

# 光栅化 Kickstart

## A. 框架及个人文件夹更新

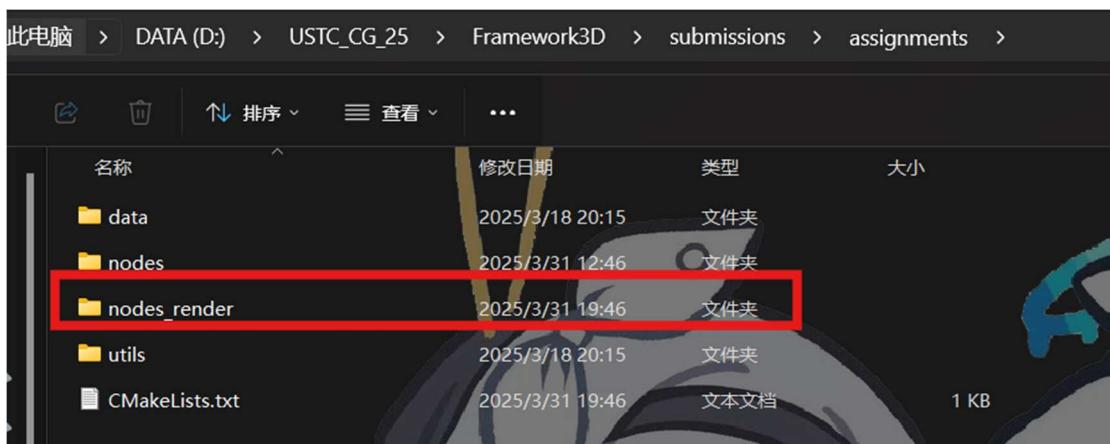
首先，拉取更新：

```
git pull upstream
```

```
git submodule update --remote .\Framework3D\Framework3D\
```

然后打开 Framework3D\submissions\assignments

你会发现里面多了个文件夹



这个文件夹是给这次光栅化作业使用的，其目录结构为：

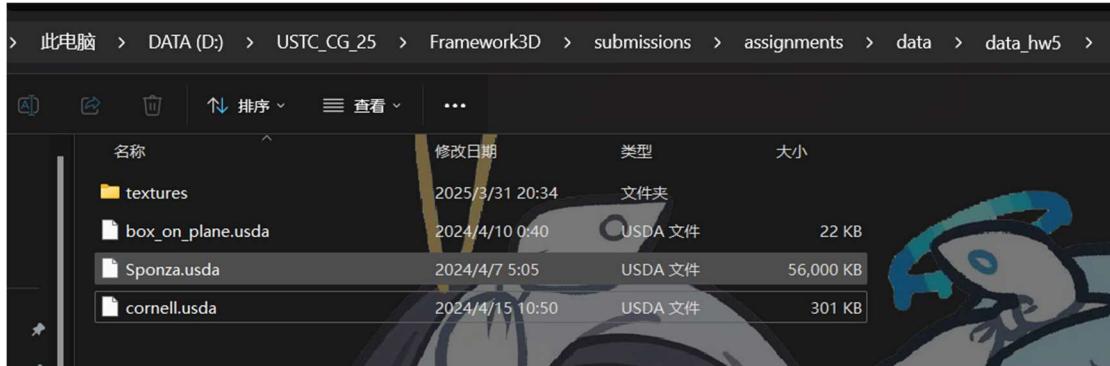


Shaders 文件夹中存放的是 Fragment shader & Vertex shader。

和之前的作业类似，将 **nodes\_render** 文件夹复制到你自己的作业文件夹中同样的位置，然后所有的修改均在你自己的 **nodes\_render** 文件夹下面进行！需要注意的是，我对 **assignments**

文件夹下面的 **CMakeLists.txt** 进行了一些修改 (添加了 `nodes_render` 的 subdir)，请结合你自己的文件夹下具体情况进行修改！  
这些节点有的本次作业不会强制你使用，如果不需要使用可以直接忽略。

