

作业二

一. 作业内容

1. 实现三角形的光栅化算法

实验要求：

- 1.1: 用 DDA 实现三角形边的绘制
- 1.2: 用 bresenham 实现三角形边的绘制
- 1.3: 用 edge-walking 填充三角形内部颜色

讨论要求：

- 1.4: 针对不同面数的模型，从实际运行时间角度讨论 DDA、bresenham 的绘制效率。

2. 实现光照、着色

实验要求：

- 2.1: 用 Gouraud 实现三角形内部的着色
- 2.2: 用 Phong 模型实现三角形内部的着色
- 2.3: 用 Blinn-Phong 实现三角形内部的着色

讨论要求：

- 2.4: 结合实际运行时间讨论三种不同着色方法的效果、着色效率。

3. 最终效果：

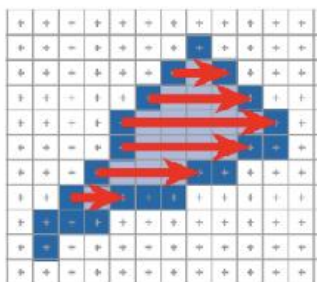
3.1：在 scene_0 中，加载单一三角形情况下，展示 DDA-EdgeWalking 与 Bresenham-EdgeWalking 两种方法下，三角形光栅化是否正确。

3.2：在 scene_1 中，加载复杂立体图形，用 Bresenham-EdgeWalking 方法，展示三种模型，在同一角度的光照效果（角度要能体现环境光、散射光、高光等）。

二：实现思路

1、三角形光栅化：

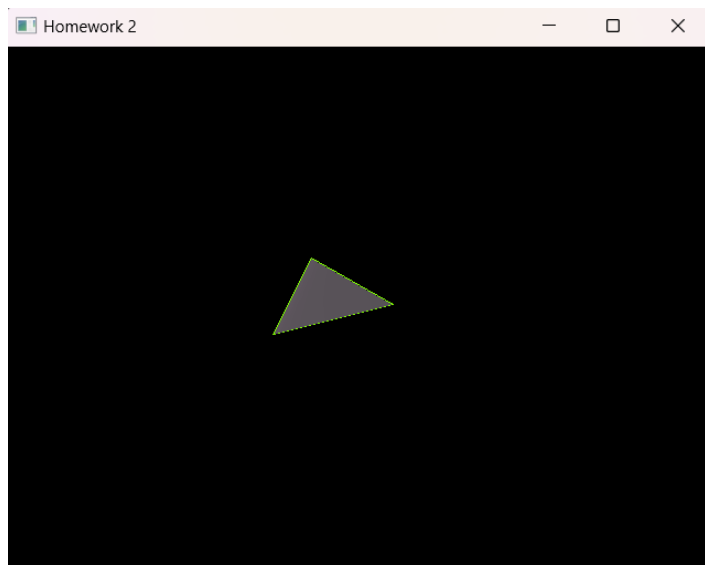
- 1.1: 首先利用两种边绘制方法将三角形的三边的片段记录下来。然后在 edge-walking 中从上到下遍历。



- 1.2: 在光栅化时可以任意填充一个固定颜色（要求 1.3 中）。
- 1.3: 默认不考虑片段在画布外的情况，若考虑到了，在展示中调整出一个物体超出画

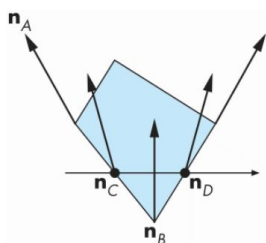
布的场景，作为加分项。

正确绘制的三角形应类似如下（bresenham 方法）：



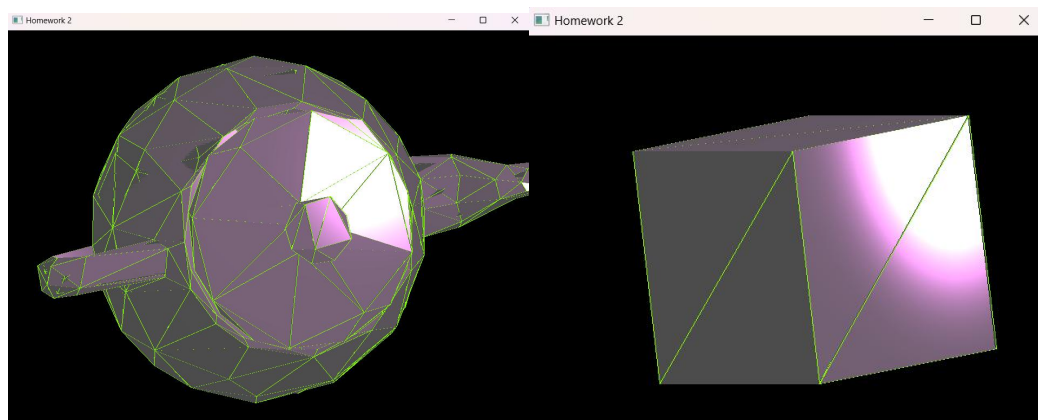
2、着色：

- 2.1: 在 edge-walking 中，计算三角形内部的每个片段的着色。
- 2.2: 着色时需要考虑每个片段在当前视角变化下的三维空间中的位置，将其与光源位置、相机位置相比较，计算相关方向向量。
- 2.3: 着色方法可能需要在边绘制、内部填充时对每个片段进行差值计算。因此在边绘制、内部填充时都需要考虑到。如下图， n_c 由 n_a, n_b 插值而来， n_c, n_d 横线内部的点由 n_c, n_d 插值得到。



