Document

开发环境

- 操作系统：Windows 11 x64
- 处理器：12th Gen Intel(R) Core(TM) i9-12900H　2.50 GHz
- IDE：Microsoft Visual Studio 2019 x64
- 运行环境：Release X64
- 依赖库：glm（向量运算），Eigen（高性能数学运算），opencv（texture贴图），tinyxml2（xml文件读取）

使用说明

数据结构

Camera

相机类，存储相机信息。成员函数可通过相机信息设定机位，通过图片像素位置获取视线，以及进行相机信息的打印。

```
class Camera
{
public:
    Camera() {}
    ~Camera() {}

    void setCamera();// set camera
    Ray getRay(float u, float v);// get ray at (u,v)
    void Print();// print info for debugging

    double fovy = 90;
    vec3 eye = vec3(278.0, 273.0, -800.0);
    vec3 lookat = vec3(278.0, 273.0, -799.0);
    vec3 up = vec3(0.0, 1.0, 0.0);
    double aspect_ratio = 1.0;

    vec3 lower_left_corner = vec3(0.0, 0.0, 0.0);
    vec3 horizontal = vec3(0.0, 0.0, 0.0);
    vec3 vertical = vec3(0.0, 0.0, 0.0);
};
```

Material

Material类存储mtl读入的材质信息，`readinMap()`用于读入纹理地址并存储纹理数据

```
class Material
{
public:
    Material() {}
    ~Material() {}
    void readinMap(); //for texture

    vec3 Kd = vec3(0.0, 0.0, 0.0); // diffuse
    vec3 Ks = vec3(0.0, 0.0, 0.0); // specular
    vec3 Tr = vec3(0.0, 0.0, 0.0); // transmittance
    float Ns = 1;                  // shiness, the exponent of phong lobe
    float Ni = 1;                  // the Index of Refraction (IOR) of transparent
object
    string map_Kd = "";            // map_Kd

    bool is_emissive = false;
    vec3 radiance = vec3(0, 0, 0);
    double area = 0.0;

    vector<Triangle> triangles; // for light sampling

    cv::Mat img;// for texture
    int map_height, map_width;
};
```

Light

Light类是为了方便光源采样时遍历光源。

Light类

```
class Light
{
public:
    Light() {}
    Light(string m, vec3 r) : mtl_name(m), radiance(r) {}
    ~Light() {}

    string mtl_name = "";
    vec3 radiance = vec3(0.0, 0.0, 0.0);
};
```

Ray

ray记录了一条射线的起点、终点和类型。

```
const int DIFFUSE      = 0;
const int SPECULAR     = 1;
const int TRANSMISSION = 2;
const int INVALID = 3;

class Ray
{
public:
    Ray() {}
    Ray(vec3 s, vec3 d) :startpoint(s), direction(d) {}
    Ray(vec3 s, vec3 d, int r) :startpoint(s), direction(d), ray_type(r) {}
    ~Ray() {}

    vec3 startpoint = vec3(0.0, 0.0, 0.0);
    vec3 direction = vec3(0.0, 0.0, 0.0);
    int ray_type = INVALID;
};
```

Triangle

存储三角面片信息。Triangle类的成员函数可以计算三角形面积和取重心坐标。

```
class Triangle
{
public:
    double calAera();            // calulate area of the triangle
    vec3 findBaryCor(vec3 hitp); // fing the barycenter coordinate of the hitpoint

    Triangle() {}

    vec3 v[3];                        // vertex
    vec3 vn[3];                       // vertex normal
    vec2 vt[3];                       // vertex texture
    vec3 normal = vec3(0.0, 0.0, 0.0); // for caculating hitpoints and distances
    vec3 center = vec3(0.0, 0.0, 0.0); // for sorting
    double area = 0.0;                // for light sampling
    std::string mtl_name = "";
    bool is_emissive = false; // choose Emissive triangle when they are overlapping
    // int id = 0; // for debug
};
```

