OpenGL图形编程

授课教师：少书师夜

创建窗口

- 1、简单图形
- 2、创建视图

2. OpenGL基本图形绘制

- 2.1glBegin/glEnd
- 2.2点的绘制
- 2.3直线的绘制
- 2.4多边形面的绘制
- 2.5OpenGL中的字符函数
- 2.6OpenGL中的反走样

2.1glBegin/glEnd

- OpenGL的图元绘制放在函数glBegin和glEnd之间，由函数glBegin的参数指定绘制图元的类型。

2.1glBegin/glEnd

表 glBegin可支持的OpenGL图元

模式	图元类型
GL_POINTS	将指定的各个顶点用于创建单个的点
GL_LINES	将指定的顶点用于创建线段。每两个顶点指定一条单独的线段。如果顶点个数是奇数，则忽略后一个
GL_LINE_STRIP	将指定的顶点用于创建线条。第一个顶点之后的每个顶点指定的是线条延伸到的下一个点
GL_LINE_LOOP	特性和GL_LINE_STRIP 相似，只不过最后一条线段是在指定的后一个和第一个顶点之间绘制。典型情况下，这用于绘制那些可能违反了 GL_POLYGON 用法规则的封闭区域
GL_TRIANGLES	将指定的顶点用于构造三角形。每三个顶点指定一个新三角形。如果顶点个数不是三的倍数，多余的顶点将被忽略

2.1glBegin/glEnd

表 glBegin可支持的OpenGL图元（续）

模式	图元类型
GL_TRIANGLE_STRIP	将指定的顶点用于创建三角条。指定前三个顶点之后，后继的每个顶点与它前面两个顶点一起用来构造下一个三角形。每个顶点三元组（在初的组之后）会自动重新排列以确保三角形绕法的 一致性。
GL_TRIANGLE_FAN	将指定的顶点用于构造三角扇形。第一个顶点充当原点，第三个顶点之后的每个顶点与它的前一个顶点还有原点一起组合。
GL_QUADS	每四个顶点一组用于构造一个四边形。如果顶点个数不是四的倍数，多余的顶点将被忽略
GL_QUADS_STRIP	将指定的顶点用于构造四条形边。在第一对顶点之后，每对顶点定义一个四边形。和 GL_QUADS 的顶点顺序不一样，每对顶点以指定顺序的逆序使用，以便保证绕法的一致
GL_POLYGON	将指定的顶点用于构造一个凸多边形。多边形的边缘决不能相交。 最后一个顶点会自动连接到第一个顶点以确保多边形是封闭的

2.2 点的绘制

点的绘制

glVertex函数用于指定顶点，可以有2，3，4个参数。

带2个参数时指定的是空间点的x，y坐标，其z坐标为默认值0，在绘制平面图形时常常使用这类函数；

带3个参数时指定的是空间点的x，y和z坐标；

带4个参数时，除了定义空间点的x，y，z坐标，还有一个不为0的w坐标。点的坐标（x，y，z，w）实际上构成了一个齐次坐标。在OpenGL中，仍然使用规范化齐次坐标以保证点的齐次坐标与三维坐标的一一对应关系，最后指定的空间点的坐标为（x/w，y/w，z/w，1），w成了坐标值的一个缩放因子。

2.2 点的绘制

- 点的绘制

```
glBegin(GL_POINTS);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(10.0f, 0.0f, 0.0f);
glEnd();
```

2.2 点的绘制

- 点的属性（大小）

在OpenGL中绘制一个点时，点大小的默认值是一个象素。可以用函数glPointSize修改这个值：

```
void glPointSize(GLfloat size);
```

