湖南科技大学课程教案
（章节、专题首页）

授课教师：王志喜　　　　职称：副教授　　　　单位：计算机科学与工程学院

课 程 名 称	计算机图形图像技术
章节、专题	基本图元的显示
教学目标及基本要求	掌握显示器的基本工作原理和基本图元的绘制方法。
教 学 重 点	画线算法，多边形区域的填充
教 学 难 点	画线算法，多边形区域的填充
教学内容与时间分配	（1）显示器的基本工作原理（0.3课时） （2）画线算法（0.9课时） （3）多边形区域的填充（0.3课时） 共计1.5课时。
习　　题	第2.5节（练习题）。

第2章　基本图元的显示

2.1　显示器的工作原理

显示器的工作原理通常都包含刷新、光栅扫描和彩色这三个关键词。这里以刷新式光栅扫描彩色CRT显示器为例介绍图形硬件系统的工作原理。

2.1.1 CRT显示器

1. CRT的基本工作原理

如图2-1所示。由电子枪发出的电子束，通过聚焦系统和偏转系统，射向屏幕上的指定位置。屏幕上涂覆荧光层，在电子束冲击的每个位置，荧光层发出一个小亮点。

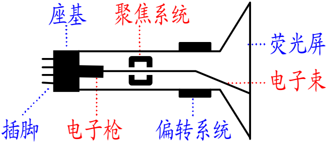


图2-1 CRT的基本工作原理

2. 电子枪的工作原理

如图2-2所示。通过给灯丝加电来加热阴极，引起受热的电子沸腾出阴极表面。带负电荷的自由电子在高正电压（由加速阳极产生）的作用下加速冲向荧光屏。

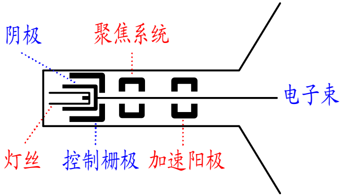


图2-2　电子枪的工作原理

3. CRT几个基本部件的作用

- 加速阳极。产生用于加速电子的高正电压。
- 控制栅极。控制电子束的强度。
- 聚焦系统。控制电子束在轰击荧光屏时汇聚成一点。
- 偏转系统。控制电子束的偏转。
- 荧光屏。当电子束的能量转移到荧光层，就在屏幕上生成亮点。

2.1.2　刷新式光栅扫描彩色显示器

1. 刷新式显示器

由于荧光层的光亮度衰减很快，必须采用某种方法保持屏幕图像。一般采用的办法是，快速控制电子束反复重画图像，这就是刷新。

图形的定义保存在称为刷新缓冲器（也可称为帧缓冲器）的存储器中。该存储器保存屏幕上所有位置的强度值。每次开始刷新时，从刷新缓冲器中取出强度值并在屏幕上画出。

2. 光栅扫描显示器

如图2-3所示。电子束横向扫描屏幕，一次一行，从顶部到底部依次进行。当电子束横向沿每一行移动时，通过电子束的强度不断变化来建立亮点的图案。每次开始刷新时，从刷新缓冲器中取出强度值并在屏幕上逐行画出。

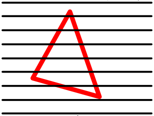


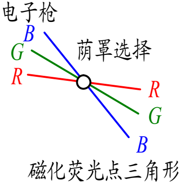
图2-3　光栅扫描

3. 彩色显示器

（1）工作原理。利用能发射不同颜色光的荧光层的组合来显示彩色图形。

（2）彩色生成技术。有穿透法和荫罩法等2种生成彩色的技术，这里只介绍目前占主导地位的荫罩法。

如图2-4所示。对每个像素位置，荫罩CRT有三个彩色荧光点，分别发射红光、绿光和蓝光。三个电子枪与每个彩色点一一对应，而荫罩栅格位于紧靠荧光层的屏幕之后。三支电子束偏转后聚焦为一组，发射到荫罩上。荫罩上有与荧光点对齐的一系列小孔。三支电子束通过荫罩上的小孔，激活一个点三角形，在屏幕上显示一个小的彩色亮点。改变三支电子束的强度等级，可改变荫罩CRT的显示颜色。



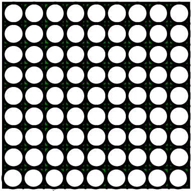


图2-4　荫罩法生成彩色

（3）亮度范围。依赖于光栅系统的能力。

- 简单黑白系统。每个像素只需使用1位来控制亮度。
- 高质量灰度系统。像素亮度分为256个等级（因为人眼无法分辨高于256个等级的亮度等级），每个像素需要使用8位来控制亮度。
- 高质量彩色系统。每个像素需要使用24位来控制颜色（红绿蓝各8位）。

