Component-Based Adaptive Sampling

Kurt Debattista and Alan Chalmers
Department of Computer Science
University of Bristol
United Kingdom
{Kurt.Debattista, Alan.Chalmers}@bristol.ac.uk

Abstract

High-fidelity renderings of virtual environments is a notoriously computationally expensive task. One commonly used method to alleviate the costs is to adaptively sample the rendered images identifying the required number of samples according to the variance of the area thus reducing aliasing and concurrently reducing the total number of rays shot. When using ray tracing algorithms, the traditional method is to shoot a number of rays and depending on the difference between the radiance of the samples further rays may be shot. This approach fails to take into account that different components of a material reflecting light may exhibit more coherence than others. With this in mind we present a component-based adaptive sampling algorithm that renders components individually and adaptively samples at the component level, finally composting the result to produce a full solution. Results demonstrate a significant improvement in performance without any perceptual loss in quality.

Keywords: high-fidelity graphics, adaptive sampling, component-based rendering.

1. Introduction

High-fidelity physically rendering of complex scenes is one of the major goals of computer graphics. Rendering algorithms that compute high-fidelity images, traditionally based on ray-tracing [31] and its extensions, or radiosity [12] are notoriously computationally expensive. While radiosity rendering is usually a function of the number of elements, traditionally patches, needed to be rendered, ray-tracing computation is a function of the number of samples traced to render a scene. Ray-traced images rely on shooting rays to calculate the radiance at a given pixel. The radiance of adjacent pixels may tend to exhibit spatial coherence, by which radiance values close to each other are similar. This property has been taken into consideration for designing adaptive sampling algorithms [31], whereby rays are not traced at each pixel but at certain intervals and only if the resulting difference in the radiance at the pixels is above a certain threshold are the intermediate pixels calculated, see Figure 1. These algorithms can improve speedup substantially by reducing the total number of rays significantly, particularly in areas of high variance.

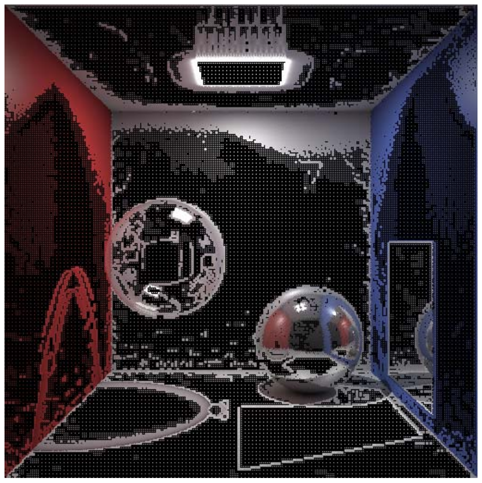

Figure 1. Traditional adaptive sampling. Samples only, no interpolation.

The traditional method of doing image plane adaptive sampling was at the level of the pixel's gross radiance computation. In this paper we introduce a new flexible mechanism for adaptive sampling based on the reflectance function of a material whereby the light that hits a surface can be divided into a number of components, see Figure 2. When using ray-tracing techniques it has long been assumed that certain reflective properties of a material exhibit different behaviours. For example, indirect diffuse computations are traditionally smooth and do not change much over space, a property that Ward *et al.*'s irradiance cache [30] takes full

