SIMULATING THE J-PET DETECTOR ON NVidia RAY TRACING HARDWARE*

P. Bialas, P. Sagało, K. Nowakowski

Faculty of Physics, Astronomy and Applied Computer Science
Jagiellonian University, Kraków, Poland

(Received November 25, 2019)

In this contribution, we present preliminary results of using graphic card with hardware support for ray tracing for physics simulation of a positron emission tomography (PET) scanner. On our simplistic setup, we notice an impressive (about 350 times) speedup compared to Geant4 code running on modern multicore CPU. We expect this speedup to come down but remain substantial also for other more complicated scenarios.

DOI:10.5506/APhysPolB.51.191

1. Introduction

Monte-Carlo methods (MC) are by now extensively used for designing hardware and software for medical imaging. The software of choice for this kind of simulations are Geant4 [1] and its extension GATE [2]. While Geant4 is a powerful simulation toolkit it can be slow as it currently does not take advantage of the vector hardware as found in modern processors and GPUs. Efforts to change this state are underway, notably in the framework of the GeantV project [3]. GeantV is a major undertaking aimed at vectorizing most of the Geant4 code and intended to be used for simulations in the next generation high-energy experiments. We have opted instead for an approach customized for a particular application: simulating our prototype positron emission tomography scanner J-PET [4–7]. This is suited for our limited resources and permitted us to experiment with the newest hardware that would not necessarily scale up to bigger projects. Restricting our focus to PET has the advantage that the energy range is also very limited, so we decided to simulate only the Compton scattering as it is dominant process in the considered energy range.

* Presented at the 3rd Jagiellonian Symposium on Fundamental and Applied Subatomic Physics, Kraków, Poland, June 23–28, 2019.

2. Ray tracing

The problems with efficient vectorization are, of course, not unique to particle physics simulations. In fact, those simulations are similar to a rendering technique for global illumination known as ray tracing [8]. In this approach, rays of light are traced usually from the camera to the object on scene and then reflected, refracted or scattered according to laws of optics. There were many attempts to implement efficient ray tracing on GPUs (*e.g.* [9, 10]). One big effort came from NVidia in the form of the OptiX engine [11]. This provided a relatively easy to use framework for GPU accelerated ray tracing and it was already used for particle detector simulations [12]. Meanwhile, NVidia has released GPUs with hardware support for ray tracing. This contribution presents results obtained using OptiX engine with NVidia RTX 2080 Ti graphics card.

3. Implementation

Programing the RTX pipeline amounts to specifying programs that will be executed automatically on the graphic hardware. Two main programs are: ray generation program that will generate initial rays, and the closest hit program that will be called when ray intersection with the triangle closest to its origin is detected. This program can in turn spawn other rays.

Physics process is defined by the mean free path and scattering angle distribution. We store the energy-dependent mean free path as 1D layered texture, one layer for each material. For sampling scattering angle probability distribution, we use the inverse cumulant method. We discretize the whole energy range and for each energy value, we calculate the inverse cumulant. The resulting array is stored as a 2D texture. The additional advantage of using textures is optimized memory access on GPU and automatic interpolation.

When the closest hit program is invoked, it checks if the particle interacted before hitting the border by comparing the interaction length to the distance traveled. If so, it calculates the energy change and scattering angle and recasts the ray in appropriate direction from the interaction point. If not, ray is recast in the same direction. Obviously, this is possible only for neutral particles or in the absence of magnetic field which is the case for PET scanners.

A common bottleneck of GPUs is the data transfer between CPU and GPU. We generate all of the rays directly on GPU so the transfer to GPU is not a problem. However, we are using quite a lot of memory on the GPU. This is due to the fact that we allocate each event its own memory slot capable of storing a fixed number of scatterings. For a detector like the J-PET, most of those slots are unoccupied. For this reason, before transferring the data back to CPU we use stream compaction to eliminate empty slots.

4. Results

We have tested our implementation on the three layers J-PET prototype [6, 13]. This detector contains 192 half meter long scintillators in total. We have compared our code to a multithreaded Geant4 implementation with all the physics processes except Compton scattering switched off. Geant4 code was executed on a six core Intel i9 CPU. The results are presented in Table I. One event corresponded to casting two rays (γ quanta) in opposite directions. The rays origins were selected randomly from a 4 mm^3 voxel, with one corner at the center of the detector.