Scene

Scene类存储场景内的Material、Triangle、Light、Camera信息，成员函数对obj/xml/mtl文件进行读取。

```
class Scene
{
public:
    Scene(/* args */) {}
    ~Scene() {}

    void readxml(string xml_path);                  // get camera & lights
    void readmtl(string mtl_path, string base_dir); // get materials
    void readobj(string obj_path);                  // get triangles

    int img_width;
    int img_height;

    vector<Triangle> triangles; // obj models
    vector<Light> lights;
    unordered_map<string, Material> materials;
    Camera camera;
};
```

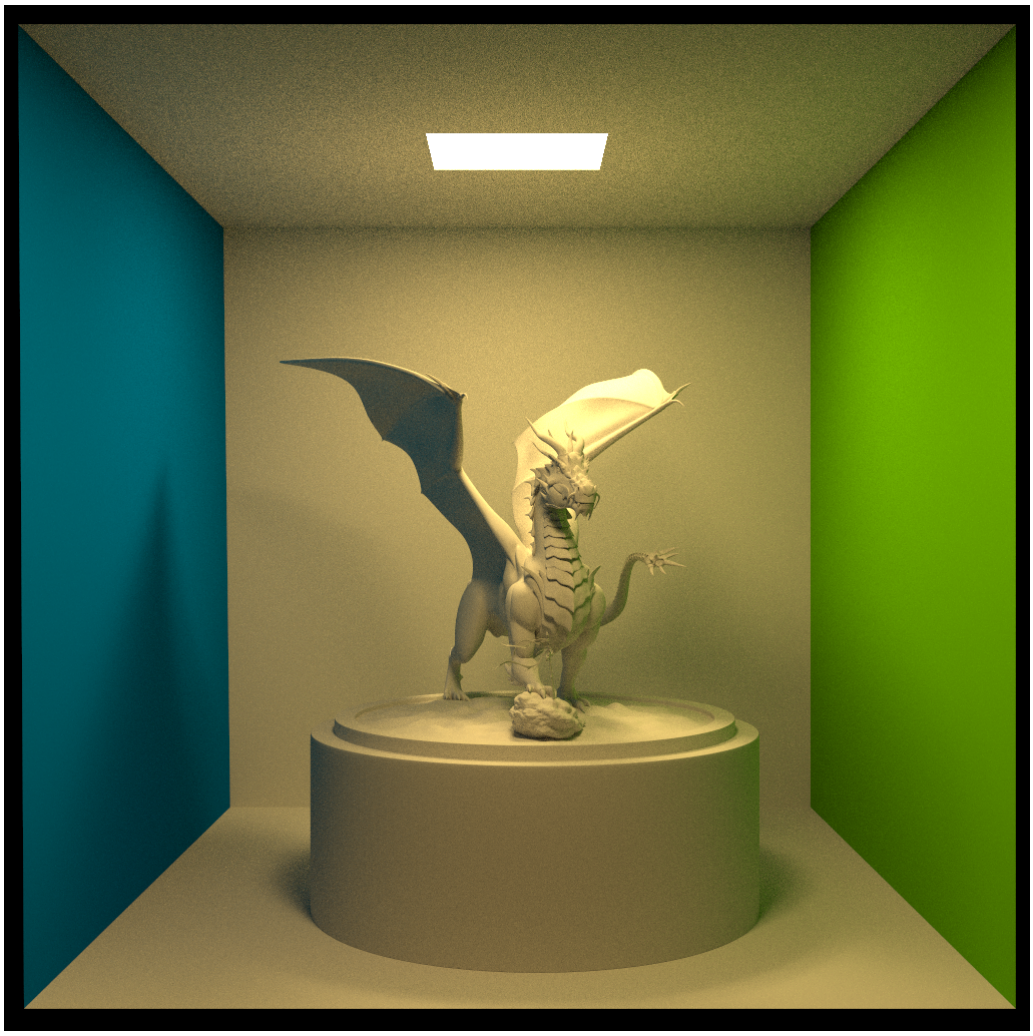
算法实现

- 蒙特卡洛积分
- 加速结构：基于SAH构建的BVH树
- 采样：bdrf重要性采样

测试结果

cornell-box

256spp, 72973.8s ≈ 20.3h (跑的时候忘了设置电脑不休眠了，时间可能有点问题)