- 2.4: 对于立体图形来说，还需要考虑插值顶点、边以及三角形内部的点的深度信息。
- 2.5: 讨论渲染效率，可以利用旋转让程序不断渲染，统计同一时间能渲染多少帧。

渲染的效果类似下图（对三角形的边可以不考虑光照，赋固定值）：



三：框架介绍

1 环境搭建：

- 1: pro 文件中, includepath 更改自己的 glew 文件位置
- 2: cmd 中打开 Template 文件夹, 输入命令 `qmake -tp vc`。然后双击打开 `vcxproj` 文件。
- 3: 编译运行, 不出错即可。

2 文件介绍

- 1: CGTemplate.pro: Qt 项目文件, 可以使用 Qt 项目文件生成 vs 项目文件 (windows)、makefile (linux)、或 xcode 项目文件 (macos) 等。
- 2: main.cpp: 主函数, 不需要修改。
- 3: myglwidget.cpp: 在此处完成你的代码实现, 请注意编程规范, 每个函数都需要写清楚注释, 包括函数的作用和大致的步骤, 实验报告内需要说明具体的实现思路并且需要有实验结果图。
- 4: myglwidget.h: 对应 cpp 的头文件, 注意更新函数声明。
- 5: utils.h: 工具头文件, 不用修改 (如有修改, 在报告中说明一下)
- 6: utils.cpp: 工具 cpp 文件, 不用修改 (如有修改, 在报告中说明一下)
- 7: glm 文件夹: 向量库, 用于方便的向量定义、计算

3 程序指导：

- 1: 在代码的“HomeWork: 1”标注部分, 给了 bresenham 的调用案例。请仿照框架, 添加 DDA 函数, 实现 DDA 与 bresenham 两种绘制方法。
- 2: 在代码的“HomeWork: 2”标注部分, 实现 edge_walking 函数内部逻辑。
- 3: 三角形画前都进行了 temp_render_buffer 的清空。每个三角形画在 temp_render_buffer 中。画完后, 框架基于 temp_z_buffer 将 temp_render_buffer 合并到 render_buffer 中。
- 4: 函数 drawTriangle 前半部分已经将三角形顶点映射到二维平面 (FragmentAttr.xy)。只需考虑二维平面上的点的绘制
- 5: 光照计算时, 需要知道每个片段在三维空间中的位置, drawTriangle 中已将位置保存在 FragmentAttr.pos_mv。可以不用其他变换, 直接与光源位置、相机位置相比
- 6: 自己调整光源位置、相机位置和其他的光照、反射参数
- 7: Utils.cpp 中包含 getLinearInterpolation 函数, 根据同一高度的两个片段, 在他们中间根据 x 轴位置进行差值, 可以使用, 也可以自己按照编程习惯实现。
- 8: 在 Scene_1 中, 选择加载不同的 model 运行。对于 teapot 模型, 根据电脑性能不同, 可以选 600 面或 8000 面文件运行
- 9: 框架中已经完成了深度测试, 只需正确赋值每个三角形的像素深度到 temp_z_buffer
- 10: 框架中, 按键 9 用于旋转, 可以比较同一时间能渲染多少帧

四. 作业提交方式

1. 提交作业内容：

(1) 项目完整文件：

- 放在一个文件夹中；
- 要求包含 `sln` 文件；

(2) 演示视频：

- 要求展示完整的功能；
- 禁止用程序功能外的内容拉长视频时长；
- 禁止重复多次展示同一功能；

(3) 实验报告：

- 要求写明功能实现思路和过程；
- 若有引用代码，要求分别明确标出引用部分和自己实现部分的代码和功能；
- 要求提交的实验报告格式为 `PDF`。

2. 文件组织方式：将上述提交作业内容中的三个文件压缩成一个压缩包，使用 `ZIP` 格式。

3. 文件命名方式：

(1) 上述提交作业内容中的三个文件分别依次命名为：

- 项目完整文件：CGHW2；
- 演示视频：V-HW2；
- 实验报告：R-HW2。

(2) 上述文件组织方式中的 `ZIP` 压缩包的命名方式：学号-姓名-作业 n，示例：22111111-李四-作业 2。

4. 提交作业网址：

请进入 <https://easyhpc.net/course/213> 后，加入课程并根据站内指引上传作业。

五. 注意事项

1. 允许讨论代码，但严禁任何形式的抄袭。

2. 未提交迟交申请，且迟交作业的同学，本次作业不得分。

3. 如遇到特殊情况导致未能按时提交作业的，可在作业截止日期前向 TA 提出 `slip days` 申请，并在 `slip days` 结束前补交作业。

4. 如遇到特殊情况导致未能按时提交作业且已用完 `slip days` 的，或者有不可抗力导致需要延迟超过 `slip days` 额度的，可提交申请说明理由，特事特办。