

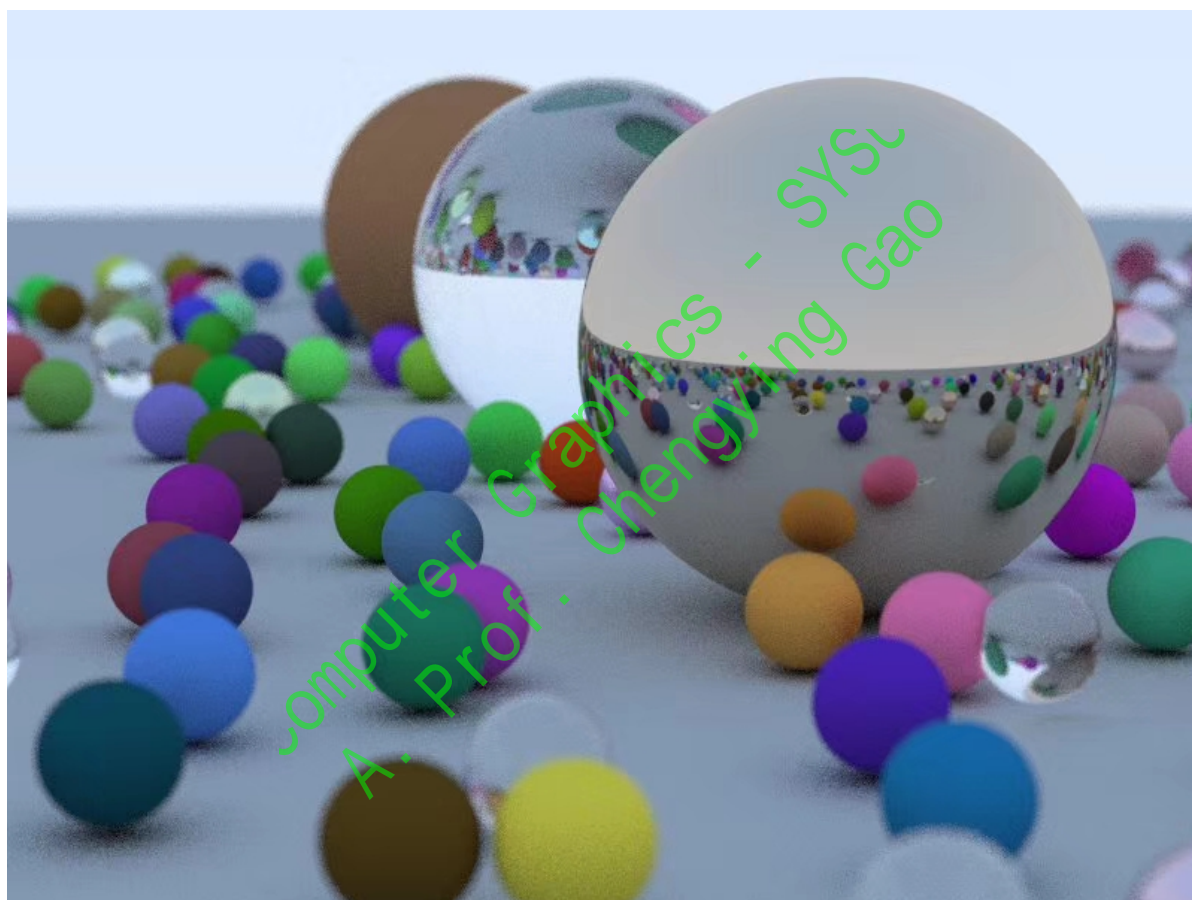
Assignment 4: Ray Tracing

Computer Graphics Teaching Stuff, Sun Yat-Sen University

Due Date: 具体截止日期见群公告

Submission: Send the report (In **PDF** Format) to mailbox (邮箱地址见群公告)

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。但在重度追求**以假乱真的影视特效领域**，基于光线追踪的渲染算法才是主流。与基于光栅化方法不同，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。此次作业的任务是要求同学们动手实现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。



本次作业你们将实现的渲染效果图

1、作业概述

本次作业的主题为基于光线追踪的渲染，要求同学们动手实现一些光线追踪渲染器的基本算法，加深对光线追踪算法的理解。本次作业要求在提供的代码框架上，跟着我们提供的教程一步一步地实现相应的效果。提供的教程为英文教程，但同学们不必担心，该英文教程直观易懂，没有太多晦涩难懂的术语，对新手友好，放心食用。

2、代码框架

关于本次作业的框架代码部署和构建，请仔细阅读 `CGAssignment4/readme.pdf` 文档，基本上与 `CGAssignment3` 没有太大的差别。本次作业依赖的第三方库为：

- **SDL2**: 窗口界面库, 主要用于创建窗口并显示渲染的图片结果, 本作业不需要你对这个库深入了解

这些第三方库同学们无需太过关注, 我们的框架代码已经构建好了相应的功能模块。目录

`CGAssignment4/src` 存放我们的渲染器的所有代码文件:

- **main.cpp**: 程序入口代码, 负责执行主要的渲染循环逻辑;
- **WindowsApp(.h/.cpp)**: 窗口类, 负责创建窗口、显示结果、处理鼠标交互事件, 无需修改;

本次的代码框架比较简单。在 `main.cpp` 中, 我们首先声明了一个二维数组作为画布, 数组的每个元素存储画布上的 `r`、`g`、`b` 三个颜色通道的颜色值, 每个颜色通道的取值范围为 `[0,1]` 的浮点数。

```
static std::vector<std::vector<color>> gCanvas;    //Canvas
```

在 `main` 函数中, 我们创建窗口句柄、启动渲染线程。渲染在另一个线程上, 这是为了防止阻塞主线程的 `GUI` 交互事件。主线程将渲染线程的画布内容显示到屏幕上。

```
int main(int argc, char* args[])
{
    // Create window app handle
    WindowsApp::ptr winApp = WindowsApp::getInstance(gwidth, gHeight,
    "CGAssignment4: Ray Tracing");
    if (winApp == nullptr)
    {
        std::cerr << "Error: failed to create a window handler" << std::endl;
        return -1;
    }

    // Memory allocation for canvas
    gCanvas.resize(gHeight, std::vector<Pixel>(gwidth));

    // Launch the rendering thread
    // Note: we run the rendering task in another thread to avoid GUI blocking
    std::thread renderingThread(rendering);

    // window app loop
    while (!winApp->shouldWindowClose())
    {
        // Process event
        winApp->processEvent();

        // Display to the screen
        winApp->updateScreenSurface(gCanvas);
    }

    renderingThread.join();

    return 0;
}
```

`rendering()` 函数是渲染循环的主函数，在这里我们遍历画布的每个像素点，并为这些像素计算颜色值，然后调用 `write_color` 函数将颜色写入到画布的相应位置上。你将在这个函数里面实现一些光线追踪渲染算法的逻辑。这里初始的实现为 [Ray Tracing in One Weekend.pdf](#) 教程里面的 Chapter 2: Output an image。

```
void rendering()
{
    double startFrame = clock();

    printf("CGAssignment4 (built %s at %s) \n", __DATE__, __TIME__);
    std::cout << "Ray-tracing based rendering launched..." << std::endl;

    // Image

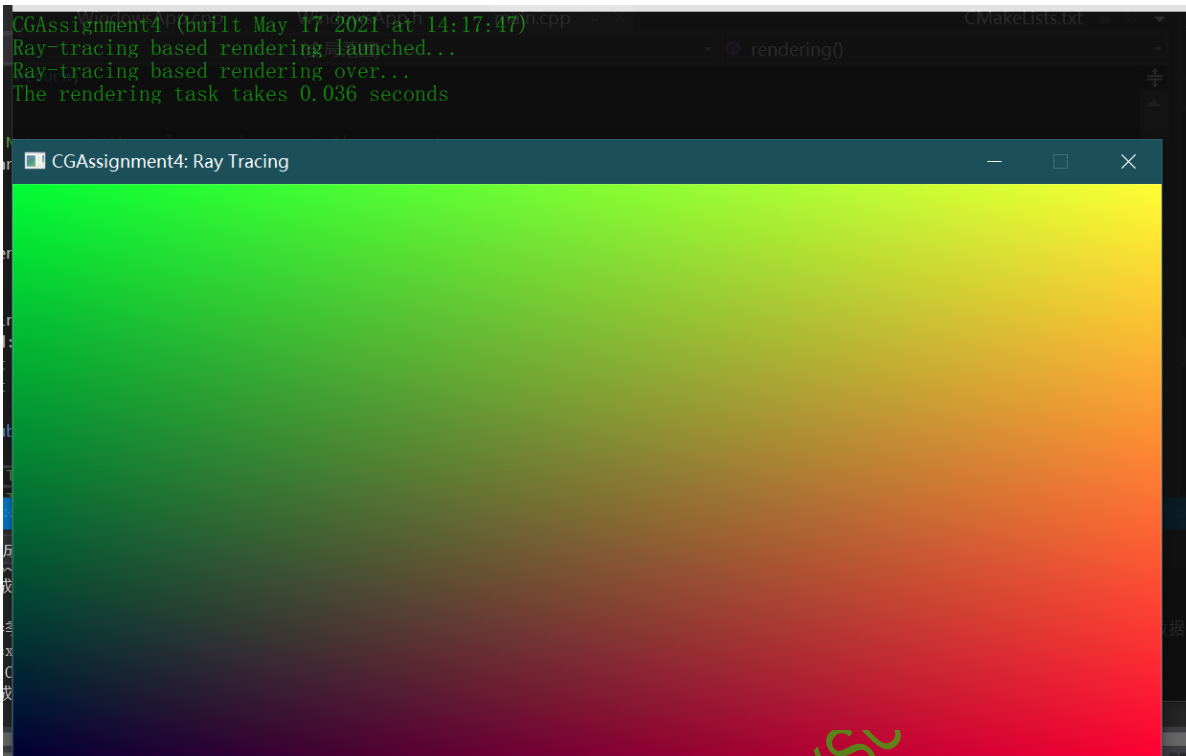
    const int image_width = gwidth;
    const int image_height = gheight;

    // Render

    // The main ray-tracing based rendering loop
    // TODO: finish your own ray-tracing renderer according to the given
    tutorials
    for (int j = image_height - 1; j >= 0; j--)
    {
        for (int i = 0; i < image_width; i++)
        {
            color pixel_color(double(i) / (image_width - 1),
                               double(j) / (image_height - 1), 0.25);
            write_color(i, j, pixel_color);
        }
    }

    double endFrame = clock();
    double timeConsuming = static_cast<double>(endFrame - startFrame) /
    CLOCKS_PER_SEC;
    std::cout << "Ray-tracing based rendering over..." << std::endl;
    std::cout << "The rendering task took " << timeConsuming << " seconds" <<
    std::endl;
}
```

