计算机图形学 —— 区域扫描线Z-Buffer算法

葛林林

2018 年 12 月 15 日

1 预备知识

1.1 *obj*文件

顶点的表示：顶点以v开头后面跟着该顶点的x, y, z三轴坐标，示例如下

$$format. \quad v\ x\ y\ z$$

$$e.g. \quad v\ -57.408021\ 196.143694\ 2.816352$$

纹理坐标的表示：纹理坐标以vt开头。

$$format. \quad vt\ tu\ tv$$

法向量的表示：法向量的表示以vn开头。

$$format. \quad vn\ nx\ ny\ nz$$

$$e.g. \quad vn\ 5.9333\ -0.4798\ -1.8985$$

面的表示：面以f开头，代表”face”的意识，格式为“f 顶点索引/ 纹理坐标索引/ 顶点法向量索引”，如下所示

$$format. \quad f\ v/vt/vn\ v/vt/vn\ v/vt/vn$$

$$e.g. \quad f\ 1/1/1\ 2/2/2\ 3/3/3$$

1.2 OpenGL预备知识

1.2.1 OpenGL的坐标系

OpenGL中常用的坐标系有局部坐标系、世界坐标系、视点坐标系、投影坐标系、规格化设备坐标系和屏幕坐标系。其中规格化设备坐标系$O_dX_dY_dZ_d$的坐标范围为$\{x, y, z \in [-1, 1]\}$，它以创建的窗口的中心为原点O_d，从左向右为X_d轴正方向，从下往上为Y_d轴正方向。屏幕坐标系$O_sX_sY_s$以屏幕左上角为坐标原点，从左往右为X_s轴正方向，从上往下为Y_s轴正方向。

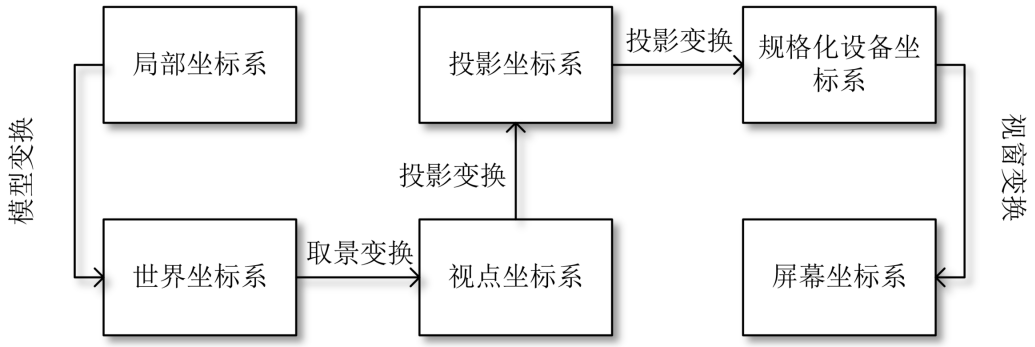

Figure 1: 常用坐标系变换

1.2.2 OpenGL工程的搭建

OpenGL是一种跨平台的图形渲染编程接口，下面总结了搭建OpenGL工程的过程。

```
void OpenGLFunc(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA | GLUT_STENCIL);
    glutInitWindowSize(800, 600);        //set window size
    glutInitWindowPosition(100, 150);    //set window position
    glutCreateWindow("Display");         //window name
    glutDisplayFunc(DisplayFunc);        //call self defined display function
    glutIdleFunc(IdleFunc);          //call self-defined idle function
    glutKeyboardFunc(KeyboardFunc); //call self-defined keyboard function
    glutSpecialFunc(SpecialFunc);        //call self-defined special function
    glutMouseFunc(MouseFunc);        //call self-defined mouse function
    glutMotionFunc(MotionFunc);      //call self-defined mouse motion function
    glutPassiveMotionFunc(PassiveMotionFunc);//call self-defined passive mouse motion function
    glutMainLoop();
}
```

上述代码是使得OpenGL程序能够正常运行的一个模板，该段程序为OpenGL产生的窗口设置回调函数：显示回调函数、空闲回调函数、键盘回调函数、特殊键回调函数，鼠标回调函数、鼠标按下移动回调函数和鼠标移动回调函数，他们分别定义在如下所示的函数中：

```
    void DisplayFunc(...){...}
    void IdelFunc(...){...}
    void KeyboardFunc(...){...}
    void SpecialFunc(...){...}
    void MouseFunc(...){...}
    void MotionFunc(...){...}
    void PassiveMotionFunc(...){...}
```

2 数据结构

- 活化多边形表
  根据y_{max}的值对多边形进行分类：
  - a, b, c, d :多边形所在平面的方程系数
  - id :多边形的编号
  - dy :多边形跨越的剩余扫描线数目
  - $color$:多边形的颜色

