華東理工大學

计算机图形学

2023年9月

奉贤校区

05

基本图形生成算法

Basic Graphics Generation

绘制流水线

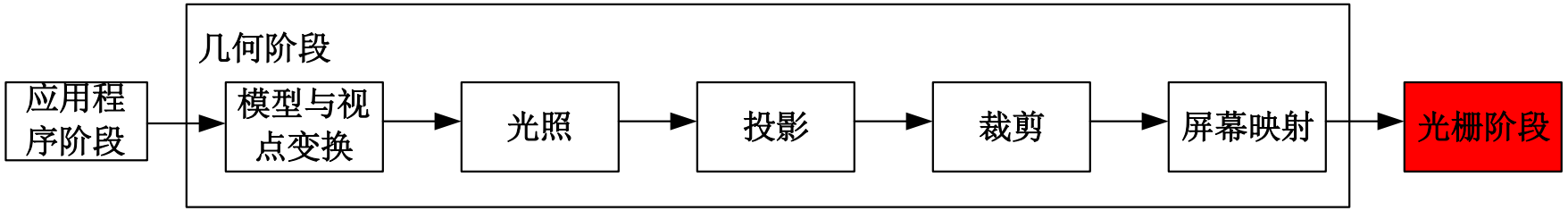
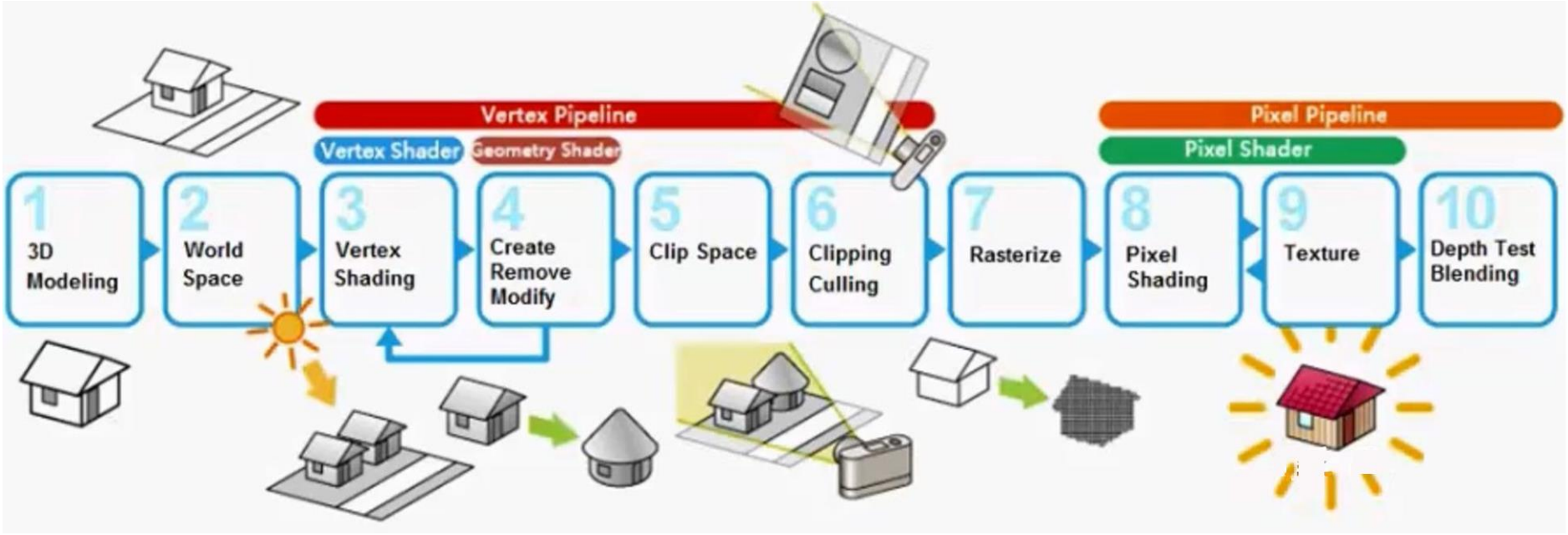


图2.22 绘制流水线的结构


Vertex Pipeline
Vertex Shader
Geometry Shader
Pixel Pipeline
Pixel Shader
1
3D Modeling
2
World Space
3
Vertex Shading
4
Create Remove Modify
5
Clip Space
6
Clipping Culling
7
Rasterize
8
Pixel Shading
9
Texture
10
Depth Test Blending

- 快速
- 真实逼近

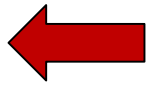
主要内容

如何在指定的输出设备上根据坐标描述构造基本二维几何图形

- 直线、圆
- 多边形域
- 字符串
- 相关属性
- 反走样

字符处理

1. 点阵字符



2. 矢量字符

“因乱用字体被方正告”事件

醉里挑灯看剑 方正兰亭大黑

梦回吹角连营 方正兰亭中黑

八百里分麾下炙 方正兰亭细黑

沙场秋点兵 方正兰亭纤细黑

马作的卢飞快，弓如霹雳弦惊。了却君王天下事，赢得生前身后名。可怜白发生！

方正兰亭黑

字符处理

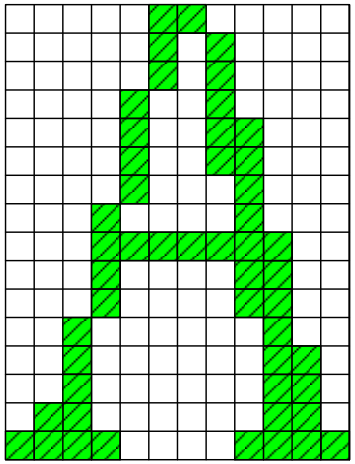
- ASCII码：“美国信息交换用标准代码集”(American Standard Code for Information Interchange)，简称ASCII码。
- 国际码：“中华人民共和国国家标准信息交换编码，简称为国际码，代号GB2312－80。
- 字库：字库中储存了每个字符的图形信息。
- 矢量字库和点阵字库。

字符处理——点阵字符

- 在点阵表示中，每个字符由一个点阵位图来表示。
- 显示时：形成字符的象素图案。

0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	0
0	0	0	0	1	0	0	1	1	0	0	0
0	0	0	0	1	0	0	1	1	0	0	0
0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	0	1	0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	1	1	0
0	1	1	0	0	0	0	0	0	1	1	0
1	1	1	1	0	0	0	0	1	1	1	1

(a)字符A的点阵位图



(a)字符A的象素图案

图5-33 字符A的点阵表示

字符处理——点阵字符

特点：

（1）定义和显示直接、简单。

（2）存储需要耗费大量空间。

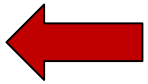
解决方法：

（1）从一组点阵字符生成不同尺寸和不同字体的其他字符。

（2）采用压缩技术

字符处理

1. 点阵字符
2. 矢量字符



字符处理——矢量字符

- 矢量字符采用直线和曲线段来描述字符形状，矢量字符库中记录的是笔划信息。
- 显示时：解释字符的每个笔划信息。

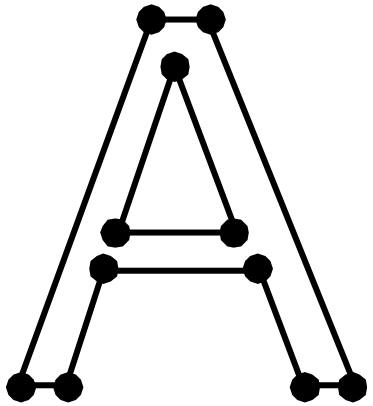


图5-34 字符A的矢量表示

字符处理——矢量字符

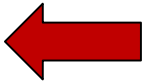
- 常用的矢量字符表示：轮廓字型法
- 轮廓字型法采用直线或二、三次曲线的集合来描述一个字符的轮廓线，轮廓线构成了一个或若干个封闭区域，显示时只要填充这些区域就可产生相应的字符。
- 显示时可以“无极放缩”
- 占用空间少，美观，变换方便

属性处理

- 图素或图段的外观由其属性决定。
- 图形软件中实现属性选择的方法是提供一张系统当前属性值表，通过调用标准函数提供属性值表的设置和修改。进行扫描转换时，系统使用属性值表中的当前属性值进行显示和输出。

属性处理

1. 线型和线宽



2. 字符属性

3. 区域填充属性

线型处理

- 线型的显示可用象素段方法实现
- 针对不同的线型，画线程序沿路径输出一些连续象素段，在每两个实心段之间有一段中间空白段，他们的长度（象素数目）可用象素模板(pixel mask)指定。
- 存在问题：如何保持任何方向的划线长度近似地相等

