全部隐藏

2.1.2 生成直线的Bresenham算法

从上面介绍的DDA算法可以看到，由于在循环中涉及实型数据的加减运算，因此直线的生成速度较慢。

在生成直线的算法中，Bresenham算法是最有效的算法之一。Bresenham算法是一种基于误差判别式来生成直线的方法。

一、直线Bresenham算法描述：

它也是采用递推步进的办法，令每次最大变化方向的坐标步进一个象素，同时另一个方向的坐标依据误差判别式的符号来决定是否也要步进一个象素。

我们首先讨论m=△y/△x，当0≤m≤1且$x_1<x_2$时的Bresenham算法。从DDA直线算法可知这些条件成立时，公式(2-2)、(2-3)可写成：

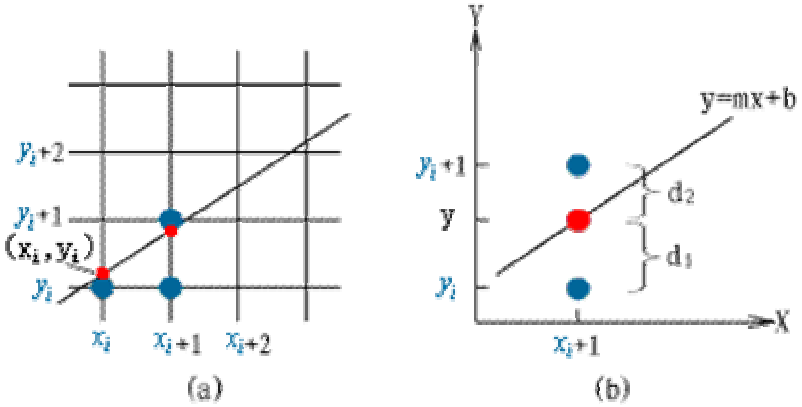
$$x_{i+1}=x_i+1 \qquad (2-6)$$

$$y_{i+1}=y_i+m \qquad (2-7)$$

有两种Bresenham算法思想，它们各自从不同角度介绍了Bresenham算法思想，得出的误差判别式都是一样的。

二、直线Bresenham算法思想之一：

由于显示直线的象素点只能取整数值坐标，可以假设直线上第i个象素点坐标为(x_i, y_i)，它是直线上点(x_i, y_i)的最佳近似，并且$x_i=x_i$(假设m<1），如下图所示。那么，直线上下一个象素点的可能位置是(x_i+1, y_i)或(x_i+1, y_i+1)。



由图中可以知道，在$x=x_i+1$处，直线上点的y值是$y=m(x_i+1)+b$，该点离象素点(x_i+1, y_i)和象素点(x_i+1, y_i+1)的距离分别是d_1和d_2：

$$d_1=y-y_i=m(x_i+1)+b-y_i \qquad (2-8)$$

$$d_2=(y_i+1)-y=(y_i+1)-m(x_i+1)-b \qquad (2-9)$$

这两个距离差是

$$d_1-d_2=2m(x_i+1)-2y_i+2b-1 \qquad (2-10)$$

我们来分析公式(2-10)：

(1)当此值为正时，$d_1>d_2$，说明直线上理论点离(x_i+1, y_i+1)象素较近，下一个象素点应取(x_i+1, y_i+1)。

(2)当此值为负时，$d_1<d_2$，说明直线上理论点离(x_i+1, y_i)象素较近，则下一个象素点应取(x_i+1, y_i)。

(3)当此值为零时，说明直线上理论点离上、下两个象素点的距离相等，取哪个点都行，假设算法规定这种情况下取(x_i+1, y_i+1)作为下一个象素点。

因此只要利用(d_1-d_2)的符号就可以决定下一个象素点的选择。为此，我们进一步定义一个新的判别式：

$$p_i=\triangle x\times(d_1-d_2)=2\triangle y\cdot x_i-2\triangle x\cdot y_i+c \qquad (2-11)$$

式(2-11)中的$\triangle x=(x_2-x_1)>0$，因此p_i与(d_1-d_2)有相同的符号；

这里$\triangle y=y_2-y_1$，$m=\triangle y/\triangle x$；$c=2\triangle y+\triangle x(2b-1)$。

