Silhouette Vertex Shader

梁晨 3180102160

- 实验环境

  本课程的所有作业均在macOS Catalina 10.15.7上运用集成开发平台XCode 12.0.1完成。我在文件中附上了XCode的原始工程文件。C++相关源码在与最上层文件夹同名的文件夹中，而GLSL源码（即各个shader）在Debug文件夹中。编译生成的在macOS上可执行的UNIX可执行文件和建模过程中需要用到的.txt, .obj, 以及. tga（相关纹理）文件，也都在Debug文件夹中。渲染的结果实时、动态地显示在GLUT窗口中，我将渲染结果做了截图，统一放进output文件夹中。

- Silhouette Vertex Shader的实现

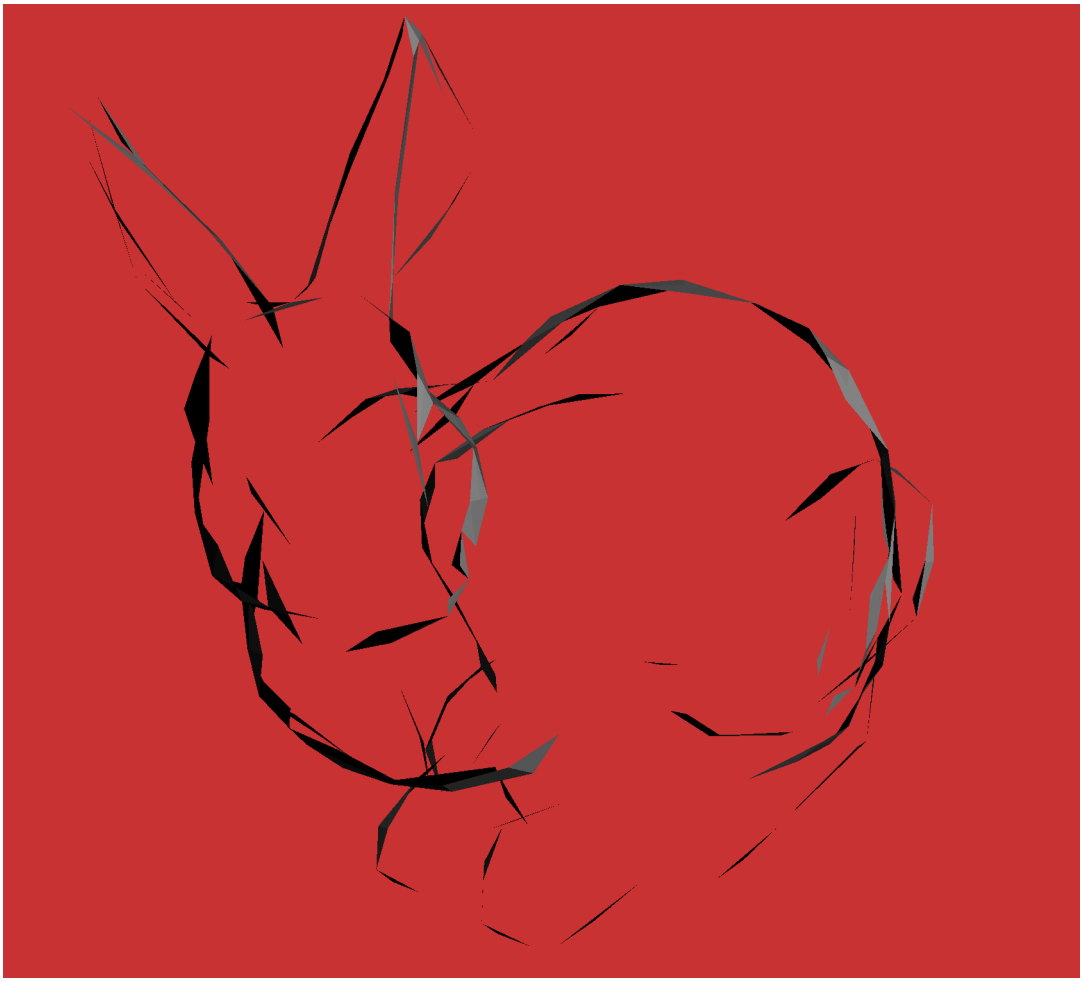
  本作业在作业1的框架下进行，只改动vertex shader。PPT中给的算法是要对顶点所在的边是否是轮廓边进行判断，而判断的方式是看共用这条边的两个相邻面片的法向量的夹角够不够大，如果够大，则认为这条边是轮廓边，对相关顶点进行绘制，反之，则不绘制，将vertView.xyz（做过ModelView的顶点坐标）设置为（0.0,0.0,0.0)。而在我搭建的框架中，渲染是以小面片为单位的（面片与面片间无关联），一个顶点对应着多条边，如果要对每条边找相邻两个面片，则违背了面片与面片间无关联的原则，需要进行大规模的代码重构。于是，我采取了《real time rendering》书中判断轮廓边的方法，即我们判断单个面片的法向量和viewVector的夹角够不够大，如果够大，则是轮廓边，绘制相关顶点，如果不是，则不绘制，将vertView.xyz设置为（0.0,0.0,0.0)。代码如下：

