Reflection Simulation of KSTAR Tungsten PFC

Seungtae Oh, Kyungmin Kim and Jieun Choi

Korea Institute of Fusion Energy, Daejeon, Republic of Korea

stoh@kfe.re.kr

Tungsten is a great candidate for plasma facing components (PFCs) of tokamaks due to its high melting temperature and high thermal conductivity. However, the reflectivity of tungsten is not negligible and it can make the diagnostic instruments work in a wrong way due to the signal distortions induced by the reflections. The reflection elimination can be possible when all reflection rays coming into a sensor are tracked and subtracted. For the first process tracking all the rays, a ray-tracing code of KSTAR, “Wray”, is successfully developed.

1. Introduction

The plasma facing components (PFCs) of KSTAR are planned to be changed into tungsten tiles in 2022. After the changes, most diagnostic instruments of KSTAR may experience new environments with the reflections of the tungsten tiles since the reflectance of tungsten was reported to be more than 90 % at 3 um of wavelength [1]. Moreover, the reflectance goes up as its temperature increases [2]. Signal distortions of the diagnostic instruments are inevitable due to the reflections. This means that most diagnostics will not work properly under the tungsten environments. This situation was already reported in Reference 3 that reflections from tungsten tiles can induce 85 % over-estimation in tile temperature measurements with an infrared camera. However, considering that most optical systems are linear systems, we might be able to discriminate the true signal from the reflections if all ray information is given. The ray paths could be found with ray-tracing simulations since the geometry structures of KSTAR tokamak are all given.

In ray-tracing technique, there are two approaches (a ray-tracing way in computer graphics and one in scientific light simulation). They are quite different since computer graphics fields use backward ray-tracing and they just concern color while scientific simulation concerns optical power through forward ray-tracing. Unfortunately, the commercial or open ray-tracing programs do not give what we want, "full ray information such as ray hitting angle, object name, its position and etc.". Keeping or saving full information costs huge memory waste and interrupts during calculations. Moreover, most ray-tracing pipelines supported by hardware or software are encapsulated, and the modules will not permit any external intervention. So, "Wray", a ray-tracing simulation code of KSTAR got to be developed, and the most distinct feature of "Wray" compared with other ray-tracing programs is to provide all information of the ray paths and hit event information. So, the outputs of the code are a synthetic radiation image on an IR camera and data files of all ray information.

2. Wray, ray-tracing code of KSTAR

The ray-tracing code of KSTAR, "Wray" is one of 3 parts of the reflection elimination technique composed of "Wemi" for PFC surface property measurement, "Wray" for ray-path information, and "Wana" for discrimination true signal as shown in Figure 1. In a program design stage, "Wray" was supposed to have the components shown in Figure 2. To lighten the programing loads, "Wray ver. 1" was developed on the Unity3D [4] since it already has most components in itself such as 3D file handlings and user interfaces. The physics engine of Unity3D has very suitable components to build ray-tracing codes for our purpose. So, the structure of the program was simplified as shown in Figure 3. The first version of "Wray" has successfully provided the full ray information, and the feasibility of the reflection elimination concept has been nicely checked. However, the physics engine of Unity does not support multi-threading. "Wray ver.1" got even slower since the ray information is saved on a SQL database. So, the first version has a limitation to extend its functionality due to this speed. In

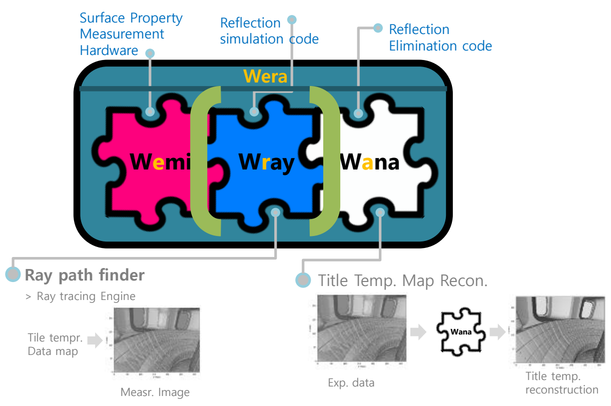


Figure 1. 3 parts of the reflection elimination technique.