线型处理

- 可根据线的斜率来调整实心段和中间空白段的象素数目

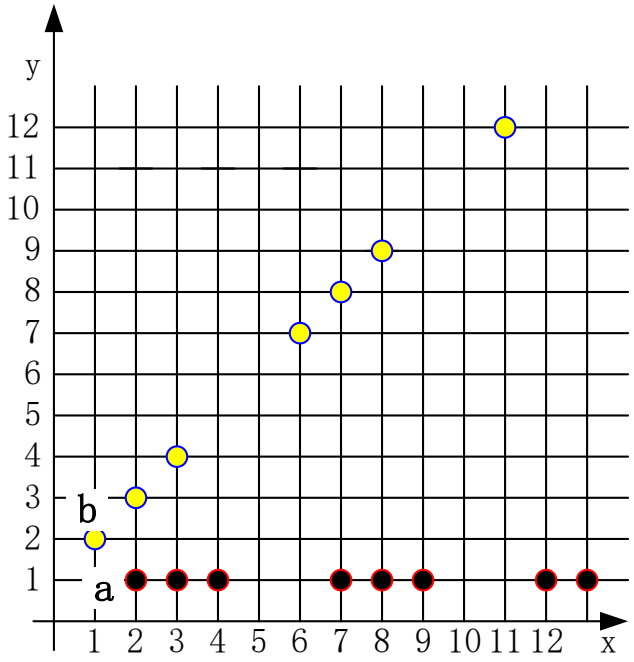


图5-35

线宽处理——线刷子

- 线刷子和方刷子处理线宽
- 线刷子：垂直刷子、水平刷子

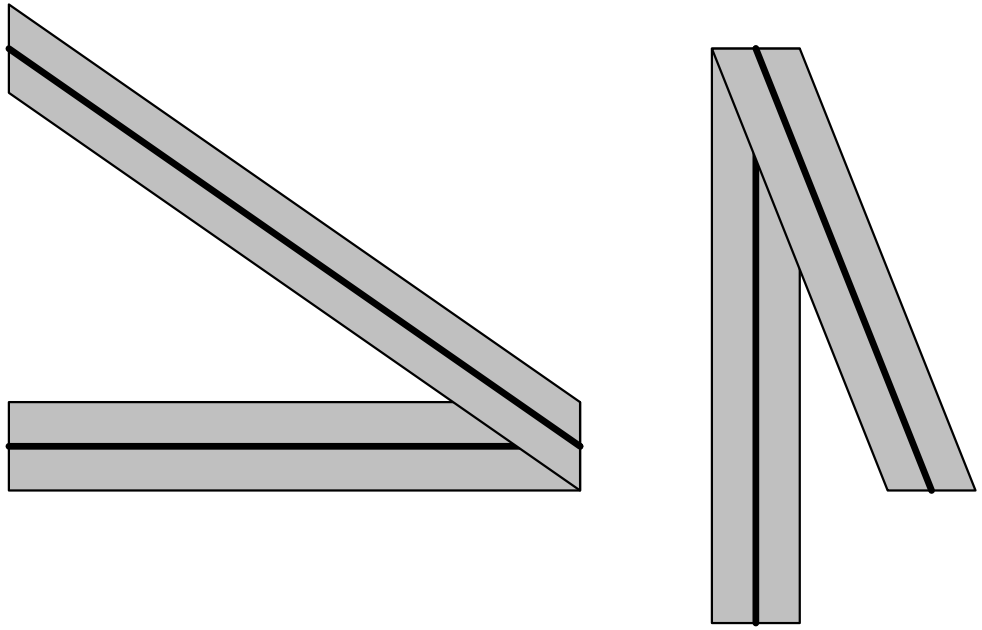


图5-36

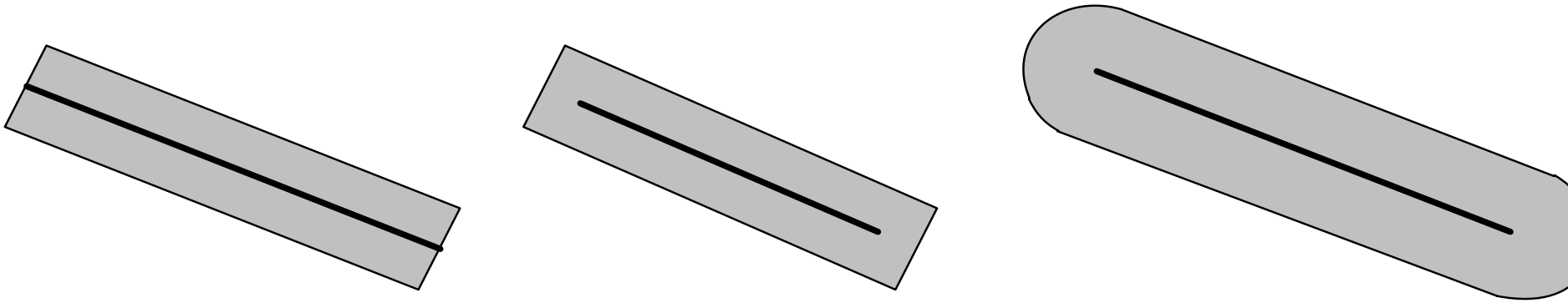
线宽处理——线刷子

特点：

- 实现简单、效率高。
- 斜线与水平(或垂直)线不一样粗。
- 当线宽为偶数个象素时，线的中心将偏移半个象素。
- 利用线刷子生成线的始末端总是水平或垂直的，需要添加“线帽（line cap）”

线宽处理——线刷子

- 方帽：调整端点位置，使粗线的显示具有垂直于线段路径的正方形端点。
- 凸方帽：简单将线向两头延伸一半线宽并添加方帽。
- 圆帽：通过对每个方帽添加一个填充的半圆得到



(a) 方帽　　(b) 凸方帽　　(c) 圆帽

图5-37 线帽

线宽处理——线刷子

- 当比较接近水平的线与比较接近垂直的线汇合时，汇合处外角将有缺口

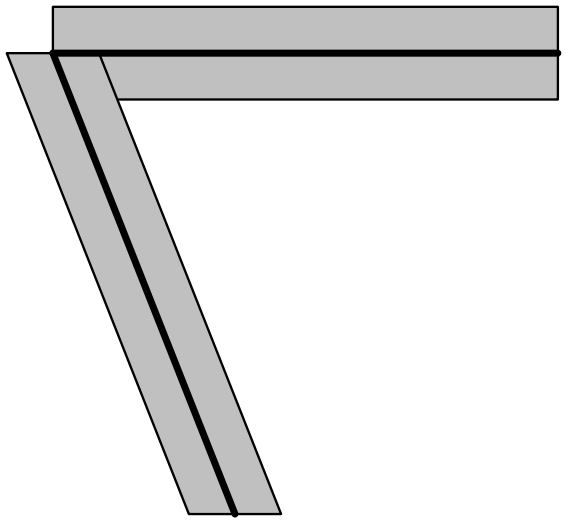


图5-38 线刷子产生的缺口

线宽处理——线刷子

- 解决：斜角连接（miter join）、圆连接（round join）、斜切连接（bevel join）

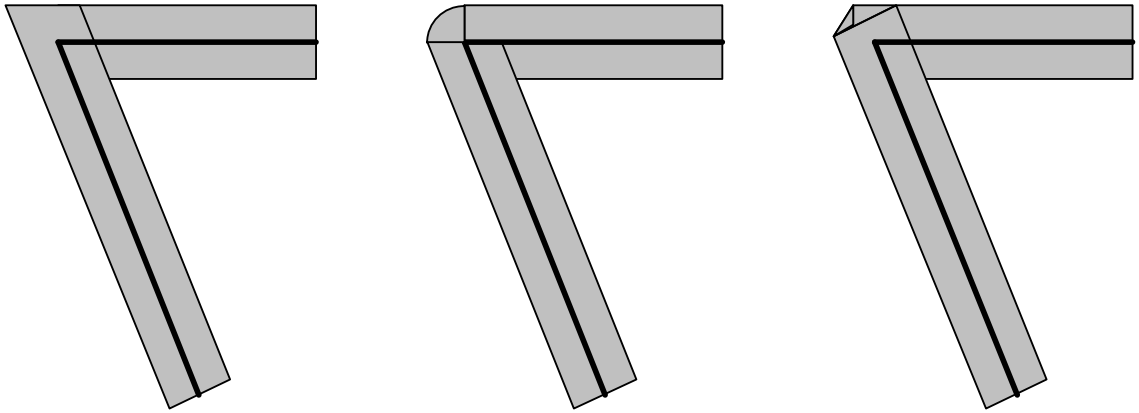


图5-39 解决线刷子产生的缺口

线宽处理——方刷子

- 方刷子绘制的线条（斜线）比用线刷子所绘制的线条要粗一些
- 方刷子绘制的斜线与水平（或垂直）线不一样粗
- 方刷子绘制的线条自然地带有一个“方线帽”

