Milestone: Raytracing Pipeline for Radiance

Shashank Anand
UC Berkeley
United States
shashank.anand@berkeley.edu

Yongye Zhu
UC Berkeley
United States
yongye.zhu@berkeley.edu

Yifei Dai
UC Berkeley
United States
yifeidai@berkeley.edu

Seyoung Kim
UC Berkeley
United States
sam2025@berkeley.edu

Abstract

We present the current progress on a high-performance path tracer featuring a CUDA-based implementation of ReSTIR GI and integrated denoising. The path tracer has been extensively rewritten to maximize performance, including the replacement of polymorphic scene structures with flat, memory-coherent representations, and the introduction of a custom RNG for GPU compatibility. We demonstrate 1spp ReSTIR GI renders denoised in under 50ms for common scenes. We also highlight ongoing efforts to support advanced scenes by replacing the incomplete Collada-based pipeline with a GLTF parser using tinyGLTF. The new parser eliminates the need for half-edge meshes, streamlining the asset pipeline and paving the way for real-time rendering of complex models.

ACM Reference Format:
Shashank Anand, Yongye Zhu, Yifei Dai, and Seyoung Kim. 2025. Milestone: Raytracing Pipeline for Radiance. In . ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Website at : `https://shorturl.at/aT0QD`
Slides at : `https://shorturl.at/uSDh9`

1 Overview

Over the course of the last weeks, we have focused on the following primary tasks:

(1) Implement ReSTIR GI
(2) Implement naive raytracing on Cuda
(3) Implement ReSTIR GI on Cuda
(4) Support advanced scenes

Points 1 through 3 are complete. Point 4 has seen significant progress, and a direction to solve point 4 has been established.

2 Cuda, ReSTIR GI and Denoising

We have the following implementation completed:

(1) ReSTIR GI CPU + Denoising
(2) Naive raytracing on Cuda
(3) ReSTIR GI on Cuda + Denoising

Implementing these has been a tremendous amount of work. More or less every aspect of the hw3 pathtracer has been rewritten. From the renderer to the pathtracer, and the data types as well. Most significantly polymorphism has been entirely eliminated, samplers have been revamped to use a custom no-library RNG compatible with Cuda (we faced issues with CuRand, and therefore dropped it). Our RNG is uses two 64 bit integers.

The only untouched part of the pathtracer code from hw3 is the Collada parser. However (spoiler alert) that will also be eliminated soon, making the pathtracer truly custom.

Primitives, BSDFs and Lights are now stored in plain structs with unions embedded in them. They are stored contiguously in memory, for better locality as we sweep through the lights in order. Sadly can't optimize primitive placement as the BVH intersections are not predictable.

Finally all fp64s have been replaced with fp32s. This gave us a speedup of around 1.5x to 2x compared to using only fp64.

We have a roughly 10x speedup over baseline (exact numbers in final paper). ReSTIR GI 1spp render takes around 20-50ms (rendering easy scenes such as bunny, spheres, dragon, etc). Denoising this theoretically gives us over 100x speedup over a 1024spp render without ReSTIR GI, however the denoised image is of course of less quality than a true 1024spp render.

3 Milestone Progress: GLTF to Collada Conversion and Initial Rendering

The initial task involved assessing the feasibility of integrating GLTF files into the existing path tracing infrastructure by converting them to the Collada (.dae) format, as the current framework supports only Collada files. Blender was employed to successfully export GLTF models to Collada format. However, the initial exports resulted in segmentation faults when rendered within the path tracer. Upon investigation, it was discovered that Blender's default triangulation setting during export was causing incompatibility with the renderer, which exclusively supports `<polylist>` elements. This issue was resolved by unchecking the triangulation option in Blender's export settings.

To further ensure compatibility, Blender's 3D Print Toolbox add-on was utilized to detect and repair non-manifold geometries. After this mesh-cleaning process, exported Collada models (such as `bed.dae`) were successfully rendered by the path tracer.

Conference'17, Washington, DC, USA
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

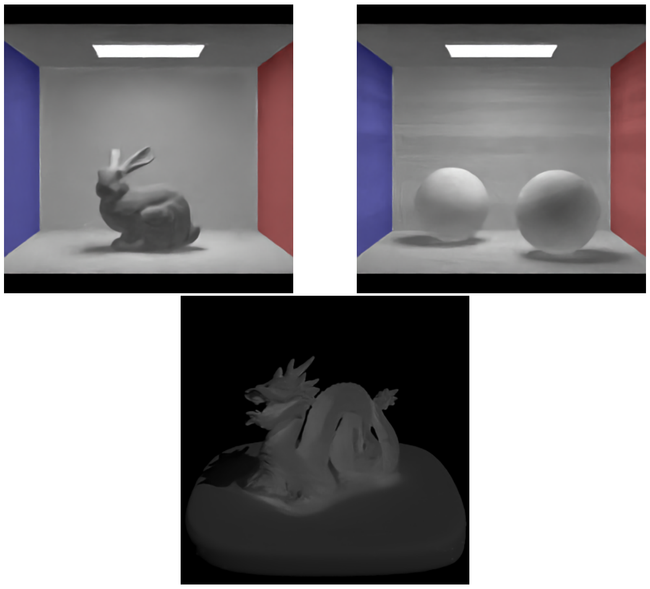

Figure 1: 1spp ReSTIR GI Denoised renders of bunny, sphere and dragon. Notice some artifacts around dragon's lower body due to spatial resampling at the surface inaccurately taking the bright dragon's body as a high probability sample due to high radiance value. Can be fixed with some delta illuminance cutoff

Subsequently, attempts to render a more complex model (`car.dae`) caused the half edge builder to complain that there were way more vertices in the vertex list, than the number of vertex indices in the polygon list. This issue was caused by the Collada parser in the hw3 pathtracer being incomplete: The collada parser does not support a single mesh having several polylists, only the first polylist is used. Additionally, there is no support for parsing triangle tags in the collada file. We cannot work around this by unchecking "triangulate", it is not feasible to manually prevent the existence of triangles (instead of polygons) in large multi-million primitives meshes.

4 What to do next?

Highest on the priority list is to get these GLTF scenes working, so we can show some cool renders. However, the collada parser is incomplete. We only have one way to work around this, write a GLTF parser. The current parser does way too much: it manually parses the XML. This will not be even remotely feasible to do manually to parse a .glb object file. Additionally, the meshes in the .dae files are converted to half-edge meshes, which are in turn converted back to a list of triangles for us to use in the pathtracer. This is unnecessary: we truly do not need the half edge mesh structure.

Therefore, the simplest path forward is to nuke the collada parser and the half edge mesh, and directly handle .glb files and extract the list of triangles, bsdfs, and lights. We will use tinyGLTF to greatly streamline this process. This will be our next goal, after long hours spent debugging the collada parser and understanding the intricacies of .dae files.

Our other parallel goal will be to convert the ReSTIR GI Cuda implementation to Vortex RISCV code. This will be slightly easier now, because over 90% of the pathtracer codebase has been nuked, and the other remaining parts almost entirely rewritten.