Rendering, Complexity, and Perception

Kenneth Chiu and Peter Shirley

Indiana University, Bloomington IN 47405, USA

1 Introduction

Over the last ten years, computer graphics researchers have labored intensely to strengthen the physical foundations of computers graphics. We now need to step back and examine the nature of scenes our end-users wish to render, and what qualities these rendered images must possess. Humans view these images, not machines, and this crucial distinction must guide the research process, lest we become an increasingly irrelevant enclave, divorced from the users we profess to serve.

In *Is Image Synthesis a Solved Problem?*[Coh92], Cohen examined the degree to which current realistic rendering methods solve problems encountered in real applications. He concluded that these methods do not adequately deal with complexities in scene geometries, reflection properties, and illumination; and with issues involving human perception. In his informal survey of rendering researchers, the two problems of image synthesis considered the "least solved" were environmental complexity and perceptual issues. In this paper, we propose a rendering system that is designed around these two problems. We do not claim to have a solution, rather we have a partial solution made from current technology, and a direction for future development.

In Section 2 we examine the qualities required by typical computer graphics applications. Section 3 discusses how these qualities impact our modeling techniques. In Section 4 the importance of perception is expanded. Section 5 summarizes the the requirements for future renderers. In Section 6, we present our framework for a renderer satisfying these requirements. Section 7 presents directions for further work.

2 Applications of Future Renderers

The primary purpose of most computer graphics imagery is communication. This communication may simply be knowledge or information about the data being rendered, such as what a traffic safety engineer might need, or it can take on a deeper, more emotional form, such as that used by the entertainment industry.

The effectiveness of this communication depends critically on realism. Different applications require different kinds of realism, but we believe that all fall under two broad categories: *perceptual realism* and *visceral realism*. Such categorization is not useful in and of itself, but only as a guide for research.

- **Perceptual realism.** An image is perceptually realistic if a viewer of the image synthesizes a mental image similar to that synthesized by the virtual viewer.[1] For example, parts of the scene too dark to be visible to the virtual viewer should also be too dark in the image. Likewise, areas of the scene where the virtual viewer is blinded by glare should be washed out in the image. These and other effects would need to be simulated accurately for applications such as safety engineering, architectural simulation, or set design. Correctly modeling them requires a full, quantitative model for human perception. Though this is not yet available, some useful incomplete models have been developed[MG88, NKON90, TR93, SW91, GFMS94, CHS+93].
- **Visceral realism.** A viscerally realistic image suspends our disbelief in the same way that a good movie convinces us that it is really happening. Viewers need to have a deep, intuitive sense that the objects in the image actually exist, even if they are fantastic or improbable. These images would be necessary for applications such as training simulations, entertainment, or artistic applications. We believe that the key quality for visceral realism is *complexity*. This complexity takes many forms, and includes complexity of the material as well as geometric complexity. Images need to be rich and detailed. Objects need to be worn, dirty, cracked, weathered. Some elements of perceptual realism such as glare are also important.

For most applications, both forms of realism are important. For example, in a military training simulator, correctly reproducing the visibility or invisibility of objects is very important. But inducing the stress that the trainee would encounter in a real situation is also very important, and for this the trainee must believe, intuitively, that what he is experiencing is real. The former is a component of perceptual realism, and the latter is visceral realism.

3 Modeling Issues

3.1 Mapping Methods

Texture-mapping, bump-mapping, and solid textures are several well-known methods for adding complexity to an image[BM93]. While computationally practical and very effective for some kinds of complexity, they suffer from some fundamental limitations.

First, they do not actually extend the geometry. So detail where the viewpoint is close enough to actually see the geometry cannot be rendered with these

[1] Unless otherwise stated, the term *viewer* refers to the real person viewing the displayed computer graphics image, and the term *virtual viewer* refers to the imaginary person from whose viewpoint the image was generated.

techniques. Second, they cannot easily handle self-shadowing. Carpets and fur are two examples that are difficult to render accurately without self-shadowing. Lastly, they have difficulty handling dynamic objects. Replacing a field of wheat waving in the wind with a texture map would require that the map be accurately recomputed for every frame.

Texels[Kaj89] are somewhat less limiting. They include self-shadowing, and thus can produce convincing images of textures like fur. Extension to geometry that is significantly non-isotropic would greatly increase memory usage, however, since the scalar density in each cell must be replaced with a higher tensor quantity. Like texture maps, texels also cannot be used in areas where the geometry is distinguishable. Texels could perhaps be used with dynamic objects, though the processing involved would limit their usefulness.

