ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 2901
WSCG Proceedings Part I

Dynamic Radiosity

Alexandr Shcherbakov
Lomonosov Moscow State University
GSP-1, Leninskie Gory
119991, Moscow, Russia
alex.shcherbakov@graphics.cs.msu.ru

Frolov Vladimir
Lomonosov Moscow State University
Keldysh Institute of Applied Mathematics
(Russian Academy of Sciences)
GSP-1, Leninskie Gory
119991, Moscow, Russia
vfrolov@graphics.cs.msu.ru

ABSTRACT

In this paper we propose a novel radiosity implementation which we called *dynamic radiosity*. By storing and updating local form-factor matrix of the patches closest to the observer we have solved 2 main problems of radiosity algorithm: (1) quadratic complexity of the algorithm and thus difficulty of applying it to a large-scale scenes, and (2) the possibility of changing geometry on the fly (dynamic geometry).

Keywords

Radiosity for large scenes, dynamic geometry.

1 INTRODUCTION

Global Illumination (GI) is a fundamentally difficult computational problem. Thus, real-time applications usually apply approximate algorithms to solve it. Such algorithms can be divided into two main classes: (1) fully dynamic and (2) precomputed radiance transfer. Dynamic GI algorithms allow changing scene (both geometry and materials) each frame. Precomputed Radiance Transfer (PRT) usually immobilizes geometry or both materials and geometry of scene and moves the most complicated computations to *precomputation stage*. PRT gets the best balance of speed and accuracy because of that. The proposed algorithm lies somewhere in between of these two classes.

2 RELATED WORK

2.1 Dynamic GI

Reflective Shadow Maps [1] is a popular solution for interactive global illumination. RSM is a variation of the Instant Radiosity [2] method — global illumination via virtual light sources. The main disadvantage of this approach is the low accuracy of the resulting solution; its advantage is high speed. Although similar methods (called *many lights*) have been developed [3], with the exception of RSM, they are not often used in real-time applications due to bad balance of speed/quality. At high speed they provide low accuracy, and at high quality, they are too far from real time applications [4, 5].

Voxel Cone Tracing [6] uses pre-integrated lighting and Final Gathering via tracing cones. The method is pretty accurate, however, it still suffers of artifacts and it is expensive in terms of computing due to several cones should be traced for each pixel of image.

Light Propagation Volumes [7] numerically solves a differential equation on a three-dimensional grid. The main disadvantage is low accuracy and high memory costs for a regular grid. Cascaded LPV [8] amortizes this cost but does not solve the problems of the method.

With the advent of Nvidia RTX technology [9], Path Tracing in combination with denoising has become a popular method [29, 30, 31, 32]. It is one of the most computationally intensive solutions, however.

Modern dynamic GI approaches compute lighting only around the observer [34]. It allows to apply these methods for a large scenes.

2.2 PRT

Among PRT methods, three main classes should be distinguished: methods based on spherical harmonics [10], radiosity [20], and neural network methods [22].

There are a lot of methods using spherical harmonics: [10, 11, 12, 13, 14, 15]. Their basic idea comes from the Radiance Caching [17]. The harmonics themselves are just a compact storage of the incident light via decomposition of the incident light in the basis of orthogonal functions. Next by analogy with relighting [18], these methods allow replacing the environment and evaluate the lighting dynamically by adding contribution from

the harmonics in the same way that relighting adds contribution from different sources, summing up individual images. These methods are well-suited for outdoor scenes, but fail for indoor ones, since harmonics believe that light comes to a point from infinitely distant objects. In the interiors, this rule is often broken, which leads to various types of artifacts: leakage, incorrect lighting and others [15]. With a proper effort, however, one can achieve a good result for interiors as well [19].

Precomputed Radiance Transfer via neural networks is a relatively new area of research. Authors of [22] show high accuracy in real-time and compact memory representation. The shortcomings of the method include a long pre-calculation and a complete static nature of the scene — neither geometry nor materials could be changed at all (in harmonic or radiosity based methods, for example, these limitations can be relaxed).