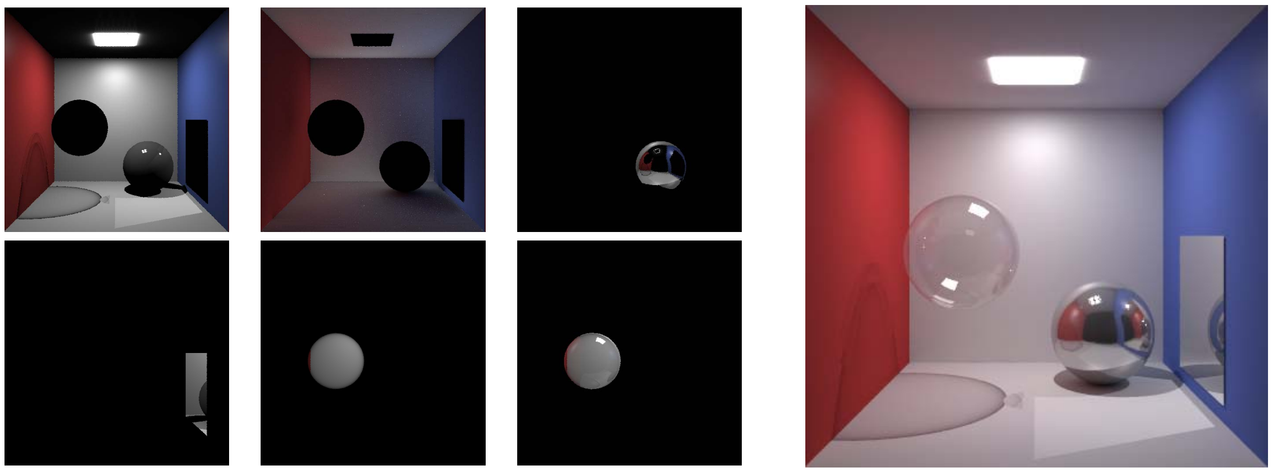

Figure 2. The Cornell box scene split into a number of *Radiance* shader-specific components: (top-left) direct, (top-middle) indirect diffuse, (top-right) pure specular, (bottom-left) specular for mirror shader, (bottom-middle) transmitted for glass shader, (bottom-right) reflected for glass shader (including transmitted) and (right) the full solution.

advantage of to obtain orders of magnitude performance over traditional distributed ray-tracing methods [7]. Similarly, specular reflections exhibit high spatial frequencies, meaning that the specular contribution to the radiance of adjacent pixels hitting the same material may be substantially different even if the direct, glossy and indirect diffuse components of the material are very similar. By taking advantage of this insight we present a novel adaptive sampling algorithm that bases its sampling criteria not on the radiance of an individual pixel but on the individual contributions of each component to the final radiance.

In this paper we present a framework for incorporating such an adaptive sampling scheme within a traditional ray-tracing renderer. To illustrate our component-based adaptive sampling we have extended the light simulation package *Radiance* [16] to support our approach as well as the more standard approach. We use a number of *Radiance* scenes to demonstrate that the component-based adaptive sampling is superior to the more traditional approach.

This paper is divided as follows. Section 2 presents related work. Section 3 introduces our novel algorithm. Section 4 presents our implementation of the algorithm in *Radiance*. Section 5 presents results using a number of scenes and evaluates our work when compared the traditional method using a visible difference predictor. Finally Section 6 concludes the paper and presents possible future directions.

2. Related Work

In this section we present an overview of the work in image sampling for ray tracing and component-based rendering.

2.1. Image Space Sampling

Methods of sampling to diminish aliasing and improve performance have been around since the earliest ray tracers. Whitted [31] used adaptive sampling by shooting rays at the corners of the pixels. If the results where within a given value, the value of the pixel was averaged from the given samples. Otherwise, another ray was shot in the centre of the pixel and the calculation was continued recursively. Cook [6] presented the concept of stochastic sampling as opposed to uniform sampling for improving antialiasing. Further work on non-uniform sampling was presented by Mitchell [17]. Painter and Sloan [20] presented a progressive approach to traditional ray tracing refining samples when necessary. Guo [13] constructed a coherence map from sparse uniform samples and used it identify discontinuities on the image plane in his progressive renderer. An overview of uniform and non-uniform sampling is provided in [11].

