2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学　数学科学学院

http://math.ustc.edu.cn/

第四节　三维图形程序

改变到三维的情形

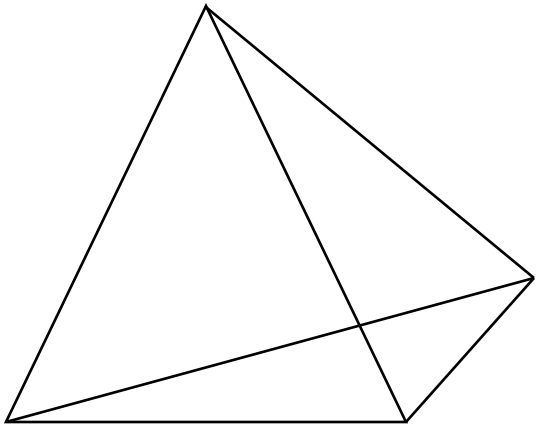
- 通过下述修改，可以很容易地把程序生成三维图形：

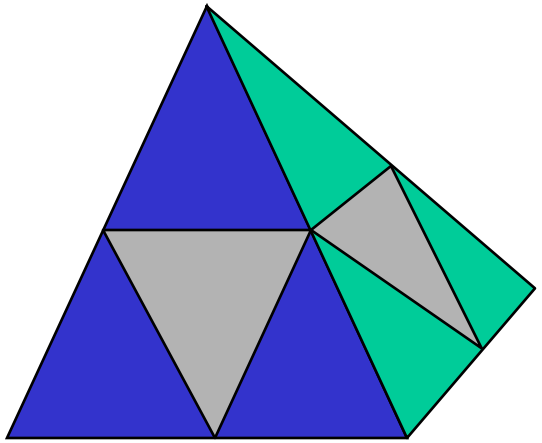
```
typedef GLfloat point3[3]
glVertex3f
glOrtho
```

- 但这并不没有多大实质性改变
- 下面从四面体开始迭代

三维Sierpinski镂垫

- 细分四面体的四个面





- 看起来好像是把四面体的中心移走，留下四个小四面体

生成三角形的代码

```
void triangle( point3 a, point3 b, point3 c)
{
//     glBegin(GL_POLYGON);
              glVertex3fv(a);
              glVertex3fv(b);
              glVertex3fv(c);
//     glEnd();
}
```

细分代码

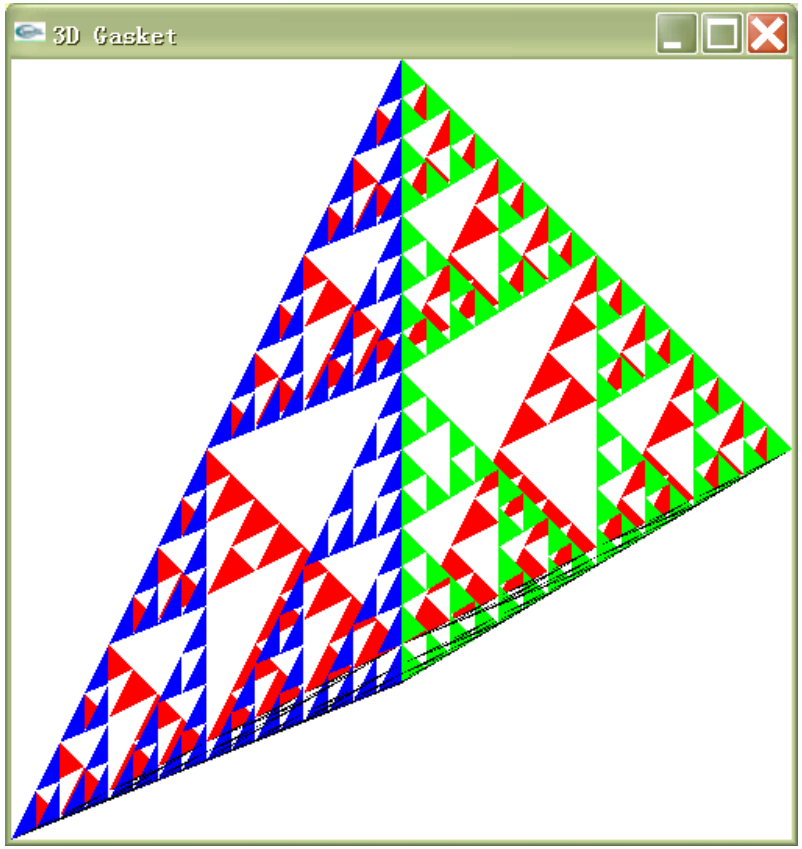
```
void divide_triangle(point3 a,point3 b,point3 c, int m)
{
        point v1, v2, v3;
        int j;
        if(m>0) {
                for(j=0; j<3; j++) v1[j]=(a[j]+b[j])/2;
                for(j=0; j<3; j++) v2[j]=(a[j]+c[j])/2;
                for(j=0; j<3; j++) v3[j]=(b[j]+c[j])/2;
                divide_triangle(a, v1, v2, m-1);
                divide_triangle(c, v2, v3, m-1);
                divide_triangle(b, v3, v1, m-1);
        }
        else(triangle(a,b,c));
}
```