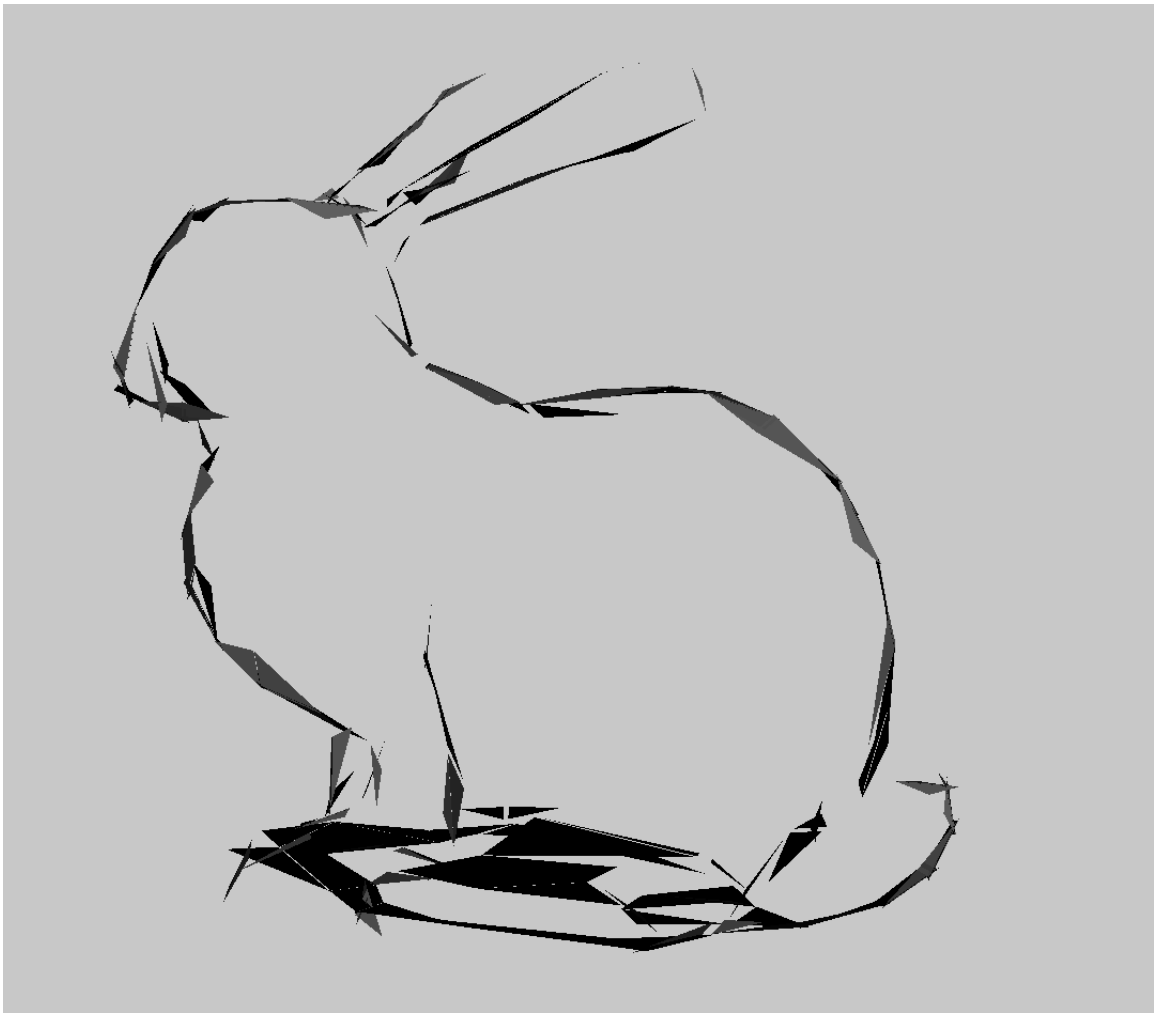
```
//原算法中需要用到的两个面，本算法中弃之不用
attribute vec3 facenorm1;
attribute vec3 facenorm2;

varying vec4 diffuseColor;
varying vec3 fragNormal;
varying vec3 lightVector;
varying vec3 viewVector;

void main()
{
  vec3 eyeSpaceLightVector = gl_LightSource[0].position.xyz;
  vec4 viewVert=gl_ModelViewMatrix*gl_Vertex;
  vec3 vert=vec3(viewVert);

  lightVector= vec3(normalize(eyeSpaceLightVector - vert));
  fragNormal = gl_NormalMatrix * gl_Normal;
  viewVector = -vec3(vert);

  //facenorm1=gl_NormalMatrix*facenorm1;
```

```
  //facenorm2=gl_NormalMatrix*facenorm2;
  //float d1=dot(facenorm1,vert);
  //float d2=dot(facenorm2,vert);

  //判断做过ModelView的normal和viewVector的夹角是否够大
  float d=dot(normalize(fragNormal),normalize(vert));
  //我设置夹角阈值为75度
  float lim=cos(75.0*3.14/180.0);
    if(d<lim&&d>-lim){
      //这里有一个收缩或扩张的效果，来加强风格化效果，减一定倍数的normal则为收缩，加一定倍数的normal即为扩张。
     vert-=0.05*fragNormal;
     viewVert.xyz=vert;
  }
  else viewVert.xyz=vec3(0.0,0.0,0.0);
  gl_Position = gl_ProjectionMatrix * viewVert;
}
```

