vcl final project report

Xinzhe Wu

2026.1.10

1 选题:C Rendering - 5. Path Tracing 基于渲染方程的全局真实感渲染

基于 [https://www.kevinbeason.com/smallpt/]smallpt 实现的 path tracing 算法相比于 lab3 中的 whitted-style ray tracing 算法，其主要优化在于 1. 对于点光源，看成球状光源利用均匀分布多次采样 2. 对于反射过程进行蒙特卡洛采样 3. 采用 Russian roulette ，以一定概率提前终止 trace，最终可以实现更加真实的渲染结果以及更柔和的阴影边缘

2 算法实现

```
1  glm::vec3 PathTrace(const RayIntersector & intersector, Ray ray, int maxDepth, int pixel_samples, int seed) {
2      glm::vec3 color(0.0f);
3      glm::vec3 throughput(1.0f);
4      const float color_rate = 2.5f;
5      const float ambient_rate = 4.5f;
6      const int samples    = 8;
7      const float light_radius = 10.0f;
8
9      std::mt19937                    gen(seed);
10     std::uniform_real_distribution<float> dis(0.0f, 1.0f); // 初始化结果，参数以及随机数生成器
11
12     for (int depth = 0; depth < maxDepth; ++depth) { // 递归深度
13         auto rayHit = intersector.IntersectRay(ray);
14         if (! rayHit.IntersectState) {
15             color += throughput * glm::vec3(0.0f, 0.0f, 0.0f);
16             break; // 未相交则中止
17         }
18
19         const glm::vec3 pos        = rayHit.IntersectPosition;
20         const glm::vec3 nl         = glm::normalize(rayHit.IntersectNormal);
21         const glm::vec3 kd         = glm::vec3(rayHit.IntersectAlbedo);
22         const glm::vec3 ks         = glm::vec3(rayHit.IntersectMetaSpec);
23         const float   shininess    = rayHit.IntersectMetaSpec.w * 256.0f;
24         const glm::vec3 n          = glm::dot(ray.Direction, nl) < 0 ? nl : -nl;
25         const float   specular_weight = glm::length(ks);
26         const float   diffuse_weight = glm::length(kd);
27         for (const auto & light : intersector.InternalScene->Lights) {
28             glm::vec3 light_total = glm::vec3(0.0f);
29             for (int s = 0; s < samples; s++) { //每条光线采样多次
30                 glm::vec3 l;
31                 float    attenuation  = 1.0f;
32                 glm::vec3 sampledLightPos = light.Position;
33                 if (light.Type == Engine::LightType::Point) { //点光源从单位球中采样offset
34                     float    u1  = dis(gen);
35                     float    u2  = dis(gen);
36                     float    z   = 1.0f - 2.0f * u1;
37                     float    r_xy = sqrt(1.0f - z * z);
38                     float    phi = 2.0f * glm::pi<float>() * u2;
39                     glm::vec3 offset(r_xy * cos(phi), r_xy * sin(phi), z);
40                     offset *= light_radius;
41                     sampledLightPos = light.Position + offset;
42                     l             = sampledLightPos - pos;
43                     attenuation   = 1.0f / glm::dot(l, l);
44                 } else if (light.Type == Engine::LightType::Directional) {
45                     l             = light.Direction;
```

```cpp
46                   sampledLightPos = pos + l * 1000.0f;
47               }
48               l = glm::normalize(l);
49
50               // Shadow ray