For convincing complexity then, actual geometry is often necessary. Unfortunately, this produces an enormous number of primitives. Just correctly modeling the carpet in a large room can involve millions of primitives.

3.2 Levels of Detail

To reduce the number of primitives, multiple levels of detail are commonly used[FS93]. In this technique, an object has more than one representation. The representations have different degrees of detail. Depending on various factors such as the projected screen size of the object, the appropriate level of detail is selected. Thus, an object that covers only one pixel will be represented with many fewer primitives than that same object when it covers half the image.

This strategy can work well with static images, but when applied to an animation, several problems arise. First, avoiding "jumps" when changing from one level of detail to another is difficult. Second, even given that we have some method to change smoothly, generating enough intermediate representations to avoid disturbing transitions from one level of detail to a much different level of detail can be considerable work. Lastly and most importantly, producing intermediate representations for dynamic objects with arbitrary reflection characteristics can be extremely difficult.

For an intermediate representation to produce good results, it must have the same average reflective properties as the actual, full resolution object. For example, suppose we are rendering a distant forest. The trees are smaller than a pixel, so they could perhaps be represented by spheres. Now suppose the viewpoint moves closer so that the trees are approximately four pixels in size. At this point, a slightly more detailed representation should be used. In order to avoid a disturbing color shift, this slightly more detailed representation must have the same average reflective properties as the sphere representation, and both of these must have the same average reflective properties as the full representation of a tree.

Although perhaps feasible for static scenes, the above requirements are prohibitively difficult for arbitrary, dynamic objects. For example, animating a plain of wheat waving in the wind with low-detail patches of wheat would be difficult

without actually modeling the structures of an individual plant. But if the structures of the individual plant are actually modeled, then the level of detail is not very low, and we have lost most of the advantage of using different levels of detail.

3.3 Full Representation

The above problems can all be eliminated by representing everything in full detail. This immediately brings another problem: storage. A detailed model of a tree might require a million primitives. Scenes containing billions if not trillions of primitives are easily imaginable. Consider a large plain covered with prairie grass. Assume that every square meter contains one hundred plants. If 10,000,000 square meters are visible, then 1 billion plants must be rendered. If each plant is modeled with 1000 primitives, then the scene contains 1 trillion primitives total.

Such an extremely large number number of primitives imply that procedural models must be used. By procedural model, we mean that not only are the primitives generated by an algorithm, but that in addition the primitives are generated only on demand, and not explicitly stored. Some examples of high complexity procedural models are [Pru93, PH89, FS90].

Procedural models have other important advantages besides storage cost. Through procedural models, users can produce highly detailed and complex scenes with relatively sparse input. Without this capability, users would be burdened with having to provide an overwhelming amount of data about the exact geometric and physical properties of each and every object. Procedural models can also communicate with each other. Thus, the vegetation model underneath a tree model can understand that it is in shade rather than sunlight, and modify its growth accordingly[AGW86].

The question arises: when do we stop? If we are modelling a plant, why stop at the leaves? Why is it not necessary to model the microfacets of surfaces? The answer is that the depth to which it is necessary to model the actual geometry depends upon the application. Three questions are important. First, is the geometry distinguishable? If we are making an animation from the viewpoint of a virus, then modeling the microfacets is important because they can actually be seen. Second, does a good analytical model exist for the BRDF of the surface? For a complex surface such as skin, modeling the cellular structure might be necessary to capture the proper appearance[HK93]. But if a suitable analytical function exists, such as for brushed steel, then the microfacets need not be modeled. Lastly, in the time frame of application, is the surface changing appreciably? If we are generating a one-minute animation in a room with a wooden floor, modeling the microstructure of the wood is probably not necessary. If we wish to generate a realistic depiction of wood weathering over a five-year period, however, modeling the microstructure might be important.

Since we are primarily interested in rendering in the human spatial and temporal scale, for our purposes we do not need to model most microstructure as long as a satisfactory analytic approximation can be found for the BRDF. Even

in those cases where such a function cannot be found, tabulated results from Monte-Carlo simulations can be substituted for the microstructure[HK93].

4 Image Metamers

The human visual process synthesizes many different signals into an internal mental image. Normally, the input to this process is the light coming from the various surfaces in the scene. Our goal is to generate an image that, when viewed on a display device, results in the same internal mental image.

