Sphere Subdivision Geometry Shader

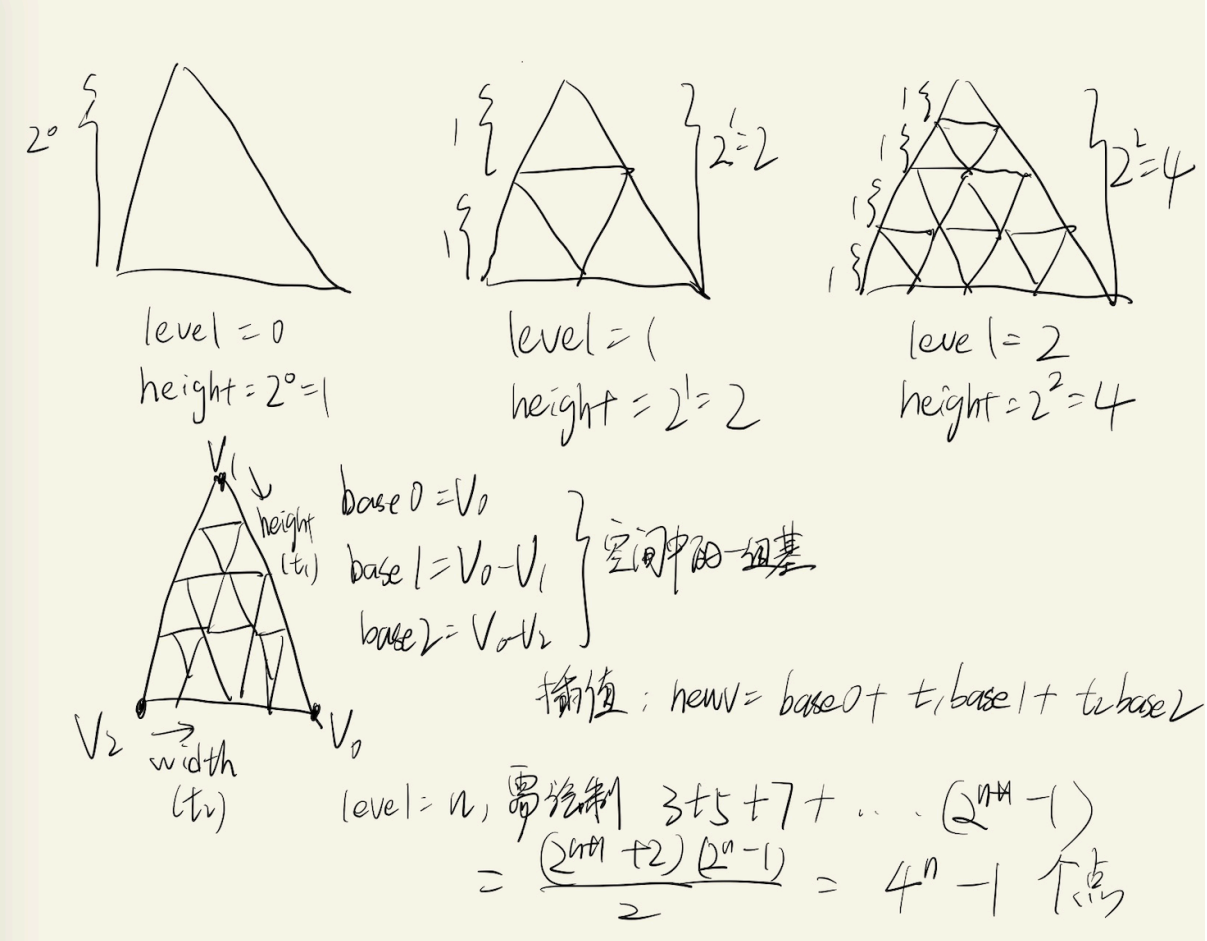
梁晨 3180102160

- 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。

- Sphere Subdivision Geometry Shade Shader的实现

  这个分形算法的思路是，我们根据三角形的三条边，计算出三维空间中的一组基，然后根据这一组基进行插值，来添加新的顶点：



  代码如下：

```
//编译宏
#version 120
#extension GL_EXT_geometry_shader4 : enable
```

```
//通过openGL传入uniform变量，即画出的圆以center为中心，以radius为半径
uniform int level;
uniform vec3 center;
uniform float radius;

//通过vertex shader传入的数据
varying in vec3 lightPos[3];
varying in vec4 vertColor[3];

//传入fragment shader的数据
varying out vec4 diffuseColor;
varying out vec3 fragNormal;
varying out vec3 lightVector;
varying out vec3 viewVector;

//给定一组基和插值量t1、t2，添加新的顶点，同时计算Blinn-Phong模型中需要用到的光照向量和法向量
void addVertex(vec3 base0, vec3 base1, vec3 base2, float t1, float t2)
{
  //计算新顶点的坐标
  vec3 newv=base0+base1*t1+base2*t2;
  newv=newv+center;
  newv=radius*normalize(newv);

  viewVector=(gl_ModelViewMatrix*vec4(newv, 1.0)).xyz;
  viewVector=-viewVector;

  //计算新顶点的光照向量
  lightVector=vec3(normalize(lightPos[0]+viewVector));

  //计算新顶点的法亮相
  vec3 norm=newv-center;
  norm=normalize(norm);
  norm=normalize(gl_NormalMatrix*norm);
  fragNormal=norm;

  //EmitVertex();
  gl_Position=gl_ModelViewProjectionMatrix * vec4(newv,1.0);
  //diffuseColor=gl_ColorIn[0];
  diffuseColor=vertColor[0];
  EmitVertex();
}

void main()
{
  //计算三组基，注意如果center不一定是原点
  vec3 base0=(gl_PositionIn[0].xyz/gl_PositionIn[0].w-center);
```

```
  vec3 base1=(gl_PositionIn[1].xyz/gl_PositionIn[1].w-
