计算机图形学
课程项目

王徐笑风[*]　　凌泽辉[†]

2020 年 11 月 3 日

[*]学号:18120193 E-mail:2208740924@qq.com

[†]学号:18120193 E-mail:785896610@qq.com

目录

摘要

查找资料，学习了解三维网格模型的相关知识。完成一个三维网格模型的显示系统。

数据输入：通过文件读取模型数据

数据存储：设计程序内用于存储模型数据的数据结构

数据输出：在窗口界面进行模型显示

编程实现三维到二维的投影变换计算

编程实现通过键盘或鼠标驱动模型的平移、缩放及旋转变换

可以使用开发工具中提供光照函数，若自己编程实现光照计算，则可获得额外加分

1 三维图形的表示与变换

1.1 向量与点

在三维空间中我们常用一个三维向量表示一个点，虽然向量本身只表达长度和方向，他是无关坐标系的，而点显然是与选取的坐标系是相关的。因此在这里将点理解为在一个给定的坐标系下，原点按某一向量移动后的位置，它写作式(1)：

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1}$$

1.2 变换

对于三维空间中的点，常用到的仿射变换和二维中的类似：平移、旋转和缩放。

1.2.1 平移

平移变换是将一个点按一个方向，移动一段距离。考虑到上面我们的点的定义即为在给点的坐标系下，原点按一个向量移动的距离。那么显然平移一个点即将原点平移两次，即点所代表的“向量”和平移向量的共同作用。

对于点$\mathbf{P}$，将其平移$\mathbf{V}$：

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ \mathbf{P}_z \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_x \\ \mathbf{V}_y \\ \mathbf{V}_z \end{bmatrix}$$

则有平移后的点$\mathbf{N}$：

$$\mathbf{N} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ \mathbf{P}_z \end{bmatrix} + \begin{bmatrix} \mathbf{V}_x \\ \mathbf{V}_y \\ \mathbf{V}_z \end{bmatrix} = \begin{bmatrix} \mathbf{P}_x + \mathbf{V}_x \\ \mathbf{P}_y + \mathbf{V}_y \\ \mathbf{P}_z + \mathbf{V}_z \end{bmatrix}$$

1.2.2 旋转

旋转指的是：点以三维空间中的某点为旋转中心，进行旋转，这里简略的认为旋转中心为坐标系原点，即此时的旋转变换是一个特殊的线性变换。在三维坐标系中对点做按原点的旋转，即是对一个向量进行旋转。只需要求取原基向量 $\hat{i}$、$\hat{j}$、$\hat{k}$ ，在旋转后的 $\hat{i'}$、$\hat{j'}$、$\hat{k'}$，可以得到旋转矩阵：

$$RotateMatrix = \begin{bmatrix} \hat{i'_x} & \hat{i'_y} & \hat{i'_z} \\ \hat{j'_x} & \hat{j'_y} & \hat{j'_z} \\ \hat{k'_x} & \hat{k'_y} & \hat{k'_z} \end{bmatrix}$$

则有旋转后的点$\mathbf{N}$：

$$\mathbf{N} = \begin{bmatrix} \hat{i'_x} & \hat{i'_y} & \hat{i'_z} \\ \hat{j'_x} & \hat{j'_y} & \hat{j'_z} \\ \hat{k'_x} & \hat{k'_y} & \hat{k'_z} \end{bmatrix} \times \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ \mathbf{P}_z \end{bmatrix}$$

特别的，绕Y轴旋转 θ 弧度的旋转矩阵，可以这么考虑：首先Y轴显然是不变的，在左手系下从Y轴逆方向向下看，Z轴X轴正好组成一个平面直角坐标系。则 $\hat{i}$ 顺时针旋转过 θ 弧度后的向量为：

$$\hat{i'} = \begin{bmatrix} cos(\theta) \\ 0 \\ -sin(\theta) \end{bmatrix}$$

同样的 $\hat{k}$ 旋转过 θ 弧度后的向量为：

$$\hat{k'} = \begin{bmatrix} sin(\theta) \\ 0 \\ cos(\theta) \end{bmatrix}$$

综上，Y轴旋转矩阵为：

$$RotateY(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{2}$$

类似的我们也可以得到X轴旋转和Z轴旋转矩阵：

$$RotateX(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \tag{3}$$

$$RotateZ(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

最后，通过上述(3)、(2)、(4)的旋转矩阵的复合矩阵即可实现任意的旋转，即：

$$Rotate(\alpha, \beta, \gamma) = RotateZ(\gamma) \times RotateY(\beta) \times RotateX(\gamma)$$

1.2.3 缩放

缩放的实现和旋转矩阵类似，计算出新的 $\hat{i'}$、$\hat{j'}$、$\hat{k'}$ 即可：

$$Scale(\alpha, \beta, \gamma) = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

则缩放后的点为：

$$\mathbf{N} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha x \\ \beta y \\ \gamma z \end{bmatrix}$$

1.2.4 复合变换

不难看出上述的变换中，旋转和缩放的变换可以简单地实现符合，将旋转矩阵和缩放矩阵进行矩阵乘法复合即可，同时这里两者是可交换的。

但平移变换就无法使用矩阵表达（矩阵变换后的点 $\mathbf{N}$ 的一个分量可以表达为 $\mathbf{N}_x = a\mathbf{P}_x + b\mathbf{P}_y + c\mathbf{P}_z$ 而平移变换可表达为 $\mathbf{N}_x = \mathbf{P}_x + d$ 显然上述公式中 $a = 1, \mathbf{P}_y + c\mathbf{P}_z = d$ 显然无法使用一个静态的矩阵实现），因此在描述一个点的平移、旋转和缩放变换时，我们使用下述公式(5)

$$\mathbf{N} = (Scale(a, b, c) \times Rotate(\alpha, \beta, \gamma)) + \mathbf{V} \tag{5}$$

但使用上述公式计算是不“简单”的，我们希望能有一种方法通过一次一种计算即可表达上述三种变换，同时又能提高运算的效率。

1.3 三维的齐次坐标

在三维中我们无法将平移、旋转和缩放变换由一个矩阵描述。不妨假设我们在四维中，并规定点 $\mathbf{P}$ 在四维中的坐标为：

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

即对于 $\forall \mathbf{P}$ 他们位于四维空间中分量 $z = 1$ 的一个三维“切片”空间中。

1.4 三角面

1.5 网格

1.5.1 .obj文件的加载

2 投影

2.1 正交投影

2.2 透视投影

3 相机变换

3.1 世界坐标系

3.2 视口坐标系

3.3 坐标系变换

4 绘制/光栅化

4.1 三角填充

4.2 绘制顺序问题

4.3 画家算法

4.3.1 画家算法的问题

4.3.2 改进方向

5 三维裁剪

5.1 性能问题与原因

5.2 三角面裁剪

5.2.1 近平面裁剪

5.2.2 视口裁剪

6 基础光照

6.1 全局光照