High performance dosimetry calculations using adapted ray-tracing

Lancelot Perrotte, Guillaume Saupin
CEA LIST, route du panorama, 94260 Fontenay aux roses, France
E-mail: lancelot.perrotte@cea.fr, guillaume.saupin@cea.fr

Abstract. When preparing interventions on nuclear sites, it is interesting to study different scenarios, to identify the most appropriate one for the operator(s). Using virtual reality tools is a good way to simulate the potential scenarios. Thus, taking advantage of very efficient computation times can help the user studying different complex scenarios, by immediately evaluating the impact of any changes. In the field of radiation protection, people often use computation codes based on the straight line attenuation method with build-up factors. As for other approaches, geometrical computations (finding all the interactions between radiation rays and the scene objects) remain the bottleneck of the simulation. We present in this paper several optimizations used to speed up these geometrical computations, using innovative GPU ray-tracing algorithms. For instance, we manage to compute every intersection between 600 000 rays and a huge 3D industrial scene in a fraction of second. Moreover, our algorithm works the same way for both static and dynamic scenes, allowing easier study of complex intervention scenarios (where everything moves: the operator(s), the shielding objects, the radiation sources).

1. Context
Preparing interventions in the nuclear field notably implies to evaluate the impact of radioactive sources on operators. In order to respect the ALARA principle, it is much relevant to find the most appropriate scenarios of intervention using simulation tools. Lots of computation codes exist to simulate the propagation of radiations, but most of them operate offline. Decreasing computation times in a noticeable way brings interactivity to the user, allowing him to interact with the scene and to intuitively study more appropriate scenarios.
Dose rate evaluation can be appraised by solving the transportation equation of Boltzmann. For radiation protection purposes, people often take advantage of simplified methods and algorithms, such as the straight line attenuation method with build-up factors [1]. Examples of radiation protection softwares using this method include MERCURAD [2] (based on the Mercure code) and VISIPLAN [3]. Besides, each volumic source is often seen as a cloud of points emitting radiations (Monte-Carlo approach of the emitting volume). The more points there are in the cloud (typically tens/hundreds of thousands of points), the more accurate is the representation of this volume.
For each generated ray, all the intersections with the objects of the scene have to be found. The final computed dose rate will depend on the depth of materials intersected by each ray. In the simulation tool we developed to perform interactive dose rate, the objects of the scene can be: analytic (sphere, box, cylinder ...), a combination of analytic objects (for instance, a pipe is an external cylinder minus an internal one), or meshed objects. The latter ones (soup of triangles) typically represent complex...
objects coming from CAD tools. We are able to account for all these kinds of objects, but we will only focus on meshed objects in this paper.

The number of objects in a scene can be quite large (some thousands), leading to a scene counting millions of triangles. The number of rays coming out of all volumic sources can also be large (up to millions of rays). Thus, a very high amount of intersections must be computed for each dose rate evaluation. Moreover, all these intersections have to be sorted along each ray. Even if additional computations are necessary before getting a final result, the bottleneck of the overall simulation always remains in these ray tracing requests.

2. Relationship with traditional ray-tracing

Our concern could be thought of as a classical ray-tracing problem, but two main differences appear between our approach and traditional ray tracing. First, this approach does not require the handling of secondary (and incoherent) rays. Indeed, each rays coming from sources converge to the measure point. Hence, it is as if all rays started from this same point, since the beginning and ending points of each ray can be switched without modifying the list of objects intersected along the ray. These highly coherent rays, usually referenced as primary rays in the ray-tracing community, allow major optimizations.

The second main difference is that each intersection along each ray has to be stored. This requirement, useless in usual ray tracing, is very demanding. Moreover, these intersections also have to be sorted, in order to compute depths of materials intersected along each ray. Most of the time, in graphical raytracing applications, only the first intersection along each ray is computed.

![Figure 1](image1.png)

**Figure 1.** nuclearCase scene (738K triangles) from two different points of view. 600,000 rays are drawn, for 6 millions of intersections sorted, in 400 ms on a NVIDIA GTX 295. Rays are coloured in white and become red when crossing an object.

3. Choice of the structure of acceleration

There also exist major similarities between our case and the traditional ray-tracing. In both cases, it would be totally inefficient to test each ray against each triangle of the scene. It is then appreciable to use acceleration structures. Building an acceleration structure consists in pre-sorting the triangles, so that, for each ray, only a very few triangles will have to be effectively tested. Different types of acceleration structures can be used, depending on the use case ([4], [5], [6]). Hunt defined a grid in perspective space [7] that turns out to be the most appropriate structure for primary rays. In our case, it consists in a regular tiling of the space (in fact the perspective space) around the point where the dose rate has to be evaluated (see Figure 2). It can be seen as a regular tiling of the image produced by a look at the scene through a camera whose point of view is the measure point. To view all of the space, six of these cameras, *e.g.* six grids, have to be constructed.
The perspective grid is particularly well-suited to our purpose: since each ray starts from the same point, each ray belongs to one unique cell. Therefore, the list of triangles to be checked for potential intersections is directly available, saving up significant computation time (see Figure 2).

