SCI INSTITUTE
TECHNICAL REPORT

Interactive Distribution Ray Tracing

Solomon Boulos[†], *Dave Edwards*[†], *J. Dylan Lacewell*[†], *Joe Kniss*[†], *Jan Kautz*[°], *Peter Shirley*[†], *Ingo Wald*[‡]
[†]School of Computing, University of Utah [°]University College London [‡]SCI Institute, University of Utah

UUSCI-2006-022

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

June 1, 2006

Abstract:

Distribution ray tracing uses multiple samples per pixel to produce antialiased images that include soft shadows, glossy reflection, motion blur, and depth-of-field. The two main potential barriers to making distribution ray tracing interactive are that many rays might be required, and that those rays are not coherent enough to derive efficiency from tracing them in packets. A new interleaved sampling approach based on the Sudoku puzzle is used to minimize the number of rays per pixel. An empirical demonstration is used to show that there is still enough coherence in the rays to allow for a per-ray cost near that of a traditional ray tracer. In addition, a demonstration is provided that participating media can also be handled interactively in a distribution ray tracer.

THE UNIVERSITY OF UTAH

Interactive Distribution Ray Tracing

Solomon Boulos[†], Dave Edwards[†], J. Dylan Lacewell[†], Joe Kniss[†], Jan Kautz[◇], Peter Shirley[†], Ingo Wald[‡]

[†]School of Computing, University of Utah [◇]University College London [‡]SCI Institute, University of Utah

Abstract

Distribution ray tracing uses multiple samples per pixel to produce antialiased images that include soft shadows, glossy reflection, motion blur, and depth-of-field. The two main potential barriers to making distribution ray tracing interactive are that many rays might be required, and that those rays are not coherent enough to derive efficiency from tracing them in packets. A new interleaved sampling approach based on the Sudoku puzzle is used to minimize the number of rays per pixel. An empirical demonstration is used to show that there is still enough coherence in the rays to allow for a per-ray cost near that of a traditional ray tracer. In addition, a demonstration is provided that participating media can also be handled interactively in a distribution ray tracer.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Ray tracing

1. Introduction

Over 20 years ago Cook et al. [CPC84] captured the graphics communities' attention with its stunning distribution ray tracing (DRT) images (Figure 1). Because DRT is based on ray tracing, it also allows natural support for participating media [KH84]. Unfortunately, DRT has remained an exclusively batch algorithm. In this paper, we attempt to make DRT interactive, or more precisely to demonstrate it can be interactive on the multicore chips already announced for production 2-3 years from now.

Unlike full DRT, Whitted-style [Whi80] ray tracing (restricted to viewing and shadow rays from point sources) has been made interactive on single CPUs for both static scenes [WSBW01, RSH05] and dynamic scenes [WBS06, WIK*06]. The performance of these systems rely on many factors including the use of ray packets and SIMD programming. There are three main potential barriers to extending such systems to DRT in the presence of participating media. First is that many rays may be needed per pixel for acceptable image quality. Second is that rays in DRT are less coherent than in traditional ray tracing, so it is not clear that ray packet techniques will continue to yield great efficiency benefits. Third is that even if DRT can be made fast for surfaces, participating media may dominate performance.

In this paper we attempt to show that those three barriers can be overcome without introducing limitations to the basic DRT algorithm. This is accomplished first by extending the use of ray packets and SIMD programming from Whitted-style ray tracing to DRT, second by using an interleaved sampling scheme that uses tiling inspired by the popular *Sudoku* logic puzzles to limit the number of samples per pixel, and third by a simple method to efficiently trace a packet of rays through a volume density. We also show how the performance of the system degrades as the packet coher-

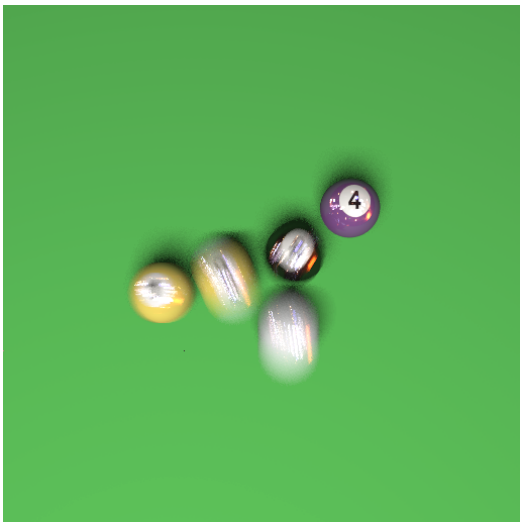

Figure 1: *A 512 by 512 pixel, 16 sample per pixel, distribution ray tracing image taken from an interactive session for an animated scene. The balls are 16,000 moving triangles and the rest of the scene is 280,000 static triangles. Generated in our interactive ray tracer running at 2-3 frames per second on a 16 core system.*

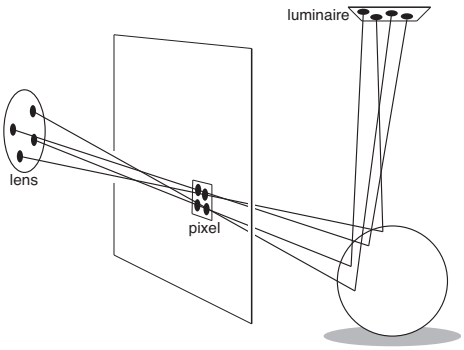

Figure 2: *The rays generated for one pixel in a DRT program. The sphere here is diffuse, so no specular rays are generated in this example. Note that there is no branching of rays. In this example, four samples per pixel are used, and six dimensions are sampled (two each for lens position, pixel position, and luminaire position). In the general case, there is an additional dimension for time, and two additional dimension for glossy specular reflection.*

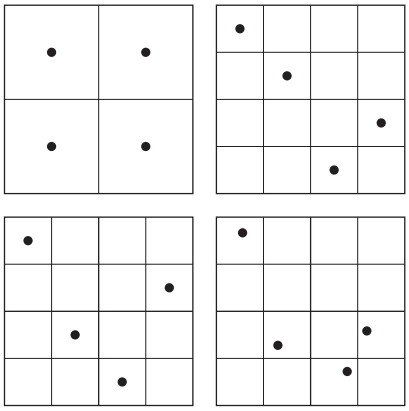

Figure 3: *Examples of four sampling strategies for four samples on the unit square. Top left: regular sampling. To right: Latin square sampling (each "block" has a sample at the center, and no two blocks are in the same row or column). Bottom left: Latin square sampling with the added constraint that one sample is in each quadrant. Bottom right: any deterministic strategy can be altered by random perturbation of a sample with a block; for example a latin square distribution can be randomized to form a "n-rooks" pattern.*

ence decreases with large lens aperture, large luminaire area, highly glossy reflection, and motion blur.

