

Assignment 3:

Basic Ray Tracing

NAME: LOU CHUANG

STUDENT NUMBER: 2023533031

EMAIL: LOUCHUANG2023@SHANGHAITECH.EDU.CN

ACM Reference Format:

Name: Lou Chuang , student number: 2023533031, email: louchuang2023@shanghaitech.edu.cn

. 2025. Assignment 3: Basic Ray Tracing. 1, 1 (November 2025), 1 page.

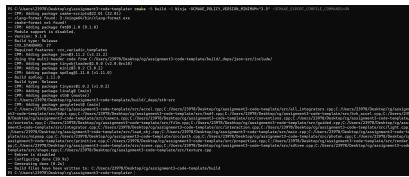
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 Introduction

In this assignment, I modified interaction.cpp, bvh_tree.h, integrator.cpp and bsdf.cpp to realize basic ray tracing.

2 Implementation Details

For compilation, I downloaded clangd and upgraded cmake. I used "cmake -B build -G Ninja -DCMAKE_POLICY_VERSION_MINIMUM="3.5" -DCMAKE_EXPORT_COMPILE_COMMANDS=ON" at the root directory and successfully compiled it. The result is as follows.



According to the formula $p_0 + u \cdot (p_1 - p_0) + v \cdot (p_2 - p_0) = o + t \cdot d$,
 $u = \frac{[d \times (p_2 - p_0)] \cdot [u \cdot (p_1 - p_0) + v \cdot (p_2 - p_0) - t \cdot d]}{(p_1 - p_0) \cdot [d \times (p_2 - p_0)]}$, $v = \frac{d \cdot (p_1 - p_0) \times [u \cdot (p_1 - p_0) + v \cdot (p_2 - p_0) - t \cdot d]}{(p_1 - p_0) \cdot [d \times (p_2 - p_0)]}$,
 $t = \frac{(p_2 - p_0) \cdot (p_1 - p_0) \times [u \cdot (p_1 - p_0) + v \cdot (p_2 - p_0) - t \cdot d]}{(p_1 - p_0) \cdot [d \times (p_2 - p_0)]}$, I calculated u, v and t through these formulas and added checks on these parameters.

Then I realized the AABB intersection, I confirmed the entering and exiting time of the ray and judged whether the intersection was valid.

After completing triangle and AABB intersection, the result of intersection test is as follows.

Author's Contact Information: Name: Lou Chuang
student number: 2023533031
email: louchuang2023@shanghaitech.edu.cn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM XXXX-XXXX/2025/11-ART
<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

```
[=====] Running 7 tests from 2 test suites.
[=====] Global test environment set-up.
[=====] 1 test from AABB
[RUN] OK ] AABB.AxisAligned_EnterExit_PositiveAndNegativeDirs
[RUN] OK ] AABB.AxisAligned_EnterExit_PositiveAndNegativeDirs (0 ms)
[=====] 1 test from AABB (2 ms total)

[=====] 6 tests from TriangleIntersect
[RUN] OK ] TriangleIntersect.Basic (0 ms)
[RUN] OK ] TriangleIntersect.MissCases (0 ms)
[RUN] OK ] TriangleIntersect.TimeWiseRejectionAndClamping (0 ms)
[RUN] OK ] TriangleIntersect.TriangleInXZPlane.Hit (0 ms)
[RUN] OK ] TriangleIntersect.TriangleInXZPlane_Miss (0 ms)
[RUN] OK ] TriangleIntersect.DegenerateTriangle_ReturnsFalse (0 ms)
[RUN] OK ] TriangleIntersect.NearIntersectionWithTriangle (0 ms)
[RUN] OK ] TriangleIntersect.MultiTriangleMesh_HitCorrectTriangle (0 ms)
[=====] 6 tests from TriangleIntersect (12 ms total)

[=====] Global test environment tear-down
[=====] 7 tests from 2 test suites ran. (21 ms total)
[PASSED] 7 tests.
```

In bvh_tree.h, I set up a stop criteria based on parameters CUT-OFF_DEPTH, span_left and span_right and sorted the nodes in [span_left, span_right) according to their centroid's 'dim'-th dimension with function 'std::nth_element'. The result of bvh test is as follows.

```
[=====] Running 3 tests from 1 test suite.
[=====] Global test environment set-up.
[=====] 3 tests from BVH
[RUN] OK ] BVH.BasicConstruction (0 ms)
[RUN] OK ] BVH.SingleObject (0 ms)
[RUN] OK ] BVH.EmptyTree (0 ms)
[=====] 3 tests from BVH (5 ms total)

[=====] Global test environment tear-down
[=====] 3 tests from 1 test suite ran. (10 ms total)
[PASSED] 3 tests.
```

In integrator.cpp, I completed the IntersectionTestIntegrator class, realizing functions like adding offsets for anti_aliasing, updating ray direction, occlude detection and assigning color.

In BSDF.cpp, I modified PerfectRefraction::sample and set the 'interaction.wi' to the direction of the "in-coming light" after refraction or reflection.

3 Results

The result is a program for basic ray tracing, the results are as follows.