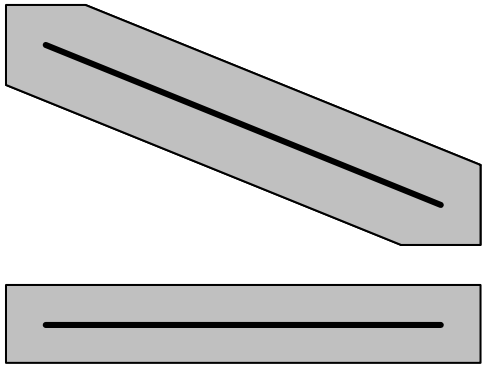


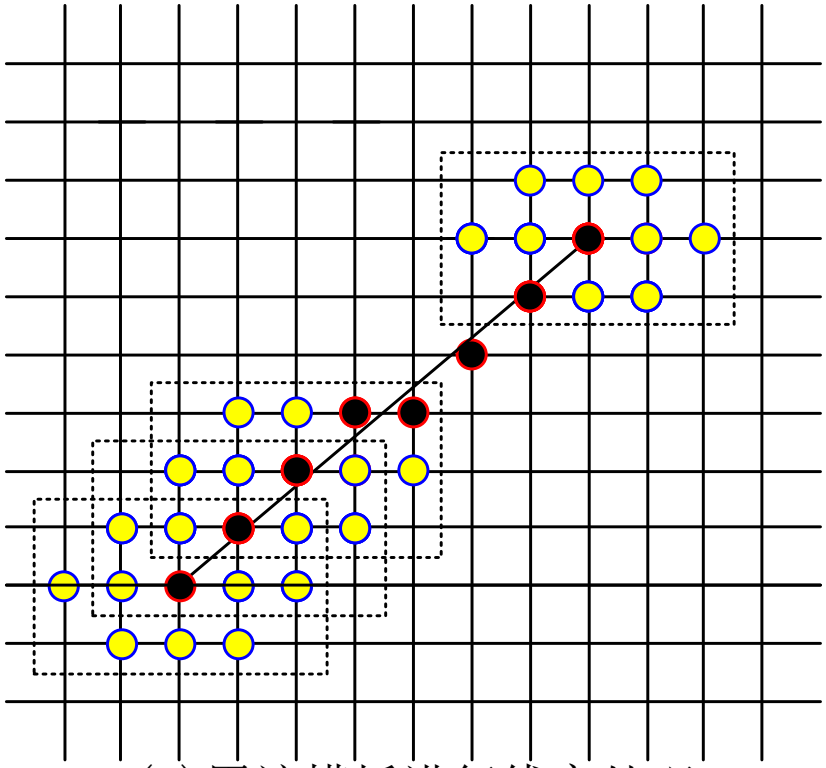
图5-40 方刷子

线宽处理——其他方式

- 区域填充
- 改变刷子形状

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

(a) 象素模板



(b) 用该模板进行线宽处理

图5-41 利用象素模板进行线宽处理

线宽处理——曲线线型

- 线型可采用象素模板的方法

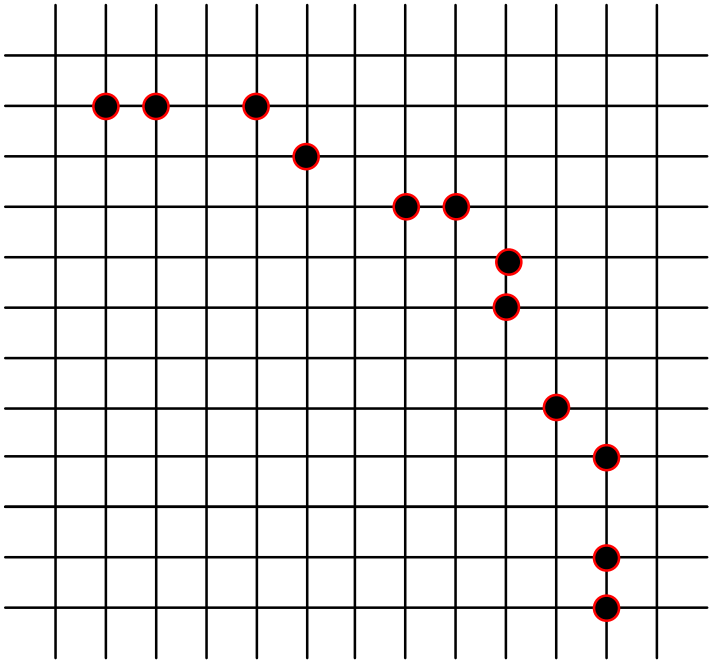


图5-42 利用模板110进行圆的线型处理

线宽处理——曲线线型

- 从一个八分象限转入下一个八分象限时要交换象素的位置，以保持划线长度近似相等。
- 在沿圆周移动时调整画每根划线的象素数目以保证划线长度近似相等。
- 改进：可以采用沿等角弧画象素代替用等长区间的象素模板来生成等长划线。

线宽处理——曲线线宽

- 线刷子

  经过曲线斜率为1和－1处，必须切换刷子。

  曲线接近水平与垂直的地方，线条更粗。

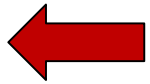
- 方刷子：接近水平垂直的地方，线条更细

  要显示一致的曲线宽度可通过旋转刷子方向以使其在沿曲线移动时与斜率方向一致

- 圆弧刷子或者采用填充的办法.

属性处理

1. 线型和线宽
2. 字符属性
3. 区域填充属性



字符属性

- 字体、字形、字号、字间距、行间距等等。
- 一般字体确定风格，字形确定外观，字号确定尺寸。

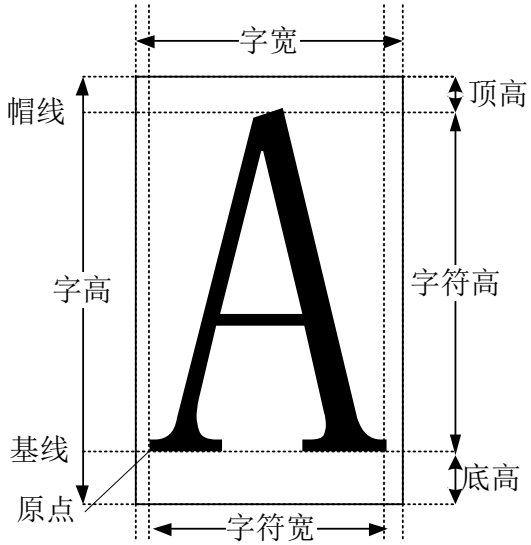


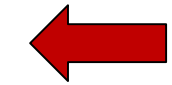
图5-43 字符的常用属性及其含义

字符串属性

- 文本高度、文本宽度（扩展/压缩因子）、字符方向、文本路径方向、对齐方式（左对齐，中心对齐，或右对齐，指定起始、终止点）、文本字体、字符的颜色属性等。
- 反绘（从右到左）、倒绘（旋转180°）、写方式（替换或与方式）等。

