这个系列我主要做两件事，一个是看 games101，另一个是玩 Vulkan SDK 和 Godot 引擎。

vulkan 先从 vulkan-tutorial 开始。

games001 看上去就很 nb，games 系列课程还是从 101 开始吧（悲

稍微渲染一些符号 owo

$$\|\vec{x}\|$$

$$\hat{a} = \frac{\vec{a}}{\|\vec{a}\|}$$

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x & y & z \\ q & \ddots & p \\ d & \dots & e \end{pmatrix}$$

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

需要注意的是数乘对向量点乘的分配律。

……右手螺旋定则非常有用！

另外，右手坐标系，也就是成立右手螺旋定则的坐标系，意味着 $\vec{x}\times\vec{y}=+\vec{z}$

万千星辰在掌中涌动 owowowowo

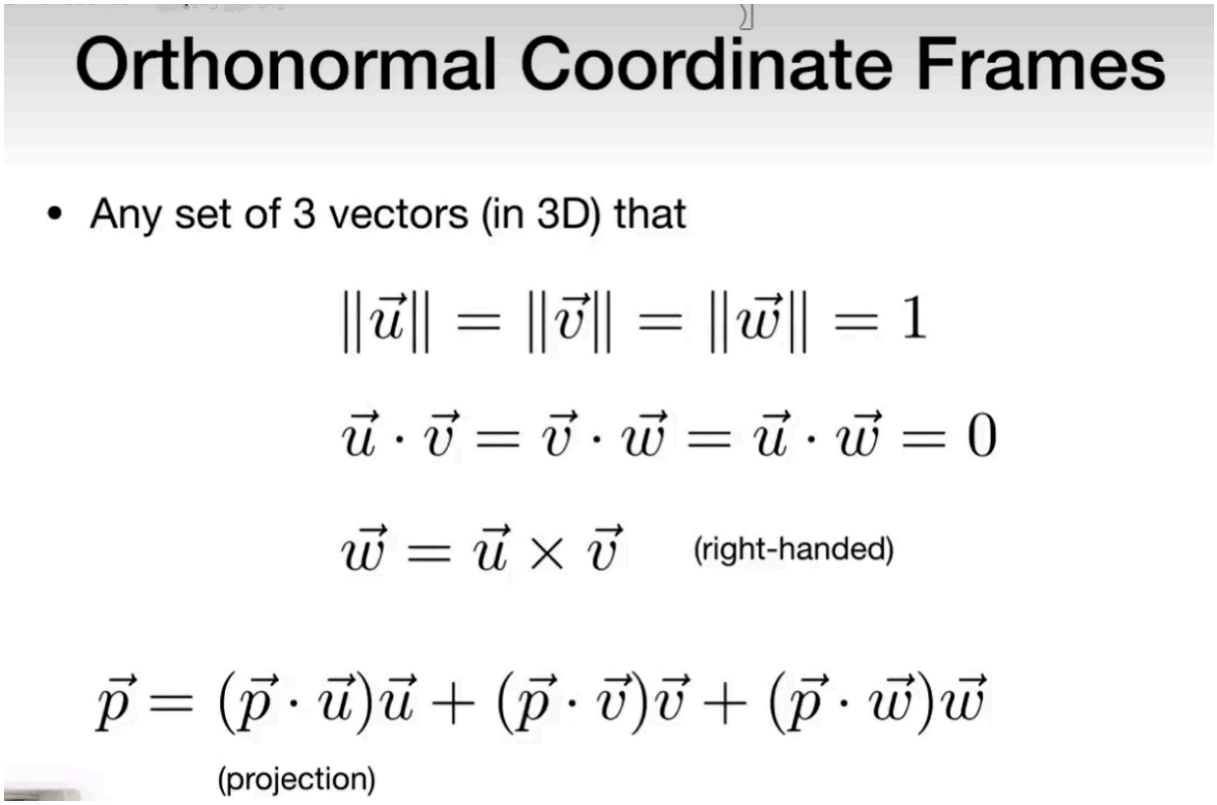


