Resolution of Self-Intersection Issue in Monte Carlo Simulations Employing Graphics Ray Tracing Technology

Sung Joon Kwon, Jaeuk Im, Han Gyu Joo*

*Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea
Corresponding author: joohan@snu.ac.kr

1. Introduction

In recent days, various advanced reactors have gained increasing interest due to several strengths such as inherent safety and sustainability. The representative advanced rectors are heat pipe cooled micro reactors such as MegaPower [1] and pebble-bed reactors such as HTR-PM [2]. These reactors have irregular geometric structures which are significantly different from the lattice-based structures of typical LWRs.

PRAGMA is a GPU-based continuous-energy MC code being developed at Seoul National University, which has a tailor-made rectilinear geometry model and its optimized features for typical PWR applications [3]. For extensibility of the scope of applications from PWRs to advanced reactors, PRAGMA exploits the hardware-accelerated NVIDIA ray tracing engine OptiX [4] rather than the constructive solid geometry (CSG) based module for efficient neutron tracking on GPUs. With the extension of geometry modeling, PRAGMA retains the general applicability by employing primitive-based geometry [5] and triangle-based unstructured mesh geometry [6] aided with OptiX.

However, there is still an obstacle, known as *the self-intersection* issue, to applying the graphics ray tracing library to physical simulations without any treatments. This issue is a well-known and notorious problem in graphics ray tracing due to round-off errors in the floating-point arithmetic. In the MC simulation, it can cause a neutron to be stuck on a surface, and it triggers performance degradation since additional manipulations are required to make the neutron deviate from the surface.

This paper presents a simple and robust ray casting algorithm to avoid self-intersection by adopting an Any-Hit module of OptiX. The irregular geometry treatment in PRAGMA is described briefly. A ray tracing pipeline and a traversal process are introduced and solutions for preventing self-intersection issues are demonstrated on unstructured triangle-based mesh geometry with fine granularity.

2. Unstructured Geometry Treatment in PRAGMA

An irregular geometry is represented based on a CAD mesh-based geometry model in PRAGMA. A nuclear reactor structure is modeled using only four types of basic mesh cells such as tetrahedron, hexahedron, wedge and pyramid cells. For an efficient modeling of a curve with meshes, a volume correction method was employed to preserve calculation accuracy using a few mesh.

The OptiX ray tracing API was adopted in PRAGMA to calculate a distance to the nearest surface for a neutron during a MC simulation. The OptiX is a CUDA-centric ray tracing API optimized for NVIDIA GPUs, which is employed in graphics or even physical simulations. For a mesh-based geometry treatment in MC simulation, the distance calculations are significant burdens since each region is divided into several mesh cells. The distance calculation performance can increase significantly by adopting the CUDA-based ray tracing API OptiX.

3. OptiX Ray Tracing Pipeline

The OptiX ray tracing API provides a simple and flexible programmable pipeline for ray tracing. The OptiX provides a program that is a block of executable code on the GPU and represents a particular shading operation. The ray tracing pipeline in OptiX mainly consists of a ray generation, an intersection program, shading, and a miss program. A virtual ray invoked from the ray generation program traverses scene geometry in the pipeline and finds the intersections with primitives as described in Fig. 1.

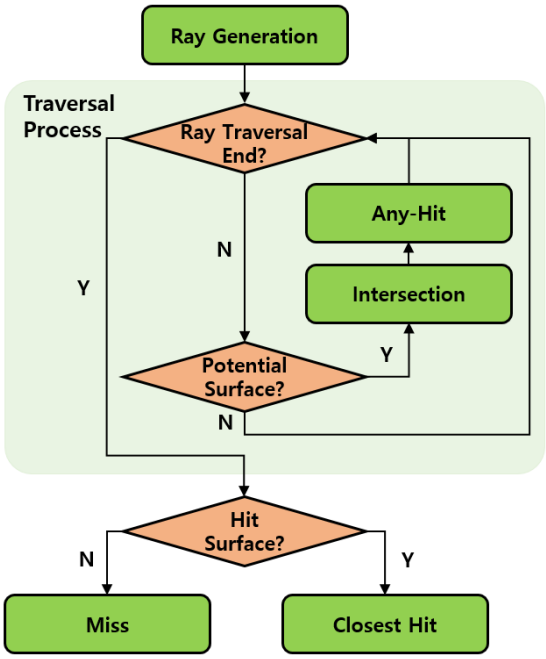

Fig. 1 Diagram of OptiX ray tracing pipeline.