属性处理

1. 线型和线宽
2. 字符属性
3. 区域填充属性

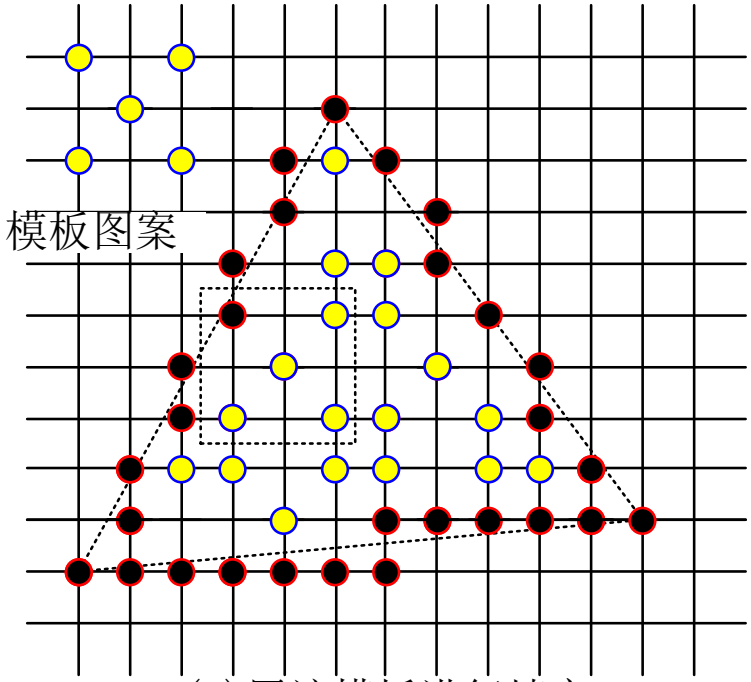


区域填充属性

- 颜色、图案和透明度

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

(a) 图案模板位图



(b) 用该模板进行填充

图5-44 用模板进行区域填充

区域填充属性

- 首先用模板定义各种图案。
- 然后，修改填充的扫描转换算法：在确定了区域内一象素之后，不是马上往该象素填色而是先查询模板位图的对应位置。

区域填充属性

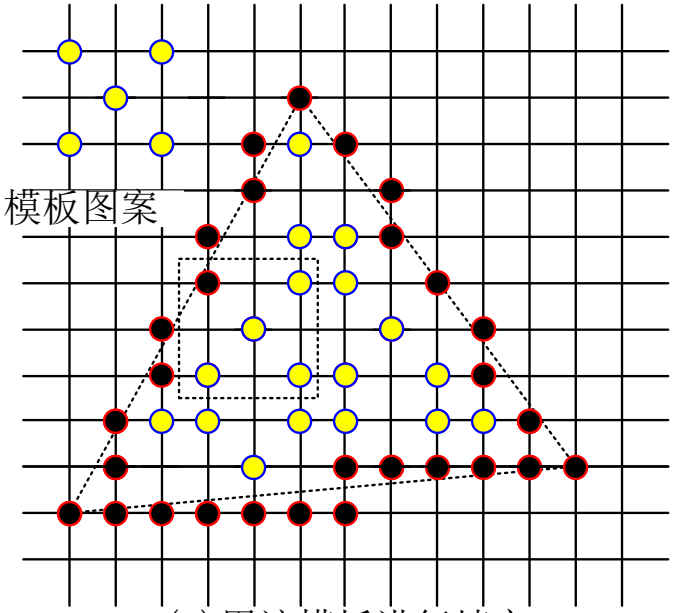
- 若是以透明方式填充图案，则当模板位图的对应位置为1时，用前景色写象素，否则，不改变该象素的值。
- 若是以不透明方式填充图案，则视模板位图对应位置为1或0来决定是用前景色还是背景色去写象素。

区域填充属性

- 确定区域与模板之间的位置关系（对齐方式）

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

(a) 图案模板位图

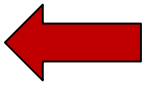


(b) 用该模板进行填充

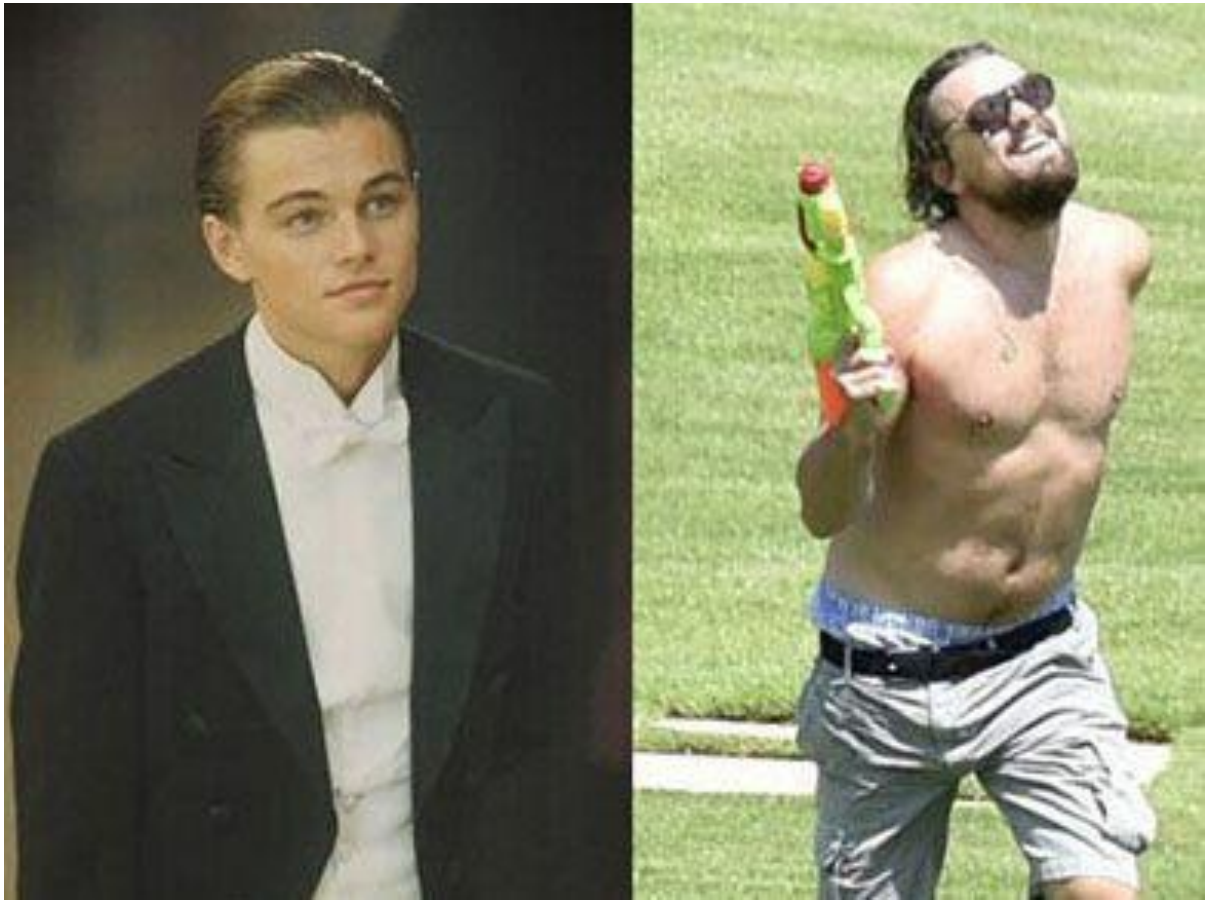
图5-45 用模板进行区域填充

反走样

1. 基本概念
2. 过取样
3. 区域取样



走样？



基本概念

- 用离散量表示连续量引起的失真，就叫做走样（Aliasing）。

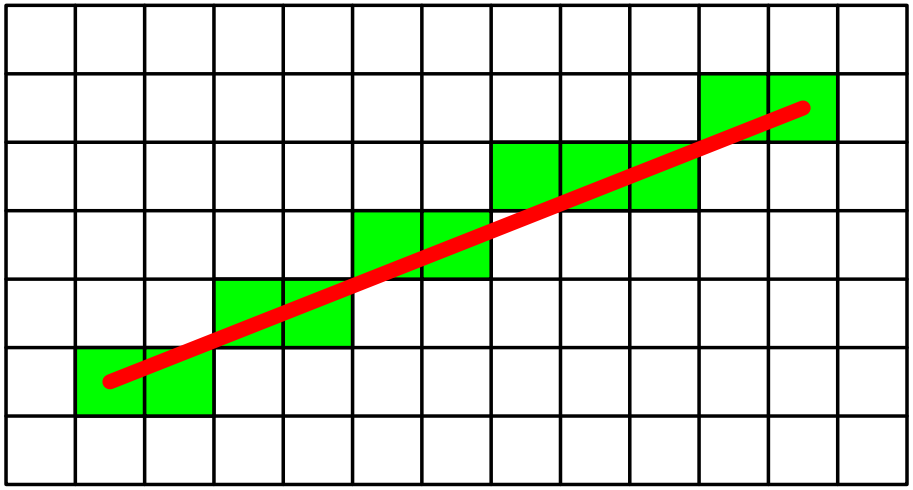


图5-46 绘制直线时的走样现象

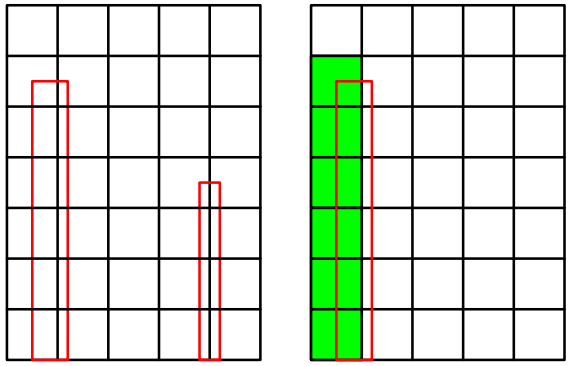
数学意义上的图形是由无线多个连续的、面积为零的点构成；但在光栅显示器上，用有限多个离散的，具有一定面积的象素来近似地表示他们。