It is the acceleration structure we chose to perform very efficient dose rate computations. Even though it is necessary to rebuild this structure regularly for static and dynamic scenes (each time the measure point moves), the construction algorithm we designed makes it possible to build it very quickly. Therefore, what seemed to be a drawback turns out to be an asset: dynamic scenes can be computed in no time thanks to this grid.

4. Results and discussion

4.1. Implementation

We bench our algorithm on an NVIDIA GeForce GTX 295 (2*896 Mo), coupled with a 4 cores, Intel(R) Xeon(R) X5550 @ 2.67GHz. After transferring the data between the CPU and the GPU, all the computations are done on the GPU. To check the results produced by our algorithm, we used traditional scenes of the ray-tracing community: Erw6, Fairy Forest, Conference and Sully.

4.2. Experimentations on a nuclear-like scene

Then, we designed a study-case (called nuclearCase) illustrating the kind of scene (in terms of complexity) potentially involved in a nuclear installation. We defined 6 groups of 100 000 points, representing the six volumic radiation sources of the nuclearCase scene. All rays thrown in the present study case start from one of the 600 000 emitting points, and end at the same point (representing the position of a virtual operator, where the dose rate is evaluated).
Table 1. Simulation times (including the building of the structure) required to measure the dose rate at a point inside various scenes. BVH is a standard acceleration structure used in traditional ray-tracing. perspGrid is the perspective grid we used. Times are given in milliseconds.

| Scene Description         | BVH CPU 1 core | PerspGrid CPU 1 core | PerspGrid CPU 8 cores | PerspGrid GPU |
|---------------------------|---------------|----------------------|------------------------|---------------|
| Erw (804 trgs)            | 810           | 561                  | 121                    | 71            |
| Conference (283K trgs)    | 3578          | 2308                 | 427                    | 152           |
| Fairy (174K trgs)         | 2864          | 1653                 | 276                    | 112           |
| Sully (804K trgs)         | 5169          | 3266                 | 550                    | 340           |
| Nuclear Case (800K trgs)  | 6145          | 3920                 | 605                    | 395           |

These rays and the overall scene are seen through an external camera. In the air, each ray is coloured white, and turns red when intersecting an object (see Figure 1).

Using our new algorithm and implementation, we compute and sort about 6 millions intersections by simulation step, in less than 400ms (see Table 1).

5. Next steps
We currently investigate some complementary optimizations, such as:
- Taking advantage of boxes bounding sources to restrict the amount of triangles potentially intersected by a group of rays related to the same source
- Dispatching the computations on several GPUs. A first check with the NVidia GTX295 card (exposing two graphical processing units) brought promising results: between 250ms and 350ms of computation, depending on the way we split all the computations on each GPU.

6. Conclusion
We briefly presented in this paper the methods and acceleration structures we use to quickly perform the geometrical computations required for the dose rate computation based on the straight line attenuation method with build-up factors. We show that important speed-up can be achieved using GPU implementation of theses methods.

We plan to further improved theses algorithms by automatically adding bounding boxed around radioactive sources, hence reducing rays triangles tests.

Then, this enhanced algorithm will be integrated to the current simulation library developped at CEA LIST, and that is part of the NARVEOS [8] industrial software from AREVA/Euriware company.

References
[1] Assad A, Chiron M, Nimal J C, Diop CMand Ridoux P 1999 Proc. of the Ninth International Conference on Radiation Shielding (Japan: Atomic Energy Society of Japan) 9 493
[2] MERCURAD http://www.canberra.com/products/13250.asp
[3] VISIPLAN http://www.visiplan.be/navigvisiplan/vud report.htm
[4] Reshetov A, Soupiokov A and Hurley J 2005 SIGGRAPH ’05: ACM SIGGRAPH 2005 Papers (New York, NY, USA: ACM) 1176-1185
[5] Wald I, Boulos S and Shirley P 2007 ACM Trans. Graph. 26 6 ISSN 0730-0301
[6] Cadet G and Lecussan B 2007 RT ’07: Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing (Washington, DC, USA: IEEE Computer Society) 63-71 ISBN 978-1-4244-1629-5
[7] Hunt W and Mark W R 2008 *IEEE/EG Symposium on Interactive Ray Tracing 2008 (IEEE/EG)* 3-10

[8] NARVEOS
   http://www.euriware.fr/scripts/home/publigen/content/templates/show.asp?p=399&l=fr