作业二也可以写了；

static bool insideTriangle(int x, int y, const Vector3f\* \_v)

{

// TODO : Implement this function to check if the point (x, y) is inside the triangle represented by \_v[0], \_v[1], \_v[2]

Vector3f v = Vector3f(x, y, 0);

Vector3f e1 = \_v[1] - \_v[0];

Vector3f e2 = \_v[2] - \_v[1];

Vector3f e3 = \_v[0] - \_v[2];

Vector3f\* p[3] = {&e1, &e2, &e3};

for (int i = 0; i < 3; ++i) {

Vector3f pvec = p[i]->cross(v - \_v[i]);

if (pvec.z() < 0) {

return false;

}

}

return true;

}

三次叉乘检验均通过说明在三角形内部

//Screen space rasterization

void rst::rasterizer::rasterize\_triangle(const Triangle& t) {

auto v = t.toVector4();

// TODO : Find out the bounding box of current triangle.

// iterate through the pixel and find if the current pixel is inside the triangle

for (int x = 0; x < width; ++x) {

for (int y = 0; y < height; ++y) {

if (insideTriangle(x, y, t.v)) {

// 计算重心坐标

auto [alpha, beta, gamma] = computeBarycentric2D(x, y, t.v);

float w\_reciprocal = 1.0 / (alpha / v[0].w() + beta / v[1].w() + gamma / v[2].w());

float z\_interpolated = alpha \* v[0].z() / v[0].w() + beta \* v[1].z() / v[1].w() + gamma \* v[2].z() / v[2].w();

z\_interpolated \*= w\_reciprocal;

// 深度测试

int index = get\_index(x, y);

if (z\_interpolated < depth\_buf[index]) {

depth\_buf[index] = z\_interpolated;

set\_pixel(Eigen::Vector3f(x, y, 0), t.getColor());

}

}

}

}

}

auto [alpha, beta, gamma] = computeBarycentric2D(x, y, t.v);

C++14好像不行，我用的C++

当像素在三角形内部时，调用computeBarycentric2D函数计算该像素相对于三角形三个顶点的重心坐标。重心坐标是用于插值计算像素颜色和深度的重要参数。

通过重心坐标，代码计算出该像素在屏幕空间中的深度值z\_interpolated。此处使用了透视除法（perspective division），并考虑了每个顶点的齐次坐标分量w，确保了正确的深度插值。

直接给的，不清楚是为什么，AI是这么说的

深度测试我的理解就是计算哪个三角形在该点更在前，就画哪个