这个函数采用一个参数来指定画点时以象素为单位的近似直径。

2.2 点的绘制

点的属性（大小）

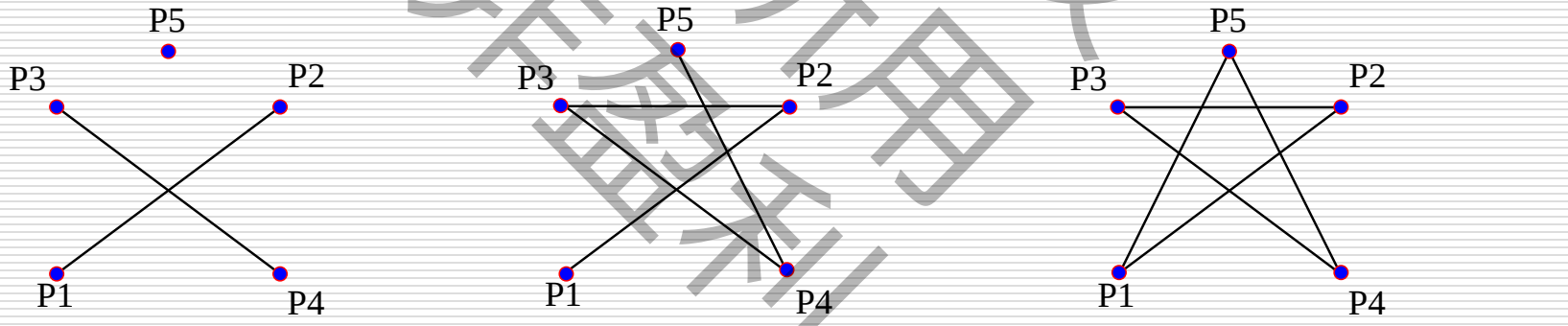
通常使用下面的代码来获取点大小的范围和它们之间最小的中间值：

```
GLfloat sizes[2];    //保存绘制点的尺寸范围
GLfloat step;        //保存绘制点尺寸的步长
glGetFloatv(GL_POINT_SIZE_RANGE, sizes);
glGetFloatv(GL_POINT_SIZE_GRANULARITY, &step);
```

在数组 seizes 中包含两个元素，分别保存了glPointSize的最小有效值和最大有效值，而变量step将保存点大小之间允许的最小增量。指定范围之外的大小不会被解释为错误，而是使用最接近指定值的可支持的最大或最小尺寸。

2.3直线的绘制

- 直线的绘制模式
  - GL_LINES
  - GL_LINE_STRIP
  - GL_LINE_LOOP



(a) GL_LINES画线模式　(b) GL_LINE_LOOP画线模式　(c) GL_LINE_STRIP画线模式

图5-57 OpenGL画线模式

2.3直线的绘制

直线的绘制

使用模式GL_LINES可以在两点之间画线，在两点(0,0,0)和(10,10,10)之间画一条直线：

```
glBegin(GL_LINES);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(10.0f, 10.0f, 10.0f);
glEnd();
```

注意，在glBegin/glEnd序列中两个顶点指定了一个直线，如果序列中指定的点为奇数个，那么最后一个顶点将被忽略。

2.3直线的绘制

直线的绘制

需要在一系列的顶点之间绘制连续直线，就要用到GL_LINE_STRIP或GL_LINE_LOOP模式。

2.3直线的绘制

直线的绘制

GL_LINE_STRIP模式可以根据指定的一系列顶点，从一个顶点到另一个顶点用连续的线段画线：

```
glBegin(GL_LINE_STRIP);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(10.0f, 10.0f, 0.0f);
    glVertex3f(20.0f, 5.0f, 0.0f);
glEnd();
```

在xy平面内绘制了两条直线(0,0,0)到(10,0,0)和(0,10,0)到(20,5,0)。

2.3 直线的绘制

- 直线的绘制

GL_LINE_LOOP模式与GL_LINE_STRIP模式类似，只是会在指定的最后一个顶点与第一个顶点之间画最后一条线。

2.3直线的绘制

- 直线的属性
  - 线宽
  - 线型

2.3直线的绘制

- 直线的属性
  - 线宽

可用glLineWidth指定线宽：

```
void glLineWidth(GLfloat width)
```

与点的大小类似，glLineWidth函数采用一个参数来指定要画的线以象素计的近似宽度 。

2.3直线的绘制

□直线的属性

■线宽

可以用下面的代码来获取线宽范围和它们之间的最小间隔：

```
GLfloat sizes[2];    //保存线宽的尺寸范围
GLfloat step;        //保存线宽尺寸的最小间隔
glGetFloarv(GL_LINE_WIDTH_RANGE, sizes);
glGetFloatv(GL_LINE_WIDTH_GRANULARITY, &step);
```