基本概念

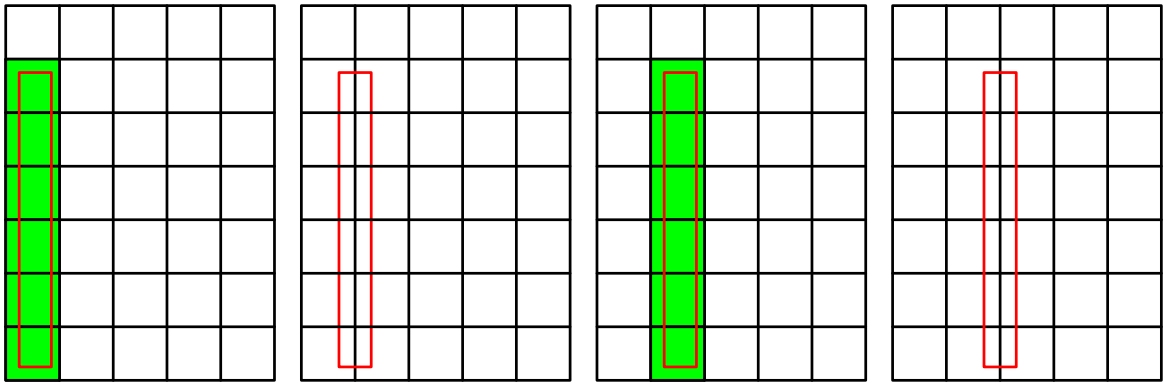
走样现象：

- 一是光栅图形产生的阶梯形。
- 一是图形中包含相对微小的物体时，这些物体在静态图形中容易被丢弃或忽略，在动画序列中时隐时现，产生闪烁。

例子



(a)需显示的矩形　(b)显示结果



(a)显示　(b)不显示　(c)显示　(d)不显示

图5-47 丢失细节与运动图形的闪烁

- 用于减少或消除这种效果的技术，称为反走样（antialiasing，图形保真）。
- 一种简单方法：

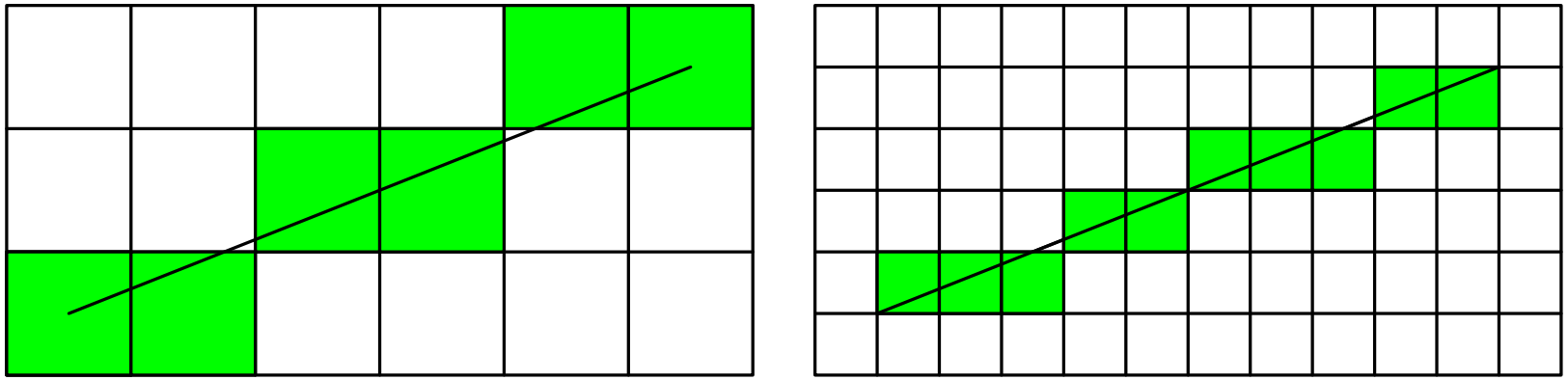


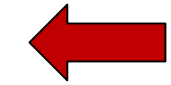
图5-48 分辨率提高一倍，阶梯程度减小一倍

过取样（supersampling），或后滤波

区域取样（area sampling），或前滤波

反走样

1. 基本概念
2. 过取样
3. 区域取样



过取样（super sampling）

- 过取样：在高于显示分辨率的较高分辨率下用点取样方法计算，然后对几个象素的属性进行平均得到较低分辨率下的象素属性。
  - 简单过取样
  - 重叠过取样
  - 基于加权模板的过取样