gl_PositionIn[0].xyz/gl_PositionIn[0].w);
  vec3 base2=(gl_PositionIn[2].xyz/gl_PositionIn[2].w-
gl_PositionIn[0].xyz/gl_PositionIn[0].w);

  //计算height和height方向的初始值、步长
  int height=1;
  for(int i=0;i<level;i++) height=height*2;
  float deltaH=1.0/float(height);
  float heigher=1.0;
  float lower=1.0-deltaH;
  for(int i=0;i<height;i++){
    //计算width和width方向的初始值、步长
    int width=i+1;
    float right_heigher=0.0;
    float right_lower=0.0;
    float deltaW_heigher=0.0;
    if(width>=1) deltaW_heigher= (1.0-heigher)/float(width-1);
    float deltaW_lower= (1.0-lower)/float(width);
    //插值添加顶点
    for(int j=0;j<width;j++){
      addVertex(base0,base1,base2,lower,right_lower);
      addVertex(base0,base1,base2,heigher,right_heigher);
      //width方向做更新
      right_heigher+=deltaW_heigher;
      right_lower+=deltaW_lower;
    }
    addVertex(base0,base1,base2,lower,right_lower);
    EndPrimitive();
    //height方向做更新
    heigher=lower;
    lower-=deltaH;
  }
}
```

同时，我们还需要在openGL的代码中实现geometryshader的导入和uniform变量的导入，注意geometry shader在vertex shader和fragment shader中间，编译和attach的顺序应该为vertex shader、geometry shader、fragment shader，不可随意颠倒，否则link时可能发生错误。在attach geometry shader后，我们需要指定geometry shader的input type和output type，本作业中为GL_TRIANGELS和GL_TRIANGLE_STRIP，还需要指定geometry可以输出的最大顶点数目，最大可以设置为1024。具体代码如下：

```
//loader.cpp void loadGeoShader(char*)
//配置geometry shader：
//指定input type
       glProgramParameteriEXT(programObject,
GL_GEOMETRY_INPUT_TYPE_EXT,GL_TRIANGLES);
```

