P2 3DMath Matrix实现

【任务】

- Matrix实现

【目的】

- 学习3D数学基础

【开始时间】

- 2019.08.09 11:00

【记录】

- 阅读任务要求 （2019.08.09 11:00 —— 2019.08.09 11:05）
- 开始查阅Matrix基础知识和线性变换Matrix的资料（2019.08.09 11:05 —— 2019.08.10 11:30）
- 开始写Matrix的代码（2019.08.10 14:30 ）
- 写完代码（2019.08.12 15:30）
- 测试Matrix的代码（2019.08.12 15:30）
- 测试完成（2019.08.12 16:15）
- 记录问题（2019.08.10 18:32）
- 记录问题（2019.08.12 16:15）

【问题】

- C++语法问题

```
};

float& Matrix::operator[]( int pos ){
    return m[pos];
}

Matrix Matrix::operator * ( const Matrix& mat ){
    Matrix re;
    re[m11] = m[m11] * mat[m11];
    return re;
}
```

这里的形参const修饰会报错， 不用const正常运行。。。。。。

- 矩阵类中的数据成员 float类型的一维数组 若不写构造函数，默认构造函数会赋值float的最小值导致矩阵乘法出错
- 《3D数学基础：图形与游戏开发》中左乘和右乘的定义是以向量的位置定义的，行向量*矩阵为左乘，矩阵*列向量为右乘，而网上大部分资料是相反定义的。

【结束时间】2019.08.12 16:31

【总结】

- C++类一定要写构造函数
- 查阅3D数学资料时中要先确定左右手坐标系，和左乘右乘的定义，然后再根据定义转换为自己需要的公式
- 变换矩阵公式：

1. 平移变换

不可能的：

$$[x \quad y \quad z \quad 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ \Delta x & \Delta y & \Delta z & 1\end{bmatrix}=[x+\Delta x$$

公式 9.10　用 4×4 矩阵实现 3D 平移

记住，即使是在 4D 中，矩阵乘法仍然是线

2. 旋转变换

$$\mathbf{r}=[0 \quad 0 \quad 1]$$

$$\mathbf{r}'=\begin{bmatrix}\mathbf{n}_x\mathbf{n}_z(1-\cos\theta)+\mathbf{n}_y\sin\theta\\ \mathbf{n}_y\mathbf{n}_z(1-\cos\theta)-\mathbf{n}_x\sin\theta\\ \mathbf{n}_z^2(1-\cos\theta)+\cos\theta\end{bmatrix}$$

注意：上面我们只使用了列向量，这样做的目的是使等式整洁清晰、易于理解。

用这些基向量构造矩阵，可得公式 8.5 所示的 $\mathbf{R}(\mathbf{n},\theta)$ 为。

$$\mathbf{R}(\mathbf{n},\theta)=\begin{bmatrix}\mathbf{p}'\\ \mathbf{q}'\\ \mathbf{r}'\end{bmatrix}=\begin{bmatrix}\mathbf{n}_x^2(1-\cos\theta)+\cos\theta & \mathbf{n}_x\mathbf{n}_y(1-\cos\theta)+\mathbf{n}_z\sin\theta & \mathbf{n}_x\mathbf{n}_z(1-\cos\theta)-\mathbf{n}_y\sin\theta\\ \mathbf{n}_x\mathbf{n}_y(1-\cos\theta)-\mathbf{n}_z\sin\theta & \mathbf{n}_y^2(1-\cos\theta)+\cos\theta & \mathbf{n}_y\mathbf{n}_z(1-\cos\theta)+\mathbf{n}_x\sin\theta\\ \mathbf{n}_x\mathbf{n}_z(1-\cos\theta)+\mathbf{n}_y\sin\theta & \mathbf{n}_y\mathbf{n}_z(1-\cos\theta)-\mathbf{n}_x\sin\theta & \mathbf{n}_z^2(1-\cos\theta)+\cos\theta\end{bmatrix}$$

公式 8.5　绕任意轴的 3D 旋转矩阵

8.3 缩　　放

我们可以通过让比例因子 k 按比例变大或缩小来缩放物体。如果在各方向应用同比例

3. 沿xyz轴的缩放：

$$\mathbf{S}(k_x,k_y,k_z)=\begin{bmatrix}k_x & 0 & 0\\ 0 & k_y & 0\\ 0 & 0 & k_z\end{bmatrix}$$

公式 8.7　沿坐标轴的 3D 缩放矩阵