本次作业为你们提供了三个模型，存放在我们的 `assignments/data/data_hw5` 文件夹里面



其中

- `box_on_plane.usda` 一个很简单的场景，没有纹理
- `cornell.usda` 也是一个很简单的场景，没有纹理
- `Sponza.usda` 一个比较大的场景，带有多种纹理贴图

## B. 框架编译

本次作业（包括下次 PT 的作业）使用的构建目标不再是

构建目标变更为：

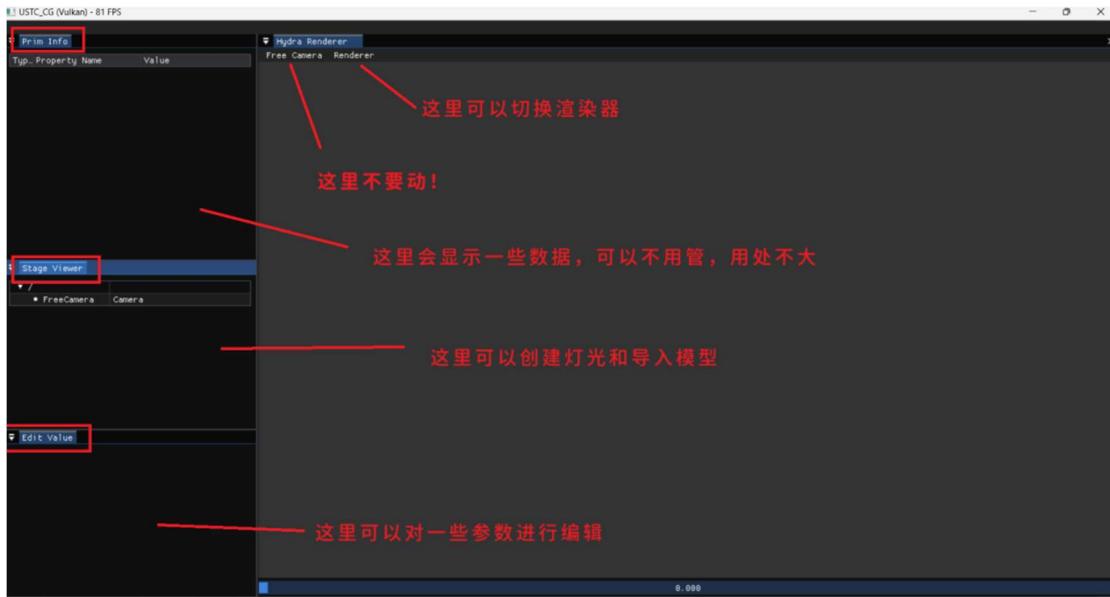
请大家在构建前**一定要选择正确的构建目标！！**

本次框架没有新的外部依赖，可以放心使用

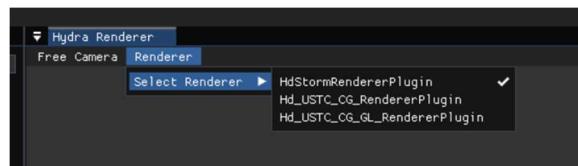
## C. 框架界面

在启动前，请**先删除 Assets 文件夹下面的 stage.usdc 文件**，防止可能存在的问题！

启动程序 `USTC(CG)_test.exe`，你会看见如下界面



需要提一嘴的是选择渲染器的部分。注意，**请仅在第一个和第三个渲染器之间进行切换**，第二个渲染器不是为了这门课设计的，它需要 Vk 的 RT 扩展，大家很多人是用不了的！