Recently sampling schemes have been developed in conjunction with the human visual system. Myszkowski [18] and Bolin and Meyer [1] used visible difference predictors, such as [8] to direct the next set of samples. Cater *et al.* [3] used the concept of the affinity of eye movements to the vi-

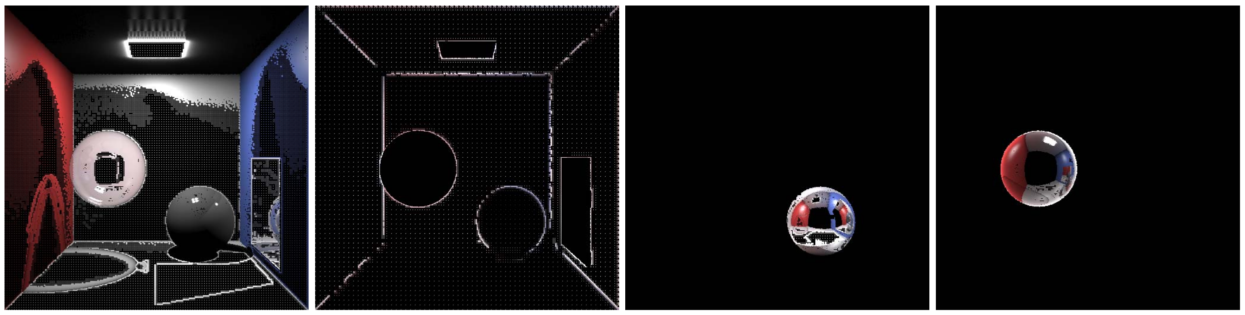

Figure 3. Samples per component without interpolation: (a) direct, (b) indirect diffuse, (c) glossy and (d) specular component of the transparency. Low resolution is used to make samples more visible.

sual task being performed [32] to direct the samples of their pixels for task based applications. O'Sullivan *et al.* [19] presents an overview of recent work in perceptually adaptive graphics.

2.2. Component-based rendering

Rendering has been divided into components on a number of occasions in order to solve the problem more efficiently. Wallace *et al.*'s [27] multipass algorithm computed the diffuse component with a rendering pass and used a z-buffer algorithm for view dependent planar reflections. Ward *et al.*'s irradiance cache [30] decoupled the computation of the indirect diffuse from the rest of the rendering equation for their distributed ray tracer. Calculated indirect diffuse samples would then be stored in an irradiance cache and future indirect requests could interpolate the results from the previous samples. Since the indirect diffuse samples are usually the most expensive part of the equation, this resulted in an order of magnitude improvement in performance over traditional distributed ray tracing. Sillion and Puech [22] adapted a technique proposed by Wallace *et al.* [27], using ray tracing for computing the specular component and the form factors of the non-planar objects, enabling multiple specular reflections. Shirley's [21] used a three pass method for varying components, path tracing from the light source was used for caustics, soft indirect illumination was obtained through radiosity and stochastic ray tracing completed the rest of the components. Other multipass algorithms that calculated components separately include Heckbert's [14] and Chen *et al.*'s progressive multipass method [5]. Slussalek *et al.* [23] presented a rendering framework for combining different algorithms together into lighting networks. Recently the component-based approach has been tied in with perceptual rendering by Stokes *et al.* [24]. They presented a perceptual metric which predicted the importance of the components for a given scene. Stokes *et al.* proposed that the perceptual metric could be used to drive a path tracing renderer, where the primary rays collected information about the scene and then used the perceptual metric to allocate the individual component calculations to resources based on their importance. The final image was then composited from the distinct components. They also indicated the possibility of allocating components to separate resources. Debattista *et al.* [9] presented a selective component-based rendering framework whereby a regular expression, termed the *crex* controls the rendering process. The *crex* could further be used in conjunction with an importance map [26] to exploit visual attention and also be applied to render images within given time constraints.

3. Component-based Adaptive Sampling

3.1. Rendering by Components

The radiance at a pixel (x, y) in direction $-\Theta$ which intersects an object in the scene at point p is given by the rendering equation [15]:

$$L(x,y) = L(p \to \Theta) =$$

$$L_e(p \to \Theta) + \int_{\Omega_p} f_r(p, \Theta \leftrightarrow \Psi)\cos(N_p, \Psi)L(p \leftarrow \Psi)\delta_{w_\Psi}$$

For convenience we set

$$L_i(p \to \Theta) = \int_{\Omega_p} f_r(p, \Theta \leftrightarrow \Psi_i)\cos(N_p, \Psi_i)L(p \leftarrow \Psi_i)\delta_{w_\Psi}$$

where Ψ_i refers to a specific direction.

