

# Assignment 3:

## Ray Tracing Fundamentals

NAME: WENHAOZHOU  
STUDENT ID: 2022533140  
EMAIL: ZHOUWH2022@SHANGHAITECH.EDU.CN

### 1 INTRODUCTION

This assignment implements the following components:

- [must] Compile the source code and configure the language server environment. [5%]
- [must] Implement ray-triangle intersection functionality. [10%]
- [must] Implement ray-AABB intersection functionality. [10%]
- [must] Implement the BVH (Bounding Volume Hierarchy) construction. [25%]
- [must] Implement the IntersectionTestIntegrator and PerfectRefraction material for basic ray tracing validation, handing refractive and solid surface interactions [25%]
- [must] Implement the DirectLightingIntegrator for direct lighting with diffuse BRDF and shadow testing. [20%]
- [must] Implement anti-aliasing via multi-ray sampling per pixel within a sub-pixel aperture. [5%]

### 2 IMPLEMENTATION DETAILS

#### 2.1 Ray-Triangle Intersection

The ray-triangle intersection is implemented in the `TriangleIntersect` function in `src/accel.cpp`. This function uses geometric methods to compute the intersection point between a ray and a triangle mesh.

```
InternalVecType P = Cast<InternalScalarType>(ray.origin);
InternalScalarType u, v, t;
InternalVecType B_A = v1 - v0;
InternalVecType C_A = v2 - v0;
InternalVecType normal = Cross(B_A, C_A);

InternalScalarType normal_length_sq
    = Dot(normal, normal);
if (normal_length_sq <
    std::numeric_limits<InternalScalarType>::epsilon())
    return false;
}
InternalVecType n = Normalize(normal);
InternalScalarType d = Dot(n, v0);

// t = (d - n · P) / (n · dir)
InternalScalarType n_dot_dir = Dot(n, dir);
if (std::abs(n_dot_dir) <
```

```
std::numeric_limits<InternalScalarType>::epsilon())
    return false; // pingxing
}
InternalScalarType n_dot_P = Dot(n, P);
t = (d - n_dot_P) / n_dot_dir;
if (t < static_cast<InternalScalarType>(ray.t_min) ||
    t > static_cast<InternalScalarType>(ray.t_max))
    return false;
}
InternalVecType Q = P + t*dir;

// inside?
InternalVecType Q_A = Q - v0;
InternalVecType Q_B = Q - v1;
InternalVecType Q_C = Q - v2;
InternalVecType cross1 = Cross(B_A, Q_A);
InternalScalarType test1 = Dot(cross1, n);
if (Dot(Cross(B_A, Q_A), n) < InternalScalarType(0))
    return false;
}
InternalVecType C_B = v2 - v1; // C-B
InternalVecType cross2 = Cross(C_B, Q_B);
InternalScalarType test2 = Dot(cross2, n);
if (test2 < InternalScalarType(0)) {
    return false;
}
InternalVecType A_C = v0 - v2; // A-C
InternalVecType cross3 = Cross(A_C, Q_C);
InternalScalarType test3 = Dot(cross3, n);
if (test3 < InternalScalarType(0)) {
    return false;
}
// barycentric
u = Dot(cross3, n) / Dot(Cross(B_A, C_A), n);
v = Dot(cross1, n) / Dot(Cross(B_A, C_A), n);
if (u < InternalScalarType(0) ||
    v < InternalScalarType(0) ||
    u + v > InternalScalarType(1))
    return false;
}
```

First, compute the intersection with the triangle's plane, then verify the intersection point lies within the triangle.

1:2 • Name: Wenhao Zhou

Student ID: 202253140

Email: zhouwh2022@shanghaitech.edu.cn

## 2.2 Ray-AABB Intersection

```
Vec3f inv_dir = ray.safe_inverse_direction;
int sign[3];
sign[0] = (ray.direction.x < 0);
sign[1] = (ray.direction.y < 0);
sign[2] = (ray.direction.z < 0);
Vec3f bounds[2] = {low_bnd, upper_bnd};
Float tmin, tmax, tymin, tymax, tzmin, tzmax;
tmin = (bounds[sign[0]].x -
         ray.origin.x) * inv_dir.x;
tmax = (bounds[1-sign[0]].x -
         ray.origin.x) * inv_dir.x;
tymin = (bounds[sign[1]].y -
         ray.origin.y) * inv_dir.y;
tymax = (bounds[1-sign[1]].y -
         ray.origin.y) * inv_dir.y;
if ((tmin > tymax) || (tymin > tmax))
    return false;
if (tymin < tmin)
    tmin = tymin;
if (tymax > tmax)
    tmax = tymax;
tzmin = (bounds[sign[2]].z -
         ray.origin.z) * inv_dir.z;
tzmax = (bounds[1-sign[2]].z -
         ray.origin.z) * inv_dir.z;
if ((tmin > tzmax) || (tzmin > tmax))
    return false;
if (tzmin < tmin)
    tmin = tzmin;
if (tzmax > tmax)
    tmax = tzmax;
*t_in = tmin;
*t_out = tmax;
return tmin < tmax &&
       tmax > ray.t_min &&
       tmin < ray.t_max;
```