生成四面体的代码

```
void tetrahedron( int m)
{
        glColor3f(1.0, 0.0, 0.0);
        divide_triangle(v[0], v[1], v[2], m);
        glColor3f(0.0, 1.0, 0.0);
        divide_triangle(v[3], v[2], v[1], m);
        glColor3f(0.0, 0.0, 1.0);
        divide_triangle(v[0], v[3], v[1], m);
        glColor3f(0.0, 0.0, 0.0);
        divide_triangle(v[0], v[2], v[3], m);
}
```

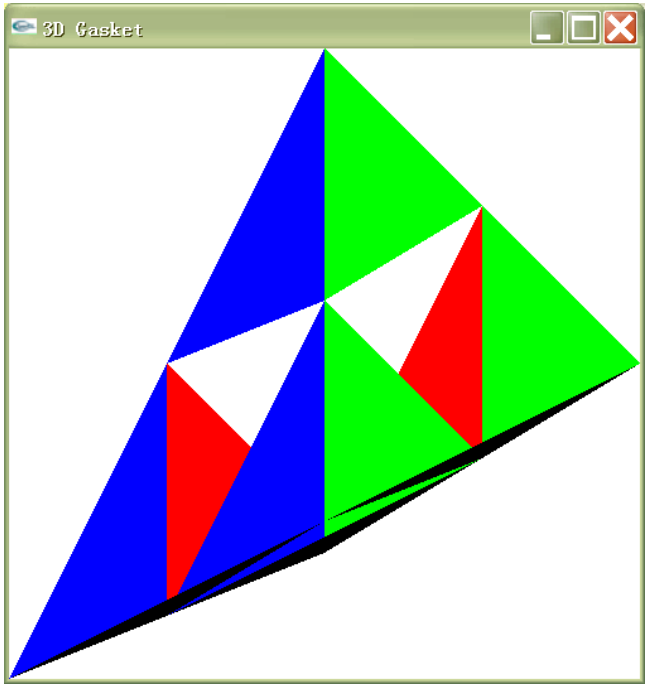
结果

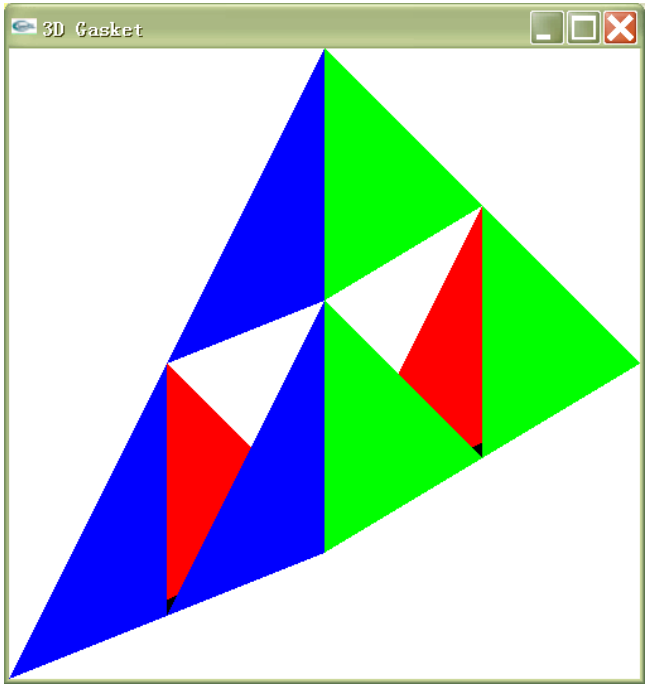
- 五次迭代后



问题

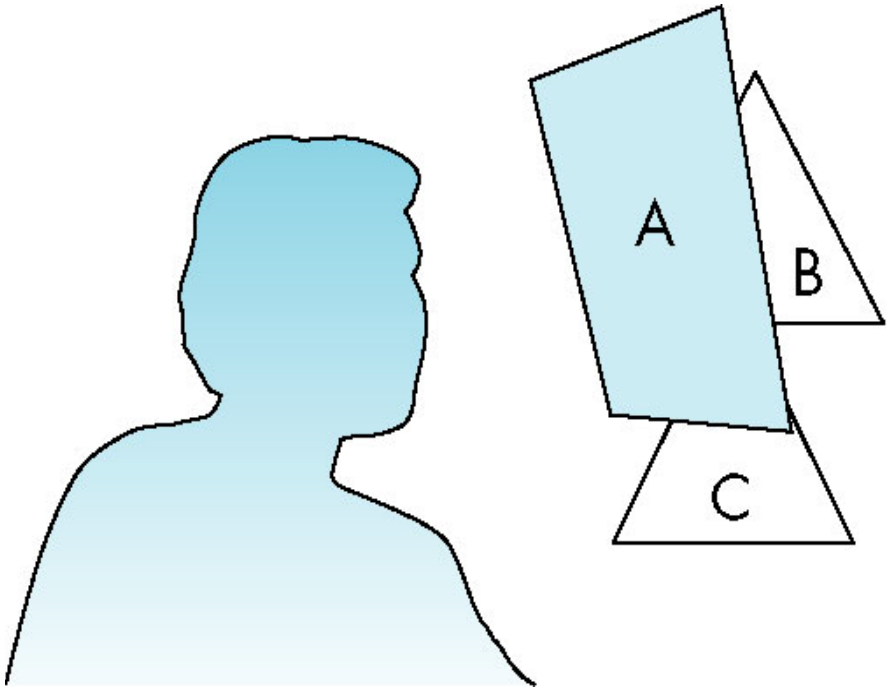
- 由于三角形是按照在程序中定义的顺序画出的，本来在前面的三角形并不是显示在位于它后面的三角形的前面





隐藏面消除

- 只想见到那些位于其它面前面的曲面或平面片
- OpenGL采用称为z缓冲区的算法进行隐藏面消除，在z缓冲区中存贮着对象的深度信息，从而只有前面的对象出现在图像中



Z缓冲区算法

- 该算法创建专门的缓冲区（称为Z缓冲区），当几何体经过流水线各步骤时，存贮着该几何体的深度信息
- 启用该算法的要素
  - 在main()中，显示模式初始化语句改为
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
  - 在init()中激活Z-Buffer隐藏面消除算法
    glEnable(GL_DEPTH_TEST);
  - 在显示回调函数中清空深度缓冲区
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

面细分 VS 体细分

- 在前面的例子中，我们对每个表面进行细分
- 也可以类似地利用中点对体进行细分
- 各边中点以及四面体的中点定义了四个小四面体，每个小四面体与大四面体的一个顶点对应
- 保留这四个小面体，把其它的去掉