这里还有一点需要明确，即为何将vertView.xyz设置为（0.0,0.0,0.0)我们就看不到与这个点相关的边了？本质上，是因为我们在做透视投影，将vertView.xyz设置为（0.0,0.0,0.0)相当于将点置于相机中心，任何以相机中心点为顶点的边，做透视投影后，都将失去一个维度，从线变成点，变得不可见。

- 实验结果：





以上两张图片，如果我们仔细看的话，还可以看到比较明显的三角形的痕迹（但效果已经很不错了，第二张有点像水墨画）。于是我们尝试在绘制时，不绘制面，而绘制线，即将polygon.h中paint()的代码修改为：

```
 void paint(){
    mtr->glSetMaterial();
    //glBegin(GL_POLYGON);
   //改为对线进行绘制，每次绘制三角形的三条边
   glBegin(GL_LINES);
    glNormal3f(norm.x(),norm.y(),norm.z());
         for(int i=0;i<vertNum;i++){
      glVertex3f(vert[i].x(),vert[i].y(),vert[i].z());
glVertex3f(vert[(i+1)%vertNum].x(),vert[(i+1)%vertNum].y(),vert[(i+1)%vertNum].
z());
         }
    glEnd();
   }
```

效果如下，第一张为做了扩张的，第二张为做了收缩的，可以看到轮廓线更细，更像是素描或简笔画：