51               Ray shadowRay(pos + n * EPS1, l); //计算shadow ray和 color 同 rat tyracing 一样
52               auto shadowHit = intersector.IntersectRay(shadowRay);
53               if (! shadowHit.IntersectState || glm::distance(pos, shadowHit.IntersectPosition) > glm::distance(pos, sampledLightPos)) {
54                   float     cosTheta     = glm::max(glm::dot(n, l), 0.0f);
55                   glm::vec3 diffuse_color = kd * cosTheta * attenuation * light.Intensity * color_rate;
56                   glm::vec3 norm_eye    = glm::normalize(-ray.Direction);
57                   glm::vec3 norm_half   = glm::normalize(l + norm_eye);
58                   float     specular_angle = glm::max(glm::dot(n, norm_half), 0.0f);
59                   glm::vec3 specular_color = ks * pow(specular_angle, shininess) * attenuation * light.Intensity * color_rate;
60                   light_total += diffuse_color + specular_color;
61               }
62           }
63           light_total /= float(samples);
64           color += throughput * light_total;
65       }
66
67       // 加载环境光
68       color += throughput * kd * intersector.InternalScene->AmbientIntensity * ambient_rate;
69
70       // 蒙特卡洛采样利用均匀分布模拟以一定概率发生镜面反射，否则发生漫反射，漫反射从以法向为中心的半球进行余弦加权采样
71       glm::vec3 newDir;
72       glm::vec3 bsdf_contribution;
73       float    r_brdf = dis(gen); %   修复: 中文分号 → 英文分号，避免编译报错
74       if (r_brdf < specular_weight && specular_weight > 0.01f) {//镜面反射
75           newDir         = glm::reflect(ray.Direction, n);
76           newDir         = glm::normalize(newDir);
77           bsdf_contribution = ks;
78
79           throughput *= bsdf_contribution / specular_weight;
80       } else { // 漫反射半球余弦加权采样
81           float r1     = dis(gen);
82           float r2     = dis(gen);
83           float phi    = 2.0f * glm::pi<float>() * r1;
84           float cosTheta = sqrt(r2);
85           float sinTheta = sqrt(1.0f - r2);
86
87           glm::vec3 tangent = glm::normalize(glm::cross(n, glm::vec3(0, 1, 0)));
88           if (glm::length(tangent) < 0.1f) tangent = glm::normalize(glm::cross(n, glm::vec3(1, 0, 0)));
89           glm::vec3 bitangent = glm::cross(n, tangent);
90
91           newDir         = cosTheta * n + sinTheta * (cos(phi) * tangent + sin(phi) * bitangent);
92           newDir         = glm::normalize(newDir);
93           bsdf_contribution = kd;
94
95           throughput *= bsdf_contribution / (1.0f - specular_weight);
96       }
97
98       ray = Ray(pos + n * EPS1, newDir);
99
100      // Russian roulette 以一定的概率提前停止
101      float p = glm::max(throughput.r, glm::max(throughput.g, throughput.b));
102      p     = glm::max(p, 0.15f);
103      if (depth > 3 && dis(gen) > p) break;
104      if (depth > 3) throughput /= p;
105     }
106     return color;
107  }
```
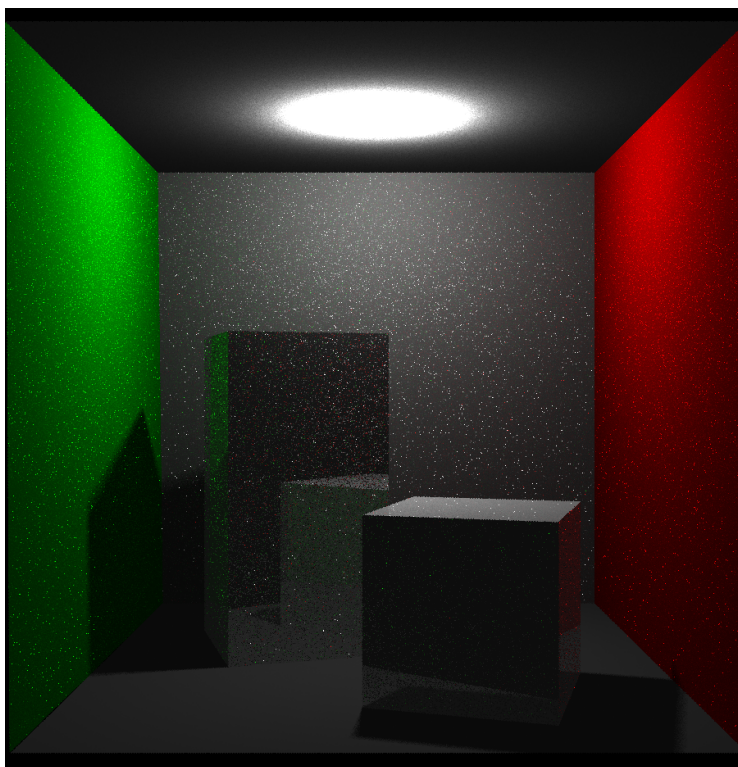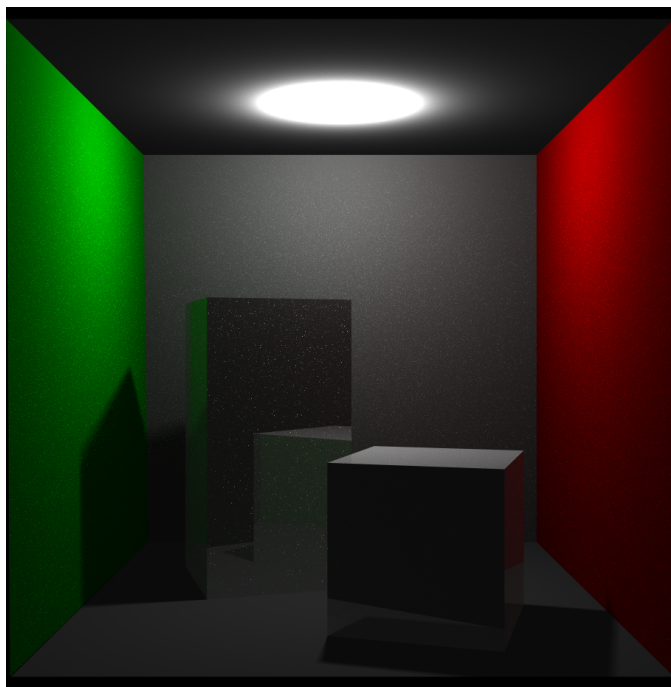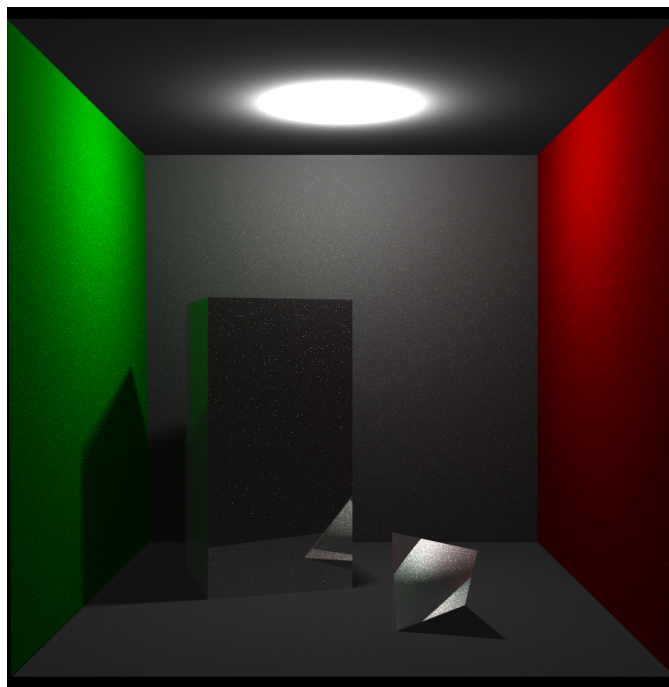
图 1: pathtracing 的噪点（sample=1，depth=3）