TABLE I

Timing (in seconds) for simulating three layers of the J-PET detector. Only Compton scattering was considered.

	Geant4/i9[1]	RTX 2080Ti[2]		GTX 1080[3]		
		HW[4]	SW[4]	HW[4]	SW[4]	
No. of threads[5]	12	8704	21760	12800	12800	
No. of events ($\times 10^6$)						Speedup
1	6.2	0.02	0.03	0.07	0.08	310
2	12.9	0.04	0.07	0.14	0.15	322
4	26.1	0.08	0.13	0.27	0.29	330
8	57.7	0.16	0.25	0.53	0.57	365

[1] 6 cores, 32GB RAM.

[2] 4352 CUDA cores, 11.75 GFLOPS, 11GB GRAM.

[3] 2560 CUDA cores, 8.20 GFLOPS, 8GB GRAM.

[4] HW — Hardware triangle-ray intersection, SW — Software triangle-ray intersection.

[5] The number of threads was chosen as to give the best performance for 8 millions events.

The obtained results clearly show the advantage of the hardware acceleration for ray tracing. The GTX 2080Ti is nominally only 1.4 times faster when considering GFLOPS than GTX 1080 but over three times faster on our simulation. Also switching off the hardware acceleration for ray triangle intersection test increases the time by 50%.

5. Discussion

We have shown that new graphics hardware is capable of running physical simulations over 350 times faster compared to standard simulation software on modern CPU. However this represents probably only the upper limit of what can be achieved. Scaling the implementation to handle multiple physical processes will most probably result in decreased performance as the thread divergence will increase. Also the processes with very short step lengths such as ionization will not benefit as much. Support for new shapes has also to be added as far we are using the built-in hardware accelerated triangle meshes. This enables to efficiently use simple shapes as boxes, but would be very wasteful for other shapes like cylinders or cones. Those shapes can be accommodated using custom written intersection programs.

We have discovered two issues with presented approach. Spawning new rays is done in recursive fashion and the recursion depth is limited currently to 31 by hardware. This was not a problem in the case of the simple model we simulated, however, in general, the recursion would have to be managed explicitly by maintaining a stack in global memory.

The second issue relates to numerical accuracy. Geant4 uses double precision and GPUs use single precision. Because of this, our physical precision was of the order of ten microns. This is adequate in our simulations but scaling up to bigger, in terms of size, detectors would entail proportional loss in precision.

We find those preliminary results very encouraging. It is obvious that realistic simulations still require a significant extension of our code which will undoubtedly result in loss of the performance but even tenfold drop in performance would still result in significant speedup. In our opinion, the ray tracing acceleration hardware seems to be suitable for simulations of PET scanners and maybe other medical devices. This, because those applications do not require simulating magnetic field, energy range of particles is limited and the detectors are generally small.

REFERENCES

[1] S. Agostinelli *et al.*, *Nucl. Instrum. Methods Phys. Res. A* **506(3)**, 250 (2003).

[2] S. Jan *et al.*, *Phys. Med. Biol.* **49**, 4543 (2004).

[3] G. Amadio *et al.*, *J. Phys.: Conf. Ser.* **664**, 072006 (2015).

[4] P. Moskal *et al.*, *Phys. Med. Biol.* **64**, 055017 (2019).

[5] P. Kowalski *et al.*, *Phys. Med. Biol.* **63**, 165008 (2018).

[6] S. Niedzwiecki *et al.*, *Acta Phys. Pol. B* **48**, 1567 (2017).

[7] P. Moskal *et al.*, *Phys. Med. Biol.* **61**, 2025 (2016).

[8] T. Whitted, *Commun. ACM* **23**, 343 (1980).

[9] J. Popov, J. Günther, H.-P. Seidel, *Comp. Graph. Forum* **26**, 415 (2007).

[10] J. Gunther, *et al.*, in :Proceedings of IEEE Symposium on Interactive Ray Tracing, September 10–12, 2007.

[11] S.G. Parker *et al.*, *ACM Trans. Graph.* **29**, 66 (2010).

[12] S. Blyth, *J. Phys.: Conf. Ser.* **898**, 042001 (2017).

[13] G. Korcyl *et al.*, *IEEE Trans. Med. Imag.* **37**, 2526 (2018).