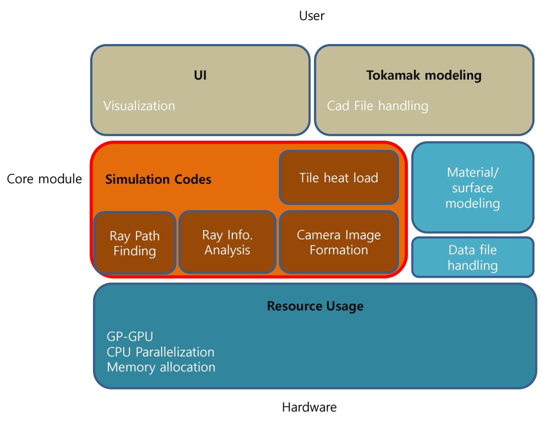


Figure 1. Components of "Wray"

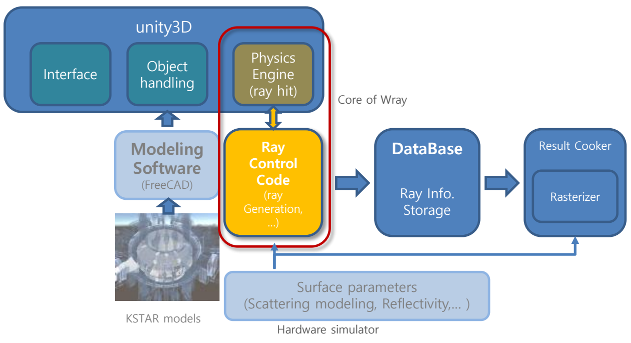


Figure 3. Structure of “Wray ver. 1”

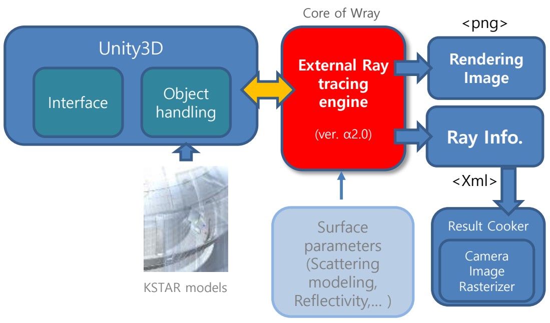


Figure 4. Structure of “Wray ver. 2”

the next version of “Wray”, its ray-tracing engine is rebuilt from scratch without any dependency on the physics engine of Unity. The ray-tracing engine of “Wray ver. 2” first allocates the camera pixels into the cores of CPUs, and each pixel shoots rays as backward ray-tracing. And, the rays are calculated by looping their hit event tracking and the child ray generations as shown in Figure 5. Finally, all cores of CPUs can be utilized in the computations, and the storage speed gets faster since the ray information is stored in XML format files by individual process. The hitting checks of a single ray are usually taken over all polygons of all objects. So, preventing the overlaps of the hitting checks, Bounding Volume Hierachy (BVH) [5] is adopted. The scattering of the surfaces is controlled by a bidirectional reflectivity distribution fuction (BRDF) of KSTAR PFC not by Phong shadings [6]. The BRDF enhances the efficieny of ray utilizations since that provides the high probability of child ray generation with high power. As shown in Figure 6, a rendering image and the ray