简单过取样

在x，y方向把分辨率都提高一倍，使每个象素对应4个子象素，然后扫描转换求得各子象素的颜色亮度，再对4个象素的颜色亮度进行平均，得到较低分辨率下的象素颜色亮度。

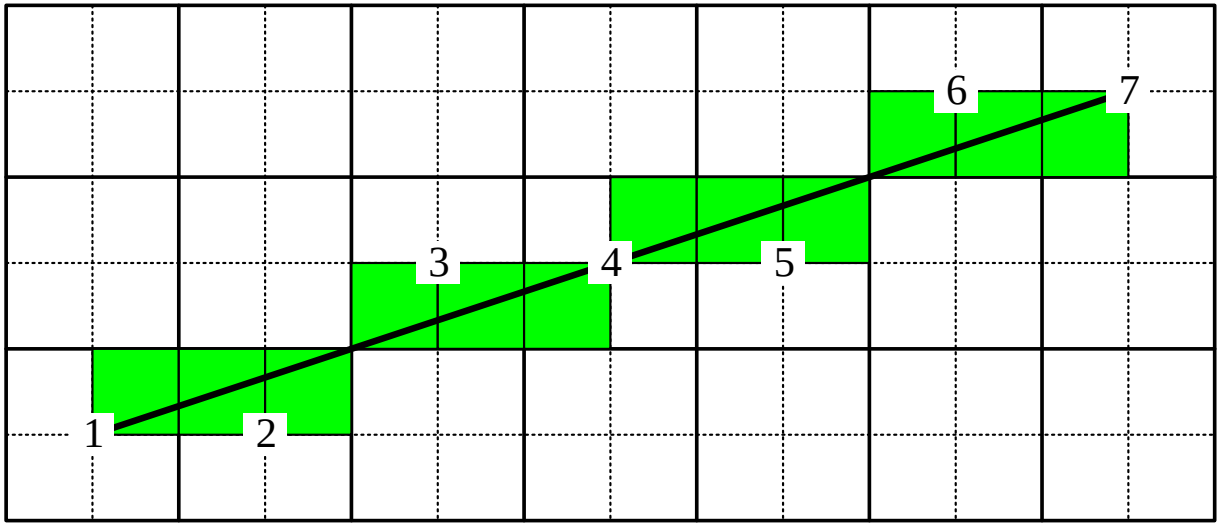


图5-49 简单的过取样方式

重叠过取样

- 为了得到更好的效果，在对一个象素点进行着色处理时，不仅仅只对其本身的子象素进行采样，同时对其周围的多个象素的子象素进行采样，来计算该点的颜色属性。

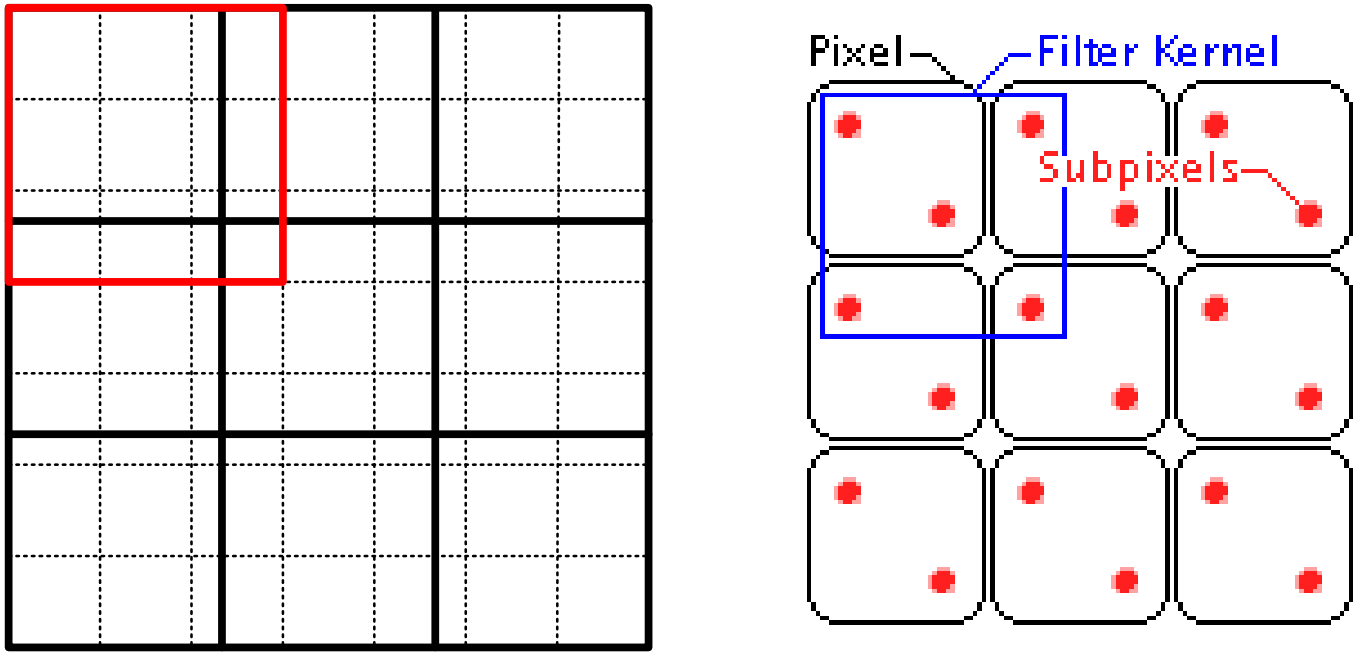


图5-50 重叠过取样

基于加权模板的过取样

- 前面在确定像素的亮度时，仅仅是对所有子像素的亮度进行简单的平均。更常见的做法是给接近像素中心的子像素赋予较大的权值，即对所有子像素的亮度进行加权平均。

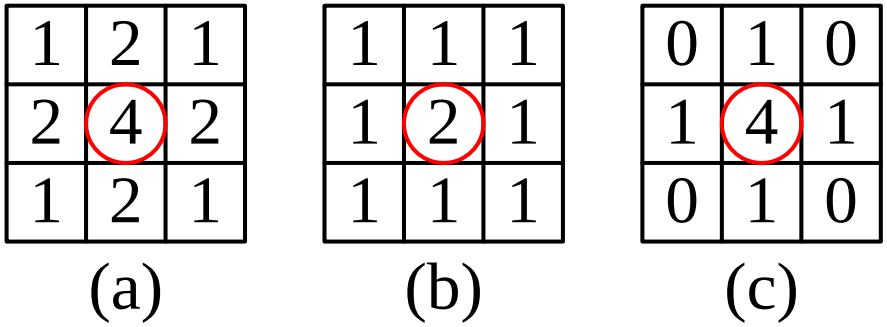
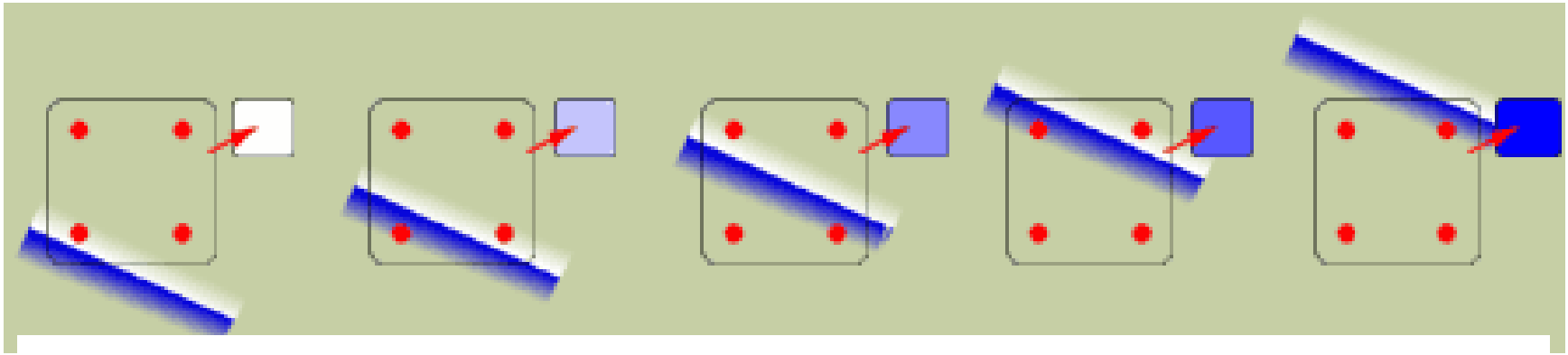


