Report 10

实验目的：

完成网格的降噪

光照的实现

Normal map的实现

算法和思路

降噪

降噪采用了，类似于极小曲面中的局部方法

计算顶点到周围点重心的向量

用顶点减去这个向量乘以一个权重（设为了0.3）

不断的进行迭代

通过attribute传入vertex shader 的数据有 坐标，法向，和偏移向量

通过 uniform 传入 的数据有权重矩阵

在vertex shader中完成position – weight\*delta的过程

光照

关照分为

Ambient， diffuse 和 specular

Ambient 为环境光，全局都设为同一个值，乘以光源颜色

Diffuse 是漫反射光，是表面法向和光源到表面的向量（light direction）的点乘，再乘以光源的颜色

Specular 是镜面反射光，不但与物体到光源的距离有关，还与观察位置和他们的距离有关

先把观察点到表面的向量（view direction）点乘前面的light direction ,然后power 这个结果的n次方，n表示光的亮度，乘以光源颜色

最后把这三个光加起来乘以物体颜色

法线贴图

法线贴图就是把法线向量存放在纹理贴图中，保存为rgb的格式，对应xyz轴

读取的时候因为rgb在[0,1]，法线在[1-,1]所以要把读取的值乘以2-1作为法线向量

贴图中的发现向量都是偏向于z轴，默认了模型的z正轴朝向，使用的时候需要将法线向量转换到世界空间中进行光照的计算

实际的操作中，我们把世界坐标转换为切线空间，然后进行光照的计算。原因在与这样大量的计算可以在顶点着色器完成，而顶点着色器的数量小于片段着色器。

实验结果

模型读取用了assimp 库

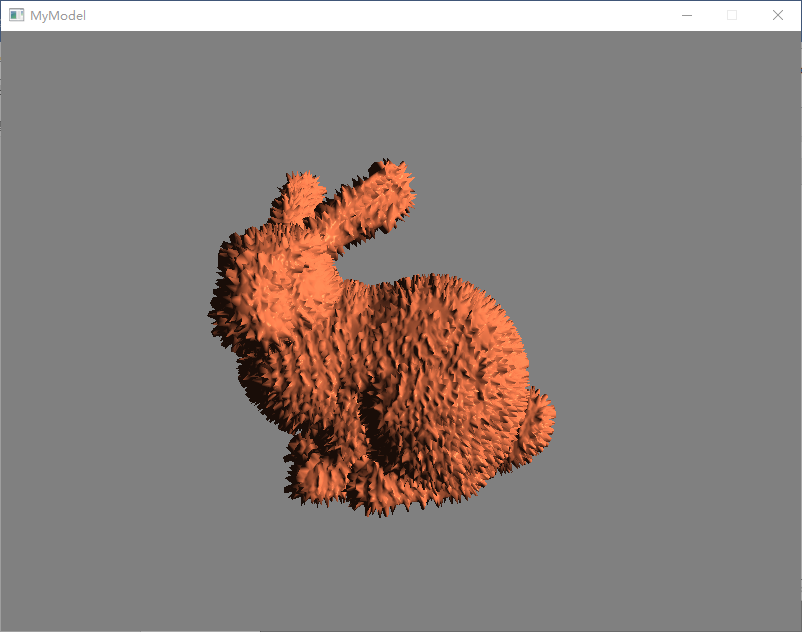
纹理读取用了soil库

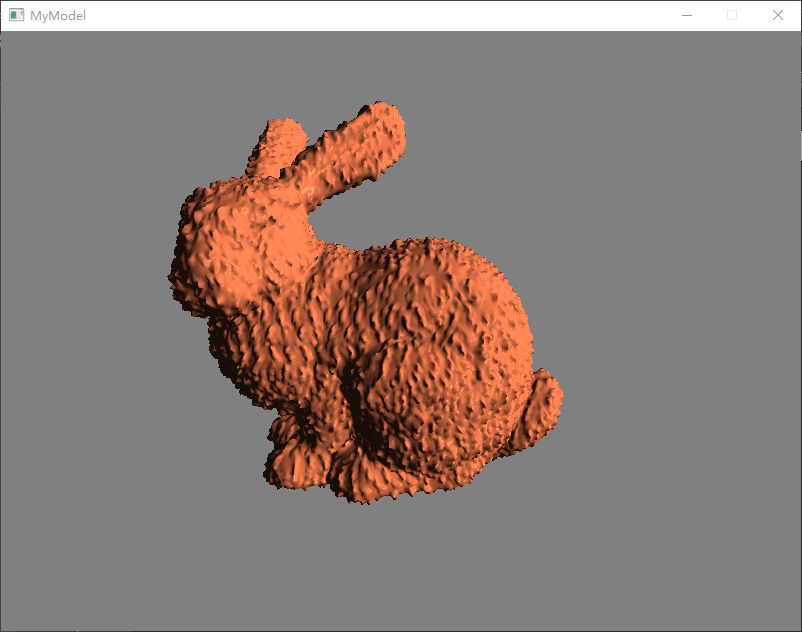
都在include 中

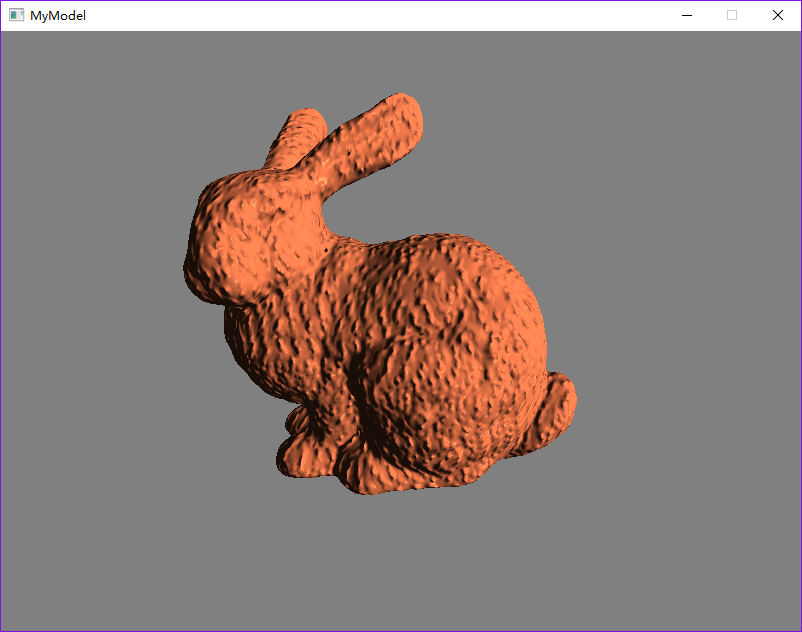
降噪因为要读取周围顶点所以还要用openmesh

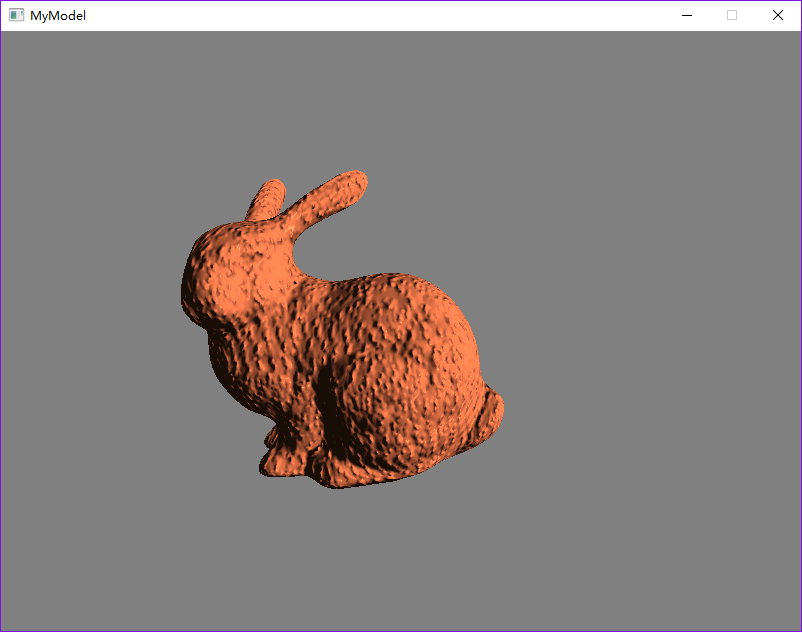
网格去噪和光照

光照以前做过类似的就直接把以前做过的拿来用了









Open Mesh 实在是太慢了所以就迭代了这么多次

法线贴图