数组 sizes中保存了glLineWidth的最小有效值和最大有效值，而变量step将保存线宽之间允许的最小增量。OpenGL规范只要求支持一种线宽：1.0。Microsoft的OpenGL实现允许线宽从0.5到10.0，最小增量为 0.125。

2.3直线的绘制

- 直线的属性
  - 线型

可以用虚线或短划线模式创建直线，需要先调用：

```
glEnable(GL_LINE_STIPPLE);
```

然后，建立用于画线的模式：

```
glLineStipple(GLint factor，GLushort pattern);
```

参数pattern是一个 16 位值，指定画线时所用的模式，逆向使用。每一位代表线段的一部分是开还是关。默认情况下，每一位对应一个象素。

参数factor充当倍数可以增加模式的宽度。

作业

通过opengl画多条线

OpenGL图形编程

授课教师：少书师夜

2.4 多边形面的绘制

- 三角形面的绘制
  - GL_TRIANGLES
  - GL_TRIANGLE_STRIP
  - GL_TRIANGLE_FAN
- 四边形面的绘制
  - GL_QUADS
  - GL_QUAD_STRIP
- 多边形面的绘制（GL_POLYGON）

2.4多边形面的绘制

三角形面的绘制

在OpenGL中，面是由多边形构成的。

三角形是最简单的多边形，有三条边。可以使用GL_TRIANGLES模式通过把三个顶点连接到一起而绘出三角形。

2.4 多边形面的绘制

- 三角形面的绘制

以下绘制了一个三角形：

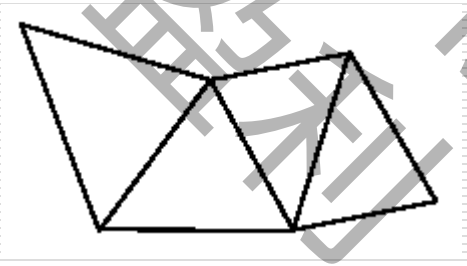
```
glBegin(GL_TRIANGLES);
    glVertex2f(0.0, 0.0);
    glVertex2f(15.0, 15.0);
    glVertex2f(30.0, 0.0);
glEnd();
```

注意：这里三角形将被用当前选定的颜色填充，如果尚未指定绘图的颜色，结果将是不确定的。

2.4 多边形面的绘制

三角形面的绘制

使用GL_TRIANGLE_STRIP模式可以绘制几个相连的三角形，系统根据前三个顶点绘制第一个多边形，以后每指定一个顶点，就与构成上一个三角形的后两个顶点绘制新的一个三角形。

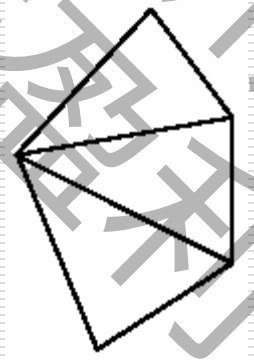


GL_TRIANGLE_STRIP模式

2.4 多边形面的绘制

三角形面的绘制

使用GL_TRIANGLE_FAN模式可以绘制一组相连的三角形，这些三角形绕着一个中心点成扇形排列。第一个顶点构成扇形的中心，用前三个顶点绘制出最初的三角形之后，随后的所有顶点都和扇形中心以及紧跟在它前面的顶点构成下一个三角形，此时是以顺时针方向穿过顶点。



