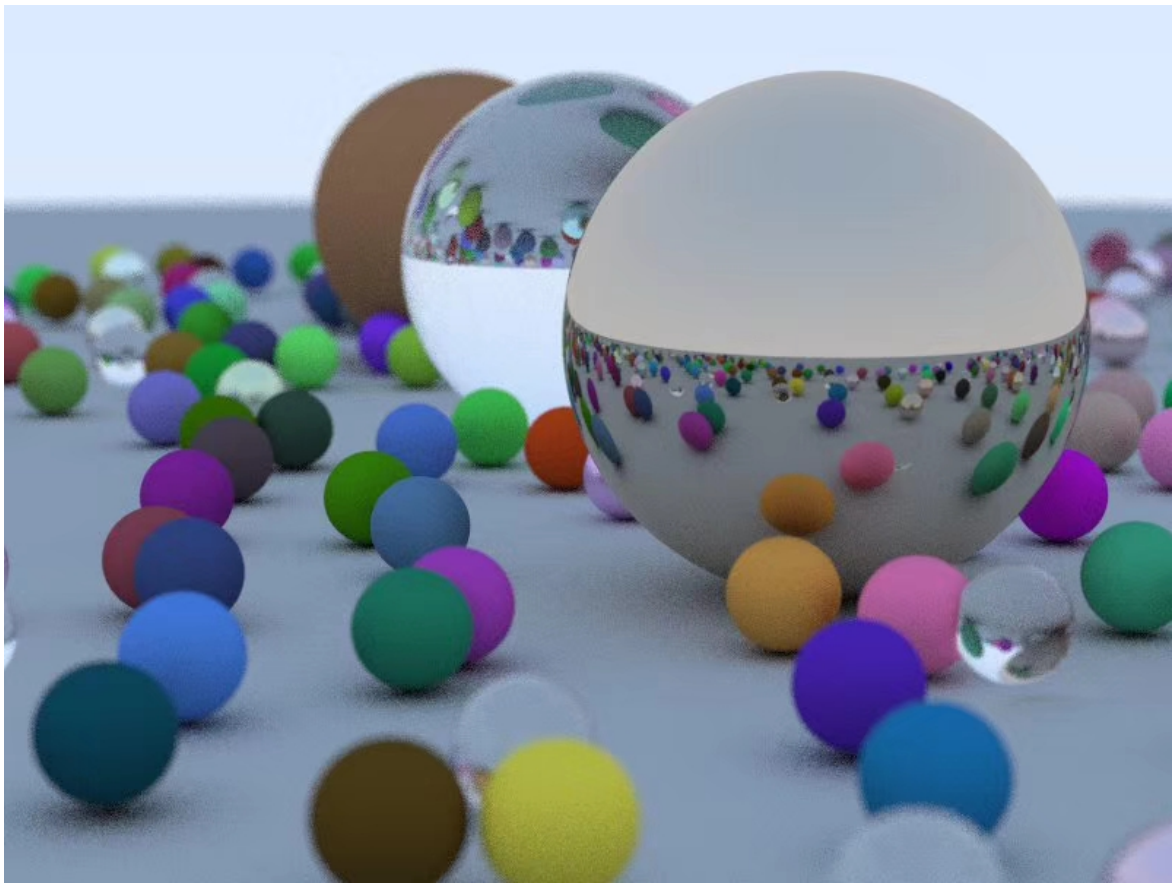
Assignment 5: Ray tracing

Computer Graphics Teaching Stuff, Sun Yat-Sen University

- Due Date: 11月15号晚上12点之前，提交到zhangzk3@mail2.sysu.edu.cn邮箱。

> 在完成了前几次的作业之后，相信同学们对基于光栅化的渲染方法和管线有了比较深入的了解。但光栅化的渲染管线仅在实时渲染领域比较火热，在影视特效领域，光线追踪才是主流。与光栅化方法不同，光线追踪算法模拟真实光线的反射、折射过程，能够实现极度真实的渲染效果。此次作业的任务你们要实现的是一个简单的光线追踪渲染器，我们有提供详细的教程一步一步带你实现这个渲染器。做完这个作业，你将深刻领会到代码和算法的艺术性，你们得到的不再是乌漆抹黑的黑框，不再是枯燥的一堆数据，而是一张非常漂亮的图片。