## 2.3 BVH Construction

```
// Stop criteria for leaf nodes
if (depth >= CUTOFF_DEPTH
    || span_right - span_left == 1) {

}

split = span_left + count / 2;
std::nth_element(
    nodes.begin() + span_left,
    nodes.begin() + split,
```

```
nodes.begin() + span_right,
[dim](const NodeType &a, const NodeType &b) {
    return a.getAABB().getCenter()[dim]
        < b.getAABB().getCenter()[dim];
});
```

## 2.4 Implement a Direct Illumination Integrator

```
//Cast multiple rays per pixel with small offsets for a
const Vec2f &pixel_sample = sampler.getPixelSample();
auto ray = camera->generateDifferentialRay(pixel_sample);

//Cast a shadow ray from the intersection point
//toward the light source to determine visibility.
SurfaceInteraction test;
Ray shadow_ray(interaction.p, light_dir,
                RAY_DEFAULT_MIN, dist_to_light);
if (scene->intersect(shadow_ray, test)) {
    return color;
}
//For each visible intersection,
//compute direct illumination from the light source.
if (bsdf != nullptr && is_ideal_diffuse) {
    Float cos_theta =
        std::max(Dot(light_dir, interaction.normal), 0.0f);
    color = bsdf->evaluate(interaction) * point_light_f
        * cos_theta / (dist_to_light * dist_to_light)
}
```

## 2.5 Integrate with Refractive Materials

```
if (is_perfect_refraction) {
    Float pdf;
    interaction.bsdf->sample(interaction, sampler, &pdf);
    ray = interaction.spawnRay(interaction.wi);
    continue;
}

// Compute diffuse lighting
Vec3f refracted_dir;
if (Refract(interaction.wo, normal, eta_corrected, refraction_ratio)) {
    interaction.wi = refracted_dir;
} else {
    interaction.wi = Reflect(interaction.wo, normal);
}
```

## 2.6 Anti-aliasing via Multi-ray Sampling

### 3 RESULTS

#### 3.1 Intersection Tests

The ray-triangle and ray-AABB intersection implementations were validated using the provided test suite:

```
cmake -B build
cmake --build build
./build/tests/intersection_tests
```

```
PS E:\assignment3-code-template> cmake --build build
[15/15] Linking CXX executable tests\bvh_tests.exe
PS E:\assignment3-code-template> ./build/tests/intersection_tests
Running main() from E:/assignment3-code-template/build/_deps/googletest-src/gtest_main.cc
[=====] Running 7 tests from 2 test suites.
[=====] Global test environment set-up.
[=====] 1 test from AABB
[ RUN ] AABB.AxisAligned.EnterExit_PositiveAndNegativeDirs
[ OK ] AABB.AxisAligned.EnterExit_PositiveAndNegativeDirs (0 ms)
[=====] 1 test from AABBox
[=====] 6 tests from TriangleIntersect
[ RUN ] TriangleIntersect.Basic
[ OK ] TriangleIntersect.Basic (0 ms)
[ RUN ] TriangleIntersect.MissCases
[ OK ] TriangleIntersect.MissCases (0 ms)
[ RUN ] TriangleIntersect.TimeWindow_RejectionAndClamping
[ OK ] TriangleIntersect.TimeWindow_RejectionAndClamping (0 ms)
[ RUN ] TriangleIntersect.TriangleInXZPlane_Hit
[ OK ] TriangleIntersect.TriangleInXZPlane_Hit (0 ms)
[ RUN ] TriangleIntersect.TriangleIntersectionAndSurface
[ OK ] TriangleIntersect.TriangleIntersectionAndSurface (0 ms)
[ RUN ] TriangleIntersect.DominanceTriangle.ReturnFalse (0 ms)
[ RUN ] TriangleIntersect.MultitriangleMesh.HitCorrectTriangle
[ OK ] TriangleIntersect.MultitriangleMesh.HitCorrectTriangle (0 ms)
[=====] 6 tests from TriangleIntersect (10 total)

[=====] Global test environment tear-down
[=====] 7 tests from 2 test suites ran. (17 ms total)
[ PASSED ] 7 tests.
PS E:\assignment3-code-template>
```