When a ray is generated from a point, potential intersections are determined based on bounding boxes of primitives as illustrated in Fig. 2. Selecting potential intersections in the traversal process is fixed-function and hardware-accelerated operations which cannot be controlled by a developer.

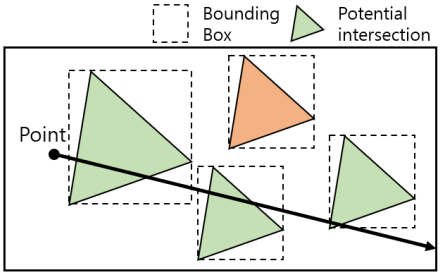

Fig. 2. Diagram of selecting potential intersections.

For all potential surfaces, an intersection and Any-Hit program are invoked during traversal. The intersection program takes a ray-primitive intersection examination and calculates a distance to the surface from the point. It is a user-provided program and can be implemented for all mathematically expressible primitives.

The Any-Hit program is called when a traced ray finds a potentially-closest intersection point for graphics shadow computation. However, in the MC application, this program is generally redundant since only the distance to the nearest surface from the point is utilized in the simulation process.

Based on the distance calculated by the intersection program, the closest surface is determined among the potential intersections. If the ray does not hit any surface along the ray trajectory, a miss program is invoked.

4. Self-Intersection Issue in Ray Tracing

In the OptiX ray tracing library, rays are represented as parametric lines. This representation calculates the endpoint of the ray using ray origin, direction, and the parameter that indicates distance. For the ray tracing simulation, it forms a ray path by starting at the light source and intersecting the ray with the scene geometry. As the surface is hit, a new ray is generated on that surface to continue the tracing.

Theoretically, this new ray should not detect the intersection with the same surface again since the zero-distance is excluded in parametric representations. However, due to errors in the floating-point arithmetic, the new ray may intersect a given point repeatedly with an infinitesimally small distance parameter. This phenomenon is called a self-intersection.

To prevent a self-intersection caused by the floating-point representation, a threshold parameter called scene epsilon is utilized in the ray tracing library. It neglects all intersections with distance parameters smaller than a preset value, which is defined to be small enough to discover actual intersections.

5. Self-Intersection Issue in MC Simulation

Nevertheless, the scene epsilon parameter is often not a remedy in the MC simulation. The physical accuracy required in neutronics compels the scene epsilon value to be considerably small, which is too microscopic to avoid self-intersections.

For the MC simulation, most self-intersections occur after the following: a collision event nearby the surface, crossing the surface, and reflecting on the surface. The common ground of these cases is when neutrons are cast extremely close to the surfaces. These can make neutrons stuck on the surface and thereby the simulation does not proceed without any additional treatment.

6. Any-Hit Approach to Avoid Self-Intersection

In this regard, a simple and robust treatment is developed for efficient neutron transport. When a ray traverses over the entire scene geometry, all ray-primitive intersection points along with the ray trajectory are considered as the potentially-closest intersection. The idea to resolve the self-intersection is to compare all potential hit-primitive indices to the hit-primitive index of an antecedent ray tracing. If the one among those indices is the same as the previous hit-primitive index, it means that the self-intersection occurs. Namely, the self-intersection can be surely prevented by “ignoring” duplicated intersection with that primitive.

To achieve this, the Any-Hit program is adopted. A new conditional operation is augmented in the Any-Hit program such that it detects and ignores the duplicated intersection point. It can achieve an original closest intersection to be selected instead of the self-intersection as illustrated in Fig. 3.

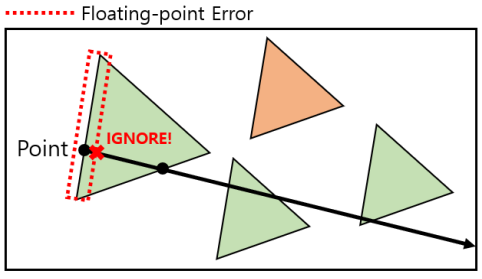

Fig. 3. Resolving self-intersection by applying Any-Hit.

This procedure does not take a further computational burden since the operation is performed within a single ray traversal process. Also, the overhead of applying the Any-Hit program to every potential intersection is negligible due to the optimized algorithms of the OptiX when constructing a ray tracing pipeline.