编译运行本次作业的代码框架，你应该得到如下的结果：



请注意，如果你要修改渲染图片的大小，请修改 `gwidth` 和 `gHeight`（而不是修改 `image_width` 和 `image_height`），确保窗口大小和渲染图片大小一致。

3、作业描述

本次作业提供了两个教程文件，分别为《Ray Tracing in One Weekend》和《Ray Tracing The Next Week》，本次作业大部分要完成的任务在《Ray Tracing in One Weekend》文件。教程文件可能看起来有点长，不必惊慌，这是因为教程里面贴了大量的代码。请同学们先打开《Ray Tracing in One Weekend》文件，花一两分钟阅读Chapter 1概述章节。然后作业的代码框架已经帮你们实现好了Chapter 2和Chapter 3，因此在阅读完这两个章节之后你无需再做任何的补充和修改。（注：教程的Chapter 2通过将渲染的结果保存到文件来查看渲染的结果，但本作业框架直接在窗口屏幕上显示渲染效果，殊途同归）

本次作业中，我们将作业分为**基础（分数占比 80%）**作业和**提升（分数占比 20%）**作业，我们鼓励大家尽可能地完成所有作业，但还请根据自身能力选择并完成任务，**因为我们杜绝任何形式地抄袭！**

请你严格按照下面的顺序完成以下的任务：

基础任务：

1、阅读《Ray Tracing in One Weekend》的Chapter 4，完成射线类和简单的摄像机构建，并渲染一个渐变的蓝色天空背景图。贴出效果图，简述你遇到的问题以及是如何解决的。

2、阅读《Ray Tracing in One Weekend》的Chapter 5、Chapter 6和Chapter 7，为场景添加并渲染一个简单的球形物体。贴出效果图，简述你遇到的问题以及是如何解决的。

3、阅读《Ray Tracing in One Weekend》的 Chapter 8、Chapter 9 和 Chapter 10，为场景中的球形物体添加漫反射材质、金属材质和电解质材质，并渲染相应的材质效果图。贴出效果图，简述你遇到的问题以及是如何解决的。

4、阅读《Ray Tracing in One Weekend》的 Chapter 11 和 Chapter 12，实现摄像机的聚焦模糊效果。贴出效果图，简述你遇到的问题以及是如何解决的。

5、阅读《Ray Tracing in One Weekend》的 Chapter 13，渲染一张炫酷真实的图片。贴出效果图，简述你遇到的问题以及是如何解决的。

特别提醒：本任务渲染需要的时间多一点（几分钟），请耐心等待！（推荐 release 模式编译运行）。教程文件没有给出场景的参数设置，因此这里列出了场景的参数设置。

6、阅读《Ray Tracing The Next Week》的 Chapter 3，为场景物体构建一颗BVH树（Bounding Volume Hierarchies，包围体层次结构），加速追踪的射线与场景的求交计算过程。对比有无BVH树的渲染时间，简述你遇到的问题以及是如何解决的。

特别提醒：本任务是需要阅读第二个教程文件《Ray Tracing The Next Week》。

7、跟着提供的教程实现了一个光线追踪渲染器，谈谈你对基于光线追踪渲染的理解以及你的困惑、感想和收获。

提升任务：

完成《Ray Tracing The Next Week》教程的剩余章节，欢迎感兴趣、有能力的同学完成。

注意事项：

- 光追渲染运算量大，请尽可能地在 release 模式下编译运行。
- 将作业文档、源代码一起压缩打包，文件命名格式为：学号+姓名+HW4，例如19214044+张三+HW4.zip。
- 提交的文档请提交编译生成的pdf文件，请勿提交markdown、docx以及图片资源等源文件！
- 提交代码只需提交源代码文件即可，请勿提交教程文件、作业描述文件、工程文件、中间文件和二进制文件（即删掉build目录下的所有文件！）。
- 禁止作业文档抄袭，我们鼓励同学之间相互讨论，但最后每个人应该独立完成。