2.1.3 坐标系统

1. 屏幕坐标

假定扫描线从屏幕底部从0开始顺序编号，像素列沿每条扫描线从左至右从0开始编号。

2. 底层程序

- setpixel(x, y)或draw(x, y)。使用当前属性绘制像素(x, y)。
- getpixel(x, y)或pixel(x, y)。获取像素(x, y)的属性值。

3. 显示线段

计算两指定端点之间的中间位置，输出设备直接按指令在端点间的这些位置填充。

2.2 DDA画线算法

水平的、垂直的和斜率为 ± 1 的线段很容易绘制，本章不讨论这些线段。

2.2.1 算法推导

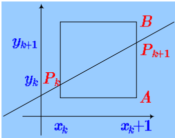
设线段的左右两个端点为 (x_L, y_L) 和 (x_R, y_R)，则斜率为 $m=(y_R-y_L)/(x_R-x_L)$。注意，这里的坐标值都是非负整数。

从左至右计算线段的中间位置 (x_k, y_k)。这里只介绍 $0<m<1$ 的情况，其他情况可以通过对称的方法做相应修改得到。

假设 (x_k, y_k) 已经确定，则下一步需要确定 (x_{k+1}, y_{k+1})。取 $x_{k+1}=x_k+1$，则由 $(y_{k+1}-y_k)/(x_{k+1}-x_k)=m$ 可得 $y_{k+1}=y_k+m$，如图2-5所示。