Traditionally it is common to subdivide the computation into direct and indirect computations, using Ψ_d to refer to the direction of the direct contribution of the light and Ψ_{id} for the indirect contribution:

$$L(p \to \Theta) = L_e(p \to \Theta) + L_d(p \to \Theta) + L_{id}(p \to \Theta)$$

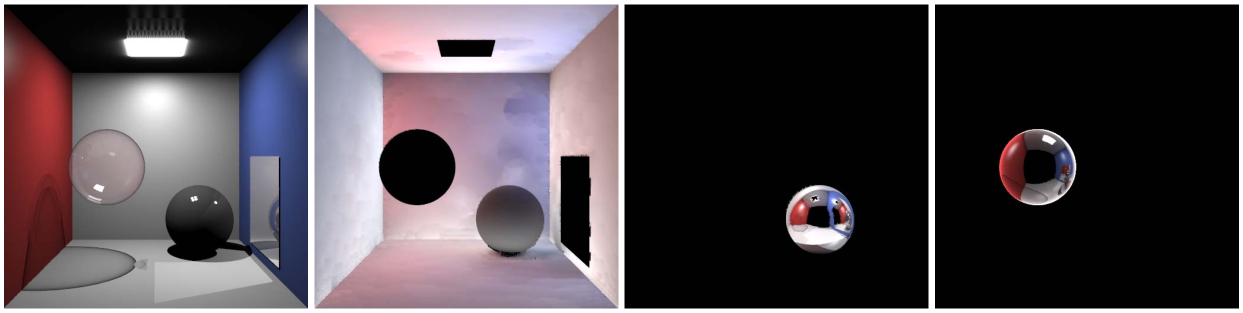

Figure 4. Interpolated components: (a) direct, (b) indirect diffuse (without material contribution), (c) glossy and (d) specular component of the transparency.

Furthermore, the indirect computation can be further divided into components. The more traditional components used are indirect indirect diffuse or ambient (a), indirect specular (is) and indirect glossy (ig):

$$L_{id}(p \to \Theta) = L_a(p \to \Theta) + L_{is}(p \to \Theta) + L_{ig}(p \to \Theta)$$

This can be abstracted further to account for shaders with more components. Assuming N_c components:

$$L_{id}(p \to \Theta) = \sum_{c}^{N_c} L_c(p \to \Theta)$$

Finally,

$$L(p \to \Theta) = L_e(p \to \Theta) + L_d(p \to \Theta) + \sum_{c}^{N_c} L_c(p \to \Theta)$$

This only means that we have split the calculation into the direct and the indirect components for the first intersection from the virtual camera. The process can be taken further and be done recursively. However, since our algorithm is based on the image plane and for design simplicity, as shall be described below, we are only interested in the first intersection.

3.2. Motivation

As outlined previously the method adopted by our novel adaptive sampling algorithm relies on adaptively sampling the individual components. The motivation behind this lies in the different spatial variances of the components and the flexibility which arises in rendering them in separate passes. From Section 3.1 it is clear that it is possible to calculate the radiance at the image plane for each of the components individually and composite them into a final result. We use this knowledge in our framework to be able to render individual components using different adaptive sampling thresholds (see Figure 3), since certain components (for example indirect diffuse) are less likely to have highlights, and finally composite the results into a single individual plane. Furthermore, we only break components up after the first intersection with an object. The system can be extended to handle component-based rendering for successive intersections, but for our purposes this complicates the rendering system unnecessary. Our philosophy is that any renderer can have enhanced speedup by a simple modification of the shaders. Furthermore, all of this can be done at a system level and is completely transparent to the user.