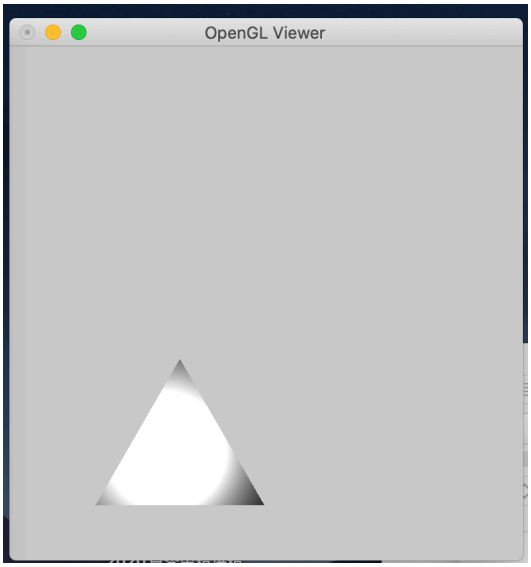
```
//指定output type
       glProgramParameteriEXT(programObject, GL_GEOMETRY_OUTPUT_TYPE_EXT,
GL_TRIANGLE_STRIP );
//指定最大输出顶点数

 glProgramParameteriEXT(programObject,GL_GEOMETRY_VERTICES_OUT_EXT,500);

//main.cpp
//传入uniform变量：
if(gO){
        glUniform1i(glGetUniformLocation(pO,"level"),2);
        glUniform1f(glGetUniformLocation(pO,"radius"),2.0);
        glUniform3f(glGetUniformLocation(pO,"center"),0.0,0.0,0.0);
    }
```

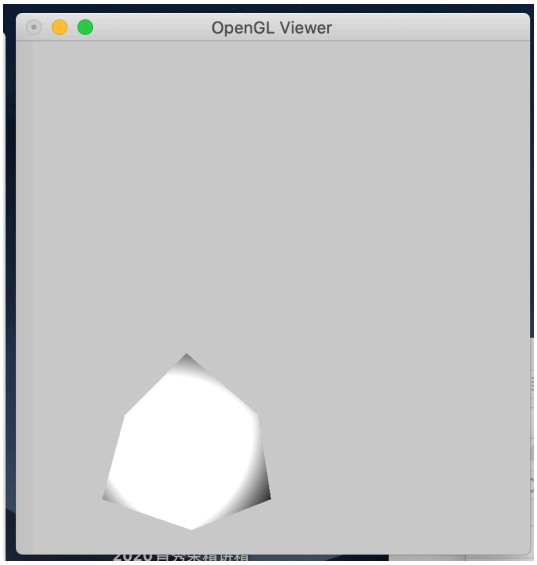
这次我们传参数时，要加上geometry shader的相关参数:

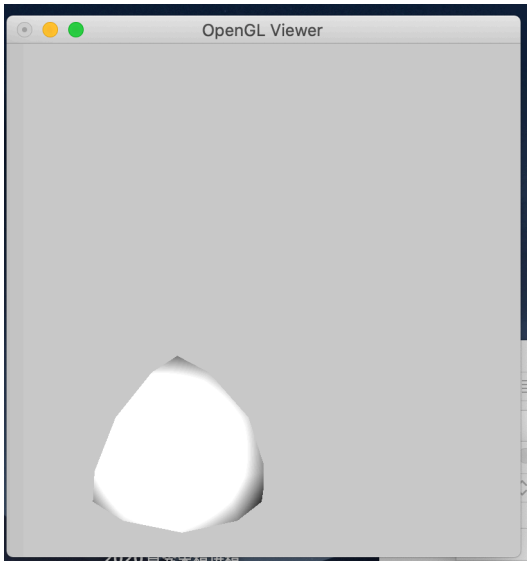
```
-input triangle.txt -vs minimal.vert -fs test.frag -gs sub.geometry
```