2.3 Radiosity

The radiosity method is a physically based approximation for diffuse global illumination [21]. Despite its venerable age, radiosity is widely used in real-time applications [23, 16] and lighting engineering [24, 25].

We chose radiosity because of its physical correctness, relatively high accuracy and speed when small number of patches are used. In addition, ones to be used on practice and well-suited for GPU and hardware implementations since it can be expressed as a single matrix-vector multiplication [26]. On the other hand, the disadvantages of the radiosity are: (1) the dependence on memory and computation as $O(N^2)$ where N is a number of patches and (2) the static geometry of the scene (in this case, unlike the method based on neural networks, materials can be easily replaced).

Hierarchical radiosity [27] subdivides polygons of the original scene into smaller patches until the form-factors for them can be computed with acceptable accuracy (the error is less than a specified threshold), or the maximum depth of the splitting is reached. Thus, a hierarchy of patches is obtained. In the calculation, part of the light is not transferred between the detailed patches, but between the top patches in the hierarchy, which reduces the amount of computation. For some scenes, a very detailed split may be required for good computational accuracy. This method can no longer be expressed in terms of lineal algebra operations and thus it is more complicated than original radiosity for specific GPU or hardware implementations.

Progressive radiosity [28] is a method for gradual calculating of the illumination. Initially, all patches are assigned the same illumination. Gradually, for individual patches, the correct illumination is calculated based on form-factors. The main disadvantage of this method is the difficulty of choosing of criterion for the next lit patch. In this method, each rotation or shift of the camera leads to a complete lighting recalculation. This is a serious disadvantage in comparison with original radiosity that is completely independent from camera parameters.

It is a well known that the radiosity problem can be expressed as a linear equation system problem. By using well known LU (low-upper) decomposition this problem can be solved in $O(N*N)$ operations where N is a number of patches. In [26] more straightforward and GPU friendly algorithm was presented. It was shown that single matrix-vector multiplication can be used to solve multi-bounce radiosity equation.

3 SUGGESTED APPROACH

We suggest a new method for calculating global illumination based on the radiosity method, which we called *dynamic radiosity*. The new method allows to effectively compute the lighting around the observer (Fig. 1), recalculating the local matrix of form-factors. Due to this, we can reduce $O(N^2)$ complexity to $O(M^2)$ (where $M < N$) and add dynamic objects to the scene. We further extend our method to local multibounce matrix [26] which is important contribution of our paper. Following to the approach in [26], we suggest to use the radiosity method for indirect illumination.

Our basic idea is as follows: at first, we restricted area of interest with a subset of patches close to observer (Fig. 1, left). We do not update patches that lie out of this area. Next, we have developed the method for reusing calculations made on previous frame if we know that observer moves only slightly (Fig. 1, right). Thus, we can perform only a small number of calculations every frame, effectively using the information that we have accumulated in time.

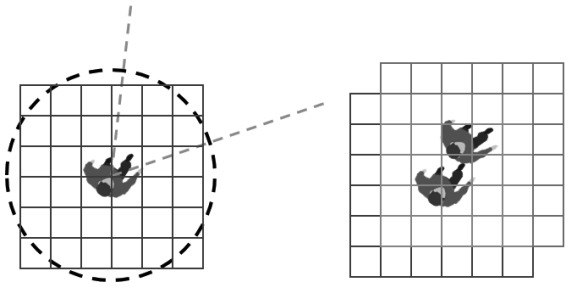


Figure 1: Concept of our method. Red patches (right part of the image) illustrate new patches that we should insert into local matrix. Green patches can be reused.

To start, assume using original radiosity in which we simply have $O(N^2)$ form-factor matrix. Select a subset of M patches and then make M^2 submatrix of form-factors (Fig. 2). We could directly evaluate several radiosity bounces using this M^2 form-factor submatrix,

 ISBN 978-80-86943-37-4

but this approach has several disadvantages. First, it requires $B * M^2$ operations per each frame where B is the number of light bounces. Second, this idea has poor utilization of modern computing architectures (both CPU and GPU) because it does not combine computations with memory operations when making local submatrix and does not use the principle of data locality. Due to that we used different approach and worked with multibounce matrix instead of simple form-factor matrix.