3.3. Framework

The algorithm we use is a modification of the classic raytracing algorithm and can be applied to any ray tracer. Furthermore, in the description of this algorithm we assume a very simple adaptive sampling scheme whereby rays are shot at the corners of a square and if the resultant calculated radiance of the rays differs by some threshold, the square is subdivided recursively up to a user-defined depth [31]. The only additional data structures that are strictly required are separate image buffers for each of the components that may be broken, see Figure 4, and a data storage for the primary rays that hit objects that have component-based materials. We will term these structures, the component image buffers and the primary ray structure. These structures are equal in size to the maximum number of samples that can be shot. The memory requirements of these structures on modern systems is negligible. The material types (or shaders) that can have components that are going to be rendered separately need to be identified. This is a system design decision and does in no way effect the user. It is added to our algorithm design since certain shaders such as a mirror shader in *Radiance* always perform a pure specular reflection and thus it is computed immediately. We term shaders that can be broken down into components as *breakable*. A description of our choice of breakable shaders for our implementation in *Radiance* will be discussed in Section 4.

3.4. Algorithm

The algorithm can be thought of as consisting of two passes. The first pass corresponds to the calculation of the radiance contribution by the direct light and the radiance contribution from the non-breakable components. The second pass corresponds to the calculation of the individual components. Initially the first pass rays are traced in the traditional method with the sampling method outlined above. When a ray initially hits an object, primarily the direct lighting is calculated and stored on the direct component image buffer. Whereas using a traditional method each ray is traced to completion, with our method when the ray first intersects an object, if the object's material properties contain a breakable shader, a flag is set inside the primary ray structure tagging that particular pixel for calculating the component, or components if more than one, for future use. Non-breakable shader computations are performed the traditional way. After each of the four corners have been calculated, the adaptive sampling criteria is consulted and if necessary further adaptive sampling is performed and the operations above are repeated for each of the direct rays. If the criteria were satisfied the underlying pixels are interpolated. When the first part terminates the direct lighting and some parts of the indirect lighting (for the non-breakable components) would have been completed and stored in the direct component image buffer.

The second phase of the algorithm begins by identifying which of the components need to be sampled. This is performed for each of the breakable components. The samples to be calculated are identified by consulting the primary ray structure. The samples use the information from the primary rays stored in the primary ray structure to launch the component ray. If the primary ray data is missing due to the component requiring finer grain sampling than the direct rays, the information can either be interpolated if all the information is available or a primary ray calculated at that stage in the algorithm. The component ray is fully recursive and computes until it terminates in the traditional ray tracing method as implemented by the underlying renderer. The resulting radiance values are stored in the component buffer for that specific component. Adaptive sampling proceeds in the traditional fashion. When the algorithm has iterated through all components, the component buffers are composited to produce the final image buffer.

4. Implementation

In this section we outline our own implementation of the algorithm described in the previous section, and show how we modified an existing renderer to do so. We will describe the decisions we took for choosing the breakable shaders and use the implementation for obtaining results in Section 5.

Our component-based adaptive sampling algorithm has been incorporated into the `rpict` renderer for still images and animations of the lighting simulation package *Radiance* [16]. *Radiance* is based on distributed ray tracing and also implements a number of well documented algorithms such as the irradiance cache [28, 30], adaptive shadow testing [28] and shaders [29]. The irradiance cache, in particular, is of interest to us since it caches indirect diffuse samples in object space, which are used for extrapolation by other indirect diffuse requests similar to our method but in object space. The irradiance cache can be further used in animations since the indirect diffuse component is view independent and does not vary for static scenes. Our component-based adaptive sampling can sit on top of an algorithm using an irradiance cache effectively providing a two-tier image space and object space interpolation method, with improved results.

