homework1–3 games101

至今（2023.2）games101课程在b站一直维持着很高的播放量，前三次课程作业有一些问题，导致新人做作业时产生困惑。在原作业框架上面修改错误很困难，因此我重写了作业框架，希望能对新人有所帮助。重写的作业内容尽量保持原作业内容不变，为简化环境，采用windows系统+visual studio2019（及以上）。

1 原作业的问题

作业1

(1) 计算投影矩阵的函数：

Eigen::Matrix4f get_projection_matrix(float eye_fov, float aspect_ratio，float zNear, float zFar)

原作业框架使用的 zNear、zFar均为正数，而闫老师课上使用的均为负数，因此使用课上推导的投影矩阵计算方式，会导致渲染的三角形上下颠倒。重写框架和课上保持一致，zNear、zFar均为负数。（注意，此时计算视锥高度 t 的时候，应该是 t = –zNear * tan(fov/2)）。

(2) 计算view变换矩阵的函数：

原作业中的函数：get_view_matrix(Eigen::Vector3f eye_pos) 只有eye_pos一个参数。实际上view变换矩阵跟 相机位置、相机看向的位置、相机y轴朝向三个参数有关。重写框架实现了完整的view变换矩阵的函数。

(3) 作业提高项：

原框架提高项，计算绕任意过原点的轴的旋转变换矩阵，对于某些轴和转动角度，三角形会被转出到视锥之外，这时候原框架的framebuffer会出现数组越界，程序崩溃。重写框架判断一旦有三角形顶点出视锥，就不渲染这个三角形。（最完整的做法应当是做 齐次坐标剪裁，请参考https://zhuanlan.zhihu.com/p/162190576）

作业2

（1）同作业1的第一个问题。重写框架和课上保持一致，zNear、zFar均为负数，此时depth-buffer里面存储的z数值为负，因此z数值大更靠近相机。

（2）原作业框架中，透视矫正插值实现有误，闫老师课上并没有详细讲透视矫正插值的做法，参考 https://zhuanlan.zhihu.com/p/144331875。重写框架改正了透视矫正插值。

作业3

（1）判断点是否在三角形内部：

原框架rasterizer.cpp中的 static bool insideTriangle，没有考虑点在三角形边上的情况，点在边上会被判定不在三角形内部。重写框架参考了https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/rasterization-stage.html，点在上边、左边的时候判定在三角形内，否则判定在三角形外。

（2）关于着色的参考系

在着色计算的时候，需要知道fragment位置的法向量，和颜色、uv坐标等信息一样，法向量也需要做重心坐标插值。但是法向量有点特殊，obj文件中保存的法向量，是在model的局部坐标系的，变换到世界坐标系（或者相机坐标系，取决于你想在哪个坐标系着色），一个简单的想法是直接用model变换矩阵 M 乘以obj文件里面的法向量，

n_world = M * n_model

但这样是不对的，因为model变换可能会在x、y、z三个方向上有不同的缩放尺度。具体可以参考这篇文章：https://zhuanlan.zhihu.com/p/477868547。简言之就是需要这样做:

n_world = M.inverse().transpose() * n_model

其中M.inverse().transpose()称为normal matrix。

原作业框架是在view 坐标系做的着色，因为从rasterizer.cpp的207行可以看出，normal matrix用的是(view * model).inverse().transpose()。那么在做着色的时候，需要的点光源的位置、方向，也都应该是在view 坐标系下的，在其着色函数中，直接给了点光源的在世界坐标系的位置，以及相机在世界坐标系的位置eye_pos。因为在view坐标系下面做着色，所以还需要将这些变换到view坐标系，因此在fragment shader中还需要view变换矩阵，原框架没有给出。在bbs上，一些做法是用eye_pos – point来计算视线方向，这里明显是不合理的，因为在view坐标系下着色，eye_pos应该是（0,0,0）而不是原框架给出的（0,0,10），原框架相当于用世界坐

标系的eye_pos减去view坐标系的point位置，这显然是不对的，因此原框架渲染出来的小牛高光位置也不对。

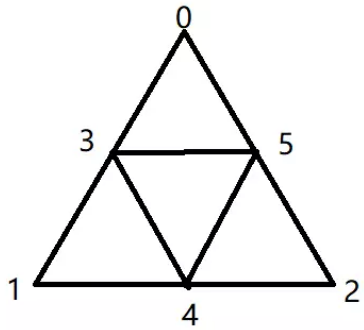
为避免这些错误，重写框架在world坐标系下面做着色，渲染图上高光的位置和原作业给出的答案不同。

（3）关于bump mapping 和 displacement mapping

原作业框架的TBN计算方式是错的，TBN是几何体局部的坐标系，需要用到局部三角形三个顶点的信息，原框架中只用了着色点的位置就计算TBN，显然是不对的。关于TBN的计算，有很多细节，我是按照https://zhuanlan.zhihu.com/p/139593847实现的，这篇文章里面也有一些细节上的讨论。把每个顶点的TBN提前计算好，然后针对每个fragment做重心坐标插值，并传入fragment shader。

原框架中 displacement mapping是错误的，课堂上提到过，displacement mapping需要改变原来的mesh几何，原框架这种实现方式并没有改变几何，因此小牛的边界出没有凹凸感。我没有查到正式商业引擎里面displacement mapping的做法，按照我自己的理解实现：

首先对每个三角形做细分：为简单，把每个三角形细分成四个，当然也可以细分成更多.



然后根据位移贴图hmap计算0-5六个点，移动后的位置和法向量：

计算每个点的Ti、Bi、Ni（i=0,1,2,3,4,5），那么位移后的位置是pi + Ni * h

计算每个点切空间的法向量：（-dh/du, -dh/dv, 1），然后计算世界坐标系的法向量：TBN *（-dh/du, -dh/dv, 1）

最后采用bump_fragment_shader对035、314、345、542四个三角形着色.

bump mapping、displacement mapping的作用都是使渲染的物体表面粗糙，所以应该在texture fragment shader的基础上加上这三种的效果，原框架没给出最终的渲染效果。

原作业框架提到过，给出的hmap.jpg并不是真正的bump mapping，原作业框架是用每个像素点的norm作为h，重写发现这样的h太大了，做displacement mapping小牛会有严重变形，因此重写框架用 h = (norm – 0.5)*0.2，这样h不仅数值比较合适，而且有正有负。