GL_TRIANGLE_FAN模式

2.4 多边形面的绘制

- 四边形面的绘制
  - GL_QUADS
  - GL_QUAD_STRIP
- 多边形面的绘制（GL_POLYGON）

2.4 多边形面的绘制

- 多边形面的属性
  - 正反（绕法）
  - 颜色
  - 显示模式
  - 填充
  - 法向量

2.4 多边形面的绘制

- 多边形面的属性
  - 正反（绕法）

指定顶点时顺序和方向的组合称为“绕法”。绕法是任何多边形图元的一个重要特性。

一般默认情况下，OpenGL认为逆时针绕法的多边形是正对着的。以逆时针顺序显示的多边形称为正面

```
glFrontFace(GL_CW);
```

CL_CW告诉OpenGL应该把顺时针缠绕的多边形为正对着的。为了改回把逆时针绕法视为正面，可以使用 CL_CCW。

2.4 多边形面的绘制

□ 多边形面的属性

■ 颜色

在绘制多边形时，常常要指定绘制的颜色。在OpenGL中，颜色实际上是对各个顶点而不是对各个多边形指定的。

多边形的轮廓或者内部用单一的颜色或许多不同的颜色来填充的处理方式称为明暗处理。

在OpenGL中，用单一颜色处理的称为平面明暗处理（Flat Shading），用许多不同颜色处理的称为光滑明暗处理（Smooth Shading），也称为Gourand明暗处理（Gourand Shading）。

2.4 多边形面的绘制

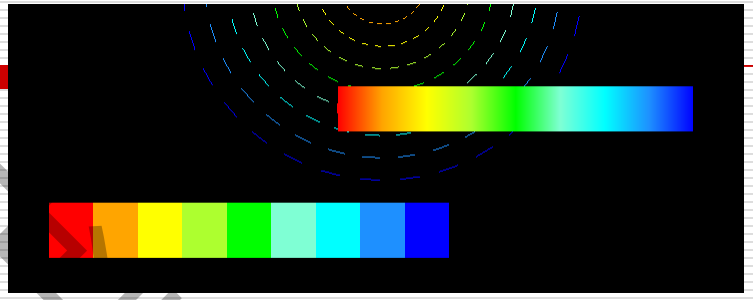
- 多边形面的属性
  - 颜色

设置明暗处理模式的函数为：

```
void glShadeModel(GLenum mode);
```

其中参数mode的取值为GL_FLAT或GL_SMOOTH，分别表示平面明暗处理和光滑明暗处理。

2.4 多边形面的绘制



- 多边形面的属性
  - 颜色

glShadeModel(GL_FLAT) 指定多边形最后一个顶点时的当前颜色作为填充多边形的纯色，唯一例外是GL_POLYGON图元，它采用的是第一个顶点的颜色。

glShadeModel(GL_SMOOTH) 从各个顶点给三角形投上光滑的阴影，为各个顶点指定的颜色之间进行插值。

2.4 多边形面的绘制

- 多边形面的属性
  - 显示模式

多边形不是必须用当前颜色填充的。默认情况下绘制的多边形是实心的，但可以通过指定把多边形绘制为轮廓或只是点（只画出顶点）来修改这项默认行为。

函数glPolygonMode允许把多边形渲染为填充的实心、轮廓线或只是点。可以把这项渲染模式应用到多边 形的两面或只应用到正面或背面。

2.4 多边形面的绘制

- 多边形面的属性
  - 显示模式

glPolygonMode(GLenum face，GLenum mode);

参数face用于指定多边形的哪一个面受到模式改变的影响，取GL_FRONT，GL_BACK 或GL_FRONT_AND_BACK。

参数mode用于指定新的绘图模式，GL_FILL是默认值，生成填充的多边形；GL_LINE生成多边形的轮廓；而GL_POINT只画出顶点。

2.4 多边形面的绘制

- 多边形面的属性
  - 法向量

法向量是垂直于面的方向上点的向量，它确定了几何对象在空间中的方向。

在OpenGL中，可以为每个顶点指定法向量。

```
void glNormal3{bsidf} ( TYPE nx, TYPE ny, TYPE nz);
void glNormal3{bsidf}v (const TYPE* v);
```