2. Background

A DRT program differs from a classic ray tracing program in that it takes multiple samples on a pixel, and each of these samples is associated with a different position on the camera lens, time, reflection direction, and luminaire position (Figure 2).

Most DRT programs generate samples on a unit hypercube rather than directly on on the nine-dimensional space the rays occupy. Further, most generate four two-dimensional patterns on $[0,1]^2$ and one one-dimensional pattern on $[0,1]$ and then use a permutation to assemble those into a set of nine-dimensional samples. These permutations can be random [KK02, SM03] or based on more sophisticated techniques [Coo86, KK02].

The samples themselves can be generated using Monte Carlo (MC) or quasi-Monte Carlo (QMC) techniques. Cook has shown that random sampling has visual errors whose noise properties are not very objectionable to viewers [Coo86]. Mitchell has argued that the sampling pattern should have certain frequency characteristics to minimize apparent error [Mit91]. Keller and Heidrich developed "interleaved sampling" and showed that creating dependent patterns across pixels could create aliasing that makes error less obvious because of local correlations [KH01, Kel04]. Interleaved sampling is based on a similar concept to that used in dithering: in the presence of unavoidable error, the error of nearby pixels should be anticorrelated, and a regular structure in the error can improve subjective image quality [Tho91].

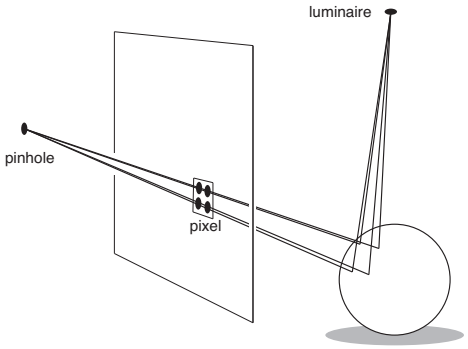

Figure 4: *In a Whitted-style ray tracer, all viewing rays start at a pinhole, and all shadow rays end at a points. These rays are more coherent than DRT rays in that they not only have common origins, but they have less directional spread in practice. Reflection/refraction rays (not shown) lack a common origin, but are more coherent in a typical Whiited-style ray tracer than in a DRT program.*

Keller has argued that 2D QMC patterns naturally have two important properties: uniform distribution in 2D as well as uniform distribution in each of the two 1D Cartesian axes [Kel04]. An example of such a pattern is shown in the bottom left of Figure 3. These good patterns can also be randomized for applications that demand unbiased solutions.

Whitted-style ray tracers produce "crisp" images because their rays follow deterministic paths without the randomized spread of DRT rays. A consequence of this is that sets of rays in a Whitted-style ray tracer are likely to be more coherent than in a DRT program (Figure 4). This is a cause for some concern for DRT efficiency because modern efficient ray tracers gain much of their speed from tracing "packets" of rays[†] together through the environment [WSBW01, WIK*06]. These packets must be somewhat coherent in what objects they visit to help efficiency, so it is not clear DRT programs can be made fast by borrowing these packet techniques.

3. Implementing Distribution Ray Tracing

For distribution ray tracing to approach interactivity we must group secondary rays into coherent packets. In our system, packets are groups of up to 16 rays traced together. One of the simplest indicators of coherency is shader type, or more specifically which component of the shading model generated the ray. Our system supports Phong shading, as well as refraction. Thus we group secondary rays so that a single packet only contains one of the following: shadow rays, reflection rays, or refraction rays. For example, a primary packet that hits a Phong-shaded plastic surface generates a shadow packet and a reflection packet. Purely diffuse or specular surfaces send only shadow or reflection packets, respectively. It is possible for the child packets to be only partially full, e.g., if a primary packet hits two overlapping surfaces, one of which is purely diffuse. There is no hard coding in our system for a particular shader model, and scene attributes such as camera aperture and light source size can be modified at runtime.

In this section we describe how we implemented each type of secondary ray packet, as well as camera rays and motion blur, and discuss requirements for acceleration structures. We assume a set of input sampling points and tile patterns. Varying these does not significantly affect the ray packet intersection and shading cost, though it can affect how many rays are needed for visual quality, as discussed in Section 4.

3.1. Camera Model

Recent interactive ray tracing systems have relied on pinhole cameras, in which all camera rays have a common origin and common signs, to achieve high performance using a kd-tree [WSBW01, Wal04, RSH05]. In distribution ray tracing, we achieve depth of field by jittering ray origins on a lens, which removes the common origin of primary rays. However, the bounding volume hierarchy presented by Wald et al. [WBS06], which we implemented in our system, does not have the common origin restriction as both the slabs test and interval arithmetic allow for differences in ray origins. We sample a disc-shaped lens by transforming uniform random variables into polar coordinates on the lens as follows:

$$\theta = 2\pi\xi_1, \; r = \frac{\alpha\sqrt{\xi_2}}{2},$$

where α is the aperture or lens diameter. The results of this are shown in Figure 5.

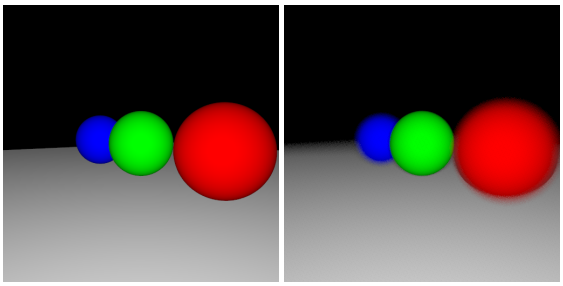

Figure 5: *Left: pinhole camera. Right: thin lens camera.*

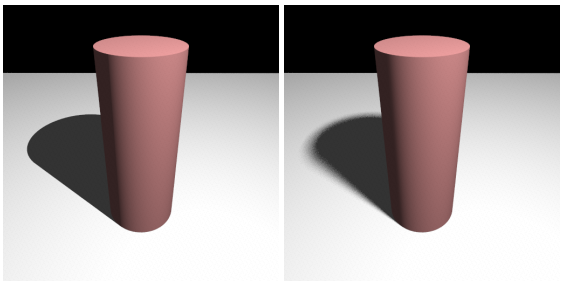

Figure 6: *Left: Hard shadows from a point light source. Right: Soft shadows from an area light source.*

For increased coherence, we group our rays in a tiled fashion to form ray packets. For example at one sample per pixel, sixteen rays would be traced in a 4x4 pixel block. When using 16 samples per pixel, all the rays within a pixel form a single packet which is assumed to be sufficiently coherent.