图 1　一堆表示，意会即可。

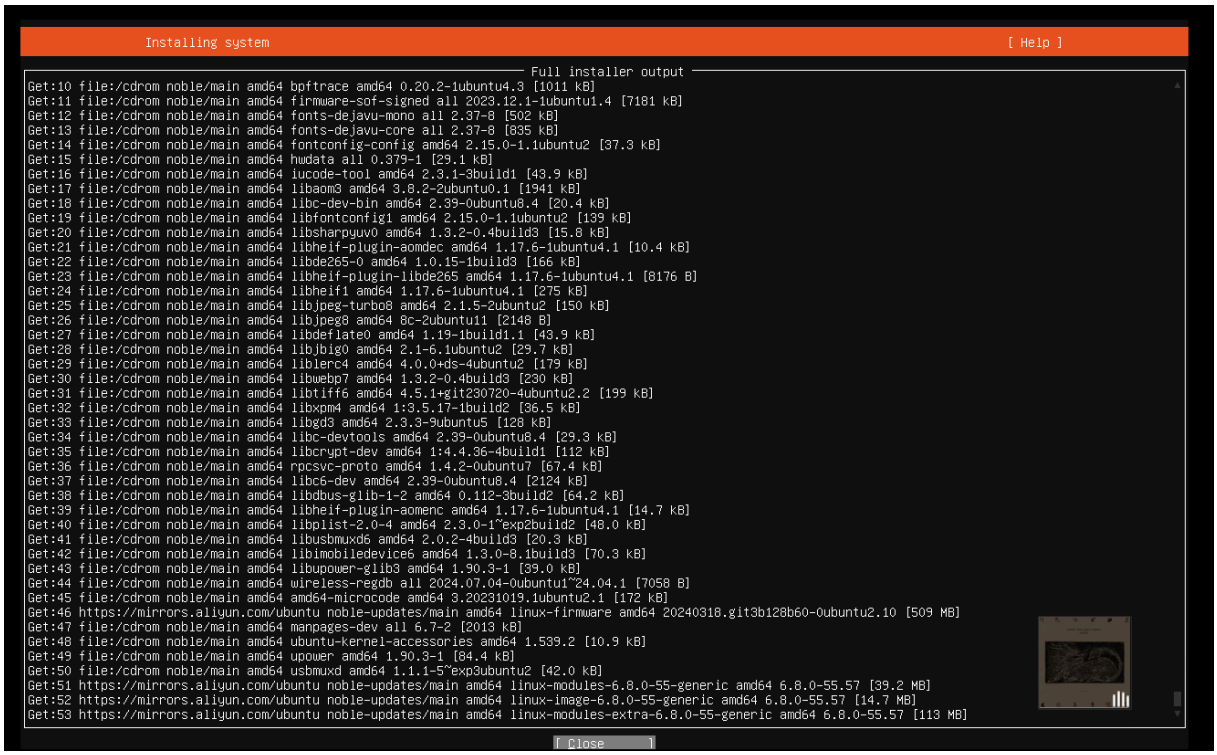


图 2　ubuntu server 24.04LTS

由于课程实在有些久远，清华云盘的资料已经失效了，所以虚拟环境就用自己的啦。

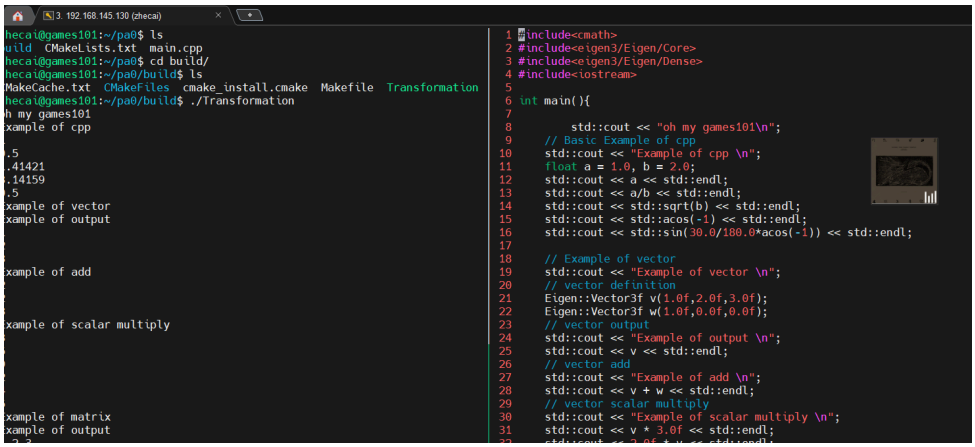
当然，用自己的虚拟环境就得手动装一些 lib。

```
zhecai@games101:~/pa0$ apt search eigen3
Sorting... Done
Full Text Search... Done
libeigen3-dev/noble-updates 3.4.0-4build0.1 all
  lightweight C++ template library for linear algebra

libeigen3-doc/noble-updates 3.4.0-4build0.1 all
  eigen3 API documentation

zhecai@games101:~/pa0$ sudo apt install libeigen3-dev
```