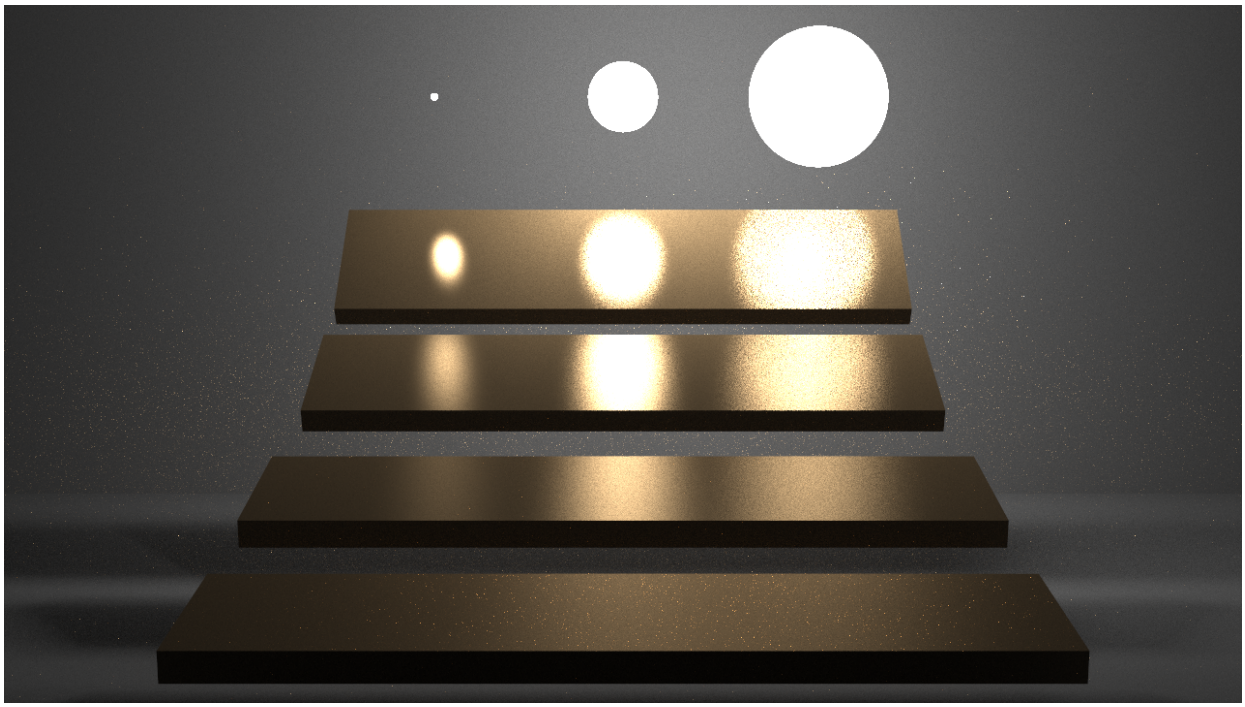
staircase

256spp, 6088.5s ≈ 1.7h

分析：天花板光和玻璃反射不对，整体场景偏亮。调了很久，分析这个场景和另外两个场景相比有更多光源，并且每种材质的光源不止有一处光源，所有点光源采样可能存在问题，导致漫反射和镜面反射的强度与方向与参考答案有较大差距。