3.2. Soft Shadows

Whitted-style ray tracing uses point light sources and produces hard shadow boundaries. Previously, packets of coherent shadow rays have been shot from point light sources towards primary ray hit positions. Soft shadows are produced by using more realistic area light sources, which again removes the common packet origin.

One way to reintroduce a common origin is to shoot a packet of shadow rays from each hit position in the primary packet towards distributed sample points on the light source. However, this oversamples the shadows (i.e., we do not need

[†] Note that these packets are less constrained than the "beams" of beam tracing [HH84]. Packets are just arrays of rays and these rays do not necessarily have any required geometric relationships to each other.

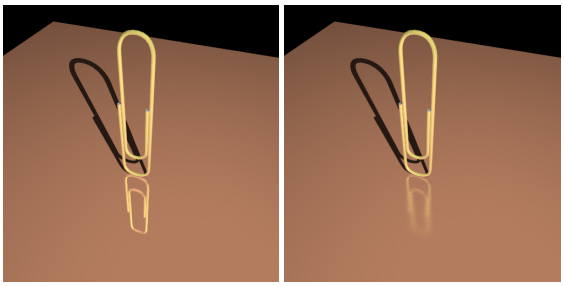

Figure 7: *Left: perfect specular Right: glossy reflection.*

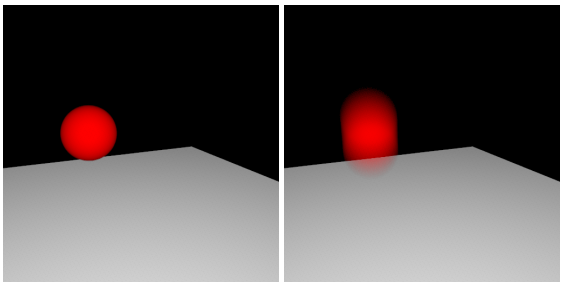

Figure 8: *Left: A moving sphere without motion blur. Right: With motion blur.*

16 visibility samples each of which uses 16 shadow rays) and violates the principles of DRT. We therefore continue to shoot shadow packets from lights toward geometry, with at most one shadow ray per primary ray.

As with lenses, sample positions on area light sources are chosen by using a simple mapping from uniform random variables to points on the luminaire. As the light source size grows, this produces a larger packet footprint that may contain rays with differing signs. For systems based on the kd-trees and other spatial subdivision techniques with strict ordering requirements, this requires splitting up what would otherwise appear to be coherent packets. Imagine a packet of 16 rays with only 1 degree of spread. This packet is certainly coherent, but for a kd-tree if the packet happens to cross an axis-aligned boundary it must be split. The BVH does not have this restriction, so this allows us to use larger packets and extract more coherence from rays that might be seen as incoherent for other acceleration structures.

For simplicity, we only usea rectangular luminaires. Although this would seem to be overly restrictive, the luminaire size is more important in determining shadow appearance than the luminaire's shape [SM03]. An example of a hard vs. a soft shadow is shown in Figure 6.

3.3. Reflections

Following Cook [CPC84], if an object has a reflective component we compute the direction of perfect reflection and perturb it to produce a glossy reflection. Recent packet based systems either switched to single ray code for secondary bounces (like OpenRT [Wal04]) or only handled planar reflections in a manner similar to beam tracing [HH84]. However, the difference between perfectly specular objects and slightly blurred is quickly apparent and desirable (See Figure 7).

To construct the blurry reflection direction, for each ray in the packet we first compute the perfect reflection direction and a coordinate frame around it:

$$\begin{aligned}\vec{R} &= \vec{V} - \hat{N}(2\hat{N}\cdot\vec{V})\\ \hat{W} &= \hat{R}\\ \hat{U} &= \hat{W}\times\hat{e_0}\\ \hat{V} &= \hat{W}\times\hat{U}\end{aligned}$$

where $\vec{V}$ is incident ray direction, $\hat{N}$ is the unit surface normal and $\hat{e_0}$ is the canonical first basis vector. If $\hat{R}$ is sufficiently parallel to $\hat{e_0}$, however, $\hat{U}$ will be close to the zero vector. In this case, $\hat{U}$ becomes $\hat{W}\times\hat{e_1}$. Each of these cross products reduces to simpler scalar calculations than a full cross product would entail due to the zeros in the canonical basis vectors. Note that this computation is done for each ray in the packet.

Once we have constructed this basis around the reflection direction, we can use the Phong model [Pho75] to perturb the direction using uniform random variables:

$$\begin{aligned}\phi &= 2\pi\xi_1\\ \cos(\theta) &= \sqrt[n+1]{\xi_2}\\ \hat{A} &= \langle\cos(\phi)\sin(\theta),\sin(\phi)\sin(\theta),\cos(\theta)\rangle\\ \vec{R} &= \hat{U}(\hat{U}\cdot\hat{A})+\hat{V}(\hat{V}\cdot\hat{A})+\hat{W}(\hat{W}\cdot\hat{A}),\end{aligned}$$

where n is the specular exponent. For computing the reflection coefficient, we use Schlick's approximation to the Fresnel equations [Sch93].

3.4. Motion Blur

Motion blur provides for substantially more realism and visual quality than framed rendering (See Figure 8). To implement motion blur, we require a random seed for a time value. If each ray is given a unique time value and multiple samples are taken within a pixel, we get an averaging effect that produces motion blur. When intersecting a primitive, each ray's time is used to create the primitive at the instance in time the ray represents. For triangles, this implies interpolating positions between frames according to the ray time value. While this approach is simple, it invalidates previous approaches for fast ray-triangle intersection [Wal04]. In our system, each

ray interpolates the triangle to its position based on the ray's individual time seed. A Moller-Trumbore style triangle test is then used on the interpolated primitive as the test does not require precomputation and is relatively fast [MT97].

