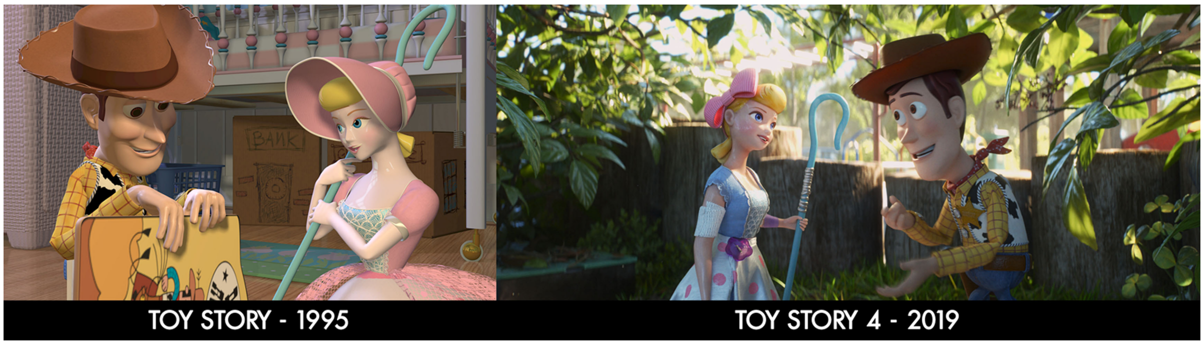

FIGURE 4. *Toy Story* (1995) and *Toy Story 4* (2019) were released 24 years apart. The characters needed to look approximately the same, but everything was rebuilt because of advances in every part of the pipeline. While story remains the most important foundation for any film, the richness of the imagery supports the story. The image complexity and lighting show the advances over the years. The cleanliness of the RenderMan Interface enabled remarkable changes under the hood. (Source: Pixar Animation Studios; used with permission © Disney/Pixar.)

ACKNOWLEDGMENTS

We would like to thank the many people involved in REYES and RenderMan over the years. First, to Loren Carpenter and Rob Cook for the fundamental contributions to the REYES architecture and algorithms, and for all the delightful time working together. Alvy Ray Smith inspired many of the directions taken at Pixar, and helped put together the image *The Road to Point Reyes* and the short film *The Adventures of André & Wally B.* The original REYES Machine team included Jeff Mock (intelligent sample memory (ISM)), Mark Leather (hider), and Adam Levinthal (CHAP/FLAP). Jim Lawson contributed to the design and implementation of the RenderMan Shading Language. Tony Apodaca was a great sounding board during the design of the RenderMan Interface, and spearheaded the productization and continued development of RenderMan for many years. Sam Leffler built the original texturing subsystem, which was improved further by Darwyn Peachey. Tom Porter led RenderMan product marketing and produced the famous image *Textbook Strike*. Mickey Mantle helped immensely by coordinating the productization of RenderMan. Charlie Gunn implemented a version of REYES on the Pixar Image Computer. Dana Batali led for many years the team that maintained and enhanced RenderMan. Finally, we are both extremely grateful to George Lucas and Steven Jobs for supporting this work in the early days and having the patience to let us bring it to completion.

And finally—the RenderMan product group. RenderMan is alive and vital because many sophisticated technical people kept adding capabilities as the state of the art advanced. After its commercial release, the community grew as RenderMan was adopted at more studios, on projects with a varied palette of looks and pipeline requirements. The RenderMan project became, and continues to be about creating a rendering technology platform that evolves to meet new production concepts, while respecting the requirements of professional production pipeline ecosystems and staying current enough to allow fresh technical concepts to be injected into the art process from the research community.