图5-51 常用的加权模板

优点：



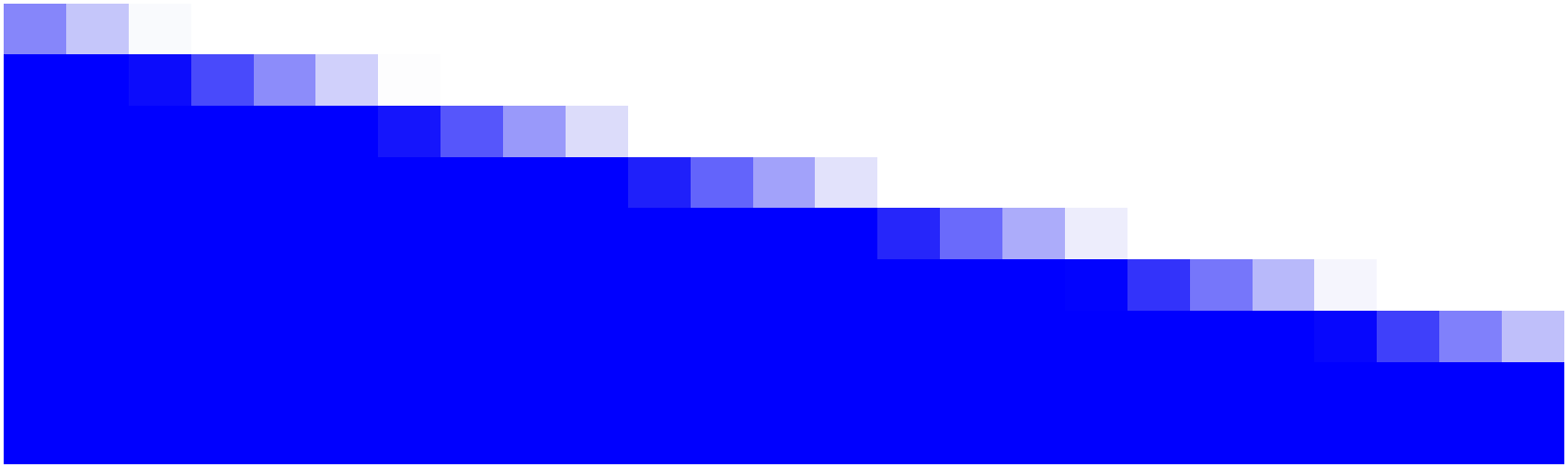


图5-52 过取样的效果

缺点：特殊角度的处理问题

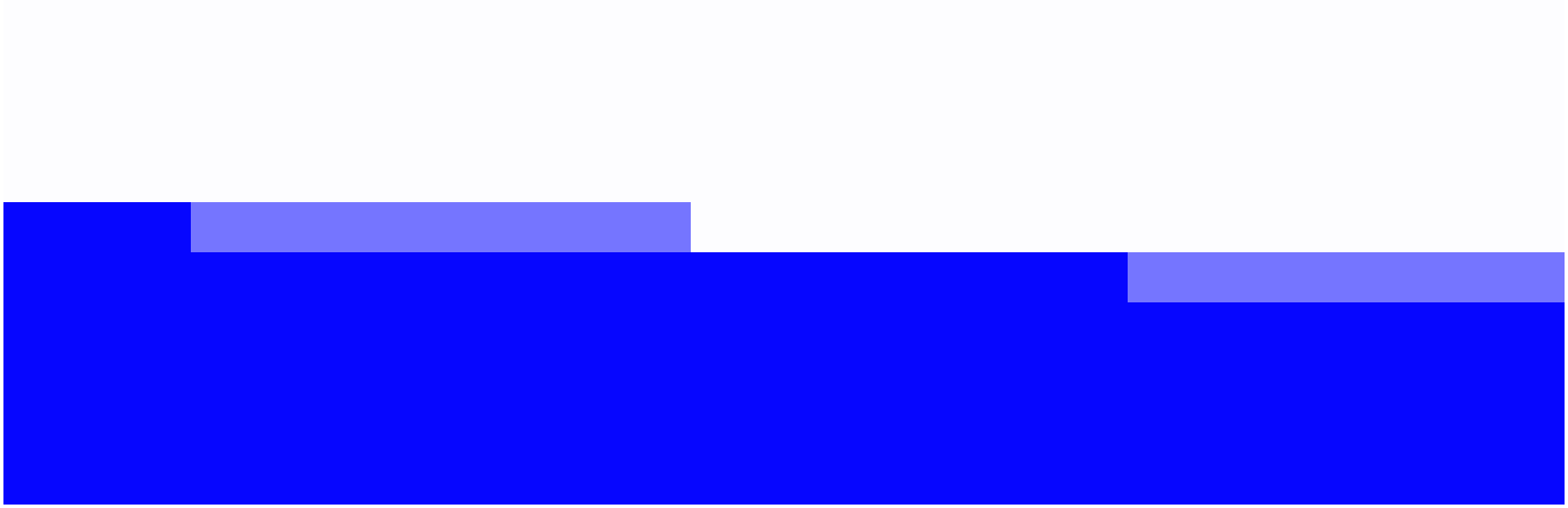
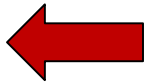


图5-53 特殊角度的问题

反走样

1. 基本概念
2. 过取样
3. 区域取样



简单的区域取样

- 在整个像素区域内进行采样，这种技术称为区域取样。又由于像素的亮度是作为一个整体被确定的，不需要划分子像素，故也被称为前置滤波。

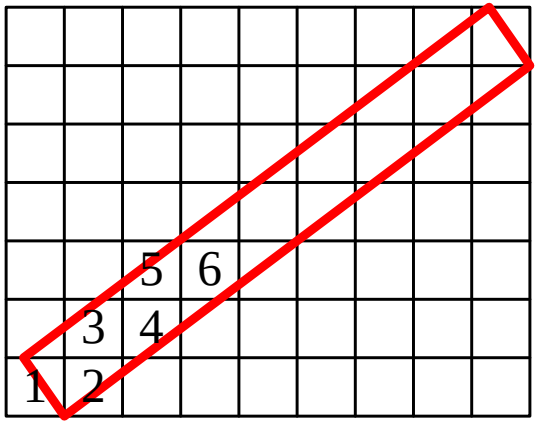


图5-54 有宽度的直线段

简单的区域取样

如何计算直线段与象素相交区域的面积？

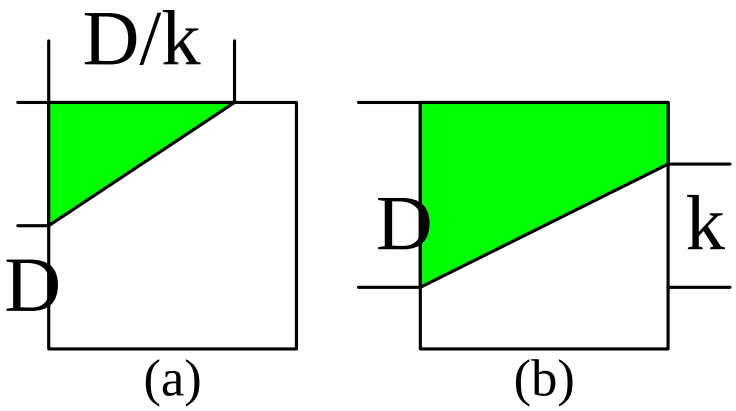


图5-55 重叠区域面积的计算

简单的区域取样

- 可以利用一种求相交区域的近似面积的离散计算方法：

  (1)将屏幕象素分割成n个更小的子象素，

  (2)计算中心落在直线段内的子象素的个数m，

  (3)m/n为线段与象素相交区域面积的近似值。

- 直线段对一个象素亮度的贡献与两者重叠区域的面积成正比。
- 相同面积的重叠区域对象素的贡献相同。

加权区域取样

- 过取样中，我们对所有子像素的亮度进行简单平均或加权平均来确定像素的亮度。
- 在区域取样中，我们使用覆盖像素的连续的加权函数（Weighting Function）或滤波函数(Filtering Function)来确定像素的亮度。

加权区域取样原理

- 加权函数W(x,y)是定义在二维显示平面上的函数。
- 对于位置为(x,y)的小区域dA来说，函数值W(x,y)（也称为在(x,y)处的高度）表示小区域dA的权值。
- 将加权函数在整个二维显示图形上积分，得到具有一定体积的**滤波器（Filter）**，该滤波器的体积为1。
- 将加权函数在显示图形上进行积分，得到滤波器的一个子体，该子体的体积介于0到1之间。用它来表示像素的亮度。