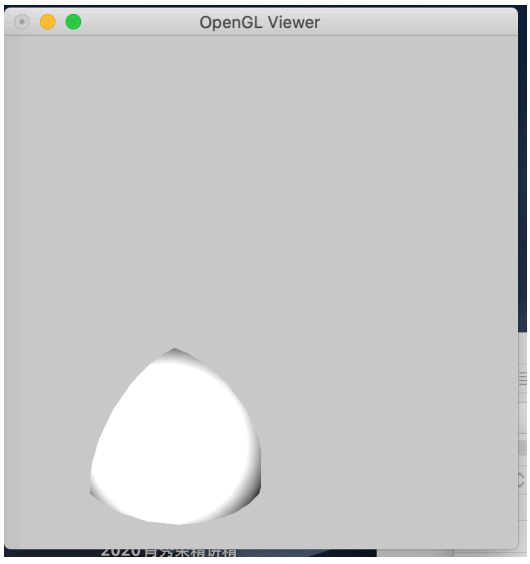
- 实验结果：

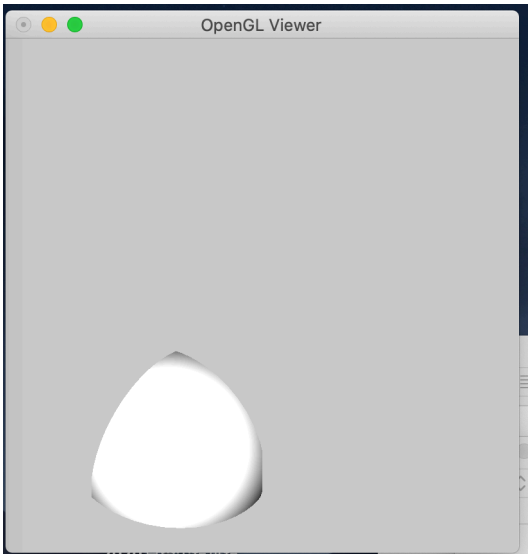
  以上各图片为level=0至level=4时的效果，fragment shader使用ivory shader：