而且其实刚开始不一定要用图形界面，cmake 生成编译框架后可以一键 make 编译，所以 tmux 也可以用的很舒服。（毕竟我没装图形界面）



前面的练习基本是比较简单

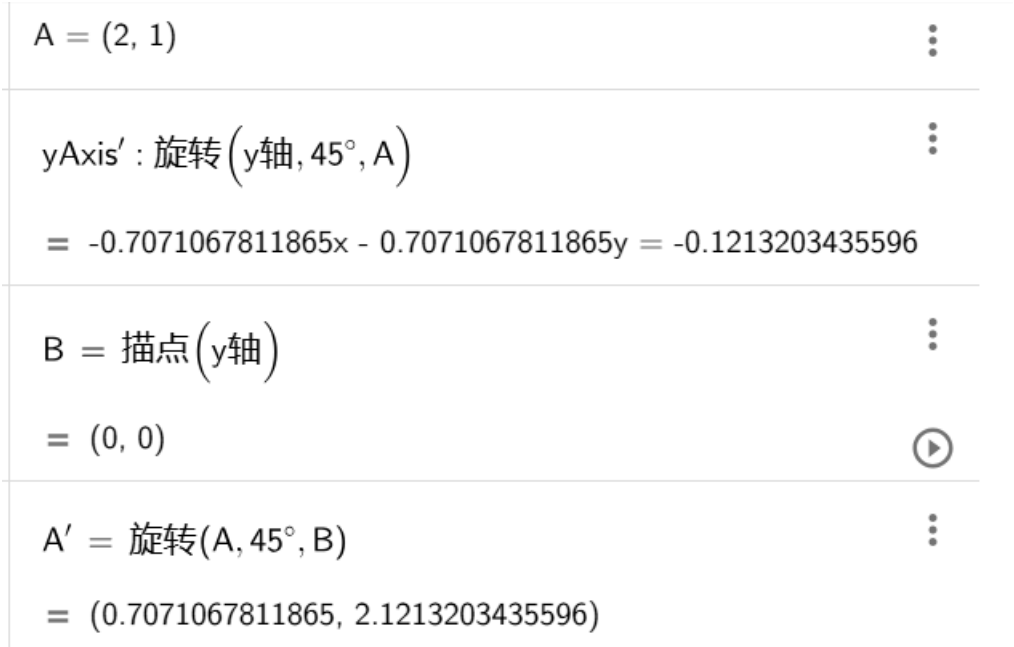
```
4
6
Example of matrix
Example of output
1 2 3
4 5 6
7 8 9
matrix add i + j
 3  5  4
 8 11 11
16 15 17
i * 2.0
 2  4  6
 8 10 12
14 16 18
matrix multiply i * j
 37  36  35
 82  84  77
127 132 119
matrix multiply vector i * v
14
32
50
vector dot vector v * v
14
zhecai@games101:~/pa0/build$
[0] 0:vim*
```

```
    std::cout << "Example of matrix \n";
    // matrix definition
    Eigen::Matrix3f i,j;
    i << 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0;
    j << 2.0, 3.0, 1.0, 4.0, 6.0, 5.0, 9.0, 7.0, 8.0;
    // matrix output
    std::cout << "Example of output \n";
    std::cout << i << std::endl;
    // matrix add i + j
    std::cout << "matrix add i + j \n";
        std :: cout << i + j << '\n';
    // matrix scalar multiply i * 2.0
    std::cout << "i * 2.0\n";
        std::cout << i * 2.0 << '\n';
    // matrix multiply i * j
    std::cout << "matrix multiply i * j \n";
        std::cout << i * j << '\n';
    // matrix multiply vector i * v
    std::cout << "matrix multiply vector i * v \n";
        std::cout << i * v << '\n';
    std::cout << "vector dot vector v * v \n";
        std::cout << v.dot(v)   << '\n';
    return 0;
}
main.cpp                                                55,18
"games101" 08
```

课上没讲齐次坐标，所以得自己学。

最简单地理解它的功用似乎就是把平移也可以用矩阵乘法完成了（因为多了个可以用于提供常量的维度）

虽然还有些东西，但网上写的语焉不详，所以就先只完成作业了，顺便用 GeoGebra 验证了一下。




```
127 132 119
matrix multiply vector i * v
14
32
50
vector dot vector v * v
14
rotate and remove
1.70711
4.12132
      1
```

```
Eigen::Matrix3f rotate, remove;
        remove << 1.0, 0.0, 1.0, 0.0, 1.0, 2.0, 0.0, 0.0, 1.0;
        rotate << std::cos(45.0/180.0*acos(-1)), -1 * std::sin(45.0/180.0*acos(-1)), 0, std::sin(45.0/180.0*acos(-1)), std::cos(45.0/180.0*acos(-1)), 0, 0, 0, 1;

std::cout << "rotate and remove\n";
std::cout << remove * rotate * Eigen::Vector3f(2.0, 1.0, 1.0)<<'\n';
// std::sin(45.0/180.0*acos(-1))
    return 0;
main.cpp                                                62,68         95%
```