Motion blur requires that acceleration structures can handle moving primitives. We are using a Bounding Volume Hierarchy as our acceleration structure, so this can easily be handled by ensuring that the bounding box of the primitive encloses the primitive for any interpolated position. For simplicity, we are using linear interpolation of positions between frames, so a primitive's bounds is simply the union the bounds from the previous frame with the current frame. Other acceleration structures that handle ray tracing of dynamic scenes might also handle motion blur in a similar fashion [WIK*06].

3.5. Refraction

If a primary packet containing N rays hits a dielectric, it splits into two secondary packets, one for reflection and one for refraction, which contain N rays between them. Rays from the primary packet reflect with constant probability P and refract with probability $(1-P)$, and receive weights R/P and $(1-R)/(1-P)$, where R is the Schlick approximation to the Fresnel term. For example, with $N = 16$ and $P = 0.25$ the first reflection packet would receive about 2 rays. In practice we found that a maximum refraction depth of 3 was sufficient for visually compelling glass, such as the ashtray in the pool table scene in Figure 9. Each bounce is attenuated according to the distance it travels (e.g, using Beer's Law). If a packet exceeds the maximum refraction depth we use its direction to lookup into a prefiltered environment map. Packets that are reflected or refracted through a dielectric have similar coherency as blurry reflection packets, but contain less rays on average due to splitting.

3.6. Participating Media

Participating media adds an additional level of realism that provides a sense of depth and atmosphere. In outdoor scenes, an atmospheric skylight model is essential for communicating distance and turbidity. Amorphic, yet dynamic phenomena like smoke, clouds, and mist are intrinsically volumetric in nature and not easily handled using surface-based primitives.

General volumetric models are rendered by integrating the volume rendering equation using discrete ray marching. In our system, volume primitives are rendered after geometric ray intersection and shading. One nice aspect of volume rendering is that it rarely requires anti-aliasing, as volume models tend to be smooth and fuzzy. As such, multi-sampling volume primitives provides little or no qualitative improvement when compared to single ray integration. When the scene is rendered using multiple samples per-pixel, a single volume ray is shot for the entire ray packet. Each ray from the packet is composited with the volume as the volume ray marches past the multi-sample ray's intersection point. Naturally, if a ray does not intersect any geometry, it's intersection point will be infinity. In this case the ray's color is composited with the complete volume ray solution.

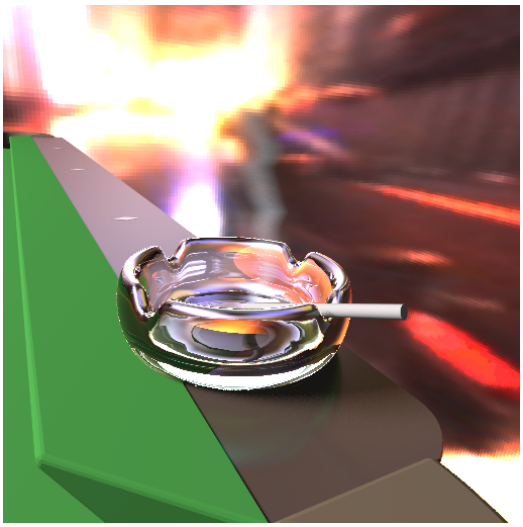

Figure 9: *A glass ashtray with maximum* 3 *refractions and a prefiltered environment map.*

Mist in the fairy scene and cigarette smoke in the pool hall are dynamically generated volume primitives. They utilize a simple analytic base volume that is perturbed by a noise vector field, which is represented as a tiled 3D texture. The mist and smoke are animated by moving the texture coordinates of the noise texture. Both of these volumetric effects represent phenomena that do not necessarily require extensive self shadowing. The mist is a thin, high albedo media, and the smoke is a small scale thin, yet low albedo media. Since the base volume structure for each volume is known, the subtle volume shading required can be computed analytically based on depth and lighting angle. Shadows cast by geometric objects are computed in the traditional way, using shadow rays. Just as with other parts of the system, we send numerous rays simultaneously as packets. Our implementation of volume ray marching is straight forward with simple optimizations such as early ray termination based on opacity, and jittered volume ray starting points to remove obvious aliasing artifacts in shadows cast through the volume.

The skylight model does not require ray marching. It is an analytic approximation that handles chromatic atmospheric extinction and skylight inscattering. The sky model uses a stratified, depth dependent atmospheric density and a simple Rayleigh and Mie scattering approximation. High-level controls include turbidity and pollution content, allowing the scene impact to vary from clear and dry to humid to smoggy.

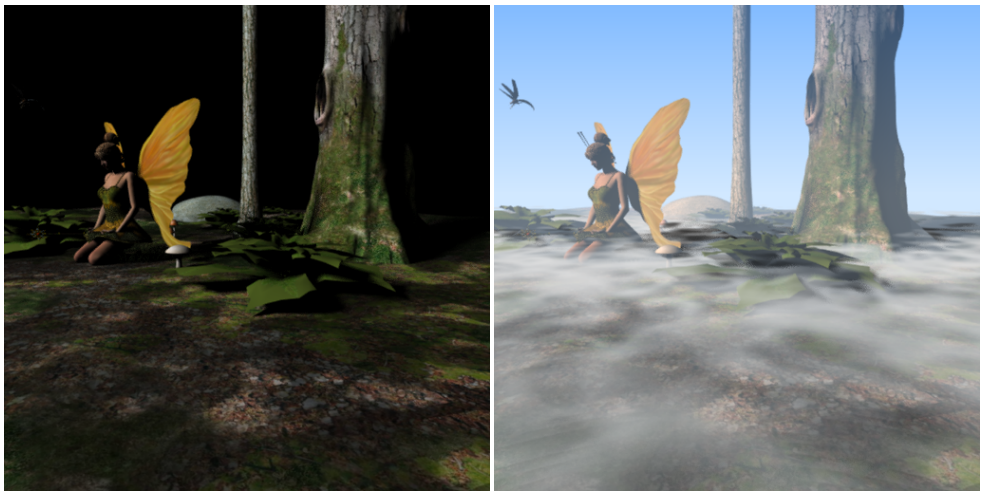

Figure 10: *Left: A scene without participating media. Right: The same scene with participating media, including an analytic skylight approximation and a dynamic mist layer.*

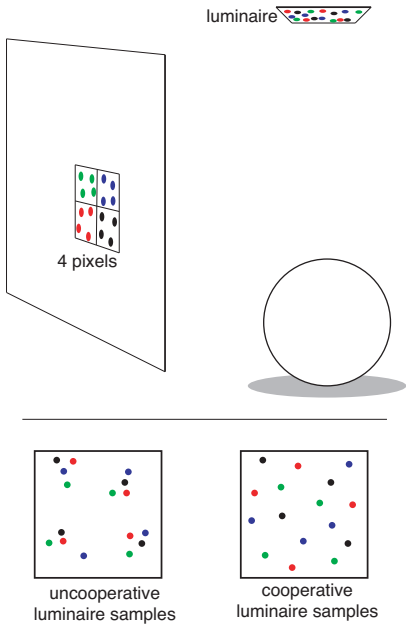

Figure 11: *Four adjacent pixels each generate samples on the luminaire. If these samples "cooperate" they can be overlaid and still make a good pattern.*

The fairy scene in Figure 10 has high turbidity and low, yet non-zero pollution.