2.4 多边形面的绘制

- 多边形面的绘制规则
  - 所有多边形都必须是平面的。

多边形的所有顶点必须位于一个平面上，不能在空间中扭曲。

  - 多边形的边缘决不能相交，而且多边形必须是凸的。

如果有任意两条边交叉，就称这个多边形与自己相交。

“凸的”是指任意经过多边形的直线进入和离开多边形的 次数不超过一次。

2.4 多边形面的绘制

- 非凸多边形

可以把它分割成几个凸多边形（通常是三角形），再将它绘制出来。

问题：如果这样的多边形被填充时看不到任何边缘，但轮廓图形状态会看到组成大表面的所有小三角形。

OpenGL提供了一个特殊标记来处理这些边缘，称为边缘标记。

2.4多边形面的绘制

□ 边缘标记

在指定顶点的列表时，通过设置和清除边缘标记，可以通知OpenGL哪些线段被认为是边线（围绕图形边界的线），哪些线段不是（不应该显示的内部线段）。

glEdgeFlag函数用一个参数把边缘标记设为True或false。当函数被设置为True 时，后面的顶点将标记出边界线段的起点。

glEdgeFlag（GL_TRUE）

glEdgeFlag（GL_FALSE）

2.6OpenGL中的反走样

- 在光栅图形显示器上绘制非水平且非垂直的直线或多边形边界时，或多或少会呈现锯齿状或台阶状外观。这是因为直线、多边形、色彩边界等是连续的，而光栅则是由离散的点组成，在光栅显示设备上表现直线、多边形等，必须在离散位置采样。由于采样不充分重建后造成的信息失真，就叫走样(aliasing)。而用于减少或消除这种效果的技术，就称为反走样(antialiasing)。
- 二种方法
  - RGBA颜色模式混合
  - 多重采样

混合

- 混合就是把两种颜色混在一起。具体一点，就是把某一像素位置原来的颜色和将要画上去的颜色，通过某种方式混在一起，从而实现特殊的效果。
  假设我们需要绘制这样一个场景：透过红色的玻璃去看绿色的物体，那么可以先绘制绿色的物体，再绘制红色玻璃。在绘制红色玻璃的时候，利用“混合”功能，把将要绘制上去的红色和原来的绿色进行混合，于是得到一种新的颜色，看上去就好像玻璃是半透明的。
- 要使用OpenGL的混合功能，只需要调用：glEnable(GL_BLEND);即可。
  要关闭OpenGL的混合功能，只需要调用：glDisable(GL_BLEND);即可。

2.6OpenGL中的反走样

- RGBA颜色模式混合

1.启用反走样

```
glEnable(primitiveType);
```

2.启用OpenGL颜色混和并指定颜色混合函数

```
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA,
GL_ONE_MINUS_SRC_ALPHA);
```

2.6OpenGL中的反走样

- OpenGL实现反走样需要满足两个条件，一是启用混合，二是启用针对几何图元的反走样处理。
- 启动混合glEnable(GL_BLEND)
- 启用几何图元的反走样glEnable(mode)
- 其中mode取值为GL_POINT_SMOOTH、GL_LINE_SMOOTH或GL_POLYGON_SMOOTH。

2.6OpenGL中的反走样

- void glHint(GLenum target,GLenum hint);
- target定义反走样的对象
- GL_POINT_SMOOTH_HINT 指定点
- GL_LINE_SMOOTH_HINT 线
- GL_POLYGON_SMOOTH_HINT 多边形的采样质量
- GL_FOG_HINT 指出雾化计算是按每个象素进行（GL_NICEST），还是按每个顶点进行（GL_FASTEST）
- GL_PERSPECTIVE_CORRECTION_HINT 指定颜色纹理插值的质量，用以纠正单纯线性插值带来的观察错误。
- hint定义了反走样的方法
- GL_FASTEST 给出最有效的选择
- GL_NICEST 给出最高质量的选择
- GL_DONT_CARE 没有选择

作业

- 做一个三角形、一个四边形、一个多边形
- 要求：颜色+反走样