4.1. Traditional Implementation

We have modified `rpict` in two further guises to benchmark the performance of our new implementation. The simplest implementation just removes the quincunx [2] sampling scheme traditionally used by *Radiance* and replaces it with a more traditional stratification scheme which renders one ray for each entry of a stratified grid. The resolution of the images is computed as a secondary pass using a separate filtering application which can downsample the calculated image. Effectively the whole process could be viewed as uniformed jittered stratified supersampling. This implementation is performed to simplify the sampling process and implementation of the other renderers, since it is the most traditional form of sampling. There is no reason apart from simplicity as to why quincunx sampling could not have also been adapted for component-based adaptive sampling. We have called this new renderer `tpict`.

The second traditional renderer is a modification of `rpict` to perform adaptive sampling by shooting pixels at the corners of the stratified structure at user-defined intervals [31] as described in Section 2. We term this renderer `apict`. Figure 1 was computed using this renderer.

4.2. Component-based Implementation

Our implementation of the component-based adaptive sampling algorithm was applied as an extension to `rpict` we term `capict`. The `capict` sampling scheme at its basic is similar to that of `apict`. However, this renderer follows the implementation of the algorithm outlined in Section 3.4. The breakable shaders chosen for this implemen-

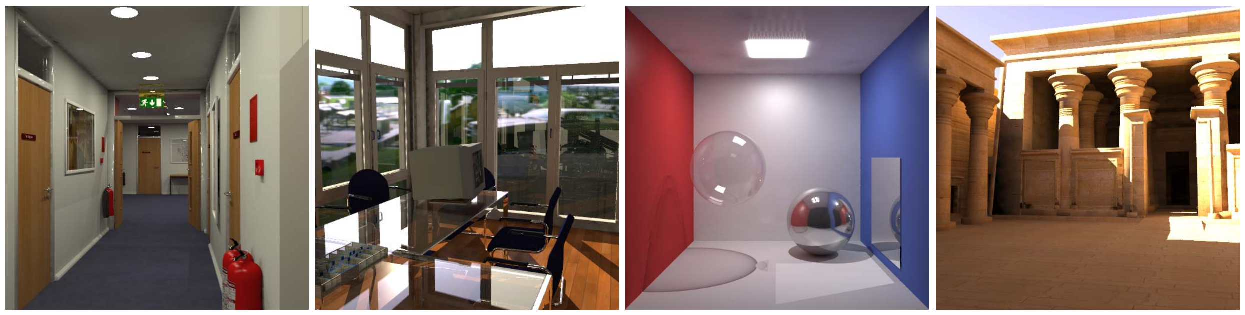

Figure 5. Views of the scenes used for results: (a) the corridor, (b) the library, (c) the Cornell box and (d) the Temple of Kalabsha [25].

tation, were the isotropic shaders [29] that contain indirect diffuse, indirect specular, indirect glossy for both reflected and transmitted materials. Each of the components in these shaders are breakable. Furthermore, the pure transparent component glass is breakable into only the reflected component. The transmitted component is computed as part of the direct computation as a non-breakable computation. This method is chosen since no direct computation is performed, so breaking the component into both reflected and transmitted would be useless. Furthermore, for similar reasons, pure specular materials as implemented by the mirror shader in *Radiance* are also bundled with the direct computation. These shaders satisfied the requirement of our test scenes so no further shader adjustments were required. However, further shader modifications would be straightforward to add. The renderings of the breakable shader modifications are illustrated in Figure 4.

5. Results and Verification

In order to demonstrate our approach we present results using four distinct scenes, see Figure 5. The scenes chosen represent a variety of different realistic and practical scenes. The corridor scene (a) represents an indoor environment with artificial lighting, the library scene (b) is an indoor scene with sun light. Our version of the Cornell box (c) is chosen for practicality and the Kalabsha scene (d) is a reconstruction of the Temple of Kalabsha [25] in Egypt . The results were computed with default *Radiance* parameters except for the resolution and indirect diffuse bounces. The four scenes were rendered with a resolution of 512×512, with a maximum of nine rays per pixel effectively rendering a stratified sampling grid of 1536×1536 and filtered using the standard *Radiance* gaussian filter. The indoor scenes were rendered with one indirect diffuse bounce and the Kalabsha scene and Cornell box with two indirect bounces. All these images were rendered with an irradiance cache. Furthermore, we present results for the Cornell box without irradiance cache computations rendered at 512×512 with a maximum of four rays per pixel and one indirect diffuse bounce. Lower values were chosen for this benchmark due to the prohibitive time it takes to render without adaptive sampling. All results were computed on a Intel Pentium Xeon CPU 2.40GHz with 3GB RAM under Linux.

