实验过程

1️⃣投影变换简述见here

2️⃣代码和详细注释见Cmera.cpp中的Matrix4f Camera::projection()函数

路径为dandelion/src/scene/camera.cpp，实验的唯一目的就是修改Matrix4f Camera::projection()

```
Matrix4f Camera::projection()
{
    // 将视角从度转换为弧度，因为tan函数需要弧度为参数
    const float fov_y = radians(fov_y_degrees);

    // 计算近裁剪面上的顶部和右侧边界
    const float top   = near * std::tan(fov_y / 2.0f);
    const float right = top * aspect_ratio;

    // 根据对称性，计算近裁剪面上的左侧和底部边界
    const float left = -right;
    const float bottom = -top;

    Matrix4f projection = Matrix4f::Zero();  // 初始化投影矩阵为零矩阵

    // 根据透视投影矩阵的公式，填写矩阵的元素
    projection(0, 0) = 2.0f * near / (right - left);  // 元素(0,0)
    projection(1, 1) = 2.0f * near / (top - bottom);  // 元素(1,1)
    projection(0, 2) = (right + left) / (right - left);  // 元素(0,2)
    projection(1, 2) = (top + bottom) / (top - bottom);  // 元素(1,2)
    projection(2, 2) = -(far + near) / (far - near);  // 元素(2,2)
    projection(2, 3) = -2.0f * far * near / (far - near);  // 元素(2,3)
    projection(3, 2) = -1.0f;  // 元素(3,2)
    projection(3, 3) = 0.0f;  // 元素(3,3)

    return projection;  // 返回填写好的透视投影矩阵
}
```

3️⃣构建：

1. 在.dandelion/build目录下执行以下命令

```
cmake -S .. -B . -DCMAKE_BUILD_TYPE=Debug
cmake --build . --parallel 4
```

2. 在.dandelion/test/build目录下执行命令

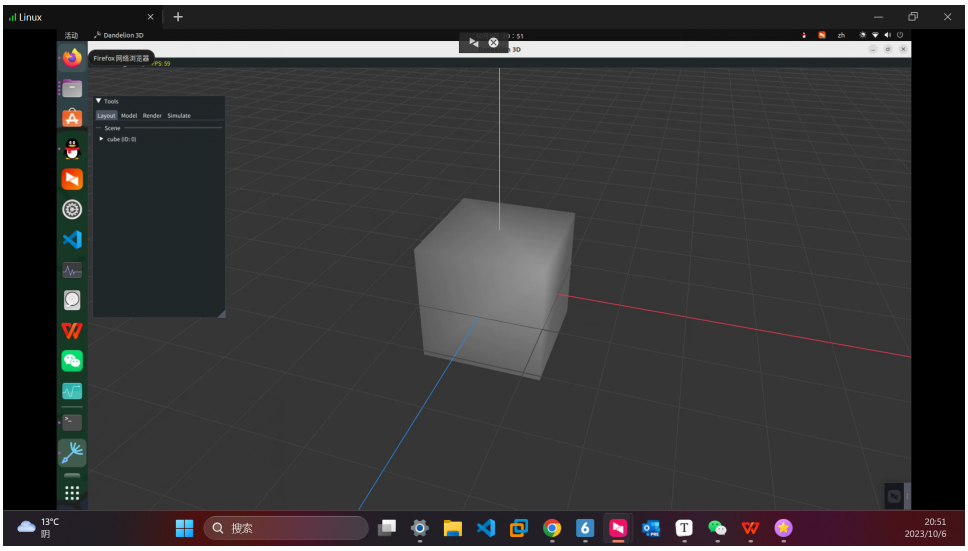
```
cmake -S .. -B . -DCMAKE_BUILD_TYPE=Release
cmake --build . --parallel 4
```

4️⃣运行：

1. 在.dandelion/build目录下运行dandelions

```
./dandelion
```

加载cube.dae



2. 在.dandelion/test/build目录下执行命令

```
./test "Perspective Projection"
```

然后终端输出

```
adminpc@admin-M6:~/桌面/dandelion/test/build$ ./test "Perspective
Projection"
[Test] [info] Dandelion 3D Unit Test, started at 2023-10-06 20:31:03+0800
Filters: "Perspective Projection"
Randomness seeded to: 3415001795
===============================================================================
All tests passed (5 assertions in 1 test case)
```

4️⃣ 文件：

1. 提交文档
2. 本次实验后的打包的dadelion文件夹