OpenGL Viewer

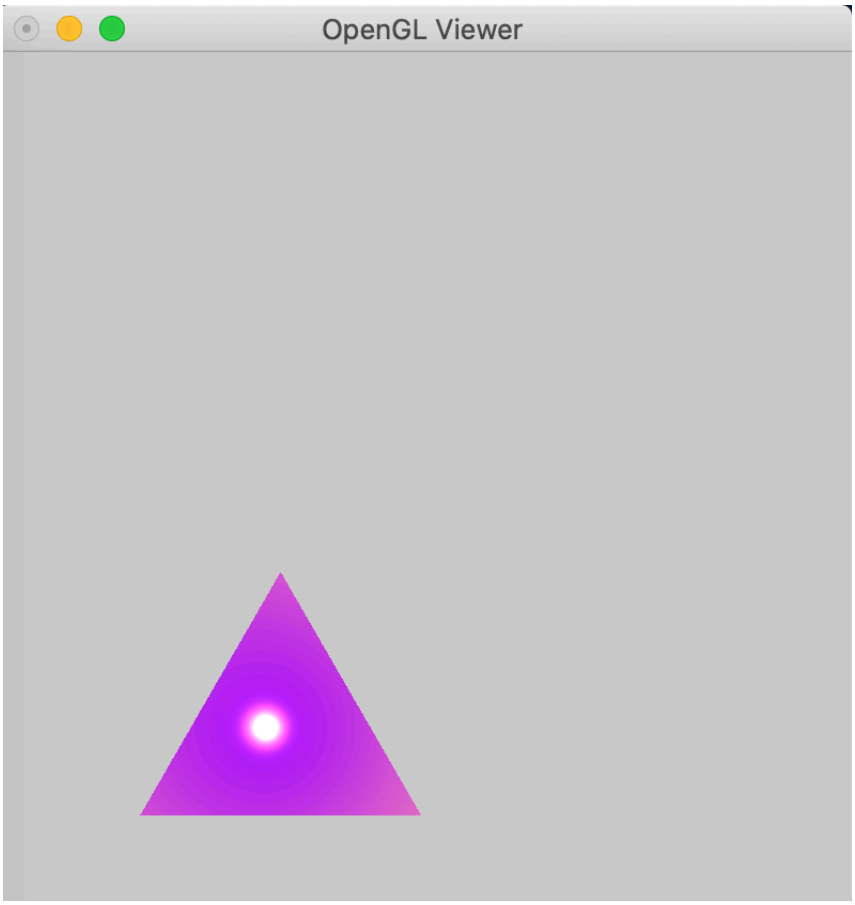

OpenGL Viewer


OpenGL Viewer

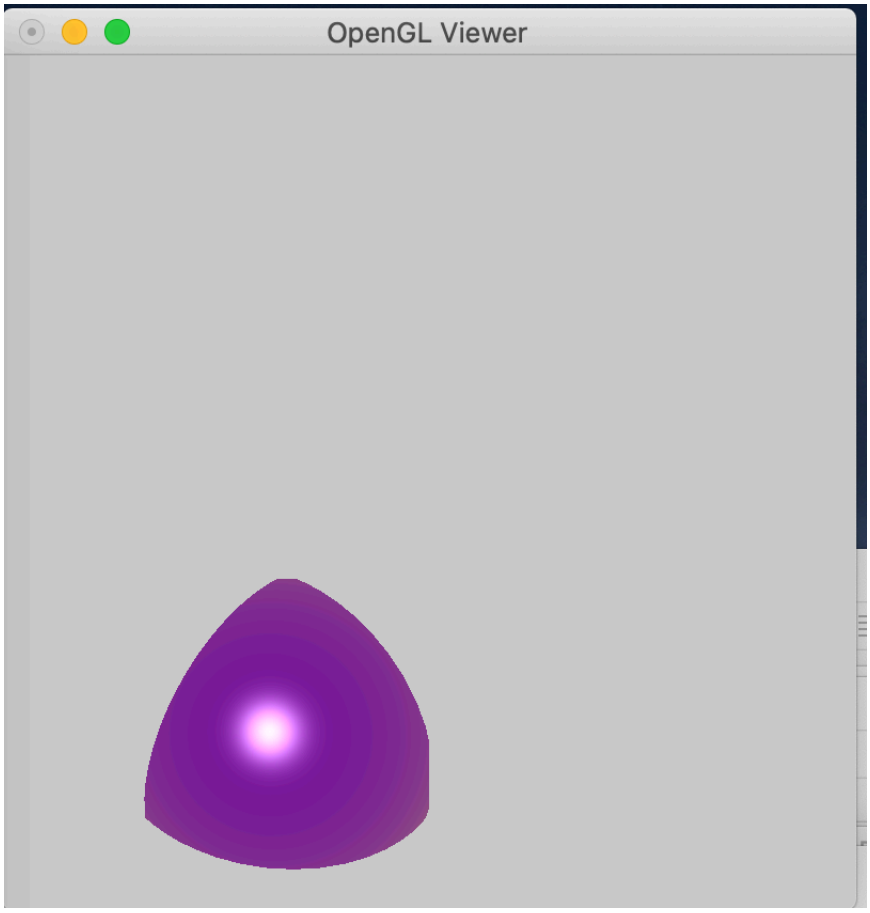


可以看出几何上，分形结果正确，为了更好地看出光照是否正确，我们将specular的指数调大，fragment shader处使用Blinn-Phong模型进行渲染：