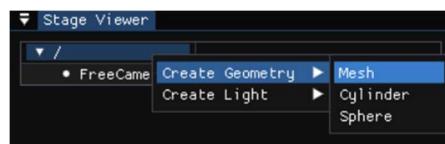
两个渲染器的分工为：

1. HdStormRendererPlugin: 进行简单的可视化，不可自定义。可以用它来调整视角
2. HdUSTC\_CG\_GL\_Renderer\_Plugin: 作业用渲染器，可以使用 shader

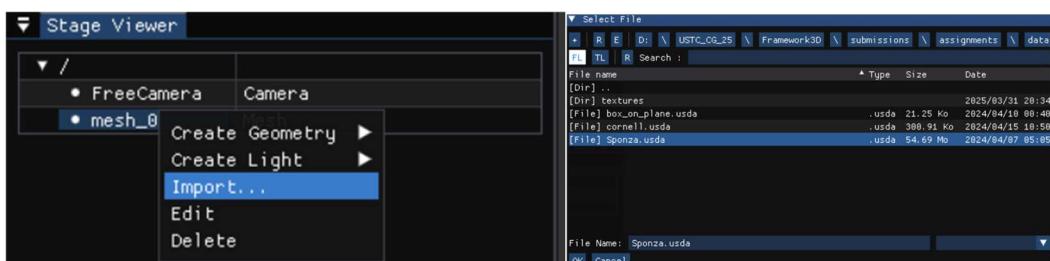
## D. 基本使用指南

下面，我们将以对 Sponza.usda 的导入作为例子，演示如何使用作业框架

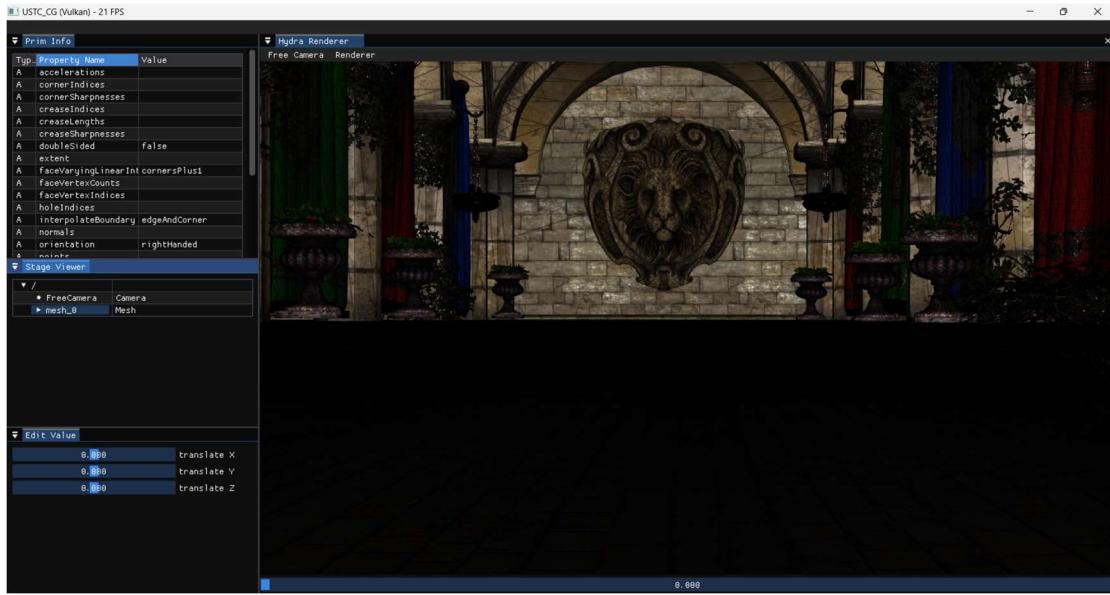
首先，**保持在第一个渲染器**（这个渲染器不会出岔子，万一你写了会使得程序崩溃的代码，这个渲染器也不会用，也可以正常显示），右键“/”，创建一个 Mesh