绘制四面体

```
typedef float point[3];

/* initial tetrahedron */
point v[]={{0.0, 0.0, 1.0}, {0.0, 0.942809, -0.33333},
          {-0.816497, -0.471405, -0.333333}, {0.816497, -
   0.471405, -0.333333}};

int n;

void triangle( point a, point b, point c)
{
    glBegin(GL_POLYGON);
       glVertex3fv(a);
       glVertex3fv(b);
       glVertex3fv(c);
    glEnd();
}
```

绘制四面体

```
void divide_triangle(point a, point b, point c, int m)
{
    point v1, v2, v3;
    int j;
    if(m>0)
    {
        for(j=0; j<3; j++) v1[j]=(a[j]+b[j])/2;
        for(j=0; j<3; j++) v2[j]=(a[j]+c[j])/2;
        for(j=0; j<3; j++) v3[j]=(b[j]+c[j])/2;
        divide_triangle(a, v1, v2, m-1);
        divide_triangle(c, v2, v3, m-1);
        divide_triangle(b, v3, v1, m-1);
    }
    else(triangle(a,b,c)); /* draw triangle at end of recursion */
}
```

细分

```
void tetrahedron( int m)
{
   /* Apply triangle subdivision to faces of tetrahedron */
   glColor3f(1.0,0.0,0.0);
   divide_triangle(v[0], v[1], v[2], m);
   glColor3f(0.0,1.0,0.0);
   divide_triangle(v[3], v[2], v[1], m);
   glColor3f(0.0,0.0,1.0);
   divide_triangle(v[0], v[3], v[1], m);
   glColor3f(0.0,0.0,0.0);
   divide_triangle(v[0], v[2], v[3], m);
}
```

显示回调函数

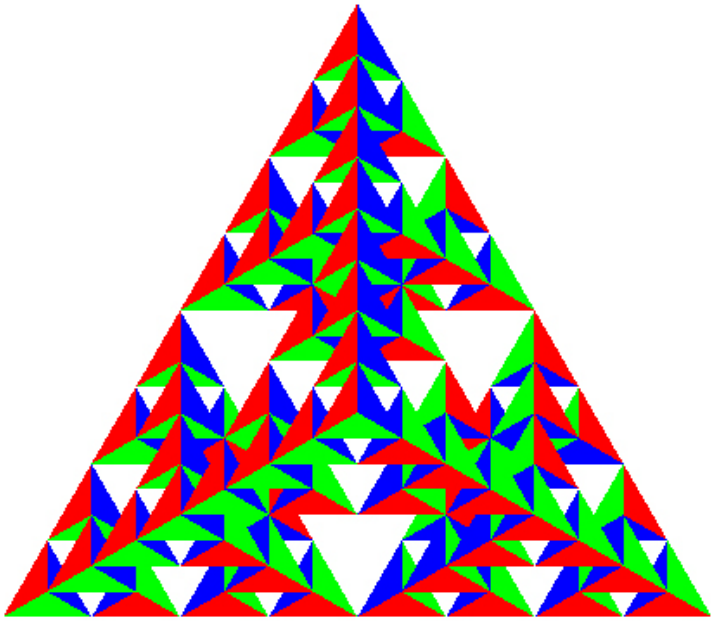
```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    tetrahedron(n);
    glFlush();
}
```

窗口改变回调函数

```
void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-2.0, 2.0, -2.0 * (GLfloat) h / (GLfloat) w,
            2.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
    else
        glOrtho(-2.0 * (GLfloat) w / (GLfloat) h,
            2.0 * (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glutPostRedisplay();
}
```

```
void
main(int argc, char **argv)
{
    n=atoi(argv[1]); /* or set number of subdivision steps here */
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("3D Gasket");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glEnable(GL_DEPTH_TEST);
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glutMainLoop();
}
```

体细分



Thanks for your attention!