4. Sample Generation

In an interactive program our random seeds cannot come from random number generators at runtime. If we were to fill the random seeds from a generator, our renderings would exhibit temporal scintillation, because the sample pattern would change whether or not the camera were still. Although reseeding the random number generator per frame sounds like a reasonable solution, it is only viable in the case of a single threaded system. In a multi-threaded system, the differences in work assignments per frame will cause a similar scintillation effect. To solve this problem, we use stable sample patterns for each dimension.

			3
2		4	
	1	3	
	2		4

1	4	2	3
2	3	4	1
4	1	3	2
3	2	1	4

Figure 12: *Left: an order-2 sudoku puzzle. Right: the solution to the puzzle where every row, column, and quadrant has exactly one of each digit.*

Current computational power does not allow enough samples to obtain convergence. It would be preferable for the error that remains to be as unobjectionable as possible. For this reason we employ interleaved sampling [KH01]. Performing interleaved sampling for antialiasing involves two basic choices: the choice of sampling patterns, and the choice of how these patterns tiled on the screen. For DRT, we must also choose how the non-screen dimensions such as luminaire position should be sampled. This section describes techniques for both sample generation and tiling. We believe that our tiling technique is the more critical innovation.

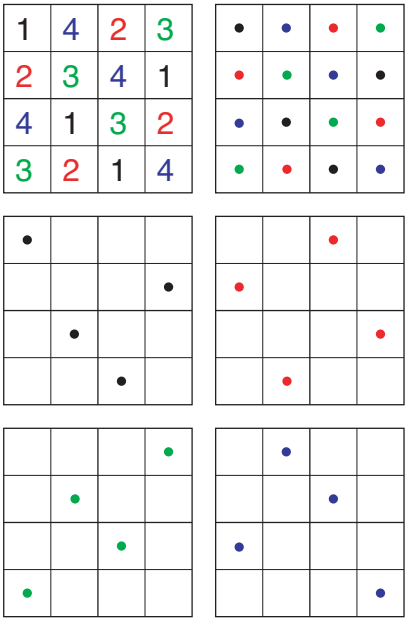

Figure 13: *An order-2 sudoku solution can be used to make four Latin square patterns that share no samples. Upper right: When all four patterns are merged they make a 16-sample regular grid. We refer to such sample patterns as "cooperative" samples.*

9	1	3	8	13	14	10	11	2	15	12	0	5	7	6	4
11	6	5	15	3	9	2	8	10	1	4	7	14	13	0	12
0	12	14	10	7	4	6	1	5	3	9	13	15	2	8	11
2	13	4	7	12	0	15	5	14	6	8	11	1	3	10	9
10	5	2	12	15	8	11	9	4	13	6	14	7	0	1	3
7	0	9	3	5	6	13	14	1	2	10	15	11	12	4	8
8	14	11	6	1	2	3	4	9	0	7	12	13	10	15	5
4	15	13	1	0	12	7	10	3	5	11	8	2	6	9	14
6	8	0	9	10	11	4	13	7	14	15	3	12	5	2	1
12	3	15	13	2	1	14	6	0	4	5	10	8	9	11	7
5	7	1	2	8	15	12	3	11	9	13	6	0	4	14	10
14	4	10	11	9	5	0	7	8	12	1	2	6	15	3	13
1	2	7	5	14	13	8	0	15	10	3	9	4	11	12	6
15	9	8	4	6	10	5	12	13	11	0	1	3	14	7	2
13	10	6	14	11	3	1	15	12	7	2	4	9	8	5	0
3	11	12	0	4	7	9	2	6	8	14	5	10	1	13	15

Figure 14: *The order-4 sudoku solution we use for both our sample distribution and our tiling. Note that each bordered block contains the numbers 0 through 15 exactly once, but other four by four blocks (such as the grey one) may include the same number twice.*

4.1. Cooperative Sampling

In the original work on interleaved sampling, motion blur was computed by using a well-distributed set of time samples over a multi-pixel tile. For example, if each pixel in a four-pixel tile used four time samples, then all sixteen time samples on the tile formed a well-distributed sample pattern. Although not explicitly explored in that work, the same principle is valuable for sampling other dimensions, as illustrated in Figure 11. We use the adjective "cooperative" to refer to sample patterns on a tile that can be merged to obtain well-distributed samples. In this sense, the the time samples used by Keller and Heidrich are cooperative [KH01]. The image space samples they used are not cooperative, although that was not a limitation for their antialiasing application.

One method for generating cooperative sample patterns in two dimensions is based on the popular Sudoku game [Hay06]. We start with a solution as shown on the right of Figure 12. If we look at an individual digit in a solution, it defines a Latin-square pattern that is also stratified in 2D. Each of the N digits defines such a pattern, and the N patterns together define a pattern of N^2 samples that is a regular lattice (Figure 13).

For an order-4 puzzle, we can get 16 cooperative patterns of 16 samples each. The particular solution we use is shown in Figure 14. These sample patterns can be used in 4x4-pixel tiles, and should be good for interleaved pixel samples as well as cooperative luminaire samples. However, we found that sampling strategies using 4x4-pixel tiles exhibit noticeable aliasing (see left half of Figure 16).

4.2. Tiling Arrangement

To remove the tiling artifacts we observe, we use the same principle that makes QMC sampling superior to uniform sampling: random noise is usually a less objectionable artifact than frequency aliasing. If we have 16 cooperative patterns of 16 samples each, we can assign these patterns to pixels in a deterministic but non-uniform way to remove some of the regularity caused by 4x4-pixel tiling. Conveniently, we can also use a sudoku puzzle solution to determine which sample pattern to use on pixels within a tile (see Figure 15 for an order-2 example). Using our method with an order-4 sudoku puzzle yields 16x16-pixel tiles, which remove some of the frequency artifacts from smaller tiles. We refer to this method of tiling sample patterns onto pixels as "sudoku tiling." In general, the results from sudoku tiling are superior to results using regular 4x4-pixel tiles (see Figure 16).