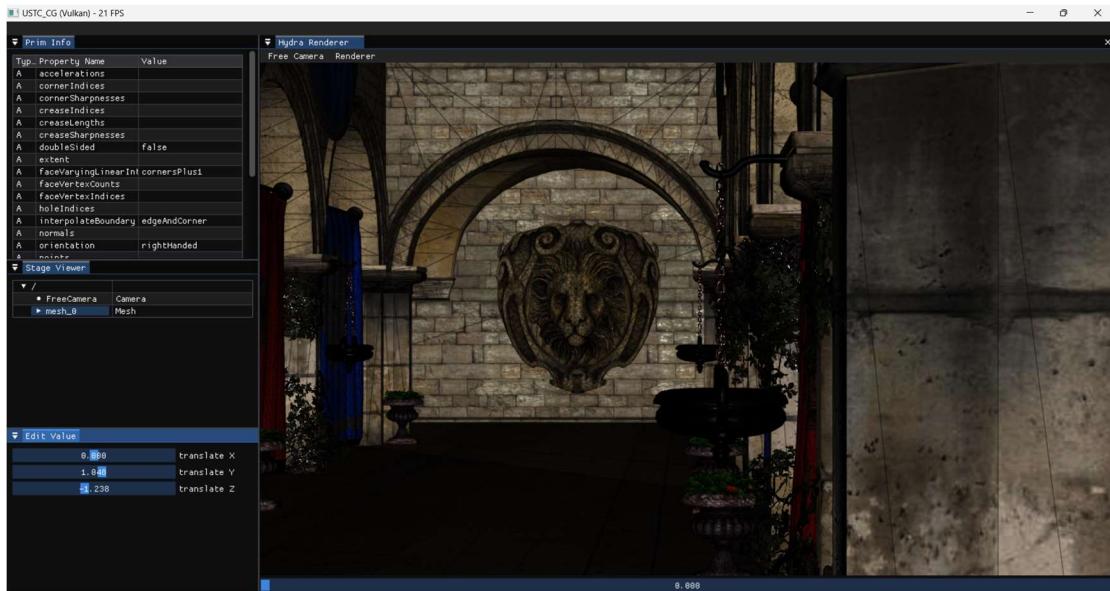
右键刚刚创建的 mesh，选 Import，并找到我们给的 usda 进行导入



打开场景后，会如下显示：



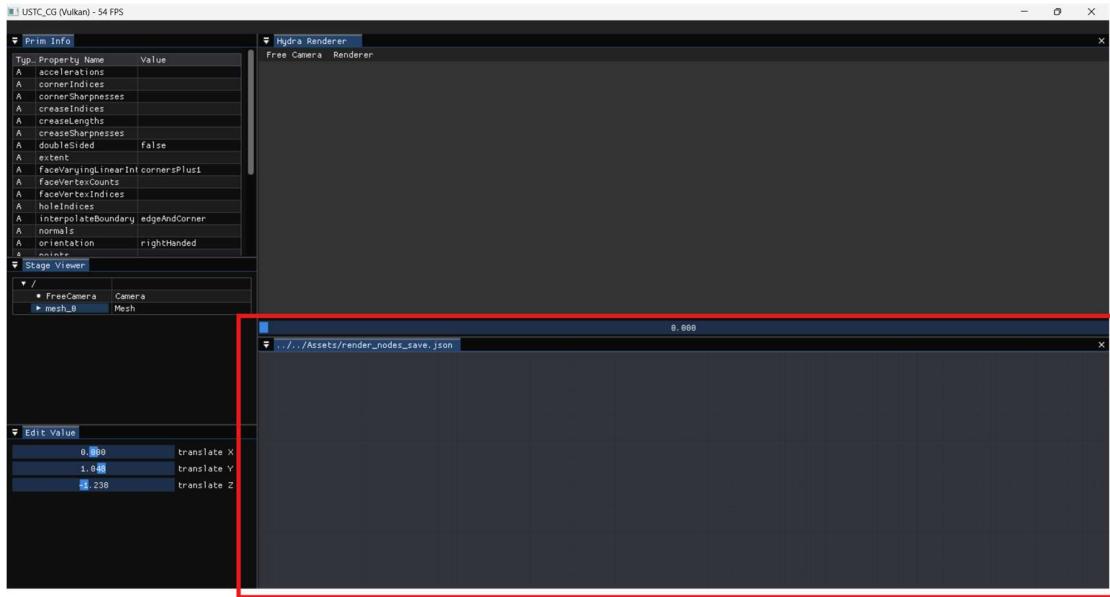
我们可以使用 **q w e a s d** 移动相机，鼠标左键拖动以改变视角。坐下的 Edit Value 部分可以修改我们选中的物体的位置（平移）。比如我们这里选中的是 **mesh\_0**（也就是刚刚打开的场景），我们就可以移动场景