Note that the two human visual systems are not necessarily identical. For example, suppose our virtual viewer is an 80-year old man encountering a pedestrian while driving a vehicle. His visual system processes the light entering his eye into an internal mental image. Now, we have a traffic engineer sitting at his workstation. He wishes to know whether or not that 80-year old man can see the pedestrian. We need to generate an image that, when processed through *his* visual system, results in the synthesis of an identical internal mental image (Figure 1).

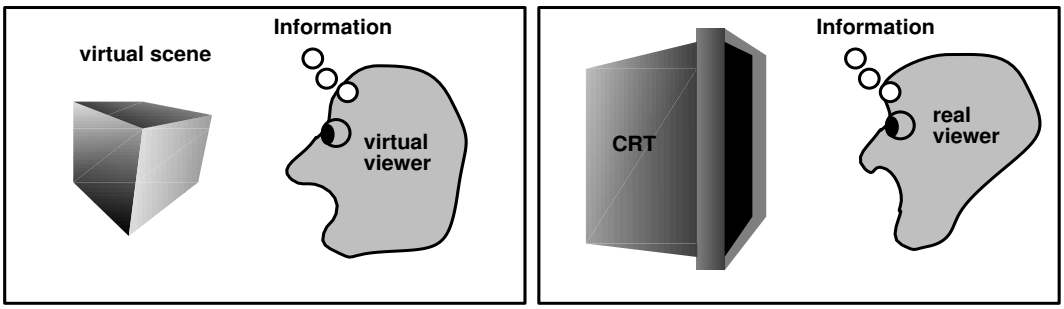

Fig. 1. Virtual person views a virtual screen and real viewer views a monitor.

Producing an exact match would probably require a perfect display device.[2] Lacking this for the forseeable future, we can only hope to produce an image that results in a similar internal mental image. Ideally, we could develop a error metric that weights differences between the two internal mental images in a way appropriate for the application. Part of the rendering process would then involve finding a perceptual transform on the computed radiance values that minimizes this error.

[2] If we had a perfect display device, and both human visual systems were identical, this would be trivial. We could simply use a physically accurate renderer to compute the light that would enter the eye of the virtual viewer. The perfect display device would then send the exact same light into the eye's of the device user. Since both viewer's eyes receive the same light, they will perceive the same image. Real display devices, however, have a limited dynamic range, a limited color gamut, and do not provide a full field-of-view.

Lacking the necessary perceptual model of the human visual system, researchers have aimed at producing somewhat weaker forms of perceptual realism. Rather than trying to duplicate the visual experience of the virtual viewer, we can try only determining which parts of the scene are visible and which parts are invisible, and displaying an image that reproduces this visibility condition for the computer user. Meyer has similar results for color-blind virtual viewers[MG88]. A viewer of the resulting image is unable to distinguish different colors where a color-blind virtual viewer is also unable to distinguish different colors. Tumblin and Rushmeier have investigated the more difficult problem of imitating the effects of dark adaptation under low-light conditions[TR93]. Nakamae et al. used glare to simulate high-intensity luminaires on a low-intensity display[NKON90]. Ultimately, all of these effects must be used in an integrated framework.

5 Requirements of Future Renderers

From the above discussion, we can summarize what capabilities a renderer must have to meet the realism requirements of current and future applications.

- **Procedural models.** The number of primitives required for convincing complexity simply cannot be stored, either now or in the foreseeable future. The modeling advantages of procedural models are also essential.
- **Arbitrary BRDFs.** Real surfaces are not some combination of perfectly diffuse and perfectly specular. Objects simply do not look real unless their surfaces are modeled and rendered with a convincing degree of physical accuracy. Near-specular surfaces such as brushed steel, semi-gloss paint, and varnish are especially important in many scenes.
- **Global illumination.** Indirect lighting is a very important component to realism. For example, the ceiling in a room lit with recessed ceiling lights would be black without including indirect lighting in the illumination computations. Other effects, such as caustics and color-bleeding, are somewhat less crucial, but can also be an important contributor to realism.
- **Many luminaires.** Indoor scenes are commonly lit with only a few lights. Outdoor scenes, however, may have thousands or even millions of lights.
- **Perceptual transform.** To meet requirements of perceptual realism, a perceptual transform must be performed at some stage of the rendering process.
- **Parallelizeable.** Current computer hardware already utilizes parallel architectures to improve performance. Future hardware is also likely to include parallel architectures, perhaps to a very high degree.
- **Physical accuracy.** In applications where the image is being used to preview a proposed design, such as set design, or architectural simulations, the calculated radiometric values must be accurate if the image is to be useful as a preview. For other applications, where perhaps visceral realism is more important, a larger degree of error is tolerable, but the accuracy must still be high enough to provide a convincing approximation to actual materials[War92, WAT92].