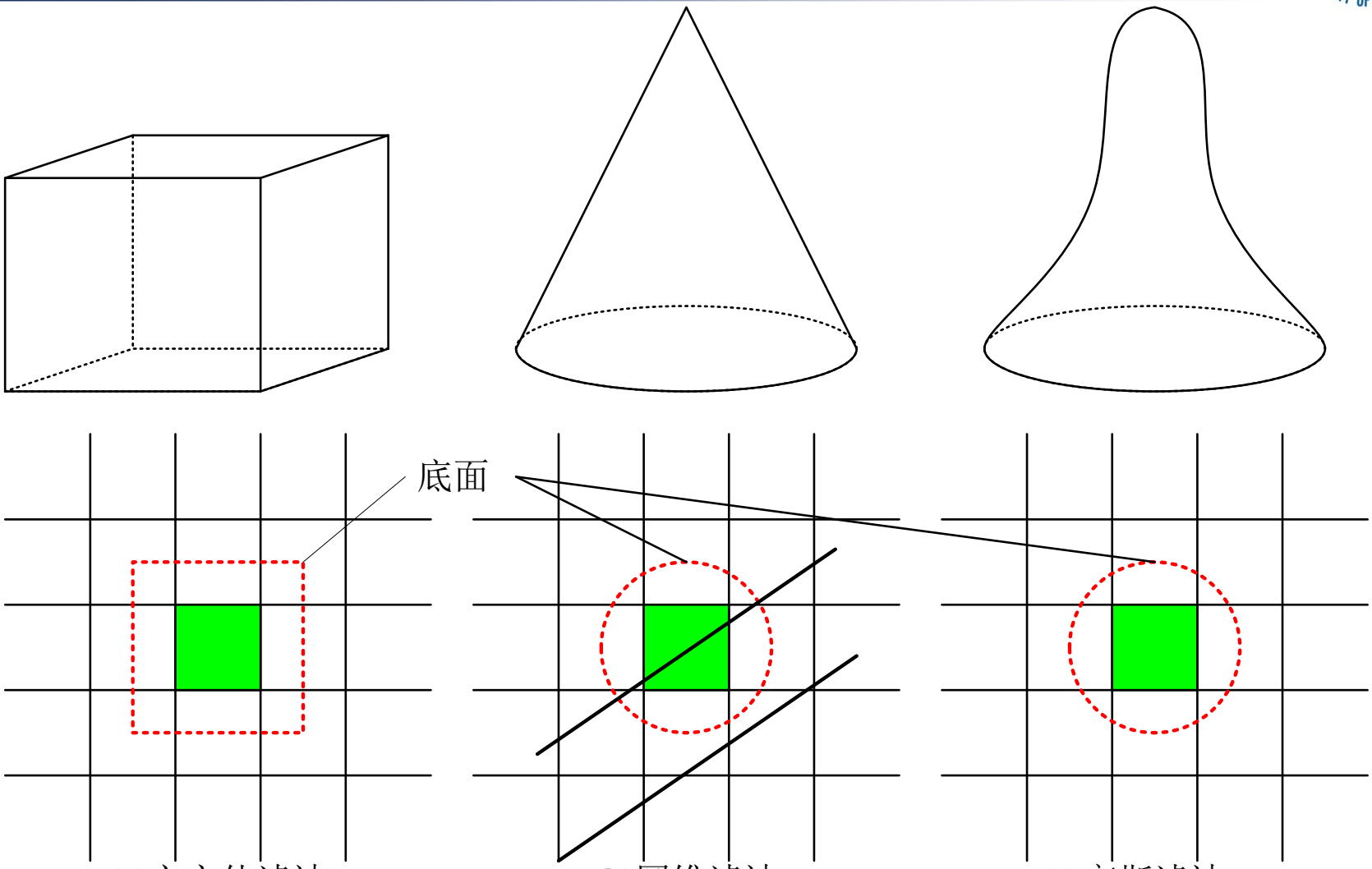


图5-56 常用的滤波函数

加权区域取样

特点：

- 接近理想直线的象素将被分配更多的灰度值；
- 相邻两个象素的滤波器相交，有利于缩小直线条上相邻象素的灰度差。

在OpenGL中绘图

1. 点和直线的绘制
2. 多边形面的绘制
3. 实体模型绘制
4. OpenGL中的字符函数

点的绘制

- 点的绘制

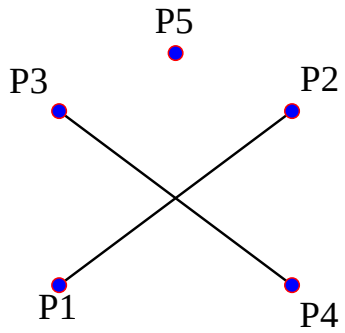
```
glBegin(GL_POINTS);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(10.0f, 0.0f, 0.0f);
glEnd();
```

- 点的属性（大小）

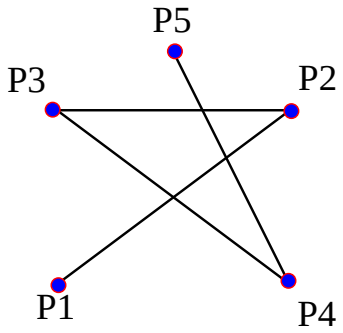
```
void glPointSize(GLfloat size);
```

直线的绘制

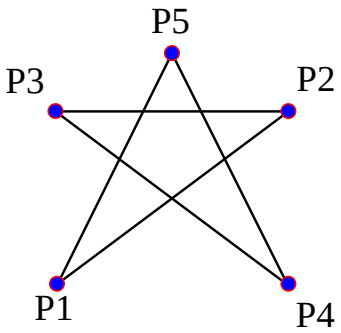
- 直线的绘制模式
  - GL_LINES
  - GL_LINE_STRIP
  - GL_LINE_LOOP



(a) GL_LINES画线模式



(b) GL_LINE_STRIP画线模式



(c) GL_LINE_LOOP画线模式

图5-57 OpenGL画线模式

直线的绘制

- 直线的属性
  - 线宽

    void glLineWidth(GLfloat width)
  - 线型

    glEnable(GL_LINE_STIPPLE)；

    glLineStipple(GLint factor，GLushort pattern)；

直线的绘制

模式：0X00FF = 255

二进制表示：0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

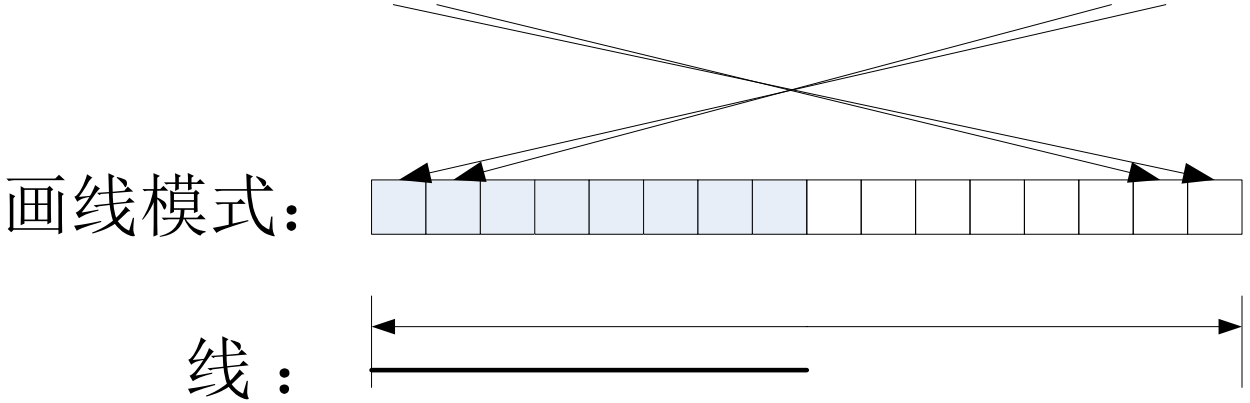
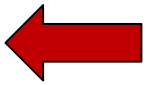


图5-58　画线模式用于构造线段

在OpenGL中绘图

1. 点和直线的绘制
2. 多边形面的绘制
3. 实体模型绘制
4. OpenGL中的字符函数



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_QUADS_STRIP
- 多边形面的绘制（GL_POLYGON）

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。
  - 多边形的边缘决不能相交，而且多边形必须是凸的。
- 解决：对于非凸多边形，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。

多边形面的绘制

- 问题：轮廓图形状态会看到组成大表面的所有小三角形。
- 处理OpenGL提供了一个特殊标记来处理这些边缘，称为边缘标记。

glEdgeFlag（True）

glEdgeFlag（False）

可以用这个函数来确定内部表面线是否可见

多边形面的属性

- 多边形面的正反属性（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。

  ```
  glFrontFace(GL_CW);
  ```

多边形面的属性

- 多边形面的颜色
  - glShadeModel(GL_FLAT) 用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。

多边形面的属性

- 多边形面的显示模式

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。
  - 参数mode用于指定新的绘图模式。

多边形面的属性

- 多边形面的填充

多边形面既可以用纯色填充，也可以用32×32的模板位图来填充。

```
void glPolygonStipple(const GLubyte *mask);

glEnable(GL_POLYGON_STIPPLE);
```

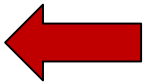
多边形面的属性

- 多边形面的法向量
  - 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);
```

在OpenGL中绘图

1. 点和直线的绘制
2. 多边形面的绘制
3. 实体模型绘制
4. OpenGL中的字符函数



OpenGL中的实体模型函数

- GLUT库中的多面体函数

表5.1 GLUT生成规则多面体的函数

函数	说明
glutSolidTetrahedron( ) glutWireTetrahedron( )	绘制中心位于世界坐标系原点的实心四面体和线框四面体，四面体的半径为$\sqrt{3}$。
glutSolidCube(size) glutWireCube(size)	绘制中心位于世界坐标系原点的实心立方体和线框立方体，立方体的半径为size，size是一个双精度浮点值。
glutSolidOctahedron ( ) glutWireOctahedron ( )	绘制中心位于世界坐标系原点的实心八面体和线框八面体，八面体的半径为1.0。
glutSolidDodecahedron( ) glutWireDodecahedron( )	绘制中心位于世界坐标系原点的实心12面体和线框12面体，12面体的半径为$\sqrt{3}$。
glutSolidIcosahedron( ) glutWireIcosahedron( )	绘制中心位于世界坐标系原点的实心20面体和线框20面体，20面体的半径为1.0。

OpenGL中的实体模型函数

- GLUT库中的二、三次曲面
  - 绘制实体或线框球面

    void glutSolidSphere/glutWireSphere (GLdouble radius, GLint slices, GLint stacks);
  - 绘制实体或线框圆锥面

    void glutSolidCone/glutWireCone (GLdouble radius, GLdouble height, GLint slices, GLint stacks);

OpenGL中的实体模型函数

- 绘制实体或线框圆环

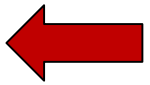
  void glutSolidTorus/ glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint slices,GLint stacks);

- 绘制实体或线框茶壶

  void glutSolidTeapot/glutWireTeapot (GLdouble size);

在OpenGL中绘图

1. 点和直线的绘制
2. 多边形面的绘制
3. 实体模型绘制
4. OpenGL中的字符函数



OpenGL中的字符函数

- GLUT位图字符

```
void glutBitmapCharacter(void *font, int character);
```

- GLUT矢量字符

```
void glutStrokeCharacter(void *font, int character);
```

谢谢！

Thank You