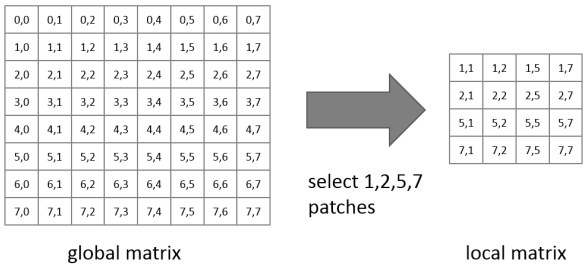


Figure 2: local matrix

The local matrix F_{local} is a multibounce matrix (by analogy with [26]), containing information only about the transfer of light for patches around the observer. The lighting is recalculated when the emission of the patches changes or the position of the observer changes. When the observer's position changes the area of interest also changes.

Light — vector of emission of patches.

Colors — vector of colors of patches.

Idx is a vector of length M, where M is a count of patches in F_{local}.

$Idx_i = k$ where k is index of patch in full form-factors matrix F and i is index of the same patch in local matrix.

$Colors^{local}$ — vector of colors for local patches. $Colors^{local}_i = Colors_{Idx[i]}$

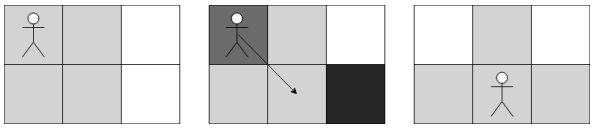


Figure 3: Green patches are included into local matrix (the nearest one to observer). Red patch is farthest out of patches in local matrix. Blue patch is the nearest of excluded patches from a local matrix.

To update the local matrix, the following algorithm is applied within each frame:

1. Select the farthest of the patches **involved** in the local matrix. Let it be indexed as i.
2. Select the patch which is the closest to the observer and **not included** into the local matrix. Let it be indexed as j.
3. If patch j is closer to the observer than patch i, then the matrix is updated, otherwise the update is not required (Fig. 3).
4. In the matrix, the column and row corresponding to patch i are cleared.
5. Calculate the row and column for the j-th patch.
6. The remaining data in the matrix is updated, taking into account the light reflected by patch j.

3.1 Form-factors composition

This method uses the idea of form-factors composition. For patches i, j, k, the value of $F_{ik} * Colors_k * F_{kj}$ is the amount of lighting that excidents from patch j to patch i through patch k.

Thus, the lighting transferred from patch j to patch i through other patches will be equal to:

$$\sum_{k=0}^{N} F_{ik} * Colors_k * F_{kj}.$$

Then, to transfer the lighting from patch i to patch j, taking into account one reflection, it is equal to:

$$\sum_{k=0}^{N} F_{ik} * Colors_k * F_{kj} + F_{ij}.$$

Similarly, it is possible to calculate form-factors that take into account the transfer of light between two patches, taking into account an arbitrary number of reflections from intermediate patches.

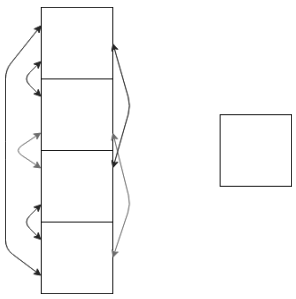


Figure 4: Patches in the local matrix with computed light transfer and the new patch.

3.2 Adding a new patch to local matrix

The process of adding a new patch i (Fig. 4) consists of two parts:

1. Calculation of form-factors for the new patch (just take them from global form factor matrix if geometry is static).
2. Update of the remaining values of the matrix.

