EECS 498-014 Graphics and Generative Models
Homework 2

Yichen Lu nechy@umich.edu

October 8, 2024

1 Task 1: Tracing the Rays

```
Vec3 Scene::trace(const Ray &ray, int bouncesLeft, bool discardEmission) {
    if constexpr(DEBUG) {
        assert (ray.isNormalized());
    }
    if (bouncesLeft < 0) return {};
    Intersection inter = getIntersection(ray);
    if (!inter.happened) return {};

    return inter.getDiffuseColor();
}
```

Listing 1: Scene.cpp

2 Task 2: Direct Illumination

```
Vec3 Scene::trace(const Ray &ray, int bouncesLeft, bool discardEmission) {
    if constexpr(DEBUG) {
        assert (ray.isNormalized());
    }
    if (bouncesLeft < 0) return {};
    Intersection inter = getIntersection(ray);

    if (!inter.happened) return {};

    Vec3 Lo(0.0f, 0.0f, 0.0f);
    if (!discardEmission) {
        Lo += inter.getEmission();
    }

    Vec3 randomDir = Random::randomHemisphereDirection(inter.getNormal());
    Ray randomRay = {inter.pos, randomDir};
    Intersection randominter = getIntersection(randomRay);

    Vec3 Li = Vec3(0.0f, 0.0f, 0.0f);
    if (randominter.happened) {
        float pdf = 1.0f / (2.0f * PI);
        Vec3 brdf = inter.calcBRDF(-randomDir, -ray.dir);
        float cosineTerm = randomDir.dot(inter.getNormal());
        Li += 1 / pdf * randominter.getEmission() * brdf * cosineTerm;
    }

    return Lo + Li;
}
```

Listing 2: Scene.cpp

2.1 Thought question : why does the image of 16 SPP look darker than 128 SPP?

Using 16 samples per pixel, means fewer rays being traced, fewer light paths are being sampled. Therefore, many light sources and surface interactions may be missed. So it's darker overall.

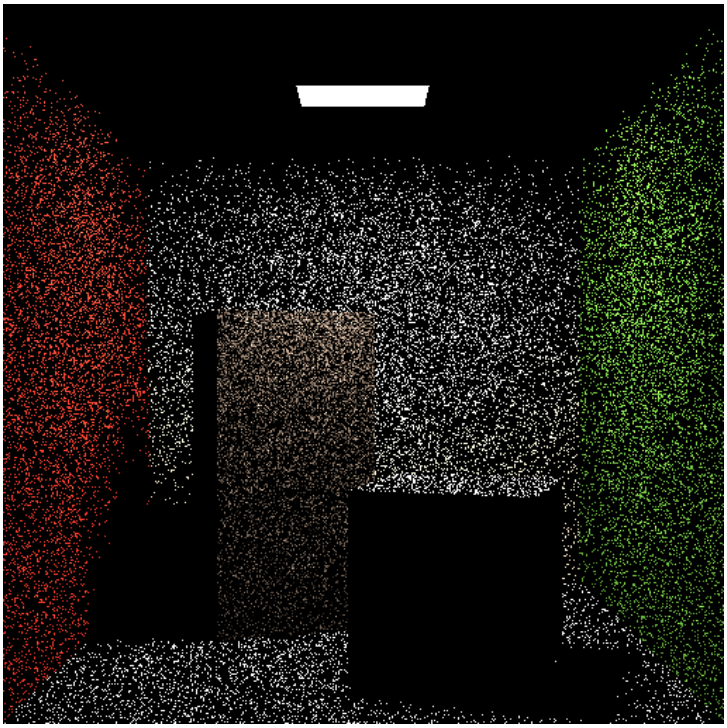

Figure 1: Task6 SPP = 32

3 Task 3: Global Illumination

```
Vec3 Scene::trace(const Ray &ray, int bouncesLeft, bool discardEmission) {
    if constexpr(DEBUG) {
        assert (ray.isNormalized());
    }
    if (bouncesLeft < 0) return {};
    Intersection inter = getIntersection(ray);

    if (!inter.happened) return {};

    Vec3 randomDir = Random::randomHemisphereDirection(inter.getNormal());
    Ray nextRay(inter.pos, randomDir);
    Intersection randominter = getIntersection(nextRay);

    Vec3 Lo(0.0f, 0.0f, 0.0f);
    if (!discardEmission) {
        Lo += inter.getEmission();
    }

    if (randominter.happened) {
        float pdf = 1.0f / (2.0f * PI);
        Vec3 brdf = inter.calcBRDF(-randomDir, -ray.dir);
        float cosineTerm = randomDir.dot(inter.getNormal());

        Vec3 Li = trace(nextRay, bouncesLeft - 1, false);
        Lo += 1 / pdf * Li * brdf * cosineTerm;
    }

    return Lo;
}
```

Listing 3: Scene.cpp

3.1 Thought question: compare the results to direct illumination only, what new features can you identify, and why do we have them? For example, how does the ceiling get illuminated?