当 (x_k, y_k) 确定以后，就可以绘制像素 $([x_k],[y_k])$ 了。其中，

$[x]$ 表示与 x 最接近的整数。



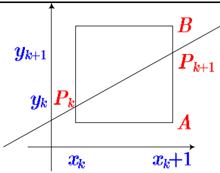


图2-5　DDA画线算法推导

2.2.2 算法描述

线段的几种情况如图2-6所示。

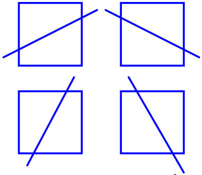


图2-6 线段的几种情况

1. 0<*m*<1

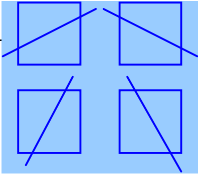
（1）输入两个端点，将左端点保存在 (x_0, y_0) 中。

（2）计算常量 m 。

（3）从 $k=0$ 开始，对每个 x_k ，绘制 $([x_k],[y_k])$ ，并作如下计算

$$x_{k+1}=x_k+1\text{，}\quad y_{k+1}=y_k+m$$

（4）重复步骤（3），直到 $x_{k+1}>x_R$ 。



2. -1<*m*<0

（1）输入两个端点，并将端点的 y 坐标反号。

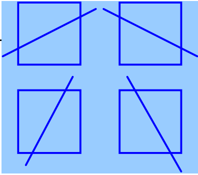
（2）将新的左端点保存在 (x_0, y_0) 中。

（3）计算常量 m 。

（4）从 $k=0$ 开始，对每个 x_k ，绘制 $([x_k],[-y_k])$，并作如下计算

$$x_{k+1}=x_k+1，\quad y_{k+1}=y_k+m 。$$

（5）重复步骤（4），直到 $x_{k+1}>x_R$。



3. $m>1$

（1）输入两个端点，并交换端点的 x 坐标和 y 坐标。

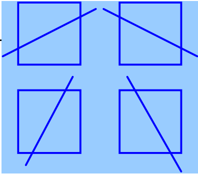
（2）将新的左端点保存在 (x_0, y_0) 中。

（3）计算常量 m 。

（4）从 $k=0$ 开始，对每个 x_k ，绘制 $([y_k],[x_k])$，并作如下计算

$$x_{k+1}=x_k+1, \quad y_{k+1}=y_k+m$$

（5）重复步骤（4），直到 $x_{k+1}>x_R$。



4. m<-1

（1）输入两个端点以后，将端点的 y 坐标反号，并交换新端点的 x 坐标和 y 坐标。

（2）将新的左端点保存在 (x_0, y_0) 中。

（3）计算常量 m。

（4）从 $k=0$ 开始，对每个 x_k，绘制 $([y_k],[-x_k])$，并作如下计算

$$x_{k+1}=x_k+1, \quad y_{k+1}=y_k+m$$

（5）重复步骤（4），直到 $x_{k+1}>x_R$。

2.2.3 举例

【问题】使用DDA算法绘制端点为 (20,10) 和 (28,16) 的线段，绘制结果如图2-7所示。

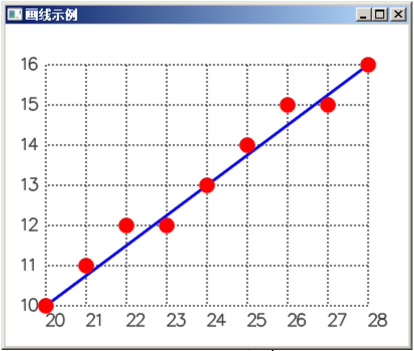


图2-7　DDA画线举例

【解答】

$$\Delta x = 8, \Delta y = 6, m = 0.75$$

$$x_0 = 20,\quad y_0 = 10,\qquad \text{draw}(20,10)$$

$$x_1 = 21,\quad y_1 = y_0 + m = 10.75,\quad \text{draw}(21,11)$$

$$x_2 = 22,\quad y_2 = y_1 + m = 11.5,\quad \text{draw}(22,12)$$

$$x_3 = 23,\quad y_3 = y_2 + m = 12.25,\quad \text{draw}(23,12)$$

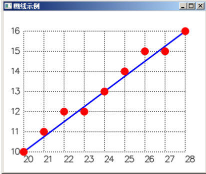
$$x_4 = 24,\quad y_4 = y_3 + m = 13,\quad \text{draw}(24,13)$$

$$x_5 = 25,\quad y_5 = y_4 + m = 13.75,\quad \text{draw}(25,14)$$

$$x_6 = 26,\quad y_6 = y_5 + m = 14.5,\quad \text{draw}(26,15)$$

$$x_7 = 27,\quad y_7 = y_6 + m = 15.25,\quad \text{draw}(27,15)$$

$$x_8 = 28,\quad y_8 = y_7 + m = 16,\quad \text{draw}(28,16)$$



2.2.4 优缺点

1. 优点

只有加法，没有乘法。

2. 缺点

- 耗时。需要取整和浮点数运算。
- 误差积累。使用了浮点数增量的累加。

2.3 中点画线算法

2.3.1 算法推导

这里只介绍 $0<m<1$ 的情况，其他情况可以通过对称的方法做相应修改得到。

如图2-8所示，假设已经确定 (x_k, y_k)，则下一点 $P_{k+1}(x_{k+1}, y_{k+1})$ 只能是 $A(x_k+1, y_k)$ 或 $B(x_k+1, y_k+1)$。

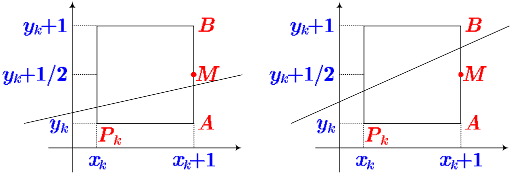


图2-8　中点画线算法推导

设 M 为 AB 的中点，若 M 在直线上方，则 A 离直线较近，故下一点为 A；否则，下一点为 B 。

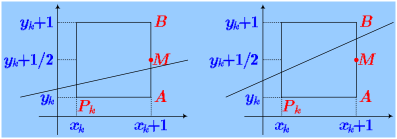
设线段的左右两个端点为 (x_L, y_L) 和 (x_R, y_R)，由

$$\frac{y-y_L}{x-x_L}=\frac{y_R-y_L}{x_R-x_L}$$

可得直线的隐式方程为（必须保证 y 系数是正数）

$$f(x,y)=-(y_R-y_L)x+\underline{\underline{(x_R-x_L)}}y+(y_Rx_L-x_Ry_L)=0$$

为了方便，记 $a=-(y_R-y_L)$， $b=(x_R-x_L)$， $c=(y_Rx_L-x_Ry_L)$。显然， a,b,c 都是整数。



平面上的点 (x, y) 与直线的相对位置可用 $f(x, y)$ 的符号检测。

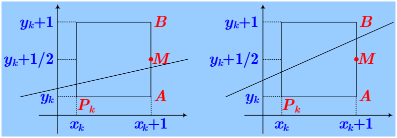
$$f(x,y)\begin{cases} >0, & (x,y)\ \text{Upside Line} \\ =0, & (x,y)\ \text{On Line} \\ <0, & (x,y)\ \text{Downside Line} \end{cases}$$