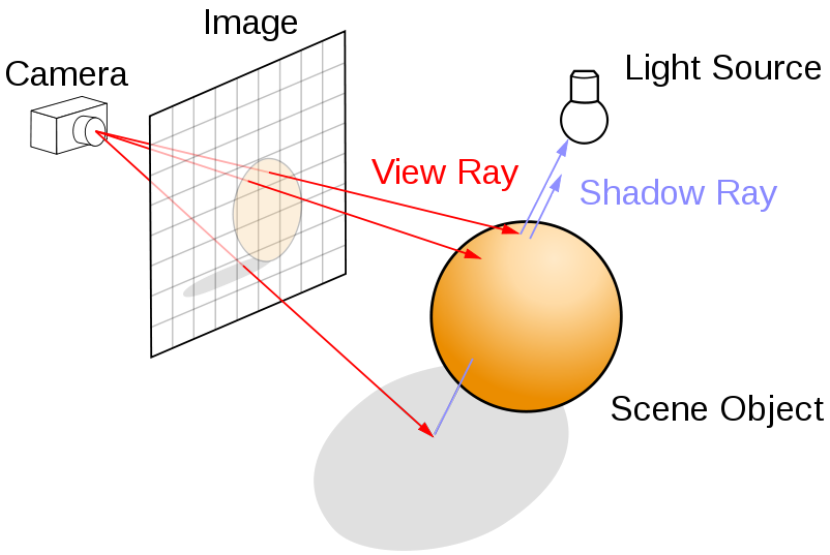


1、总览

光线追踪 (Ray Tracing) 算法是一种基于真实光路模拟的计算机三维图形渲染算法，相比其它大部分渲染算法，光线追踪算法可以提供更为真实的光影效果。此算法由 Appel 在 1968 年初步提出，1980 年由Whitted 改良为递归算法并提出全局光照模型。直到今天，光线追踪算法仍是图形学的热点，大量的改进在不断涌现。基于对自然界光路的研究, 光线追踪采取逆向计算光路来还原真实颜色。追踪的过程中涵盖了光的反射、折射、吸收等特性 (精确计算)， 并辅以其它重要渲染思想 (进一步模拟)。 其中包含了重要方法，诸如冯氏光照模型 (Phong Shading)、辐射度(Radiosity)、光子映射 (Photon Mapping)、蒙特卡罗方法 (Monte Carlo) 等等。**鉴于光线追踪算法对场景仿真程度之高，其被普遍认为是计算机图形学的核心内容， 以及游戏设计、电影特效等相关领域的未来方向。** 近年来由于硬件系统

的迅速改良， 基于分布式、GPU， 甚至实时渲染的光线追踪显卡（例如英伟达的RTX系列显卡）也纷纷出现。我们平时在影视中看到的特效，几乎都是采用光线追踪渲染得到的。

光线追踪算法是一种非常自然的技术，相比于光栅化的方法，它更加简单、暴力、真实。与光栅化根据物体计算所在的像素的方式不同，光线路径追踪的方法是一个相反的过程，它在于用眼睛去看世界而不是世界如何到达眼中。如下图所示，从视点出发向屏幕上每一个像素发出一条光线，追踪此光路并计算其逆向光线的方向，映射到对应的像素上。通过计算光路上颜色衰减和叠加，即可基本确定每一个像素的颜色。



光线追踪是一个递归的过程。发射一束光线到场景，求出光线和几何图形间最近的交点，如果该交点的材质是反射性或折射性的，可以在该交点向反射方向或折射方向继续追踪，如此递归下去，直到设定的最大递归深度或者射线追踪到光源处（或者背景色），如此便计算处一个像素的着色值。