Ceilings and shadowed areas that are indirectly lighted by light sources. This is because indirect lighting occurs when light bounces off surfaces and indirectly illuminates other parts of the scene. In the case of the ceiling, it gets illuminated because

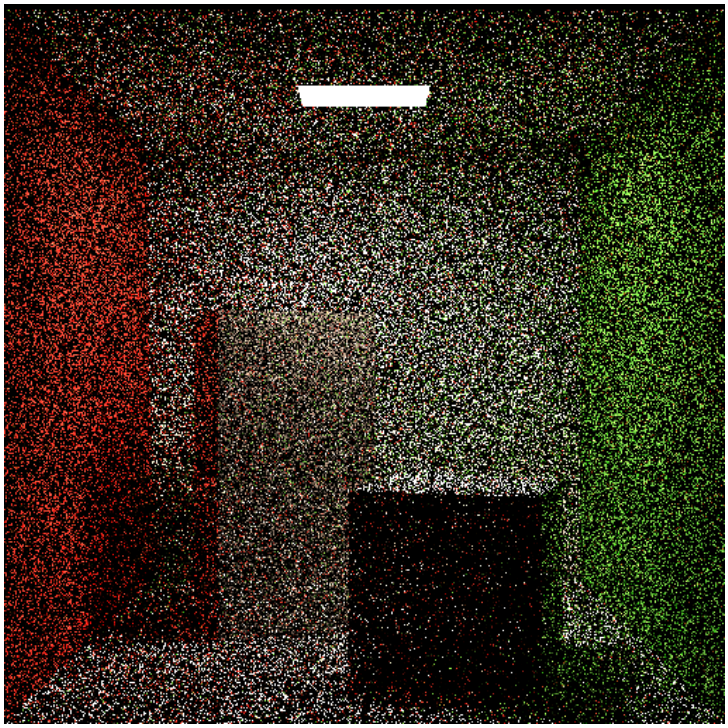

Figure 2: Task7 SPP = 32 Depth = 8

light bounces off the surfaces like the boxes and floor.

4 Task 4: Acceleration

```
Vec3 Scene::trace(const Ray &ray, int bouncesLeft, bool discardEmission) {
    if constexpr(DEBUG) {
        assert (ray.isNormalized());
    }
    if (bouncesLeft < 0) return {};
    Intersection inter = getIntersection(ray);

    if (!inter.happened) return {};

    Vec3 Lo(0.0f, 0.0f, 0.0f);
    if (!discardEmission) {
        Lo += inter.getEmission();
    }

    Vec3 indirectRadiance(0.0f, 0.0f, 0.0f);

    Vec3 randomDir = Random::cosWeightedHemisphere(inter.getNormal());
    Ray randomRay(inter.pos, randomDir);
    Intersection bounceInter = getIntersection(randomRay);

    if (bounceInter.happened) {
        Vec3 Li = trace(randomRay, bouncesLeft - 1, true);
        Vec3 brdf = inter.calcBRDF(-randomDir, -ray.dir);
        float cosineTerm = randomDir.dot(inter.getNormal());
        float pdf = cosineTerm / PI;
        indirectRadiance = (Li * brdf * cosineTerm) / pdf;
    }

    Vec3 directRadiance(0.0f, 0.0f, 0.0f);

    Intersection lightSample = sampleLight();
    Vec3 lightDir = lightSample.pos - inter.pos;
    float distanceToLight = lightDir.getLength();
    lightDir.normalize();

    Ray rayToLight = {inter.pos, lightDir};
    Intersection lightInter = getIntersection(rayToLight);
```

```
38
39      if (lightInter.happened && (lightInter.object == lightSample.object)) {
40
41          Vec3 brdf = inter.calcBRDF(-lightDir, -ray.dir);
42          float cosineTerm = lightDir.dot(inter.getNormal());
43          float lightCosineTerm = -lightDir.dot(lightSample.getNormal());
44
45          float pdfLightSample = 1.0f / lightArea;
46          float attenuation = 1.0f / (distanceToLight * distanceToLight);
47
48          directRadiance = (lightSample.getEmission() * brdf * cosineTerm * lightCosineTerm * attenuation) /
     pdfLightSample;
49      }
50
51      Lo += indirectRadiance + directRadiance;
52
53      return Lo;
54  }
```

Listing 4: Scene.cpp

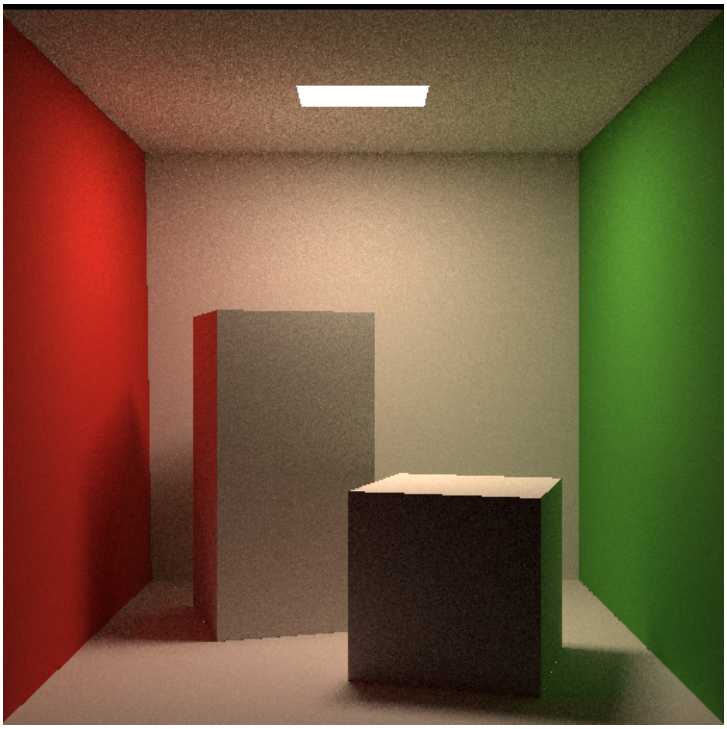

Figure 3: Task8 SPP = 64 Depth = 8