A balanced perspective must be used when evaluating the last requirement. While calculations should have a solid grounding in physics, we must not produce a renderer that reduces the complexity of the scene to make possible the generation of accurate radiometric values. For visceral realism especially, complexity is often more important that physical accuracy, as can be seen by the commercial success of movies such as *Jurassic Park*.

Different algorithms divide their computational resources between physical accuracy and complexity in different ways. A radiosity algorithm, for example, can produce extremely accurate physical results, but only for a fairly limited number of perfectly diffuse primitives. A classical ray-tracing program can produce images of much lower physical accuracy, but for a correspondingly much greater number of primitives. We propose that more effort should devoted to developing a renderer that sacrifices some of the accuracy of radiosity methods for complexity. Figure 2 diagrams the trade-offs between physical accuracy and complexity.

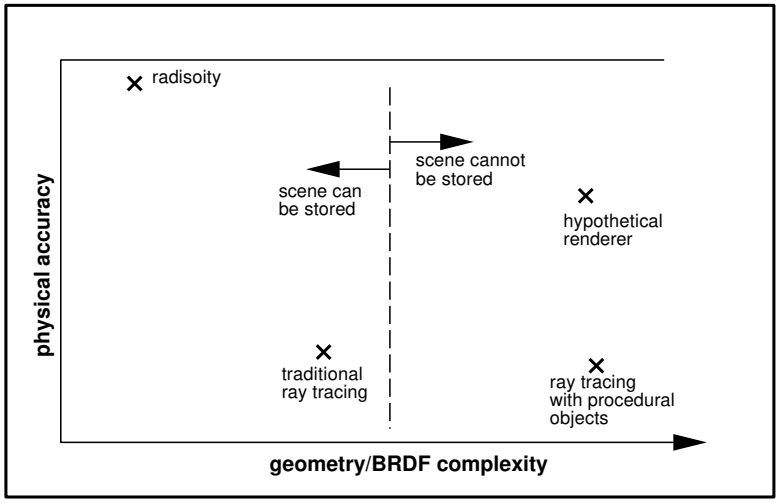

Fig. 2. Qualitative graph illustrating the trade-offs between physical accuracy and complexity for some algorithms.

6 A Framework

In this section we outline what we believe a renderer that meets the above requirements will look like. Our framework is only a proposal, and other frameworks may certainly meet our requirements.

Any method that needs all primitives ino memory at the same time will fail on scenes with sufficiently rich procedural objects. This includes most, if not

all, radiosity methods. Hierarchical schemes may manage to reduce the working set size of a renderer, but the introduction of general BRDFs into most radiosity methods greatly increases the memory storage required. We propose that a raytracing-based scheme will be more flexible, easier to program, and ultimately faster. This is an extension of arguments made by Kajiya at SIGGRAPH '88[Kaj88].

To expend resources where they will provide the most benefit, some kind of multi-stage adaptive sampling[vWvNJ91] is probably required. An initial set of samples through a pixel gathers information about what is seen through that pixel and determines whether more rays need to be sent. The difficulties of adaptive sampling are subtle[KA91a], but it can still work well in practice[Gla94].

To adaptively sample we need to know the accuracy desired at a given pixel. However, we assume that the acceptable error at a pixel depends not on the strict radiance of that pixel, but rather on the perceived brightness as produced by the visual system. If we know how a pixel radiance value will be mapped to a display intensity then we know the approximate accuracy needed for that pixel and can then adaptively sample that pixel. We have a "chicken-and-egg" problem because we don't know the mapping to display intensities for any of the pixels until all pixel radiances have been calculated. But if we had a approximate idea of what the mapping to display intensities was, this would provide information to allow us to conservatively apply adaptive sampling. This suggests the strategy of taking the initial samples at every pixel, and using this noisy image to estimate the mapping to display intensity for every pixel. Then pixels that have insufficient accuracy can be sampled in an adaptive phase. The architecture for such a strategy is illustrated in Figure 3.