ISBN 978-80-86943-37-4

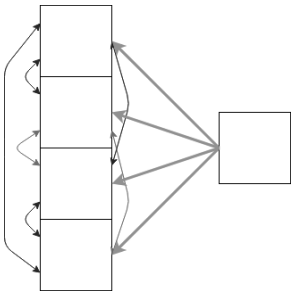


Figure 5: f_c – form-factors from i-th patch to other. Orange arrows are the directions corresponding to outcome form-factors for a new patch.

f_c is a form-factors *column* for the new patch. It contains information about the transfer of light from the i-th patch to the rest (Fig. 5).

f_r is a form-factors *line* for the new patch. It contains values showing which part of lighting is transferred from other patches to the i-th one. (Fig. 6)

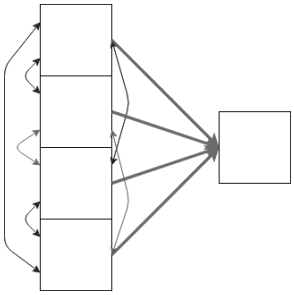


Figure 6: f_r – form-factors from all patches to i-th patch. Red arrows are the directions corresponding to income form-factors for new patch.

For the new patch, we compute the form-factor for lighting emitted by i-th patch to the rest and reflected by them to i-th (Fig. 7).

$$double\_reflection = \sum_{j=0}^{M} f_c[Idx[j]] * f_r[Idx[j]] * Colors_j^{local}. \quad (1)$$

M is the number of patches in the local matrix.

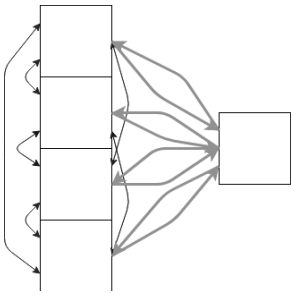


Figure 7: *double_reflection* form-factor for the new patch. Orange arrows shows direction of double light bounce.

double_reflection is taken into account in the vectors g_c and g_r. g_c is a column that takes into account the lighting, moving from the new patch to others, including three light reflections (Fig. 8). g_r is row, which takes into account the lighting that passes to the new patch, with three light reflections too (Fig. 9).

$$g_c = f_c + Colors_i * double\_reflection$$
$$g_r = f_r + Colors_i * double\_reflection$$

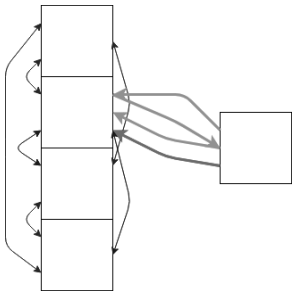


Figure 8: g_c vector of form-factors for excident lighting with information about three reflections. Orange arrows are the path of light in a triple bounce. Red arrow - light from single bounce (f_c).

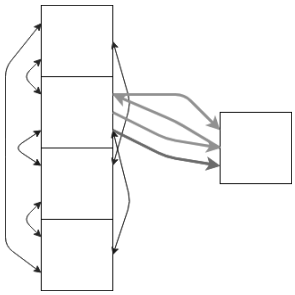


Figure 9: g_c vector of from-factors for incident lighting with information about three reflections. Orange arrows are the paths of light in triple bounce. Red arrow - light from single bounce (f_r).

Additionally, in g_c and g_r, we can add information about one reflection from the new patch and following re-reflection inside the local matrix (Fig. 10, 11):

$$g'_c = g_c + F_{local} \cdot (Colors^{local} \circ g_c)$$
$$g'_r = g_r + (Colors^{local} \circ g_r) \cdot F_{local}$$

g'_c and g'_r are the column and row in the local form-factor matrix for patch i, which take into account at least 3 reflections.

The local matrix of form-factors changes as follows:

$$F'_{local} = F_{local} + g'_r \cdot (Colors_i * g'_c).$$

This transformation adds reflections from the new patch to the matrix.

ISBN 978-80-86943-37-4

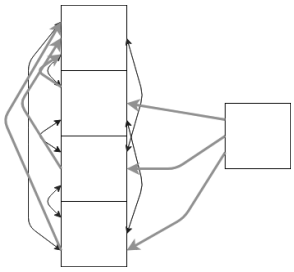


Figure 10: g'_c vector of from-factors for excident lighting from new patch through other patches in local matrix.

