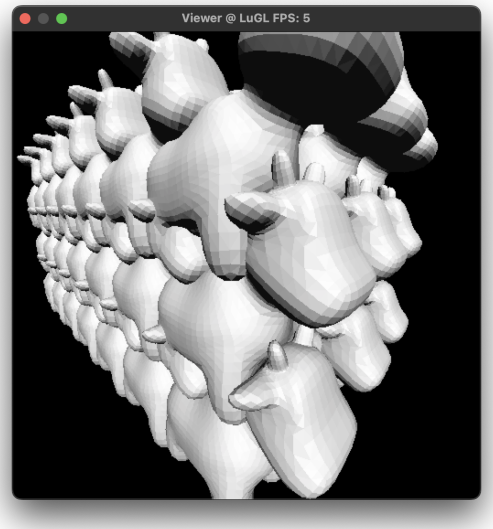
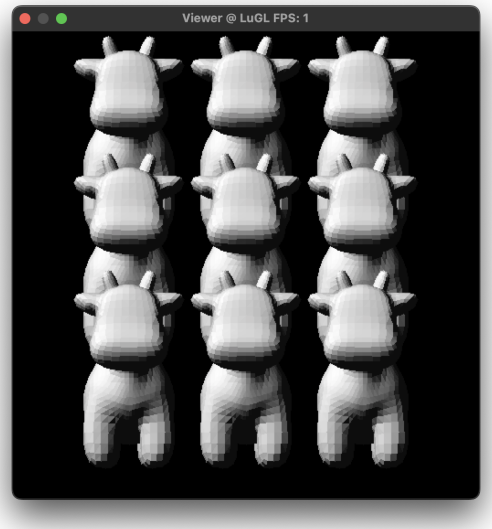
Z-Buffer说明文档

Created by	
Created time	@November 20, 2022 4:25 PM
Last edited by	
Last edited time	@November 20, 2022 4:36 PM
Tags	

Z-Buffer





概述

2022-2023秋冬学期《计算机图形学》课程大作业

本项目实现了以下功能：

- 多边形扫描转换
- 扫描线Z-Buffer
- 普通Z-Buffer（全画幅Z-Buffer）
- 层次Z-Buffer

- 八叉树加速的层次Z-Buffer

编译项目

多边形扫描转换

多边形扫描转换的实现位于`polygon_scanline.cpp`文件中，通过如下指令编译并运行后结果将储存为文件名为`result.png`的图像：

```
g++ -std=c++17 -Og polygon_scanline.cpp -o out && ./out
```

Z-Buffer

Z-Buffer相关算法在窗口程序中展示，使用make编译并运行，目前支持MacOS和Windows平台。**本程序未使用多线程或者GPU加速**，编译使用参数为`-std=c++17 -O3`，如果需要编译参数可以修改`makefile`文件：

MacOS

Compile in zsh `make`

Windows

1. Install MinGW https://www.mingw-w64.org/
2. Compile in cmd `mingw32-make`

运行程序

编译后使用命令行选择需要加载的模型和绘制模式：

MacOS

示例：

```
./viewer -i meshes/spot.obj
./viewer -i meshes/spot.obj -c 3 3
./viewer -i meshes/spot.obj -c 3 3 -z hiez
./viewer -i meshes/spot.obj -c 5 3 -z scanline -p o -m b 10
```

Windows

示例：

```
viewer.exe -i meshes/spot.obj
```

启动参数：

- `-i` 需要加载的模型（必填）
- `-z` 需要使用的Z-Buffer算法：
  - `simple` 简单Z-Buffer算法（默认）
  - `scanline` 扫描线Z-Buffer算法
  - `hiez` 层次Z-Buffer算法
  - `octz` 空间八叉树加速的Z-Buffer算法
- `-c` 绘制数量：
  - `1 n` 绘制1*1*n个模型（默认 1 1）
  - `3 n` 绘制3*3*n个模型
  - `5 n` 绘制5*5*n个模型
- `-m` 绘制模式：
  - `r` 实时模式（默认）
  - `b n` Benchmark模式，绘制n帧并输出计时结果
- `-p` 投影模式：
  - `p` 透视投影（默认）
  - `o` 正交投影

窗口操作指南

鼠标拖拽：旋转视角

鼠标滚轮：在透视投影下调整FOV

空格键：重置视角

ESC键：退出程序