7. Results

Adopted problems are the minicore designed by ANL and the one-sixth symmetrical MegaPower 3D core without control drums. Note that the vapor region of each heat pipe is replaced by a void pipe with reflective boundary in this work. The tracking method without the Any-Hit was adopted as a standard case to confirm the effect of the Any-Hit approach in a MC simulation.

Fig. 4 illustrates the specifications and the mesh granularity of the minicore problem. The calculation conditions of PRAGMA employed for this simulation are shown in Table I.

.

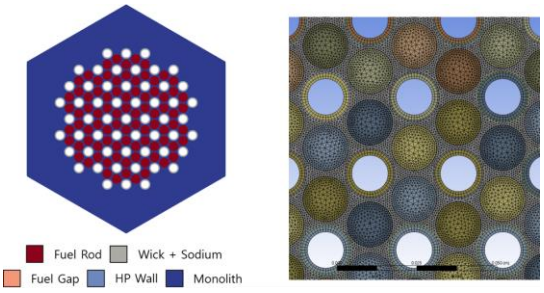

Fig. 4. Specifications and mesh granularity of minicore.

Table I: Calculation conditions for minicore.

Number of Inactive Cycles	25
Number of Active Cycles	100
Number of Neutrons / Cycle	4,000,000
Libraries (K)	900 / 1000

Fig. 5 illustrates the number of events per cycle for the minicore simulation and Table II summarizes calculation results of two cases. It is observed that a neutron event behavior of standard case shows much higher value and fluctuation compared to that of the Any-Hit case For a standard case, the average number of events appeared to be about 60 times larger than that of the Any-Hit case. The reason for this tendency is that neutrons stuck on the surfaces by self-intersections lead to an abnormal population tail effect. But, the Any-Hit approach can properly resolve this effect and thereby retain ordinary performance.

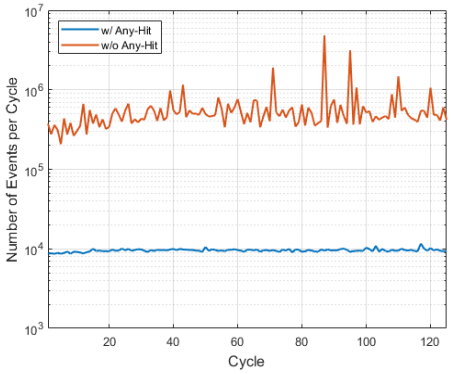

Fig. 5. Neutron events at each cycle for minicore.

Table II: Comparison of calculation results for minicore.

Scheme	**w/ Any-Hit**	**w/o Any-Hit**
Multiplication factor	1.04684 (2)	1.04682 (2)
Average Number of Events per Cycle	9,546	572,847

The next problem was the 1/6 symmetrical MegaPower 3D core illustrated in Fig. 6. This problem also has similar mesh granularity compared with the minicore problem with larger geometry size. The calculation conditions of PRAGMA are listed in Table III.

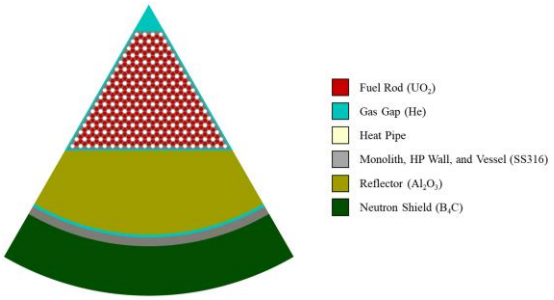

Fig. 6. Specifications of MegaPower without control drums.

Table III. Calculation conditions for MegaPower.

Number of Inactive Cycles	50
Number of Active Cycles	100
Number of Neutrons / Cycle	4,000,000
Libraries (K)	900 / 1000 / 1100

Unfortunately, the simulation did not proceed for the standard case in this problem. As shown in Fig. 7, the number of alive neutrons decreased to some extent and then became saturated in the standard case. Some particles did not disappear even after several hundred thousand events. On the other hand, the number of alive neutrons rapidly diminished when adopted the Any-Hit approach, and the simulation successfully proceeded.

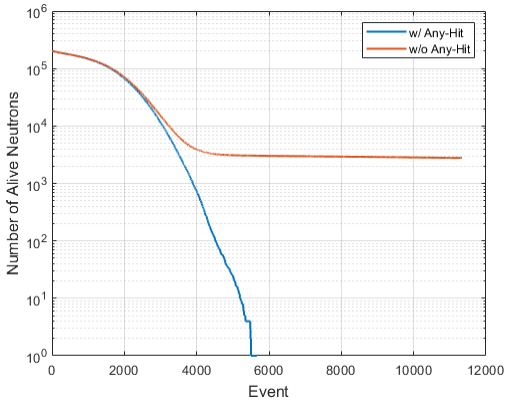

Fig. 7. Number of alive neutrons at each event of a cycle for MegaPower.