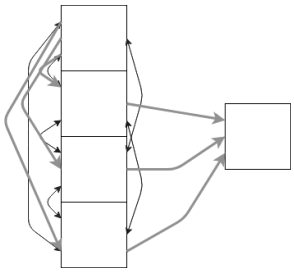


Figure 11: g'_r vector of from-factors for incident lighting from patches in local matrix through their inner reflections to new patch.

Supposed operations allow to recompute lighting in the area of interest, when the observer is moving. This approach can be applied for dynamic objects, but it needs to precalculate and store form-factors for each position of the object.

3.3 Lighting recalculation by local matrix

Lighting is recalculated only for the patches in a local matrix. The calculation itself is similar to the calculations for the multibounce matrix [26]: the local matrix is multiplied by the corresponding emission values of the patches in vector Light.

4 IMPLEMENTATION DETAILS

4.1 Eliminate duplicate reflections

If algorithm deletes a patch from the local matrix, and then adds it back, some patches in the local matrix will receive the reflection information from this patch again. To avoid such errors, we suggest storing a list of patches which lighting is already taken into account for each patch in local matrix. For this, the bitset container can be used. At the same time, it should be updated every time a new patch is added.

4.2 Form-factors streaming for large scenes

Traditional form-factors are used in new patch adding. For large scenes containing a huge amount of non-zero values it may be the problem to store all these numbers in memory.

Proposed method can be combined together with streaming technique for form-factors and load necessary values from HDD/SSD on demand.

5 EXPERIMENT RESULTS

The proposed method was compared with the naive radiosity method and path tracing. Comparisons were made for a scene with a ready-made patching (*rungholt house* from [33]), so the hierarchical radiosity is not applicable to it. The scene consists of 63125 patches. The multibounce matrix takes more than 44 GB for the scene. Such amount of data is not applicable in real-time applications. The local matrix needs less memory (96 MB for $M = 4096$). If fact the size of the memory required (and thus M) can be specified by the user.

5.1 Local multibounce matrix and form-factors sub-matrix

In the worst case our method needs the computation time equal to 3 bounces in naive radiosity for sub-matrix with dimensions $M \times M$. But multibounce can take into account more bounces. Moreover, multibounce matrix needs the same computation as usual form-factors matrix for a single reflection if observer doesn't move too far for matrix recompute.

5.2 Comparison and conclusion

For a local matrix, the lighting computation takes 400 times less time, and updating the local matrix takes 200 times less time than performing the naive radiosity algorithm (Fig. 15, 12). All tests were performed on Intel Core i7-7700HQ CPU in a single thread mode and gain real-time performance with same precision as original radiosity which is relatively close to the reference solution (path tracing, Fig. 12).

We have presented a concept of *Dynamic radiosity*. However, several problems were not considered in this paper and we believe this is our future research: (1) dynamic geometry will involve fast form-factor evaluation, which we didn't implement, (2) GPU implementation is our next area of interest and (3) effitient streaming of data from storages are not well studied yet. Finally (4), our algorithm does not account lighting from excluded patches at all. This can be fixed via cascades by analogy with Cascaded Light Propagation Volumes.