5.1. Performance Results

The results are presented in Table 1. It is clear from the speedup that `capict` outperforms both `apict` and `tpict` always obtaining close to 50% speedup and more over the traditional method. The slowest improvement is for the library scene and is due to the glass on the tables being both at the top and bottom of the tables and this weakens the component-based approach for those areas of the image. Yet the results are still an improvement over the traditional adaptive approach. Of particular interest is the result of the Cornell box without the use of an irradiance cache. The result demonstrates an order of magnitude improvement in performance over the standard technique signifying that the novel rendering method can be used as an easy to implement substitute to the irradiance cache for systems that do not support irradiance caching. It also suggests that the component-based adaptive sampling algorithm is useful for other global illumination algorithms such as path tracing where the integration of an irradiance cache is non-trivial.

5.2. Verification

We provide verification of the results using the Visible Differences Predictor [8], a metric that creates a grey scale image of the perceptual differences between two images, highlighting only differences that are visible by a human observer. Since the resultant images between any two of our images is mostly blank we summarise the results as the average pixel error between the two adaptive sampling ren-

	corridor		library		Cornell box		Cornell box (no IC)		Kalabsha	
Renderer	Time	Speedup	Time	Speedup	Time	Speedup	Time[‡]	Speedup	Time	Speedup
`tpict`	2,340	1	2,700	1	313	1	238	1	901	1
`apict`	1,687	1.39	2,085	1.29	199	1.57	61	3.9	870	1.04
`capict`	1,107	2.11	1,900	1.42	159	1.96	22	10.81	610	1.48

Table 1. Results for the various renderers. Time in seconds. [‡] Time in minutes.

Renderer	corridor	library	Cornell box	Cornell box (no IC)	Kalabsha
`apict`	0.2%	1.3%	0.2%	0.1%	1.3%
`capict`	0.2%	0.5%	0.2%	0.08%	1.3%

Table 2. Visual Differences Predictor [8] results of the adaptive sampling renderers compared the traditional approach.

derers and `tpict` in Table 2. The results demonstrate that there is practically no percentage difference between any of the adaptive renderers and the traditional approach. The small percentage difference is probably due to the stochastic nature of our rendering algorithm. In fact when comparing two distinct images rendered with the same conditions under `tpict` the error is around 0.2% similar to that of the adaptive renderers.

6. Conclusions and Future Work

We have presented a novel adaptive sampling algorithm that uses a component-based approach to speed up rendering times without any perceivable differences in the resultant images. Our new algorithm is transparent to a user and can be included into existing ray traced renderers through a modification of the required shaders. The results also show that, as expected, the component-based adaptive sampling algorithm can be used in conjunction with an irradiance caching scheme providing a two-tier interpolation mechanism and can also achieve an order of magnitude performance increase when using strict distributed ray tracing only. While we have presented the algorithm in terms of one of the simplest adaptive sampling schemes, the algorithm could potentially be used with other more complex adaptive sampling schemes [2, 13]. Since it relies on ray tracing and samples at the image plane, the component-based sampling approach is also trivial to parallelise using an image tiling demand driven approach [4].

Future work will investigate the usefulness of this algorithm for other global illumination algorithms in particular path tracing [15]. Furthermore, the use of this algorithm in conjunction with perceptually adaptable renderers [19] and techniques for visual masking [10] will be investigated.

7 Acknowledgements

We would like to thank Veronica Sundstedt for the Temple of Kalabsha model and the corridor scene and Peter Apian-Bennewitz and Stefan Jankowski for the library scene. We would also like to thank Greg Ward and Luis Paulo dos Santos for technical discussions. The work reported in this paper has formed part of the Rendering on Demand (RoD) project within the 3C Research programme whose funding and support is gratefully acknowledged.