在本次作业中，我们提供了一个教程文件（即Ray Tracing in a Weekend.pdf），这是一个外国人写的非常好的Ray tracing入门资料。该教程深入浅出，一步一步地叫你如何写一个光线追踪渲染器，跟着这个教程你将能够深入理解光线追踪算法的基本原理，并渲染出非常漂亮的图片。该pdf文件虽然有41页，但大部分都是代码，同学们不必担心，教程讲解简单直白，相信同学们都能够得到不少的收获。请同学们阅读教程并独立完成每个步骤，有任何问题或者遇到什么奇怪的bug可在群里相互讨论。

2、代码框架

教程的Chapter 1是让大家学会如何输出图片，而Chapter 2是让大家写一个简单的三维向量库，我们提供的代码框架已经完成这两部分的内容，大家阅读的教程的时候不用再去实现了。我们提供的框架代码有以下的几个文件：

- `stb_image_write.h`：一个简单的图片保存库，提供将给定的像素矩阵保存成图片的功能，不用太过深究；
- `vec3.h`：一个简单的三维向量库，跟教程Chapter 2的无差别，基本上不用修改；
- `PathTracer.h`和`PathTracer.cpp`：光线追踪器的类文件，在这个类中你将实现相应的光线追踪算法；
- `main.cpp`：程序入口，调用光线追踪器并将渲染得到的像素矩阵保存成图片，基本上不用修改。

这里重点介绍一下`PathTracer`类，在这个类中，我们声明了一个一维数组`m_image`，用以保存像素，图片的像素默认是RGBA格式（即四个通道），每个通道占据一个字节，也就是一个`unsigned char`大小。渲染得到的像素将保存到这个像素数组当中：

```
class PathTracer
{
```

```
private:
    // RGBA format.
    int m_channel;
    // image's size.
    int m_width, m_height;
    // image's pixel buffer.
    unsigned char *m_image;

public:
    // Ctor/Dtor.
    PathTracer();
    ~PathTracer();

    // Getter.
    int getWidth() const { return m_width; }
    int getHeight() const { return m_height; }
    int getChanne() const { return m_channel; }
    unsigned char *getImage() const { return m_image; }

    // Must call it for the first of all.
    void initialize(int width, int height);

    // Render a frame.
    unsigned char *render(double &timeConsuming);

private:
    // Draw one pixel in (y,x).
    void drawPixel(unsigned int x, unsigned int y, const vec3 &color);
};
```

`PathTracer`类的几个重点方法介绍如下所示，更多的具体细节请看源代码：

- `initialize`函数：根据给定的宽高，分配相应大小的像素数组；
- `drawPixel`函数：将输入的颜色`color`写入到图片`(x,y)`处，因为用的是一维数组，因此需要对其下标变换一下；
- `render`函数：在该函数中实现光线追踪算法，一个一个像素地渲染并保存到像素数组中，返回最终渲染得到的像素数组。

相应地，在`main`中我们创建一个`PathTracer`对象，初始化后调用渲染函数进行渲染，最后将渲染的结果保存到`result.png`文件当中，这里基本上不需要你修改什么：

```
int main(int argc, char *argv[])
{
    // initialization.
    PathTracer tracer;
    tracer.initialize(800, 600);

    double timeConsuming = 0.0f;
    // rendering.
    unsigned char *pixels = tracer.render(timeConsuming);

    // save pixels to .png file using stb_image.
    stbi_flip_vertically_on_write(1);
    stbi_write_png("./result.png",
        tracer.getWidth(),
        tracer.getHeight(),
        4,
```

```
        static_cast<void*>(tracer.getImage()),
        tracer.getWidth() * 4);

    std::cout << "Rendering Finished.\n";
    std::cout << "Time consuming: " << timeConsuming << " secs.\n";

    return 0;
}
```