Through the trend of decreasing particles over events, it was expected that a non-negligible number of particles was trapped on the surface in the existing tracking algorithm such that they could have significant impacts on the accuracy of simulation results.

Thus, the multiplication factors were compared to the reference solution to confirm the impact of the stuck particles on the simulations. The comparison was performed with the standard tracking algorithm with forced termination as 7,067 events per cycle and the tracking algorithm with the Any-Hit approach. Here, the maximum event limit was selected to the average number of events of the new scheme with the Any-Hit module. And the reference solution was obtained by the delta-tracking scheme which is relatively free from self-intersections.

Table IV presents calculation results of each tracking scheme. The reference solution obtained by the delta-tracking scheme is 1.13063 (3). The multiplication factor calculated by the new tracking scheme with the Any-Hit approach is reasonable compared to the reference solution. However, the multiplication factor solved by the standard tracking scheme without the Any-Hit module was significantly lower than other results despite the same average number of events as the tracking scheme with the Any-Hit program. It confirms that the effect of stuck neutrons for a simulation cannot be ignored and the Any-Hit program is in harmony with MC simulations as resolving self-intersections.

Table IV. Comparison of results for MegaPower**.**

Scheme	**w/ Any-Hit**	**w/o Any-Hit**
Multiplication factor	1.13070 (3)	1.12538 (3)
Average Number of Events per Cycle	7,067	10,000
Average Tracking Time of Cycle [s]	29.63	29.70

Also, it was expected that the additional operation to all potential intersections by the Any-Hit program increases the computational burden. However, the simulation performance increased with the Any-Hit program due to the reduced cumulative particle count despite the additional operation.

8. Conclusions

In this paper, a robust and straightforward algorithm was developed to resolve the self-intersection issues in MC simulations exploiting graphics ray tracing library. The Any-Hit program, which was originally used for graphics shadow computation, was successfully augmented to the existing OptiX pipeline in PRAGMA. By adopting the Any-Hit program, the duplicated hit-primitive can be ignored by force while a ray traverse over all potential intersections.

As the result, most self-intersections were resolved by the Any-Hit approach as retaining accuracy even in the case that induces severely high surface detection events. Moreover, it was confirmed that the accuracy was reasonably retained with reduced computational burden, unlike other additional manipulation. Now, it is planned to verify the unstructured geometry treatments of PRAGMA through various realistic problems. In addition, it should be verified whether the self-intersections are totally resolved through verifications with various advanced reactor problems.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021M2D6A1048220).

REFERENCES

[1] P. R. Mcclure, D. I. Poston, V. R. Dasari, R. S. Reid, “Design of Megawatt Power Level Heat Pipe Reactors, Los Alamos, New Mexico,” USA, LA-UR-15-8840, 2015.
[2] Z. Zhang *et al.*, The Shandong Shidao Bay 200 MWe High Temperature Gas-Cooled Reactor Pebble-Bed Module (HTR-PM) Demonstration Power Plant: An Engineering and Technological Innovation, Engineering, Vol.2, p. 112, 2016.
[3] N. Choi, K. M. Kim, H. G. Joo, “Initial Development of PRAGMA – A GPU-Based Continuous Energy Monte Carlo Code for Practical Applications,” Korean Nuclear Society Autumn Meeting, Goyang, Korea, Oct.24-25, 2019.
[4] S. G. Parker *et al.*, “OptiX: A General Purpose Ray Tracing Engine,” ACM Transactions on Graphics, Vol.29(4), pp.1-13, 2010.
[5] J. Im *et al.*, “Flexible Geometry Treatment in PRAGMA Using NVIDIA® Ray Tracing Engine OptiX™,″ Transactions of the Korean Nuclear Society Virtual Spring Meeting, Jul.9-10, 2020.
[6] J. Im *et al.*, “Multiphysics Analysis System for Heat Pipe Cooled Micro Reactors Emplying PRAGMA-OpenFOAM-ANLHTP,” Proceedings of International Conference on Physics of Reactors, Pittsburgh, PA, United States, May.15-20, 2022.