实验内容

1. 实验内容

填写 Object::model 函数
1.写出3旋转+1平移+1缩放的矩阵
2.几个矩阵按你这张图下一页开头的说法连乘
3.return这个矩阵即可
注意要将[x_angle, y_angle, z_angle]需要从角度换成rad弧度

2. 思路梳理

2.1. 关于Object::model

1️⃣大缺大德的课题组不会告诉你他在dandelion/src/scene/object.cpp里面

2️⃣这个函数不需要传入任何参数，文档里说了“根据物体的 center , rotation 和 scaling三个属性计算出
model matrix”，在object.cpp中三者是object类的成员变量

2.2. math.cpp

1️⃣路径为dandelion/src/utils/math.cpp
2️⃣quaternion_to_ZYX_euler返回的角度是角度值，要将其转化为弧度制需要用到math.cpp中的radians (T degrees)函数
3️⃣多嘴一句，就是math.cpp中已经包含了所以函数中sin/cos可以自由使用
4️⃣还有，object.cpp中包含了#include "../utils/math.hpp"，所以不用纠结有没有math.cpp了

2.3. Eigen库别忘了

3. 开始实验

3.1. 第一步：环境配置

先按照实验一中所记录的把实验环境重写配一遍，此外还需要在根目录执行以下命令

```
$ cd test
$ mkdir build
$ cd build
$ cmake -S .. -B . -DCMAKE_BUILD_TYPE=Release
$ cmake --build . --parallel 8
```

3.2. 第二步：填写函数

```
Matrix4f Object::model()
{
 // 第一步：将四元数表示的旋转转换为 ZYX 欧拉角 (角度制)
 const Quaternionf& r = rotation;
 auto [x_angle_deg, y_angle_deg, z_angle_deg] =
quaternion_to_ZYX_euler(r.w(), r.x(), r.y(), r.z());
```

```
    // 将角度转换为弧度
    auto x_angle_rad = radians(x_angle_deg);
    auto y_angle_rad = radians(y_angle_deg);
    auto z_angle_rad = radians(z_angle_deg);
    // 第二步: 使用欧拉角(弧度制)构建旋转矩阵
    Eigen::Matrix4f rotationMatrix = Eigen::Matrix4f::Identity();
    Eigen::Matrix4f rotationX = Eigen::Matrix4f::Identity();
    Eigen::Matrix4f rotationY = Eigen::Matrix4f::Identity();
    Eigen::Matrix4f rotationZ = Eigen::Matrix4f::Identity();
    // 设置绕x轴的旋转矩阵
    rotationX <<
        1, 0, 0, 0,
        0, cos(x_angle_rad), -sin(x_angle_rad), 0,
        0, sin(x_angle_rad), cos(x_angle_rad), 0,
        0, 0, 0, 1;
    // 设置绕y轴的旋转矩阵
    rotationY <<
        cos(y_angle_rad), 0, sin(y_angle_rad), 0,
        0, 1, 0, 0,
        -sin(y_angle_rad), 0, cos(y_angle_rad), 0,
        0, 0, 0, 1;
    // 设置绕z轴的旋转矩阵
    rotationZ <<
        cos(z_angle_rad), -sin(z_angle_rad), 0, 0,
        sin(z_angle_rad), cos(z_angle_rad), 0, 0,
        0, 0, 1, 0,
        0, 0, 0, 1;
    // 将三个旋转矩阵组合成一个完整的旋转矩阵
    rotationMatrix = rotationX * rotationY * rotationZ;
    // 第三步: 构建缩放矩阵
    Eigen::Matrix4f scalingMatrix = Eigen::Matrix4f::Identity();
    scalingMatrix(0, 0) = scaling.x();
    scalingMatrix(1, 1) = scaling.y();
    scalingMatrix(2, 2) = scaling.z();
    // 第四步: 构建平移矩阵
    Eigen::Matrix4f translationMatrix = Eigen::Matrix4f::Identity();
    translationMatrix(0, 3) = center.x();
    translationMatrix(1, 3) = center.y();
    translationMatrix(2, 3) = center.z();
    // 将缩放、旋转和平移矩阵组合成模型变换矩阵，并返回
    return translationMatrix * rotationMatrix * scalingMatrix;
}
```

3.3. 填写好后重新Cmake

1️⃣在/build目录下执行

```
$ cmake -S .. -B . -DCMAKE_BUILD_TYPE=Debug
$ cmake --build . --parallel 8
```

2️⃣在/test/build目录下执行

```
$ cmake -S .. -B . -DCMAKE_BUILD_TYPE=Release
$ cmake --build . --parallel 8
$ ./test Transformation
```

执行完后的根目录dandelion打包在这里

4. 实验结果

4.1. 在/test/build目录下执行./test Transformation

```
adminpc@admin-M6:~/桌面/dandelion/test/build$ ./test Transformation
[Test] [info] Dandelion 3D Unit Test, started at 2023-09-24 14:05:42+0800
Filters: "Transformation"
Randomness seeded to: 2037821475
===============================================================================
All tests passed (10 assertions in 1 test case)
```

4.2. 在/build目录下运行dandelion

打开cow.dae然后调整三种变换，注意旋转是角度制的