在你还没有编写好光栅化渲染器的时候，可以依靠第一个渲染器来调整视角。

## D.1. 检查光栅化渲染器安装结果

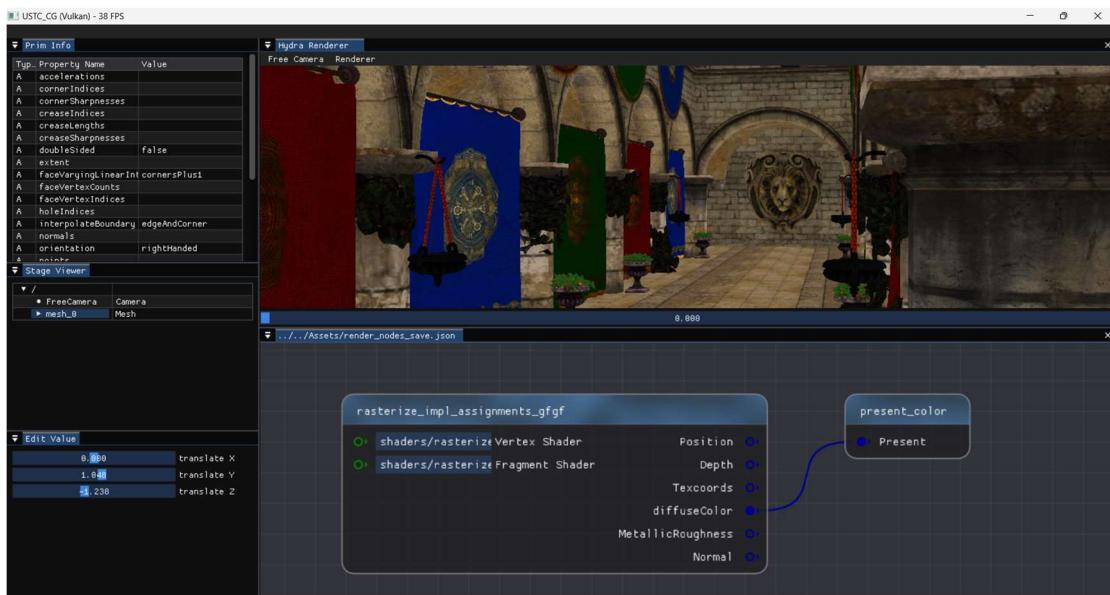
选中第三个渲染器，下面会弹出一个渲染节点的编辑窗口：



类似我这样添加节点并进行链接（除了 `present_color` 之外的节点都用你自己的！）



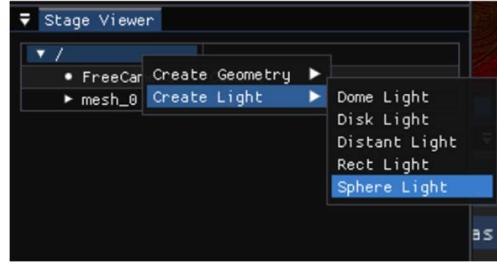
你会得到大概这样的效果（我这里移动了相机的）：



你还可以连一些其他的输出来可视化，这里不再赘述。

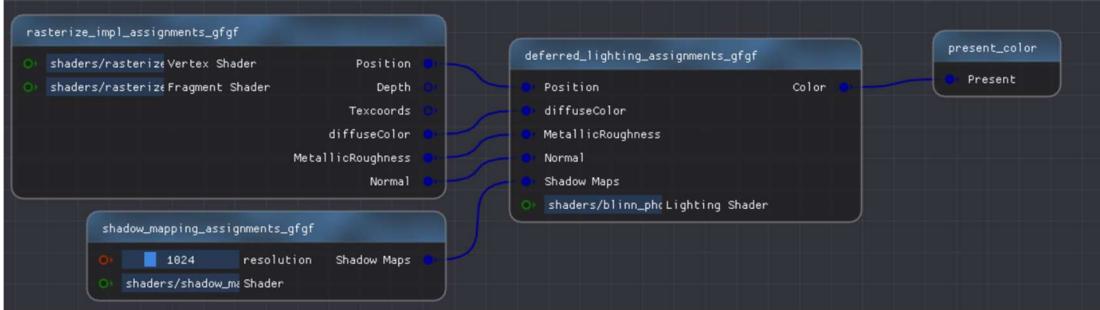
## D.2. 检查 Shadow map 计算模块是否正常

首先，右键“/”，向场景中添加一个 Sphere light

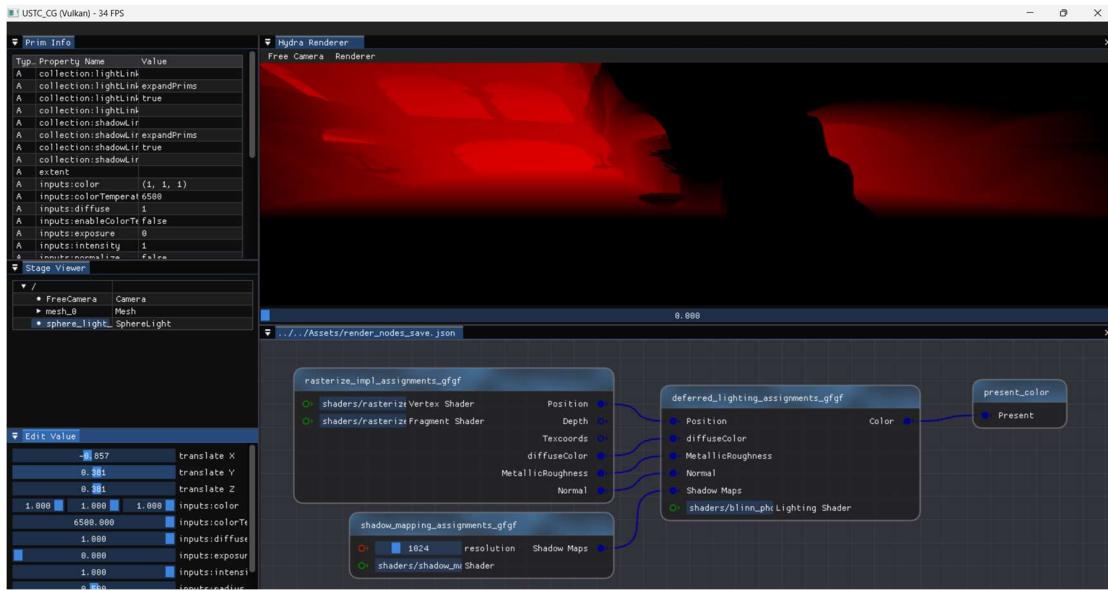


需要提一下的是，程序默认只对 **Sphere Light** 进行了支持。如果想要支持其他类型的光源，请自行对节点进行修改！且 Sphere Light 默认的 FoV 是 120 deg，且总是看向 (0, 0, 0)。

像我这样连接节点：



并对光源进行一些移动 (Edit Value 窗口)，你可以获得类似结果：



你们自己的 **deferred\_lighting\_assignments\_gfgf** 节点默认会为你们进行 Shadow map 的可视化，但这并不是最终的结果。你们要对 **shaders** 文件夹中的着色器进行修改，对你们自己的节点文件进行修改，来达到作业要求！请发挥自己的主观能动性，自己去分析代码的逻辑和依赖关系。当然，文件里面有注释提示你们要怎么做

注意！更改渲染器的时候不需要关闭程序，程序会重新对你们的渲染器进行加载。但是建议更改的时候断开和 **present\_color** 的连接，否则可能会出问题

## E. 新增提交要求

唯一的与之前有区别的地方，就是要把 **Assets** 文件夹下面的这两个文件一块拷贝（原来是只拷贝 stage.usdc 一个，这个 json 是新增的！）

名称	修改日期	类型	大小
render_nodes_save.json	2025/3/31 21:07	JSON 源文件	4 KB
stage.usdc	2025/3/31 20:39	USDC 文件	1 KB

祝大家找虫愉快 😊