取

$$p_k = 2f(x_k+1, y_k+1/2) = 2a(x_k+1) + 2b(y_k+1/2) + 2c$$

即 AB 中点 M 的函数值的2倍。

- 若 $p_k > 0$，M 在直线上方，下一点为 $A(x_k+1, y_k)$。
- 若 $p_k \le 0$，M 在直线下方，下一点为 $B(x_k+1, y_k+1)$。



寻找 p_{k+1} 与 p_k 的关系

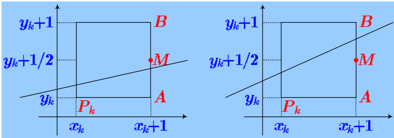
$$\begin{aligned} p_k &= 2a(x_k+1)+2b(y_k+1/2)+2c \\ p_{k+1} &= 2a(x_{k+1}+1)+2b(y_{k+1}+1/2)+2c \\ p_{k+1}-p_k &= 2a+2b(y_{k+1}-y_k) \end{aligned}$$

由此，可得如下递推关系

- 若 $p_k > 0$，则 $p_{k+1}-p_k = 2a$，即 $p_{k+1} = p_k + 2a$。
- 若 $p_k \le 0$，则 $p_{k+1}-p_k = 2a+2b$，即 $p_{k+1} = p_k + (2a+2b)$。

计算 p_0

$$\begin{aligned} p_0 &= 2f(x_0+1, y_0+1/2) \\ &= 2a(x_0+1)+2b(y_0+1/2)+2c \\ &= 2(ax_0+by_0+c)+2a+b \\ &= 2f(x_0,y_0)+2a+b \\ &= 2a+b \end{aligned}$$



2.3.2 算法描述与算法优点

1. 0<*m*<1

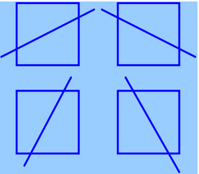
（1）输入两个端点，将左端点保存在 (x_0, y_0) 中。

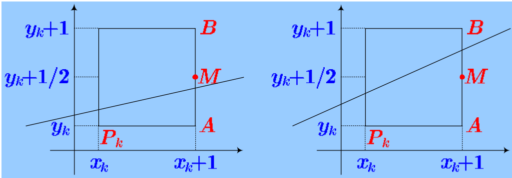
（2）计算 $a=-(y_R-y_L)$，$b=x_R-x_L$，$2a$，$2a+2b$，$p_0=2a+b$。

（3）从 $k=0$ 开始，对每个 x_k，绘制 (x_k, y_k)，并进行下列检测

- 若 $p_k>0$，则下一点为 (x_k+1, y_k)，且 $p_{k+1}=p_k+2a$。
- 若 $p_k\le 0$，则下一点为 (x_k+1, y_k+1)，且 $p_{k+1}=p_k+(2a+2b)$。

（4）重复步骤（3），直到 $x_{k+1}>x_R$。





2. -1<*m*<0

（1）输入两个端点，并将端点的 y 坐标反号。

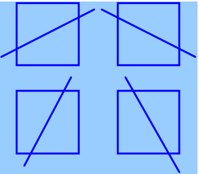
（2）将新的左端点保存在 (x_0, y_0) 中。

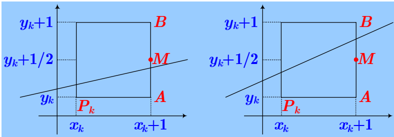
（3）计算 $a=-(y_R-y_L)$，$b=x_R-x_L$，$2a$，$2a+2b$，$p_0=2a+b$。

（4）从 $k=0$ 开始，对每个 x_k，绘制 $(x_k, -y_k)$，并进行下列检测

- 若 $p_k>0$，则下一点为 (x_k+1, y_k)，且 $p_{k+1}=p_k+2a$。
- 若 $p_k\le 0$，则下一点为 (x_k+1, y_k+1)，且 $p_{k+1}=p_k+(2a+2b)$。