#### 3.2 BVH Construction Tests

The BVH construction was tested with the following command:

```
cmake -B build
cmake --build build
./build/tests/bvh_tests
```

```
PS E:\assignment3-code-template> ./build/tests/bvh_tests
Running main() from E:/assignment3-code-template/build/_deps/googletest-src/gtest_main.cc
[=====] Running 3 tests from 1 test suite.
[=====] Global test environment set-up.
[=====] 3 tests from BVH
[ RUN ] BVH.BasicConstruction
[ OK ] BVH.BasicConstruction (0 ms)
[ RUN ] BVH.SingleObject
[ OK ] BVH.SingleObject (0 ms)
[ RUN ] BVH.EmptyTree
[ OK ] BVH.EmptyTree (0 ms)
[=====] 3 tests from BVH (2 ms total)

[=====] Global test environment tear-down
[=====] 3 tests from 1 test suite ran. (5 ms total)
[ PASSED ] 3 tests.
```

### 3.3 Visual Results

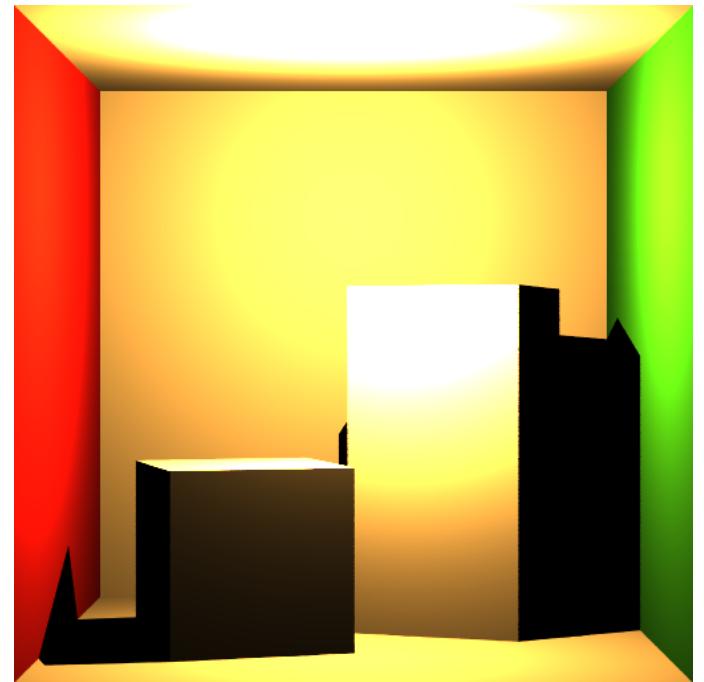


Fig. 1. Cbox No Light Refract

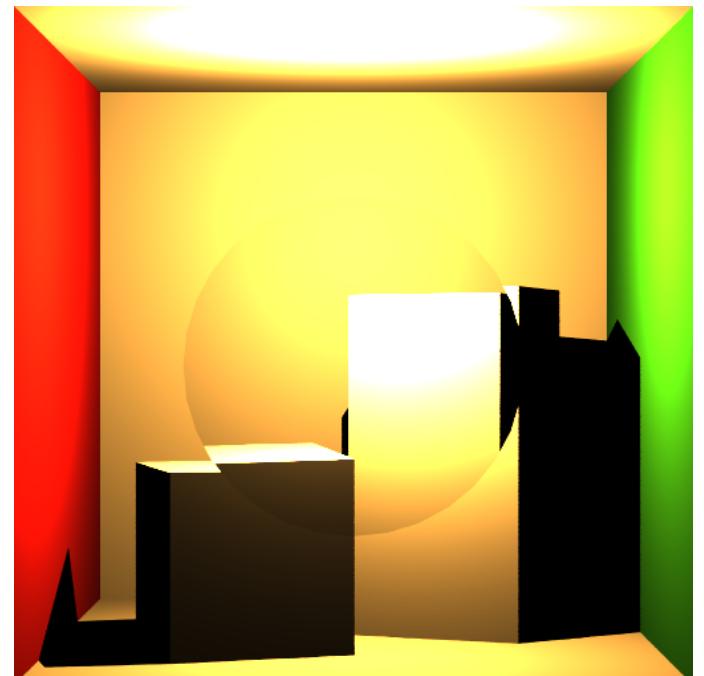


Fig. 2. Cbox No Light Refract