We show several combinations of sampling and tiling strategies for luminaire sampling in Figure 16. In our experience, the tiling method affects the final image more than the sampling method, as long as the sample sets are not uncooperative. All of the figures are generated using 16 patterns of 16 samples each (i.e., 16 samples per pixel). The left column

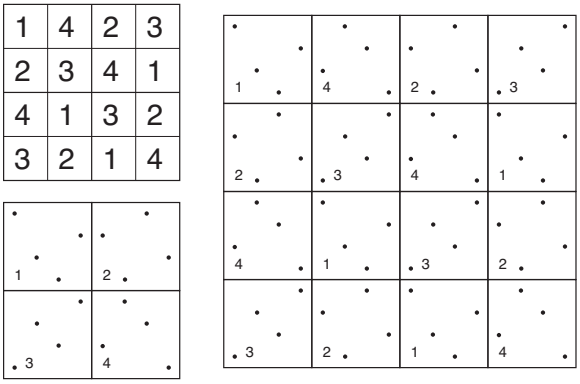

Figure 15: *The four sample patterns from the order 2-sudoku solution from Figure 12 can be trivially arranged into a 2x2-pixel tile (bottom left). Instead, we use the puzzle solution again to arrange the sample sets into a 4x4-pixel tile. Sample set 1 (generated from the locations of 1s in the puzzle) is placed wherever there is a 1 in the puzzle (right).*

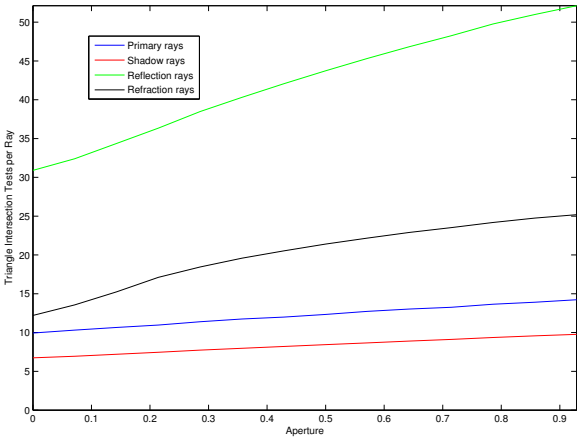

Figure 17: *Number of triangle intersections per ray type as aperture is increased from a pinhole through extreme blur.*

uses 4x4-pixel tiles; each pixel in the tile uses one of the 16 patterns. The right column uses 16x16-pixel tiles, where the sample pattern used is determined by the index in the Sudoku solution shown in Figure 14. The particular sampling startegies shown are:

Uncooperative Hammersley: A 256-sample Hammersley pattern is generated on the unit square $[0,1]^2$. The unit square is divided regularly into 16 square cells (i.e., 4 cells by 4 cells), each of which contain 16 samples. A sample pattern consists of all samples within a given cell, rescaled to fit within the unit square. These patterns turn out to be particularly uncooperative.

Independent jittered: 16 independently generated jittered (stratified random) sample sets. These are not explicitly cooperative, but in practice they work better than the previous sample patterns.

Independent sudoku: 16 Latin-square stratified sudoku patterns taken from 16 different sudoku puzzle solutions. Because they come from different puzzles, these patterns are not necessarily cooperative.

Cooperative sudoku: 16 Latin square stratified sudoku patterns taken from the same puzzle analogous to the four patterns in Figure 13. These points cooperate to form a regular 256-sample lattice when merged.

Cooperative Latin-square sudoku: Here the points in the cooperative sudoku solution are slightly perturbed, so that when merged they form a 256-sample Latin square pattern.

Cooperative Hammersley: The same as the initial method, except that sample patterns consist of one unique sample point from each cell in the unit square (rather than all samples from one cell). These16 patterns cooperate to form a 256-point Hammersley pattern.

It is not clear which strategy in Figure 16 produces the best images, but the uncoopertive Hammersley patterns on the top row are clearly the worst. All of the methods shown are stable, in that they do not produce any scintillation when the viewpoint and objects are static. This is superior to jittering with 16 new random samples per pixel in every frame, which has objectionable time-dependent noise in our experience. The bottom two rows of the figure are both similarly good; this implies that there are probably many good ways to generate cooperative samples. It is likely that other tiling methods work well, but we have not investigated that.

5. Empirical Evaluation of Ray Coherence

In this section, we examine our system with respect to each feature of distribution ray tracing over a 200 frame animation path of the billiards scene. This allows us to investigate ray coherence in an empirical setting. Each of these tests was chosen to examine a particular feature that would be expected to greatly reduce ray coherence and therefore increase the number of primitive intersections. All tests in this section manipulate a single variable to differ from the settings used in the video. Please see the accompanying video for a demonstration of this path with what we believe to be suitable settings for aperture, light source size and Phong exponents.

5.1. Depth of Field

As aperture increases from a pinhole to an extreme blur, coherence is decreased due to separated ray origins . These incoherent primary rays produce incoherent hitpoints, which produce incoherent secondary rays. Shadow rays are not as strongly influenced, because incoherent hitpoints are still being linked to coherent locations on the luminaire. Figure 17