（5）重复步骤（4），直到 $x_{k+1}>x_R$。





3. $m>1$

（1）输入两个端点，并交换端点的 x 坐标和 y 坐标。

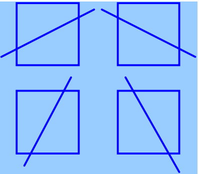
（2）将新的左端点保存在 (x_0, y_0) 中。

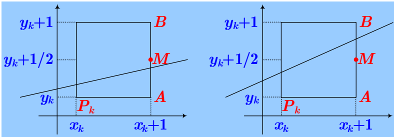
（3）计算常量 $a=-(y_R-y_L)$，$b=x_R-x_L$，$2a$，$2a+2b$，$p_0=2a+b$。

（4）从 $k=0$ 开始，对每个 x_k，绘制 (y_k, x_k)，并进行下列检测

- 若 $p_k>0$，则下一点为 (x_k+1, y_k)，且 $p_{k+1}=p_k+2a$。
- 若 $p_k\le 0$，则下一点为 (x_k+1, y_k+1)，且 $p_{k+1}=p_k+(2a+2b)$。

（5）重复步骤（4），直到 $x_{k+1}>x_R$。





4. $m<-1$

（1）输入两个端点以后，将端点的 y 坐标反号，并交换新端点的 x 坐标和 y 坐标。

（2）将新的左端点保存在 (x_0, y_0) 中。

（3）计算常量 $a=-(y_R-y_L)$， $b=x_R-x_L$， $2a$， $2a+2b$， $p_0=2a+b$ 。

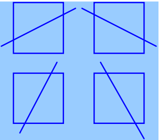
（4）从 $k=0$ 开始，对每个 x_k ， 绘制 $(y_k, -x_k)$，并进行下列检测

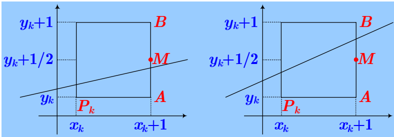
- 若 $p_k>0$，则下一点为 (x_k+1, y_k)，且 $p_{k+1}=p_k+2a$
- 若 $p_k\le 0$，则下一点为 (x_k+1, y_k+1)，且 $p_{k+1}=p_k+(2a+2b)$

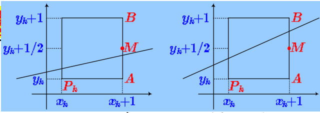
（5）重复步骤（4），直到 $x_{k+1}>x_R$ 。

5. 算法优点

消除了乘法和取整运算。







2.3.3 举例

【问题】使用中点画线算法绘制端点为(20, 10)和(28, 16)的线段，绘制结果如图2-7所示。

【解答】

$$a=-6, b=8, 2a=-12, (2a+2b)=4$$

$$(x_0, y_0)=(20,10),\ \ p_0=2a+b=-4$$

$$(x_1, y_1)=(21,11),\ \ p_1=\ p_0+(2a+2b)=0$$

$$(x_2, y_2)=(22,12),\ \ p_2=\ p_1+(2a+2b)=4$$

$$(x_3, y_3)=(23,12),\ \ p_3=\ p_2+2a=-8$$

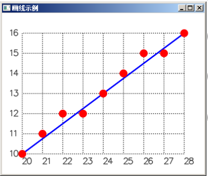
$$(x_4, y_4)=(24,13),\ \ p_4=\ p_3+(2a+2b)=-4$$

$$(x_5, y_5)=(25,14),\ \ p_5=\ p_4+(2a+2b)=0$$

$$(x_6, y_6)=(26,15),\ \ p_6=\ p_5+(2a+2b)=4$$

$$(x_7, y_7)=(27,15),\ \ p_7=\ p_6+2a=-8$$

$$(x_8, y_8)=(28,16)$$



$$p_{k+1}=\begin{cases} p_k+2a, & p_k>0 \\ p_k+(2a+2b), & p_k\le 0 \end{cases}$$

2.4 多边形区域的填充

2.4.1 扫描线算法的一般步骤

扫描线多边形填充算法是最常用的多边形区域填充算法。

如图2-9所示，扫描线多边形填充算法的一般步骤如下。

（1）对每条穿过多边形的扫描线，确定扫描线与多边形的交点（不考虑水平边）。

（2）在交点表中将交点从左至右存储（如图2-9右侧所示，只需要存储交点的 x 坐标）。

（3）将每对交点之间的点（不含交点）设置为指定颜色。

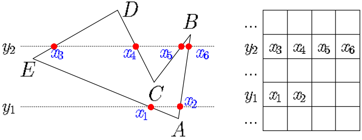


图2-9 扫描线算法

2.4.2 顶点处扫描线交点的处理方法

- 如果两相交边都位于扫描线的下侧（如图2-10中的顶点 B ），则不将该交点存入交点表。
- 如果两相交边都位于扫描线的上侧（如图2-10中的顶点 C ），则将该交点存入交点表2次。
- 如果两相交边位于扫描线的两侧（如图2-10中的顶点 E ），则将该交点存入交点表1次。
- 可以通过在交点表中不存储每条边的高端点来实现。

