第十次作业报告

PB21010362 汪兆辰

2023 年 5 月 27 日

1 算法介绍

在得到正交投影之前，需要先对给定的物体进行变形，将需要渲染的物体映射到 x,y,z 轴的 [-1,1] 范围内. 为了定义正交投影变换，需要先将相机坐标固定在世界坐标中，定义长方体视锥体，并将视锥体移动到相机坐标的中心并归一化. 由于在 OpenGL 中 z 轴的正方向指向屏幕外，但视觉上我们认为距离越远的地方 z 坐标应该越大，所以需要将 z 轴反向变为左手系.

首先定义平移矩阵：

$$M_T = \begin{pmatrix} 1 & 0 & 0 & -\frac{left+right}{2} \\ 0 & 1 & 0 & -\frac{top+bottom}{2} \\ 0 & 0 & 1 & -\frac{zFar+zNear}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

再定义缩放矩阵：

$$M_S = \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & 0 \\ 0 & \frac{top-bottom}{2} & 0 & 0 \\ 0 & 0 & \frac{zFar-zNear}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

从而得到：

$$M_{Ortho} = M_{-z} M_S M_T \tag{3}$$

即：

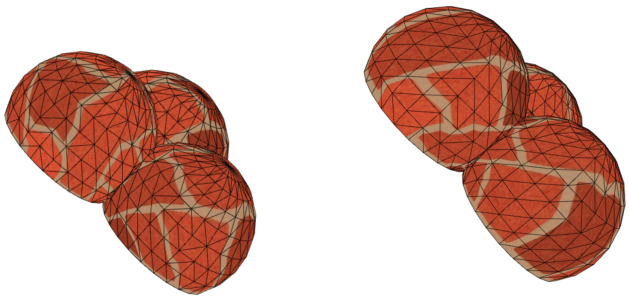
$$M_{Ortho} = \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{left+right}{2} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{2} \\ 0 & 0 & \frac{-2}{zFar-zNear} & -\frac{zFar+zNear}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

其中 M_{-z} 使得 z 轴反向.

在代码中只需要将原来的透视投影所用的矩阵变为 (4) 式中的矩阵即可，注意到视锥的宽与高需要满足比例为 aspect，在给定参数时只需要给定宽与 aspect 即可, 代码实现如下：

```
Eigen::Matrix4f orthographic(float left,float right,float aspect,
 float zNear, float zFar) const{
    assert(zFar > zNear);
    float top = left / aspect;
    float bottom = right / aspect;
    Eigen::Matrix4f res = Eigen::Matrix4f::Zero();
    res(0, 0) = 2 / (right - left);
    res(1, 1) = 2 / (top - bottom);
    res(2, 2) = 2 / (zNear - zFar);
    res(3, 3) = 1;
    res(0, 3) = -(right + left) / 2;
    res(1, 3) = -(bottom + top) / 2;
    res(2, 3) = -(zFar + zNear) / 2;
    return res;
}
```

2 效果展示



(a) Orthographic (b) Perspective

可以看到，正交投影不考虑物体的远近，在相同角度下远处的半球明显大于透视投影.

参考文献

[1] https://www.bilibili.com/video/BV12t4y1c7zd/

[2] https://blog.csdn.net/u011418943/article/details/128052420