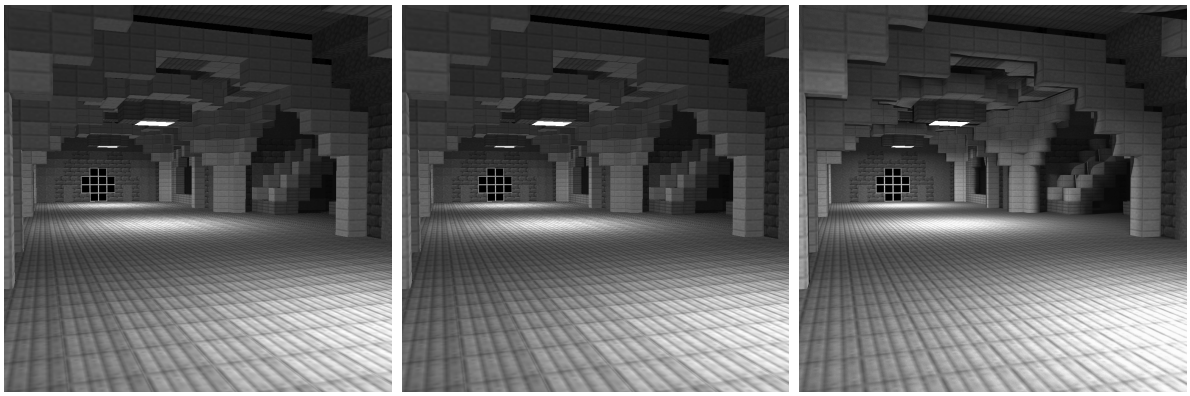
5.3 Acknowlegments

This work is sponsored by RFBR 18-31-20032.

6 REFERENCES

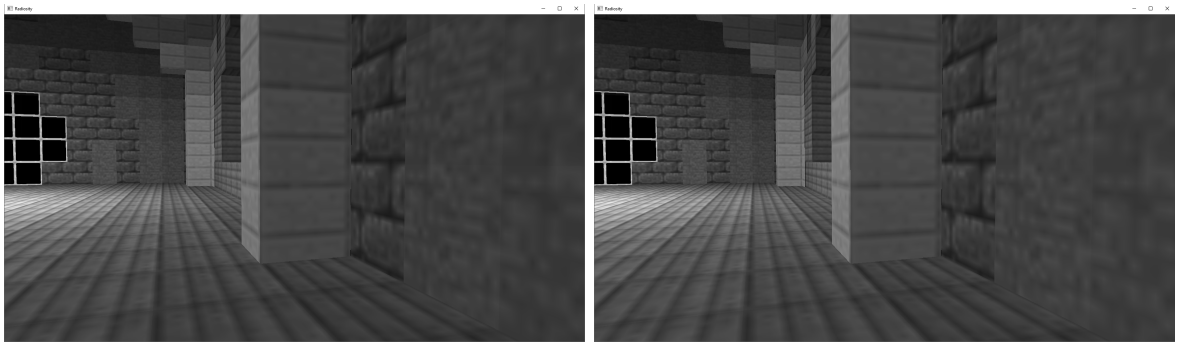
[1] Dachsbacher C., Stamminger M. *Reflective shadow maps* // Proceedings of the 2005 symposium on Interactive 3D graphics and games. — ACM, 2005. — Đ¡. 203-231.

 ISBN 978-80-86943-37-4



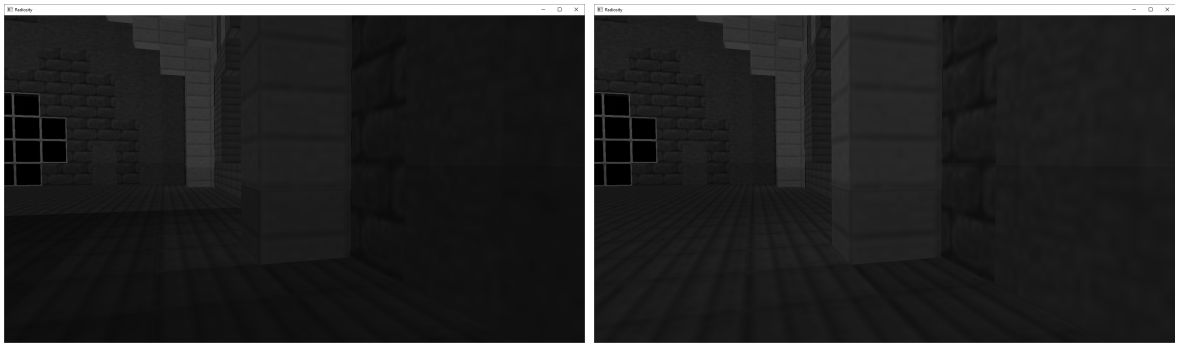
Our method 1024 patches (12 ms) Naive radiosity (1561 ms) Path Tracing (10 minutes)

Figure 12: Comparison of our method to naive radiosity and reference (path tracing).