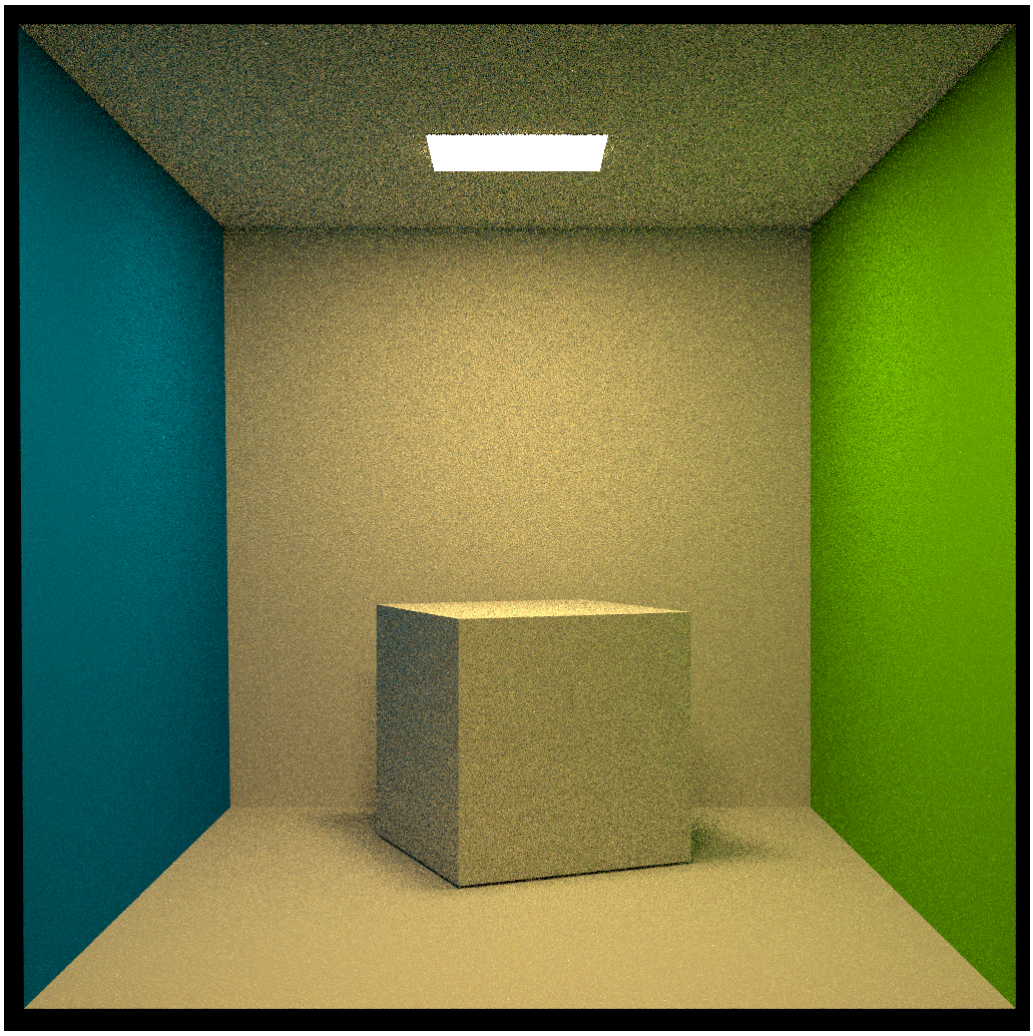
veach-mis

256spp, 268.1s ≈ 4.5min



own scene

10spp，主要是测试用的



参考

[1] *Ray Tracing in One Weekend*

[2] *光线追踪渲染实战（二）：BVH 加速遍历结构*

[3] *tinyxml2读写XML文件的例程*

[4] *蒙特卡洛路径跟踪 C++实现*

[4] Lafortune E P, Willems Y D. Using the modified phong reflectance model for physically based rendering[M]. Katholieke Universiteit Leuven. Departement Computerwetenschappen, 1994.

[5] *GAMES101: 现代计算机图形学入门*