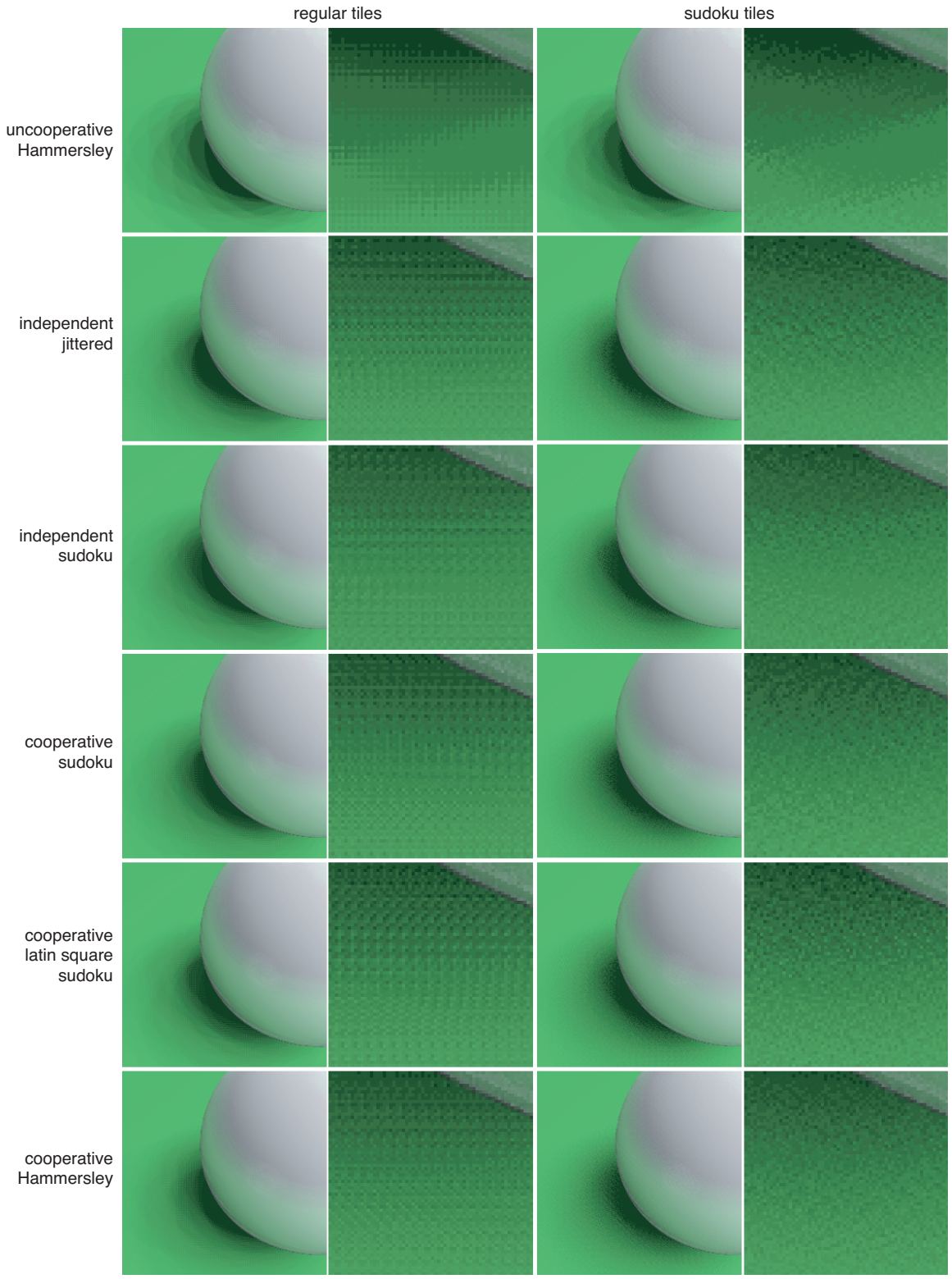

Figure 16: *Left two columns: using regular 4x4 tiles with 16 sample patterns. Right two columns: using 16x16 sudoku tiles with 16 sample patterns. Please see full resultion image in supplementary materials.*

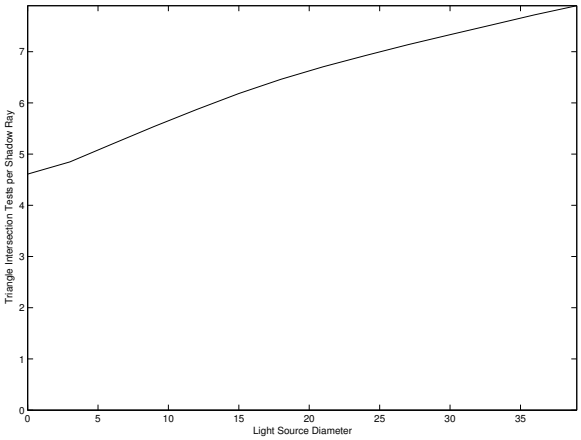

Figure 18: *Number of triangle intersections for shadow rays as luminaire diameter is increased from 0 to approximately the scene box size.*

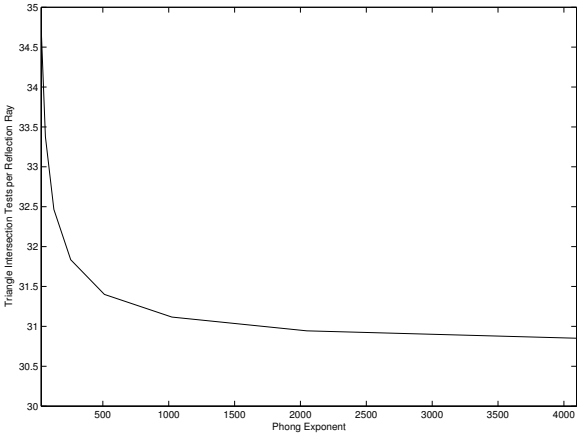

Figure 19: *Number of triangle intersections for reflection rays as Phong exponent is increased from 32 to 4096 by doubling.*

demonstrates an approximately linear increase in the number of triangles intersected for each ray type with respect to the aperture size.

5.2. Soft Shadows

Increasing the size of luminaires produces softer shadows, but also spreads the ray origins for shadow rays (shadow rays are shot from luminaires to hit points). While an increase in aperture decreases the coherence of all ray types, larger luminaires only affect shadow rays. It should be noted, however, that despite this wide range of luminaire size the number of primitives intersected by shadow rays does not increase wildly (See Figure 18).

5.3. Reflection

Perfectly specular reflection, especially for planes, is highly coherent. As the specular exponent decreases divergence would be expected for reflection rays. We tested the divergence of reflection rays by setting the Phong exponent for the billiard balls between 32 and 4096. For reference, the exponent used in the video is 2048. Figure 19 demonstrates that coherence does not change as greatly as we would expect. However, the performance of reflection rays may be negatively impacted by the motion blur of the billiard balls.

5.4. Motion Blur

Motion blur is the most interesting of the features as it relates to ray coherence. Motion blur, like camera aperture, is able to extend its influence past primary rays to both shadow rays and reflection rays (See Figure 20). For these rays the impact of motion blur is approximately a factor of two. It should be noted that without motion blur, the reflection rays in the scene are approximately as coherent as other types of rays.