Our method 1024 patches (12 ms) Naive radiosity (1561 ms)

Figure 13: Comparison of our method to naive radiosity. Patches around the observer.



Our method 1024 patches (12 ms) Naive radiosity (1561 ms)

Figure 14: Comparison of indirect lighting computed by our method and naive radiosity. Patches around the observer.

[2] Keller A. *Instant radiosity.* — 1997.

[3] Carsten Dachsbacher, Jaroslav Krivanek, Milos Hasan, Adam Arbree, Bruce Walter, and Jan Novak. 2014. *Scalable Realistic Rendering with Many-Light Methods.* Comput. Graph. Forum 33, 1 (February 2014), 88-104. DOI=http://dx.doi.org/10.1111/cgf.12256

[4] Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. (2005, July). *Lightcuts: a scalable approach to illumination.* In ACM Transactions on graphics (TOG) (Vol. 24, No. 3, pp. 1098–1107). ACM.

[5] Ou, J., and Pellacini, F. (2011). *LightSlice: matrix slice sampling for the many-lights problem.* ACM Trans. Graph., 30(6), 179-1.

[6] Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E. (2011, September). *Interactive indirect illumination using voxel cone tracing.* In Computer Graphics Forum (Vol. 30, No. 7, pp. 1921–1930). Oxford, UK: Blackwell Publishing Ltd.

[7] Kaplanyan A. *Light propagation volumes in cryengine 3* // ACM SIGGRAPH Courses. 2009. Vol. 7. p. 2.

 ISBN 978-80-86943-37-4

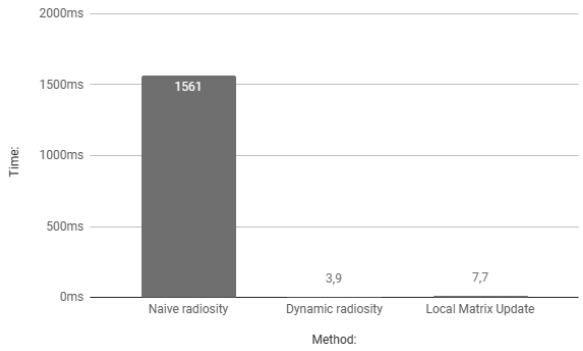


Figure 15: Detailing of computation time.

[8] Kaplanyan A., Dachsbacher C. *Cascaded light propagation volumes for real-time indirect illumination* // Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. — ACM, 2010. — pp. 99–107.

[9] KELLER A., MCGUIRE M., MUNKBERG J., PHARR M., SHIRLEY P., WALD I., WYMAN C. *Ray Tracing Gems. HIGH-QUALITY AND REAL-TIME RENDERING WITH DXR AND OTHER APIS.* . 2019. ISBN-13 (pbk): 978-1-4842-4426-5 ISBN-13 (electronic): 978-1-4842-4427-2 https://doi.org/10.1007/978-1-4842-4427-2.

[10] Green R. *Spherical harmonic lighting: The gritty details* //Archives of the Game Developers Conference. 2003. Vol. 56. p. 4.

[11] Papaioannou G. *Real-time diffuse global illumination using radiance hints* //Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics. ACM, 2011. pp. 15-24.

[12] Peter-Pike Sloan, Jan Kautz, and John Snyder. *Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments.* ACM Trans. Graph. 21, 3 (July 2002), 527-536. DOI: https://doi.org/10.1145/566654.566612

[13] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. *Clustered principal components for precomputed radiance transfer.* ACM Trans. Graph. 22, 3 (July 2003), 382-391. DOI: https://doi.org/10.1145/882262.882281

[14] Kautz, Jan, Jaakko Lehtinen, and Peter-Pike Sloan. *SIGGRAPH Precomputed Radiance Transfer: Theory and Practice course*. Aug. 2005.

[15] Akenine-Moller T., Haines E., Hoffman N. *Real-time rendering, 4th Edition*. AK Peters/CRC Press, 2018. pp 480 – 490.

[16] Akenine-Moller T., Haines E., Hoffman N. *Real-time rendering, 4th Edition*. AK Peters/CRC Press, 2018. pp 482.