The initial set of samples in the adaptive sampling gives enough information to approximate the perceptual transform. This tells us what the error metric is in a pixel space and can give us an estimate of sampling goals. The final renderer decides the effort needed to render each pixel based on the adapted image and the symbolic information passed by the initial renderer. The rendered image then filters through the perceptual module to produce the displayed image. Images are computed with full spectral resolution, and are stored as XYZ chromaticities and a fourth channel containing the rod response as detailed in the Appendix.

We have assumed that the perceived brightness of a pixel depends on nearby pixels. Though we believe this assumption to be true, the perceptual transform may be independent of nearby pixels, in which case the required accuracy can be determined without processing the entire image. In this case, no feedback loop would necessary, and the architecture would be a much simpler linear pipeline.

7 Further Work

The calculation of radiances can be formalized by the integral

$$L_\gamma(i,j) = \int_\lambda \int_t \int_{(x,y)\in P} \int_{(r,\theta)\in C} k w_P(x,y) w_\gamma(\lambda) L_\lambda(p) dA_C dA_P dt d\lambda$$

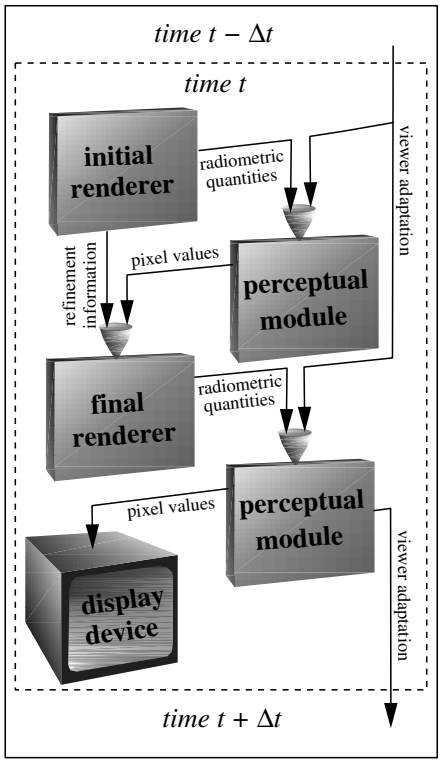

Fig. 3. Modules active during the generation of one frame.

where $L_\gamma(i,j)$ is the γ "channel"[3] of pixel (i,j), t is time, (r,θ) is the polar position on the lens C, (x,y) is the position on the film plane and P is the support of the filter for pixel (i,j), λ is the wavelength of light, and $L_\lambda(p)$ the the spectral radiance of the point p as seen by the point on the lens. Note that p is determined by the camera position at time t, (r,θ), and (x,y). This six dimensional integral is already difficult, but it is yet more difficult because $L_\lambda(p)$ is itself an integral equation:

$$L_\lambda(p,\psi) = L^E_\lambda(p,\psi) + \int_{\psi'} \rho(\psi,\psi') L_\lambda(p,\psi') \cos(\theta') d\omega'$$

Kajiya noted that incoming spectral radiance $L_\lambda(p,\psi')$ is just the outgoing radiance $L_\lambda(p',\psi')$ for some point p' seen by p in direction ψ'. The equation for $L_\lambda(p,\psi)$ is a Fredholm equation of the second kind. The result is a six-dimensional integral on the outside, and an inner term that is a recursive integral equation with a two-dimensional domain. Most authors assume no dispersion, and handle the wavelength using traditional quadrature (sample points). This reduces the dimension of the outer integral to five.

[3] Channel might mean a red, green, blue, CIE X, CIE Y, or CIE Z component.

One way to proceed is to integrate over the outer five dimensions of the integral using tradional Monte Carlo methods (distribution ray tracing) and use a different numerical technique for the inner recursive integral. Qualities of the integrand in the inner integral affect what techniques can work well. Generally, it will be mostly small, but will have very sharp spikes where the lights are. The location of the lights can be often be used to find the spikes, but this may not be possible in scenes where reflections from specular surfaces are an important contributor.

Zonal methods (radiosity) are common, but sometimes the integration domain is too complex for them to be practical. To handle arbitrary BRDFs, two-pass methods are also very numerous, and these are surveyed in [JC93]. Many of the methods have generated quite successful scenes (e.g., [KJ92]), but they still use shadow rays and will fail in highly complex scenes where virtual luminaires are important.