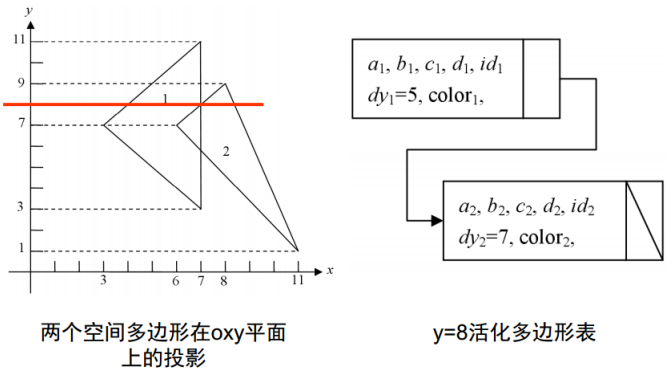

Figure 2: 分类多边形表

- 活化边表
  根据y_{max}将边进行分类：
  - x_l :左交点的x坐标。
  - dx_l :左交点边上两相邻两条扫描线交点的x坐标差。
  - dy_l :以和左交点所在边相交的扫描线数为初值，以后向下没处理一条扫描线减一。
  - x_r, dx_r, dy_r :右边的交点的三个对应分量。
  - id :边所属多边形的编号。
  - dz_x :沿扫描线向右一个像素，多边形所在平面的深度增量。$dz_x = -\frac{a}{c}(c \neq 0)$
  - dz_y :沿y方向向下移动一根扫描线时，多边形所在平面的深度增量$dz_y = \frac{b}{c}(c \neq 0)$。
  - id :交点所在多边形的编号。

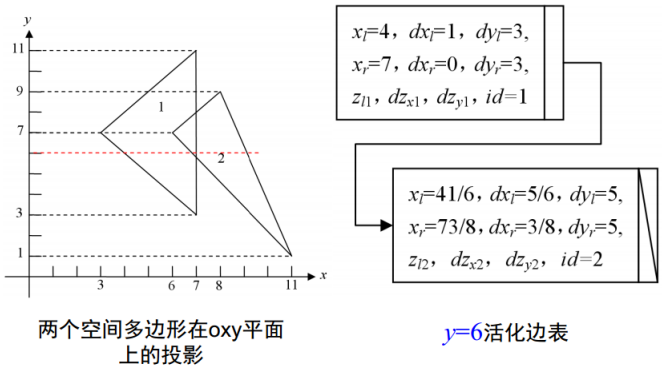

Figure 3: 分类边表

3 算法

加载文件→获取深度值→消隐算法

3.1 扫描线中in和out的讨论

3.1.1 in和out的判断

如下图所示是线段P_1P_2为in状态的情况，假设P_1, P_2, P_3点对应的坐标分别为$(x_1, y_1), (x_2, y_2), (x_3, y_3)$。则线段$P_1P_2$的斜率为

$$k = \frac{y_1 - y_2}{x_1 - x_2}$$

而线段P_1P_2对应的直线方程为：

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

令

$$f(x, y) = \frac{y - y_1}{y_2 - y_1} - \frac{x - x_1}{x_2 - x_1}$$

则当满足如下公式时则为in状态

$$sign(y_2 - y_1)kf(x_3, y_3) < 0$$

既

$$(x_2 - x_1)(y_3 - y_1) + (y_1 - y_2)(x_3 - x_1) < 0$$

Figure 4: $f(x_3, y_3) > 0, k < 0$

Figure 5: $f(x_3, y_3) < 0, k > 0$

3.1.2 in和out特点

- in和out的个数必须对应

4 测试案例的设计

在实际调试过程中会出现很多问题，为了方便调试，设计了如下所示的一些简单实例帮助调试：

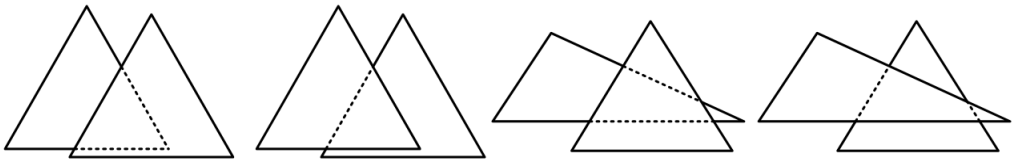

Figure 6: 示意图

Figure 7: 测试效果

5 实验结果

5.1 实验环境

本次实验的环境如下：

Table 1: 环境参数

	参数
System	Windows 10 64bit
CPU	Intel(R) Core(TM) i5-4590 CPU @3.30GHz 3.3GHz（4核）
RAM	16GB
GPU	GeForce GTX 960 (2GB显存)

5.2 程序分析

本次使用的是称为*bunny.obj*的文件，该文件中包含了v和f开头的数据，代码中*load_obj.h*和*load_obj.cpp*两个文件定义了加载*obj*文件的方法。

6 经验总结

- 用特殊的案例对结果进行测试