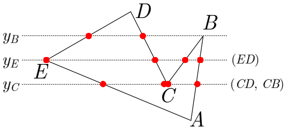


图2-10 顶点处扫描线交点的处理方法

2.4.3 计算交点的x坐标

对于多边形的每一条边，使用整数增量的方法从下往上计算出每个交点的整数坐标。

1. 方法

设 $q=|\Delta x|\,\mathrm{idiv}\,\Delta y$， $r=|\Delta x|-q\Delta y$， $\varepsilon=\mathrm{sgn}(\Delta x)$ ①。

（1）计数器 $k=0$ ②；在交点表中存储低端点。

（2）每当移向一条新的扫描线时， $k=k+2r$。

（3）若 $k<\Delta y$，则 $x=x+\varepsilon q$；否则， $x=x+\varepsilon(q+1)$， $k=k-2\Delta y$；在交点表中存储获得的交点。

（4）重复（2）、（3），直到边界的高端点（不存储每条边的高端点）。

---

① idiv 是整数除法运算，结果为整数。sgn()是符号函数，结果只能是-1、0 或 1。

② k 用于累加 $|\Delta x|/\Delta y$ 的小数部分，为了避免小数计算，使用 $k=k+2r$ 累加，$k<\Delta y$ 表示和小于 0.5， $k=k-2\Delta y$ 表示和减少 1。

2. 举例说明

边 (7,2) ～ (1,7) 的计算过程如下。

$$|\Delta x| = 6, \quad \Delta y = 5, \quad 2\Delta y = 10$$

$$q = 1, \quad r = 1, \quad 2r = 2$$

$$\varepsilon q = -1, \quad \varepsilon(q+1) = -2$$

$$y = 2, \quad k = 0, \quad x = 7$$

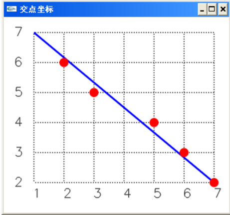
$$y = 3, \quad k = 2, \quad x = x - 1 = 6$$

$$y = 4, \quad k = 4, \quad x = x - 1 = 5$$

$$y = 5, \quad k = 6, \quad x = x - 2 = 3, \quad k = 6 - 10 = -4$$

$$y = 6, \quad k = -2, \quad x = x - 1 = 2$$

计算结果如图2-11所示。



（2）每当移向一条新的扫描线时，$k=k+2r$。
（3）若$k<\Delta y$，则$x=x+\varepsilon q$；否则，$x=x+\varepsilon(q+1)$，$k=k-2\Delta y$；在交点表中存储获得的交点。

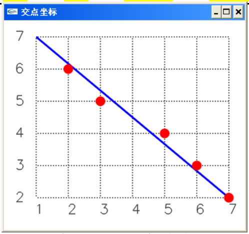


图2-11　交点坐标

2.5 练习题

1. 假设RGB光栅系统的设计采用8×10英寸的屏幕，每个方向的分辨率为每英寸100个像素。如果每个像素使用8位，存放在帧缓冲器中，则帧缓冲器至少需要多大存储容量（字节数）？

2. 假设计算机字长为32位，传输速率为1 MIP（每秒百万条指令）。300 DPI（每英寸点数）的激光打印机，页面大小为8.5×11英寸，要填满帧缓冲器至少需要多长时间。

3. 考虑分辨率为1024×768的光栅系统。若刷新速率为每秒60帧，则每秒钟应访问多少像素，每个像素的访问时间至少是多少？

4. 假设某真彩色（24位）RGB光栅系统有1024×768像素的帧缓冲器，则该系统可以有多少种不同的彩色选择（强度级），在任一时刻至多可以显示多少种不同的颜色?

5. 分辨率为1024×768的高质量彩色系统（32位）至少需要多少*MB*帧缓冲器？

6. 使用DDA算法绘制端点为(5, 6)和(13, 12)的线段。

7. 使用中点画线算法绘制端点为(5, 6)和(13, 12)的线段。

8. 使用整数增量的方法计算边(0, 1)~(11, 6)与各扫描线交点的x坐标。