Interestingly, the only rendering algorithm that can compute the global illumination in an arbitrarily complex environment is Kajiya's path-tracing algorithm[Kaj86] and even his algorithm breaks down for scenes with many luminaires. Ward's *Radiance* program performs fairly well in the presence of dozens or hundreds of luminaires, but degenerates to Kajiya's algorithm for global illumination if the variation in surface normals is high, as is common in outdoor scenes.

We are aware of only two methods that do not use shadow rays: Glassner's[Gla93], and Kirk and Arvo's [KA91b]. But both of these force sampling of luminaires on the hemisphere. Virtual luminaires,[4] however, especially non-planar virtual luminaires, makes determining where the luminaires are in the first place computationally expensive. We believe that in a highly complex environment, treating anything as a special case is only marginally beneficial. So we reach the surprising conclusion that *Kajiya's path-tracing method is not too brute-force, but rather not brute-force enough.* We need a method such as Kajiya's path-tracing with no shadow rays, which is really just the random-walk methods of the 1940s. One such method is to first sample the incident radiance hemisphere to detect areas of interest. These areas are then resampled adaptively by sending out more incident radiance rays. Figures 4–5 are images created with this method.

Our current research focuses on improving brute-force methods that maintain the principle of no shadow rays. The "smartness" of the method will undoubtably focus on the use of low-resolution environments[RPV93, Kok93]. We believe most future rendering methods will use such low-resolution environments for at least the indirect lighting calculation.

[4] A virtual luminaire is a luminaire that does not generate its own light, but rather specularly reflects or refracts the light from other luminaires.

Fig. 4. Adaptive brute-force rendering with N incident radiance rays per pixel. This is treated no differently than if the light directly illuminated the chair.

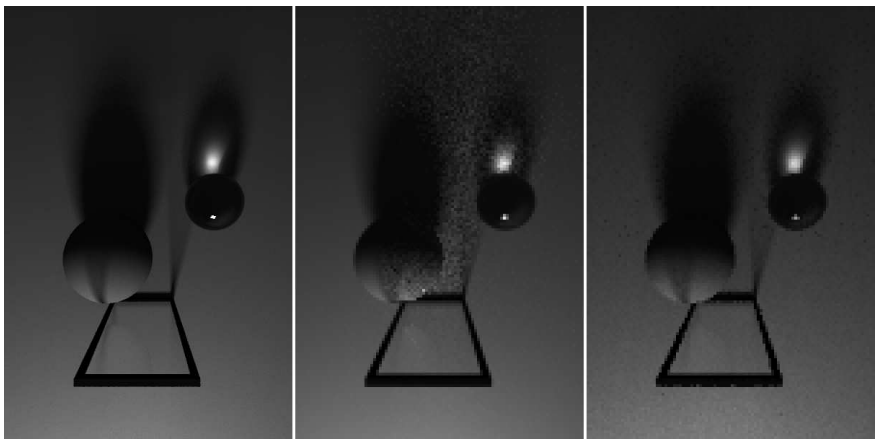

Fig. 5. Scene of light passing through a pane of glass. The image on the left is a reference image. The image in the middle was rendered with Kajiya's path-tracing algorithm. The image on the right was generated with adaptive brute-force rendering. Both images use the same number rays per pixel.

8 Conclusion

We believe that future applications will increasingly require perceptual accuracy in addition to physical accuracy. Without it, we cannot assure an architect that the walkthrough we have generated is anything at all like what a real person walking through the real building will experience. This is certainly not a new idea, but is one that needs to influence the focus of our rendering research. This is the fundamental difference between rendering research and heat-transfer research.

Future applications will also require a primitive database with too many primitives to store explicitly. We believe that this will be the only viable way to realize the goal of generating an animation that is as effective at suspending the viewer's disbelief as a normal motion picture. Such a large number of primitives probably means that physical accuracy must be selectively sacrificed.

To some degree, this implies that brute-force methods will become increasingly more useful. A field of grain waving gently in the wind would be difficult to model with multiple scales of resolution, especially as we change parameters such as moisture on the grain, species, time-of-day, cloud cover, season, soil type, terrain, etc. General techniques that consider all primitives at full resolution will become increasingly attractive as hardware performance increases, and as physical and biological models become increasingly common.

Note that we are not asserting that current work is useless. The techniques that we are proposing will not become practical, except perhaps for one week batch jobs, for many years to come. We merely wish to encourage researchers to reflect on the ultimate application of their work, now and in the future.