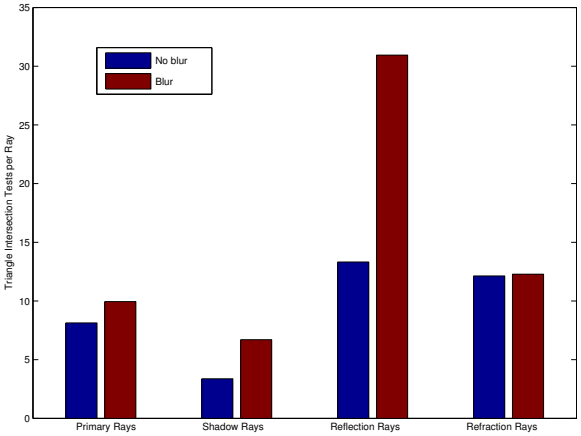

Figure 20: *Number of triangle intersections for each ray type both with and without motion blur.*

5.5. Participating Media

The average impact of our dynamic volumes on rendering performance is approximately a factor of two. The performance varies depending on the portion of the view filled by the volume and it's density. Higher density volumes will benefit from early ray termination, while low density volumes may require full ray marching throuth the volume domain. The average number of volume ray march steps in both scenes is approximately 200. Since the skylight model does not require marching, it's impact on rendering performance is low; approximately ten percent reduction in framerate.

6. Conclusion

Interactive distribution ray tracing, including rich visual effects such as depth of field, soft shadows, glossy reflec-

tions, motion blur and participating media is available now on high-end multicore systems. The main contributions of our work are a novel tiling scheme that generates reasonable animation quality at 16 samples per pixel, and a demonstration that a careful DRT implementation that groups rays into packets based on ray type alone can derive almost the same benefits from ray packets that Whitted-style ray tracers do. By using an acceleration structure that does not rely on common origin, common sign, or other restrictions that are required by the features of DRT, we achieve high performance. Currently our performance is limited to around 1-20 frames per second (depending on the model and scene settings) at resolutions of 512^2 for 16 samples per pixel. Because multicore systems are becoming the de facto architecture, we hope that an order of magnitude increase in performance will be delivered by the parallelism from new, larger multi-core architectures.

It is not clear whether our sampling techniques have reached diminishing returns at low sampling densities. Better tiling or QMC patterns might yield further image quality benefits. In our system, we currently use a simple box filter for reconstruction of samples. A higher order filter may produce better images. Similarly, sophisticated ray scheduling based on more than just ray type may be able to improve performance; however, this is certainly an area for future work. We are interested in ambient occlusion and diffuse interreflection as future extensions to this work. Diffuse interreflection would likely require many more samples per pixel than we take, but it seems that localized ambient occlusion might be practical because the associated rays are coherent in spatial extent if not direction.

We have not yet compared directly with the interleaved sampling that employs incremental QMC sampling for multiple dimensions as used by Keller et al. [Kel04]. Their techniques can implicitly provide coopertive sampling patterns for shadows and reflection. We plan to perform an empirical comparison with Keller et al.'s techniques.

We believe the most important open question is what applications would benefit from DRT. If DRT turns out to be highly desirable in video games for example, there would be incentive to design special purpose DRT hardware. If DRT is desired by high-end applications, then our approach can yield full screen fluid interactivity today given a more expensive machine. In the near future, we anticipate this performance to be accessible at workstation-level pricetags.

Acknowledgements

This work was partially supported by NSF grant 03-06151 and the State of Utah Center of Excellence Program. The first author was also supported by the Barry M. Goldwater Scholarship. The last author was supported by the U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions under grant W-7405-ENG-48.

References

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Transactions on Graphics 5*, 1 (1986), 51–72.

[CPC84] COOK R., PORTER T., CARPENTER L.: Distributed Ray Tracing. *Computer Graphics (Proceeding of SIGGRAPH 84) 18*, 3 (1984), 137–144.

[Hay06] HAYES B.: Unwed numbers - the mathematics of sudoku. *American Scientist 94*, 1 (2006).

[HH84] HECKBERT P. S., HANRAHAN P.: Beam tracing polygonal objects. In *Proceedings of SIGGRAPH* (1984), pp. 119–127.

[Kel04] KELLER A.: Myths of computer graphics. In *Monte Carlo and Quasi-Monte Carlo Methods*, Talay D., Niederreiter H., (Eds.). 2004.

[KH84] KAJIYA J. T., HERZEN B. P. V.: Ray tracing volume densities. In *Proceedings of SIGGRAPH* (1984), pp. 165–174.

[KH01] KELLER A., HEIDRICH W.: Interleaved Sampling. *Rendering Techniques* (2001), 269–276. (Proceedings of the 12th Eurographics Workshop on Rendering).

[KK02] KOLLIG T., KELLER A.: Efficient Multidimensional Sampling. *Computer Graphics Forum 21*, 3 (2002), 557–563. (Proceedings of Eurographics 2002).

[Mit91] MITCHELL D. P.: Spectrally optimal sampling for distributed ray tracing. In *Proceedings of SIGGRAPH* (1991), pp. 157–164.

[MT97] MÖLLER T., TRUMBORE B.: Fast, minimum storage ray triangle intersection. *JGT 2*, 1 (1997), 21–28.

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317.

[RSH05] RESHETOV A., SOUPIKOV A., HURLEY J.: Multi-level ray tracing algorithm. In *(Proceedings of SIGGRAPH* (2005), pp. 1176–1185.

[Sch93] SCHLICK C.: A customizable reflectance model for everyday rendering. In *Fourth Eurographics Workshop on Rendering* (1993), pp. 73–84.

[SM03] SHIRLEY P., MORLEY R. K.: *Realistic Ray Tracing*, second ed. A K Peters, 2003. ISBN 1-56881-198-5.

[Tho91] THOMAS S. W.: Color dithering. In *Graphics Gems II*. 1991, pp. 72–77.

[Wal04] WALD I.: *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Saarland University, 2004.

[WBS06] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using bounding volume hierarchies. *ACM Transactions on Graphics (conditionally accepted, under revision)* (2006).

[Whi80] WHITTED T.: An improved illumination model for shaded display. *CACM 23*, 6 (1980), 343–349.

[WIK*06] WALD I., IZE T., KENSLER A., KNOLL A., PARKER

S. G.: Ray tracing animated scenes using coherent grid traversal. *ACM Transactions on Graphics (to appear)* (2006). (Proceedings of ACM SIGGRAPH 2006).

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. In *Proceedings of Eurographics* (2001), pp. 153–164.