3、编译

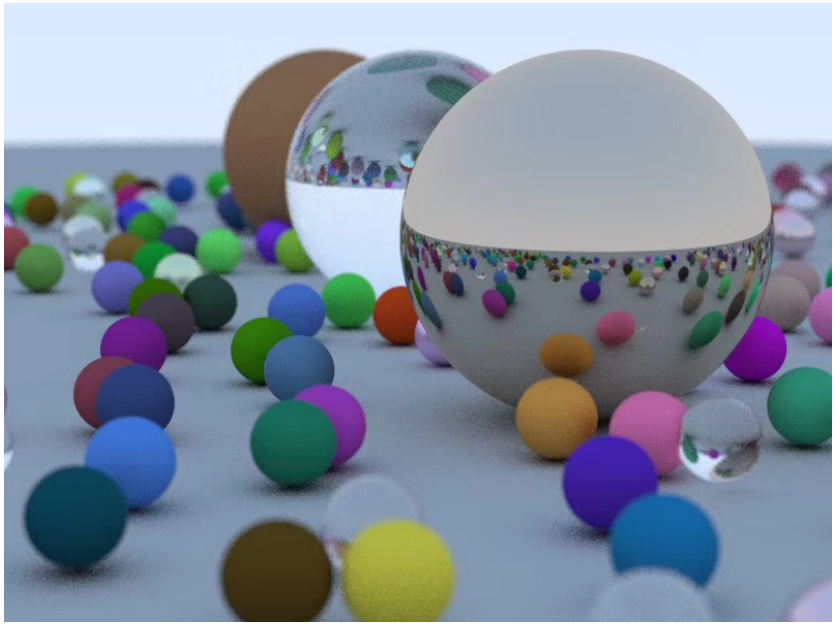
这次我们提供的代码并不提倡在虚拟机上运行，因为光线追踪算法比较耗时，所以希望同学们拿到框架代码之后在自己常用的IDE上构建项目（我们没有提供Makefile文件）。我们提供的框架代码并不依赖于任何外部库，所以只需将提供的所有代码文件添加到项目中，然后运行即可。不出意外运行成功的话，在代码的同级目录下你应该能够得到一个`result.png`文件，打开这个图片文件你看到是下面这样的：

编写代码的平台不限，取决于自身，因为我们提供的代码框架理论上不依赖于任何平台。如果构建项目出现了任何问题，请及时联系助教。构建成功之后，**这里稍稍建议一下把构建模式由debug模式改成release模式。通常情况下，debug模式要慢很多，因此在不debug的时候，建议调成release模式（特别是在后面部分）。**

到后面可能渲染一张图片需要的时间多一点（几分钟），请耐心等待！

4、注意事项

最后需要渲染如下的场景：



该场景对应的摄像机参数为：

```
int nx = 1200;
int ny = 800;
int ns = 10;
std::cout << "P3\n" << nx << " " << ny << "\n255\n";
hitable *world = random_scene();

vec3 lookfrom(13,2,3);
vec3 lookat(0,0,0);
float dist_to_focus = 10.0;
float aperture = 0.1;

camera cam(lookfrom, lookat, vec3(0,1,0), 20, float(nx)/float(ny), aperture, dist_to_focus);
```

5、作业描述与提交

（1）、作业提交

将PDF报告文件和代码压缩打包提交到zhangzk3@mail2.sysu.edu.cn邮箱，邮件标题和压缩包命名格式为：hw5_姓名_学号。

（2）、作业描述

本次作业要求同学们完成的工作如下所示：

Task 1、跟着教程实现Chapter 2 ~ Chapter 6的内容。（40分）

将遇到的问题以及是如何解决的写出来，并贴上你实现的效果。

Task 2、跟着教程实现Chapter 7~ Chapter 12的内容。（50分）

将遇到的问题以及是如何解决的写出来，并贴上你实现的效果。

Task 3、跟着教程实现了一个精简的光线追踪渲染器，谈谈你的感想和收获。（10分）