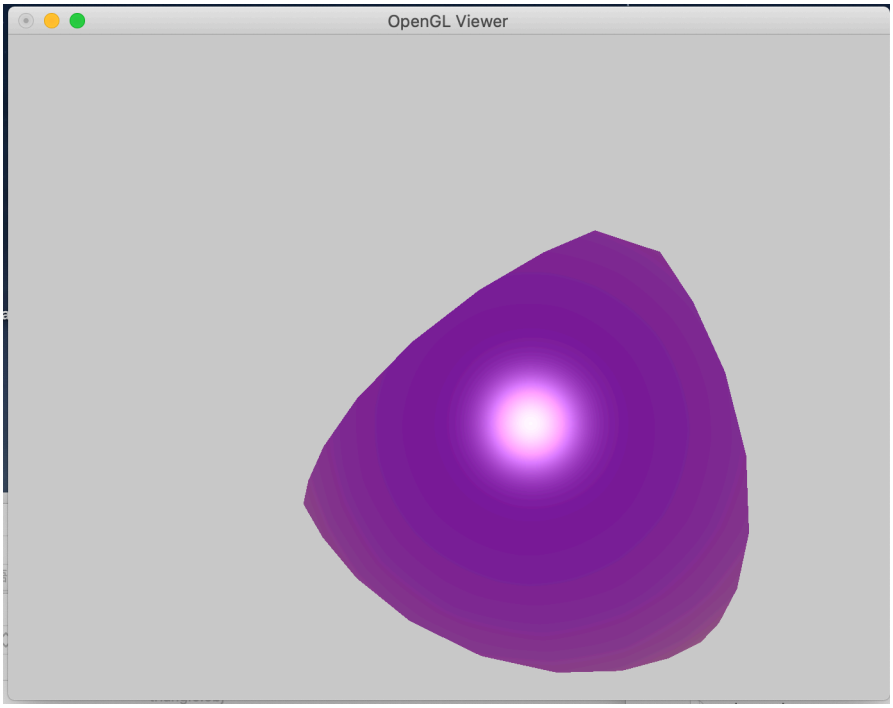
level==0时：



level==3时：

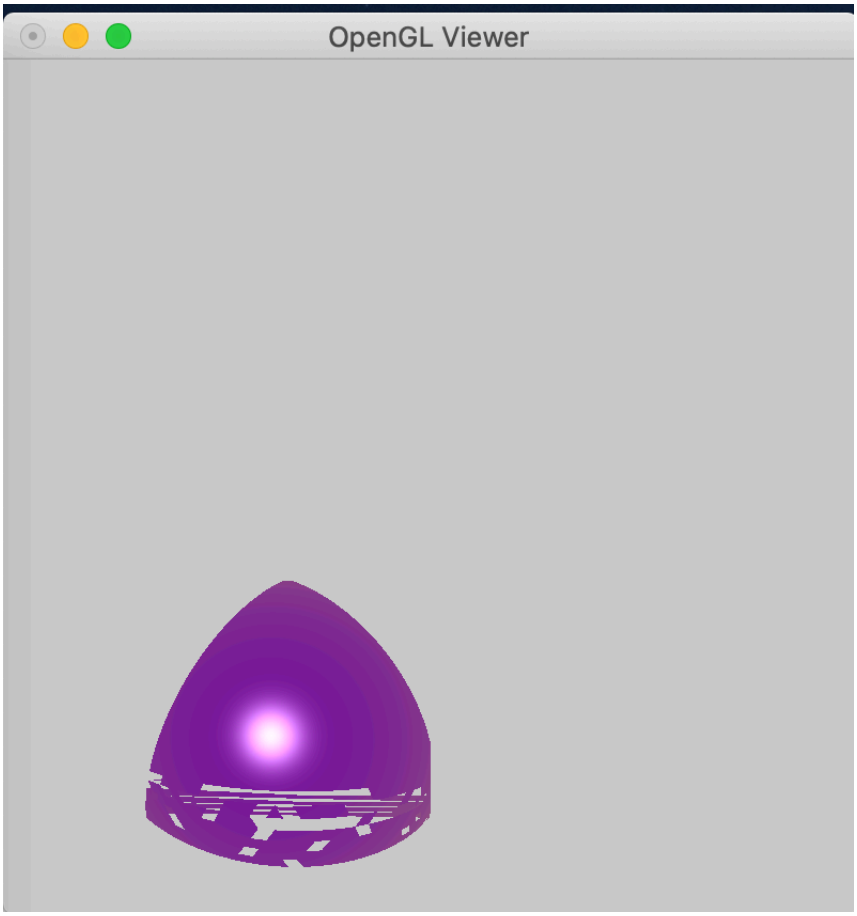


level==3时变换观察角度，可以看到高光也跟着变换位置：



另外，因为geometry shader一次最多只能输出1024个顶点，在level==5时，输出的顶点个数恰好达到1023个，但渲染结果已经出现有空洞的情况，可见geometry shader的负载不可太大。

level==5：


OpenGL Viewer