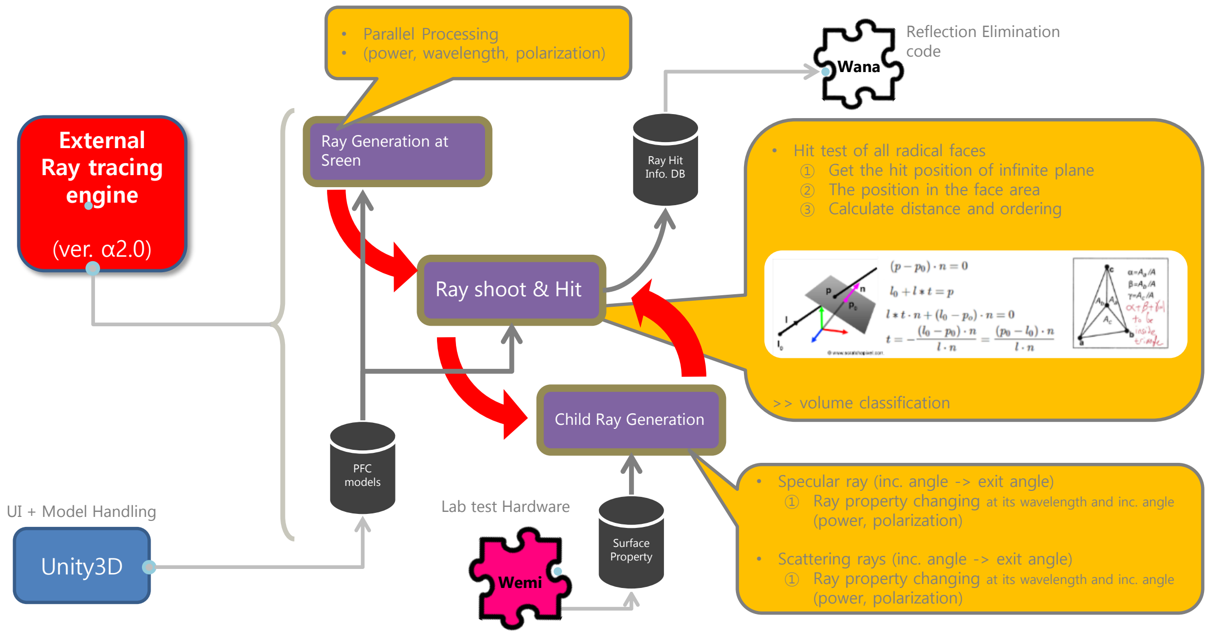
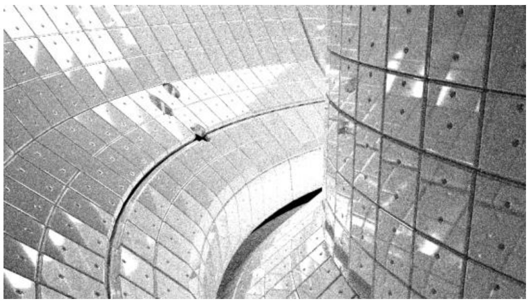
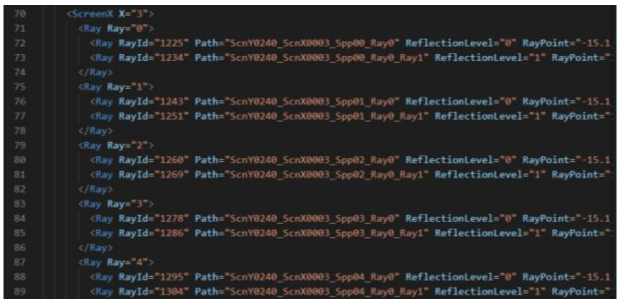