References

[1] M. R. Bolin and G. W. Meyer. A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 299–309. ACM Press, 1998.

[2] C. Bouville, P. Tellier, and K. Bouatouch. Low sampling densities using a psychovisual approach. In *EUROGRAPHICS 1991*, 1991.

[3] K. Cater, A. Chalmers, and G. Ward. Detail to Attention: Exploiting Visual Tasks for Selective Rendering. In *Proceedings of the Eurographics Symposium on Rendering*, pages 270–280, 2003.

[4] A. Chalmers, T. Davis, and E. Reinhard. AK Peters Ltd, July 2002.

[5] S. E. Chen, H. E. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 165–174. ACM Press, 1991.

[6] R. L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5(1):51–72, 1986.

[7] R. L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145. ACM Press, 1984.

[8] S. Daly. The visible differences predictor: an algorithm for the assessment of image fidelity. pages 179–206, 1993.

[9] K. Debattista, V. Sundstedt, L. P. dos Santos, and A. Chalmers. Selective rendering by components. Technical Report CSTR-05-002, University of Bristol, January 2005.

[10] J. A. Ferwerda, P. Shirley, S. N. Pattanaik, and D. P. Greenberg. A model of visual masking for computer graphics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 143–152, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[11] A. S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[12] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222, New York, NY, USA, 1984. ACM Press.

[13] B. Guo. Progressive radiance evaluation using directional coherence maps. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 255–266, New York, NY, USA, 1998. ACM Press.

[14] P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 145–154. ACM Press, 1990.

[15] J. T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150. ACM Press, 1986.

[16] G. W. Larson and R. Shakespeare. *Rendering with radiance: the art and science of lighting visualization*. Morgan Kaufmann Publishers Inc., 1998.

[17] D. P. Mitchell. Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 65–72. ACM Press, 1987.

[18] K. Myszkowski. The Visible Differences Predictor: Applications to global illumination problems. In *Eurographics Workshop on Rendering*, pages 223–236, 1998.

[19] C. O'Sullivan, S. Howlett, R. McDonnell, Y. Morvan, and K. O'Conor. Perceptually adaptive graphics. In *Eurographics State of the Art Reports*, 2004.

[20] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 281–288. ACM Press, 1989.

[21] P. Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–12, Toronto, Ontario, 1990. Canadian Information Processing Society.

[22] F. Sillion and C. Puech. A general two-pass method integrating specular and diffuse reflection. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 335–344. ACM Press, 1989.

[23] P. Slusallek, M. Stamminger, W. Heidrich, J.-C. Popp, and H.-P. Seidel. Composite lighting simulations with lighting network. *IEEE Computer Graphics and Applications*, 18(2):22–31, /1998.

[24] W. A. Stokes, J. A. Ferwerda, B. Walter, and D. P. Greenberg. Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. *ACM Trans. Graph.*, 23(3):742–749, 2004.

[25] V. Sundstedt, A. Chalmers, and P. Martinez. High fidelity reconstruction of the ancient egyptian temple of kalabsha. In *AFRIGRAPH 2004*. ACM SIGGRAPH, November 2004.

[26] V. Sundstedt, K. Debattista, P. Longhurst, A. Chalmers, and T. Troscianko. Visual attention for efficient high-fidelity graphics. In *Spring Conference on Computer Graphics (SCCG 2005)*, May 2005.

[27] J. R. Wallace, M. F. Cohen, and D. P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 311–320. ACM Press, 1987.

[28] G. J. Ward. Adaptive shadow testing for ray tracing. pages 11–20.

[29] G. J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, New York, NY, USA, 1992. ACM Press.

[30] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92. ACM Press, 1988.

[31] T. Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980.

[32] A. Yarbus. Eye movements during perception of complex objects. In *Eye Movements and Vision*, pages 171–196, 1967.