Realism of some kind has always been the goal of computer graphics, and physical accuracy has been implicitly assumed to be synonymous with realism. Perhaps this would be true in a world with fantastic computing power and display technologies, but in our world we must remember that our viewers are humans, not measuring instruments. As such, we believe that much current research efforts fail to reflect that for many applications, perception is as important as physics, and that for many other applications, accuracy should be subordinate to complexity. Understanding and attention to physical accuracy are undoubtedly crucial, but maintaining a balance between perceptual transforms, scene complexity, and physical accuracy must become our primary concern. In our quest for realism, we should not let ourselves become slaves of physical accuracy to the detriment of other important qualities.

Appendix: Implementation Issues

Because we want our rendering software to be applied to real problems, we must make an effort to support as many standards of measure and file format as possible.

For input, we must support luminaire distribution data that is supplied by manufacturers. The most common format in use now is the IES file format for far-field photometry[oNA91]. This format is also reviewed in Glassner's book[Gla94].

To approximate near-field photometry, we assume the power leaves the surface of the luminaire with equal radiance at each point.

Traditionally, integer-valued output files such as *ppm* have been used for computer graphics. For output format, we store floating point values. The new ray tracer instead outputs files with floating point values (4 bytes—3 bytes for 3 mantissas and one byte for a shared exponent). The file I/O routines are based on Greg Ward's *Radiance* format, and have proven to be very convenient to use, especially given the tools Ward makes freely available. Instead of display dependent RGB files, we output tristimulous XYZ response in SI units ($lm/(m^2\,sr)$). This unit is sometimes called the *nit*[Ins86]. You will also see luminances reported in *Lamberts*, which are $\pi/1000$ nits each. There is also a concept of scotopic (night/rod) luminance, which has the same units as photopic luminances. Because we are interested in viewing night scenes, we produce a second floating point file containing the scotopic luminance at each pixel. This brings up the question of whether a four mantissa format is needed.

A question we should have is when do we use the photopic XYZ values for display, and when do we use photopic? At luminances above 3 cd/m^2 we are in the photopic region, where colors can be distinguished and fine detail can be seen. From 0.01 to 3 cd/m^2 we are in the mesopic region, where some color and detail is visible. Below 0.01 cd/m^2 we are in the scotopic region where cones are inactive so there is no color vision[Rea93].

Another question we should ask is whether the one byte mantissa and exponent give us enough dynamic range. The brightest commonly seen object is the Sun, which has an average luminance of is about 1600×10^6 cd/m^2 viewed from sea level[Rea93]. This is well within the capabilities of the format.

The biggest hole in standardization now is scene file format. However, we do not yet understand how material properties or procedural models and textures are likely to be specified at an abstract level, so it is probably too early to work on a standard unless it is extensible.

9 Acknowledgements

Thanks to Changyaw Wang and Kurt Zimmerman who helped develop code used in this work. Thanks to Jack Tumblin and Holly Rushmeier for making their tone mapping work available to us early. Thanks to many members of the Cornell Program of Computer Graphics who gave valuable feedback on a presentation related to this work. Thanks to Greg Ward for software and guidance. Thanks to Andrew Glassner for helping us to ask the right questions about our own work. Thanks to Eric Lafortune for corrections. Thanks also to several excellent anonymous SIGGRAPH reviewers who commented on an earlier version of this paper.

This work was supported by Indiana University and by NSF grant *True Virtual Reality Systems* NSF-CCR-92-09457.

References

[AGW86] Phil Amburn, Eric Grant, and Turner Whitted. Managing geometric complexity with enhanced procedural models. *Computer Graphics*, 20(4):189–195, August 1986. ACM Siggraph '86 Conference Proceedings.

[BM93] Barry G. Becker and Nelson L. Max. Smooth transitions between bump rendering algorithms. *Computer Graphics*, pages 183–190, August 1993. ACM Siggraph '93 Conference Proceedings.

[CHS+93] K. Chiu, M. Herf, P. Shirley, S. Swamy, C. Wang, and K. Zimmerman. Spatially nonuniform scaling functions for high contrast images. In *Graphics Interface '93*, pages 245–244, May 1993.

[Coh92] Michael F. Cohen. Is image synthesis a solved problem? In *Proceedings of the Third Eurographics Workshop on Rendering*, pages 161–167, 1992.