[17] Krivanek J. et al. *Radiance caching for efficient global illumination computation* //IEEE Transactions on Visualization and Computer Graphics. 2005. Vol. 11. Number. 5. pp. 550–561.

[18] Dorsey, J., Arvo, J., Greenberg, D. *Interactive design of complex time dependent lighting. IEEE Computer Graphics and Applications.* 1995. 15(2), 26-36.

[19] McGuire, M., Mara, M., Nowrouzezahrai, D., Luebke, D. (2017, February). *Real-time global illumination using precomputed light field probes.* In Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (p. 2). ACM.

[20] Sillion F. X. et al. *Radiosity and global illumination.* San Francisco : Morgan Kaufmann, 1994. Vol. 1.

[21] Cohen M., GreenBurg D. *The Hemi-cube: A Radiosity solution for complex it is hard to achieve environments* // Proceedings of SIGGRAPH 85 in Computer Graphics, 1985. 19. number 3. pp. 31-40.

[22] Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. *Global illumination with radiance regression functions.* ACM Trans. Graph. 32, 4, Article 130 (July 2013), 12 pages. DOI: https://doi.org/10.1145/2461912.246200

[23] Sam Martin. *Enlighten real-time radiosity*. SIGGRAPH 2011.

[24] Mangkuto, R. A. *Validation of DIALux 4.12 and DIALux evo 4.1 against the Analytical Test Cases of CIE 171*. 2006. Leukos, 12(3), 139-150.

[25] Yu X., Su Y., Chen X. *Application of RELUX simulation to investigate energy saving potential from daylighting in a new educational building in UK* // Energy and Buildings. 2014. Vol. 74. pp. 191–202.

[26] Shcherbakov, A., and Frolov, V.*Accelerating radiosity on gpus.* In WSCG'2017 Full papers proceedings (2017), 2701, Computer Science Research Notes 2701 Pilsen, Czech Republic, pp. 99–105.

[27] Pat Hanrahan, David Salzman, and Larry Aupperle. *A rapid hierarchical radiosity algorithm.* SIGGRAPH Comput. Graph. 25, 4 (July 1991), 197-206.

[28] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. *A progressive refinement approach to fast radiosity image generation.* In Proceedings of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH '88), Richard J. Beach (Ed.). ACM, New York, NY, USA, 75-84. DOI:

https://doi.org/10.1145/54852.378487

[29] Martin Stich. *Real-time raytracing with NVIDIA RTX*. GAMESCOM, KOLN. 2018. URL = http://on-demand.gputechconf.com/gtc-eu/2018/pdf/e8527-real-time-ray-tracing-with-nvidia-rtx.pdf

[30] Michael Mara, Morgan McGuire, Benedikt Bitterli, and Wojciech Jarosz. *An efficient denoising algorithm for global illumination.* In Proceedings of High Performance Graphics (HPG '17). ACM, New York, NY, USA, Article 3, 7 pages. DOI: https://doi.org/10.1145/3105762.3105774

[31] A. M. Gruzdev, V. A. Frolov, and A. V. Ignatenko. 2015. *Practical approach to the fast Monte-Carlo ray-tracing.* Program. Comput. Softw. 41, 5 (September 2015), 253–257. DOI=http://dx.doi.org/10.1134/S0361768815050035

[32] D. D. Zhdanov, I. S. Potemin, V. A. Galaktionov, B. Kh. Barladyan, K. A. Vostryakov, and L. Z. Shapiro. *Spectral Ray Tracing in Problems of Photorealistic Imagery Construction* // Programming and Computer Software, Vol. 37, No. 5, 2011, pp. 236–244. DOI: 10.1134/S0361768811050069

[33] Morgan McGuire. *Computer Graphics Archive.* 2017. URL = https://casual-effects.com/data.

[34] A. Yudintsev. *Scalable Real-Time Global Illumination for Large Scenes.* 2019. URL = https://www.gdcvault.com/play/1026469/Scalable-Real-Time-Global-Illumination