Parallel Processing
(power, wavelength, polarization)
Reflection Elimination code
Wana
External Ray tracing engine
(ver. α2.0)
Ray Generation at Sreen
Ray Hit Info. DB
Hit test of all radical faces
① Get the hit position of infinite plane
② The position in the face area
③ Calculate distance and ordering
Ray shoot & Hit
$(p-p_0)\cdot n=0$
$l_0+l*t=p$
$l*t\cdot n+(l_0-p_0)\cdot n=0$
$t=-\frac{(l_0-p_0)\cdot n}{l\cdot n}=\frac{(p_0-l_0)\cdot n}{l\cdot n}$
α=A_a/A
β=A_b/A
γ=A_c/A
α+β+γ=1 to be inside triangle
>> volume classification
Child Ray Generation
PFC models
UI + Model Handling
Unity3D
Lab test Hardware
Wemi
Surface Property
Specular ray (inc. angle -> exit angle)
① Ray property changing at its wavelength and inc. angle (power, polarization)
Scattering rays (inc. angle -> exit angle)
① Ray property changing at its wavelength and inc. angle (power, polarization)

Figure 5. Process of ray tracing engine of “Wray ver. 2”





```
<ScreenX X="3">
  <Ray Ray="0">
    <Ray RayId="1225" Path="ScnY0240_ScnX0003_Spp00_Ray0" ReflectionLevel="0" RayPoint="-15.1
    <Ray RayId="1234" Path="ScnY0240_ScnX0003_Spp00_Ray0_Ray1" ReflectionLevel="1" RayPoint="
  </Ray>
  <Ray Ray="1">
    <Ray RayId="1243" Path="ScnY0240_ScnX0003_Spp01_Ray0" ReflectionLevel="0" RayPoint="-15.1
    <Ray RayId="1251" Path="ScnY0240_ScnX0003_Spp01_Ray0_Ray1" ReflectionLevel="1" RayPoint="
  </Ray>
  <Ray Ray="2">
    <Ray RayId="1260" Path="ScnY0240_ScnX0003_Spp02_Ray0" ReflectionLevel="0" RayPoint="-15.1
    <Ray RayId="1269" Path="ScnY0240_ScnX0003_Spp02_Ray0_Ray1" ReflectionLevel="1" RayPoint="
  </Ray>
  <Ray Ray="3">
    <Ray RayId="1278" Path="ScnY0240_ScnX0003_Spp03_Ray0" ReflectionLevel="0" RayPoint="-15.1
    <Ray RayId="1286" Path="ScnY0240_ScnX0003_Spp03_Ray0_Ray1" ReflectionLevel="1" RayPoint="
  </Ray>
  <Ray Ray="4">
    <Ray RayId="1295" Path="ScnY0240_ScnX0003_Spp04_Ray0" ReflectionLevel="0" RayPoint="-15.1
    <Ray RayId="1304" Path="ScnY0240_ScnX0003_Spp04_Ray0_Ray1" ReflectionLevel="1" RayPoint="
```

Figure 6. Outputs of “Wray ver. 2” : a) rendering image, b) ray information in XML format

information are nicely provided by “Wray ver. 2”.

3. Summary

3 functional modules for the reflection elimination of tungsten walls of KSTAR are being prepared under the concept that true signals can be discriminated with full ray information. The ray information can be derived by the synthetic ray-tracing program. So, for KSTAR, the ray-tracing program was developed as “Wray” providing a rendering image and ray path information in XML format. “Wray” adopts multi-threading algorithms, BVH, probability sampling for efficient usages of rays and computing resources.

Reference

1. Z. Sárosi, W. Knapp, A. Kunz, & K. Wegener, “Evaluation of reflectivity of metal parts by a thermo-camera,” *InfraMation 2010 proceedings*, 475-486 (2010).

2. A. Huber, et al. "Real-time protection of the JET ITER-like wall based on near infrared imaging diagnostic systems," *Nuclear Fusion* 58.10 (2018)

3. Aumeunier, M-H., et al., "Impact of reflections on the divertor and first wall temperature measurements from the ITER infrared imaging system.", Nuclear Materials and Energy, 12 (2017): 1265-1269.

4. https://en.wikipedia.org/wiki/Unity_(game_engine)

5. https://en.wikipedia.org/wiki/Bounding_volume_hierarchy

6. https://en.wikipedia.org/wiki/Phong_shading