[FS90] Mark Friedell and Jean-Louis Schulmann. Constrained grammar-directed generation of landscapes. In *Graphics Interface '90*, pages 244–251, May 1990.

[FS93] Thomas A. Funkhouser and Carlo H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics*, pages 247–254, August 1993. ACM Siggraph '93 Conference Proceedings.

[GFMS94] Andrew Glassner, Kenneth P. Fishkin, David H. Marimont, and Maureen C. Stone. Device-directed rendering. *ACM Transactions on Graphics*, 0(0):0–0, November 1994. To appear.

[Gla93] Andrew S. Glassner. Dynamic stratification. In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 5–14, 1993.

[Gla94] Andrew S. Glassner. *Principles of Image Synthesis*. Morgan-Kaufman, 1994.

[HK93] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. *Computer Graphics*, pages 165–174, August 1993. ACM Siggraph '93 Conference Proceedings.

[Ins86] American National Standard Institute. Nomenclature and definitions for illumination engineering. ANSI Report, 1986. ANSI/IES RP-16-1986.

[JC93] Frederik W. Jansen and Alan Chalmers. Realism in real time? In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 27–46, 1993.

[KA91a] David Kirk and James Arvo. Unbiased sampling techniques for image sysnthesis. *Computer Graphics*, 25(4):153–156, July 1991. ACM Siggraph '91 Conference Proceedings.

[KA91b] David Kirk and James Arvo. Unbiased variance reduction for global illumination. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.

[Kaj88] James T. Kajiya. An overview and comparison of rendering methods. *A Consumer's and Developer's Guide to Image Synthesis*, pages 259–263, 1988. ACM Siggraph '88 Course 12 Notes.

[Kaj89] James T. Kajiya. Rendering fur with three dimensional textures. *Computer Graphics*, 23(3):271–280, July 1989. ACM Siggraph '89 Conference Proceedings.

[KJ92] Arjan J. F. Kok and Frederik W. Jasen. Adaptive sampling of area light sources in ray tracing including diffuse interreflection. *Computer Graphics forum*, 11(3):289–298, 1992. Eurographics '92.

[Kok93] Arjan F. Kok. Grouping of patches in progressive radiosity. In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 221–231, 1993.

[MG88] Gary W. Meyer and Donald P. Greenberg. Color-defective vision and computer graphics displays. *IEEE Computer Graphics and Applications*, 8(9):28–40, September 1988.

[NKON90] Eihachiro Nakamae, Kazufumi Kaneda, Takashi Okamoto, and Tomoyuki Nishita. A lighting model aiming at drive simulators. *Computer Graphics*, 24(3):395–404, August 1990. ACM Siggraph '90 Conference Proceedings.

[oNA91] Illumination Engineering Society of North America. Ies standard file format for electronic transfer of photometric data and related information. IES Lighting Measurement Series, 1991. IES LM-63-1991.

[PH89] Ken Perlin and Eric M. Hoffert. Hypertexture. *Computer Graphics*, 23(3):253–262, July 1989. ACM Siggraph '89 Conference Proceedings.

[Pru93] Przemyslaw Prusinkiewicz. Modeling and visualization of biological structures. In *Graphics Interface '93*, pages 128–137, May 1993.

[Rea93] Mark S. Rea, editor. *The Illumination Engineering Society Lighting Handbook*. Illumination Engineering Society, New York, NY, 8th edition, 1993.

[RPV93] Holly Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. In *Graphics Interface '93*, pages 227–236, May 1993.

[SW91] Maureen C. Stone and William E. Wallace. Gamut mapping computer generated imagery. In *Graphics Interface '91*, pages 32–39, June 1991.

[TR93] Jack Tumblin and Holly Rushmeier. Tone reproduction for realistic computer generated images. *IEEE Computer Graphics and Applications*, 13(7), 1993.

[vWvNJ91] Theo van Walsum, Peter R. van Nieuwenhuizen, and Frederik Jansen. Refinement criteria for adaptive stochastic ray tracing of textures. In *Eurographics '91*, pages 155–166, September 1991.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. *Computer Graphics*, 26(4):265–272, July 1992. ACM Siggraph '92 Conference Proceedings.

[WAT92] Stephen H. Westin, James R. Arvo, and Kenneth E. Torrance. Measuring and modeling anisotropic reflection. *Computer Graphics*, 26(2):255–264, July 1992. ACM Siggraph '92 Conference Proceedings.

This article was processed using the LaTeX macro package with LMAMULT style