3 结果分析

1. 使用我实现的 path tracing 算法相比于 whitted-style 渲染真实感更明显，相比于 whitted 的锐利阴影阴影更加柔和

2. 在 sample pixels 较小时，图像会有较多的噪点 (图一)，这是因为采样数太少导致导致算法中的随机过程扰动明显 (均匀分布可进一步优化)

3.path tracing 得到的结果相比于 whitted-style 会暗一些，这可能是因为，漫反射余弦采样导致的能量各方向损耗，Russian roulette 的提前中止

在最后一页附上 path tracing（sample=70,depth=12）和 whitted-style ray tracing 的效果对比

4 结果对比展示
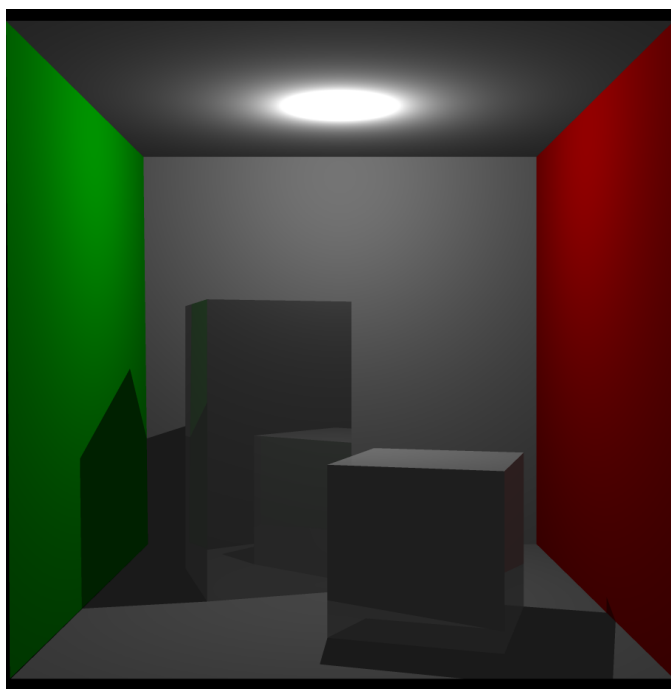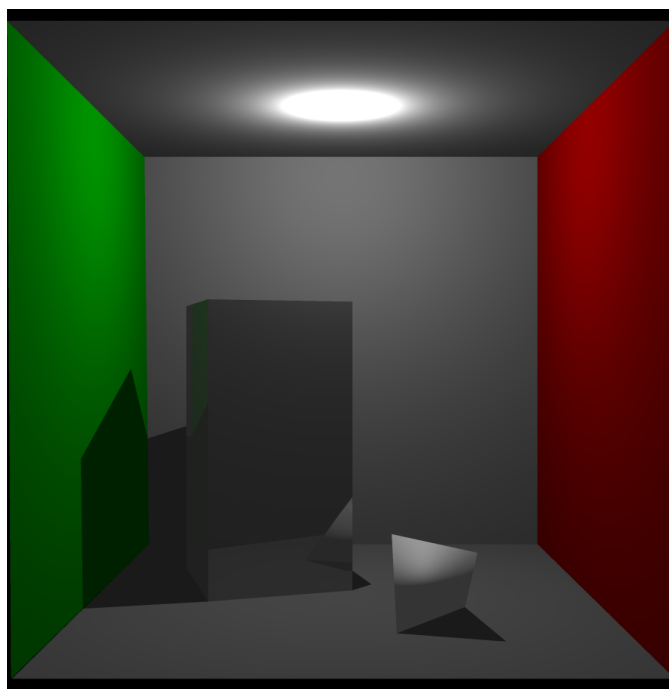

(a) pathtracing

(b) pathtracing

(c) whitted-style

(d) whitted-style

图 2: pathtracing 和 whitted-style 对比