REFERENCES

1. J. M. Lane, L. C. T. Carpenter, T. Whitted, and J. F. Blinn, "Scan line methods for displaying parametrically defined surfaces," *Commun. ACM*, vol. 23, no. 1, pp. 23–34, 1980.
2. A. Fournier, D. Fussell, and L. Carpenter, "Computer rendering of stochastic models," *Commun. ACM*, vol. 25, no. 6, pp. 371–384, 1982.
3. L. Carpenter, "The a-buffer, an antialiased hidden surface method," in *Proc. 11th Annu. Conf. Comput. Graph. Interactive Techn.*, 1984, pp. 103–108.
4. R. L. Cook, L. Carpenter, and E. Catmull, "The Reyes image rendering architecture," in *Proc. 14th Annu. Conf. Comput. Graph. Interactive Techn.*, 1987, pp. 95–102.
5. J. I. Yellott, "Spectral consequences of photoreceptor sampling in the rhesus retina," *Science*, vol. 221, no. 4608, pp. 382–385, 1983.
6. R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," in *Proc. 11th Annu. Conf. Comput. Graph. Interactive Techn.*, 1984, pp. 137–145.
7. J. T. Kajiya, "The rendering equation," in *Proc. 13th Annu. Conf. Comput. Graph. Interactive Techn.*, 1986, pp. 143–150.
8. R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. Graph.*, vol. 5, no. 1, pp. 51–72, 1986.

9. D. Peachey, "Texture on demand," Techn. Rep., PIXAR, Richmond, CA, USA, 1990.
10. E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," *Comput.-Aided Des.*, vol. 10, no. 6, pp. 350–355, 1978.
11. T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Techn.*, 1998, pp. 85–94.
12. R. L. Cook, "Shade trees," in *Proc. 11th Annu. Conf. Comput. Graph. Interactive Techn.*, 1984, pp. 223–231.
13. J. Bentley, "Programming pearls: Little languages," *Commun. ACM*, vol. 29, no. 8, pp. 711–721, Aug. 1986.
14. P. Hanrahan and J. Lawson, "A language for shading and lighting calculations," in *Proc. 17th Annu. Conf. Comput. Graph. Interactive Techn.*, 1990, pp. 289–298.
15. M. Segal and K. Akeley, "The design of the OpenGL graphics interface," Techn. Rep., *Silicon Graph. Comput. Syst.*, 1994.
16. E. Catmull and J. Clark, "Proposal for a procedural 3D graphics standard," Tech. Rep., PIXAR and SGI, 1987.
17. K. Akeley, "Achieving near-correct focus cues using multiple image planes," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2003.
18. W. B. Lampson, "Hints for computer system design," in *Proc. 9th ACM Symp. Oper. Syst. Principles*, 1983, pp. 33–48.
19. F. P. Brooks, Jr., *The Mythical Man-Month: Essays on Software Engineering*. London, U.K.: Pearson Education, 2nd ed., 1995.
20. F. P. Brooks, Jr., "No silver bullet essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987.
21. K. Proudfoot, W. R. Mark, and P. Hanrahan, "A real-time procedural shading system for programmable graphics hardware," in *Proc. 28th Annu. Conf. Comput. Graph. Interactive Techn.*, 2001, pp. 159–170.
22. I. Buck *et al.*, "Brook for GPUS: Stream computing on graphics hardware," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 777–786, 2004.
23. "NVIDIA ampere GA102 GPU architecture," Techn. Rep., NVIDIA, Santa Clara, CA, USA, 2021.

PAT HANRAHAN is the Canon Professor of Computer Science and Electrical Engineering, Computer Graphics Laboratory, Stanford University. His research focuses on rendering algorithms, graphics processing units, as well as scientific illustration and visualization. He received the Ph.D. degree in biophysics from the University of Wisconsin-Madison in 1985. In 1989, he joined the faculty of Princeton University. In 1995, he moved to Stanford University. As a founding employee at Pixar Animation Studios in the 1980s, he was part of the design of the RenderMan Interface Specification and the RenderMan Shading Language. More recently, he has served as a co-founder and CTO of Tableau Software. In 2019, he received the prestigious ACM Turing-Award. Contact him at hanrahan@cs.stanford.edu.

ED CATMULL is a co-founder of Pixar Animation Studios, and served as President of Pixar and Disney Animation Studios. Prior to this, he was Vice President of the Computer Division of Lucasfilm Ltd., where he managed development in the areas of computer graphics, video editing, video games, and digital audio. He has written a book, *CREATIVITY, INC.: Overcoming the Unseen Forces That Stand in the Way of True Inspiration*, which describes the creative management principles developed at Pixar and Disney. He is an ACM Turing Award Laureate; he has also been honored with two Oscars for his achievements. He is active in several professional organizations, including ACM SIGGRAPH, The Academy of Motion Picture Arts and Sciences, the National Academy of Engineering, and the Visual Effects Society. Since his retirement, he continues to consult and lecture.