下面对式(2-11)作进一步处理，以便得出误差判别递推公式并消除常数c。

将式(2-11)中的下标i改写成i+1，得到：

$$p_{i+1}=2\triangle y\cdot x_{i+1}-2\triangle x\cdot y_{i+1}+c \qquad (2-12)$$

将式(2-12)减去(2-11)，并利用$x_{i+1}=x_i+1$，可得：

$$p_{i+1}=p_i+2\triangle y-2\triangle x\cdot(y_{i+1}-y_i) \qquad (2-13)$$

再假设直线的初始端点恰好是其象素点的坐标，即满足：

$$y_1=mx_1+b \qquad (2-14)$$

由式(2-11)和式(2-14)得到p_1的初始值：

$$p_1=2\triangle y-\triangle x \qquad (2-15)$$

这样，我们可利用误差判别变量，得到如下算法表示：

$$\begin{aligned}&\text{初始 } p_1=2\triangle y-\triangle x\\&\begin{cases}\text{当}p_i\geqslant 0\text{时：}\ y_{i+1}=y_i+1,\\ x_{i+1}=x_i+1,\\ p_{i+1}=p_i+2(\triangle y-\triangle x)\end{cases}\\&\begin{cases}\text{否则：}\ y_{i+1}=y_i,\\ x_{i+1}=x_i+1,\\ p_{i+1}=p_i+2\triangle y\end{cases}\end{aligned} \qquad (2-16)$$

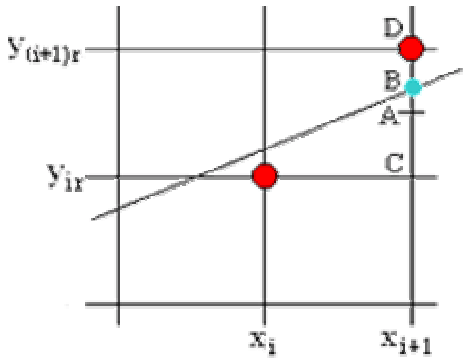
从式(2-16)可以看出，第i+1步的判别变量p_{i+1}仅与第i步的判别变量p_i、直线的两个端点坐标分量差$\triangle x$和$\triangle y$有关，运算中只含有整数相加和乘2运算，而乘2可利用算术左移一位来完成，因此这个算法速度快并易于硬件实现。

三、直线Bresenham算法思想之二：

由于象素坐标的整数性，数学点(x_i, y_i)与所取象素点(x_i, y_{ir})间会引起误差(ε_i)，当x_i列上已用象素坐标(x_i, y_{ir})表示直线上的点(x_i, y_i)，下一直线点$B(x_{i+1}, y_{i+1})$，是取象素点$C(x_{i+1}, y_{ir})$，还是$D(x_{i+1}, y_{(i+1)r})$呢？

设A为CD边的中点，正确的选择：

若B点在A点上方，选择D点； 否则，选C点。



用误差式描述为：

$$\varepsilon(x_{i+1})=BC-AC=(y_{i+1}-y_{ir})-0.5 \quad (2-8')$$

求递推公式：

$$\varepsilon(x_{i+2})=(y_{i+2}-y_{(i+1)r})-0.5 = y_{i+1}+m-y_{(i+1)r}-0.5 \quad (2-9')$$

当$\varepsilon(x_{i+1})\geqslant 0$时，选D点，$y_{(i+1)r} = y_{ir}+1$

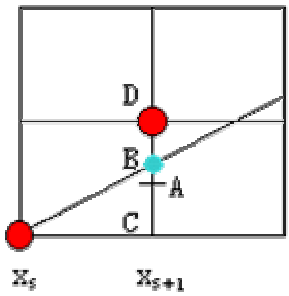
$$\varepsilon(x_{i+2})= y_{i+1}+m-y_{ir}-1-0.5=\varepsilon(x_{i+1})+m-1 \quad (2-10')$$

当$\varepsilon(x_{i+1}) < 0$时，选C点，$y_{(i+1)r} = y_{ir}$

$$\varepsilon(x_{i+2})= y_{i+1}+m-y_{ir}-0.5=\varepsilon(x_{i+1})+m \quad (2-11')$$

初始时：

$$\varepsilon(x_{s+1})=BC-AC=m-0.5 \quad (2-12')$$



为了运算中不含实型数，同时不影响不等式的判断，将方程两边同乘一正整数。

令方程两边同乘$2 \cdot \Delta x$，即$d=2 \cdot \Delta x \cdot \varepsilon$，则：

初始时：

$$d = 2 \cdot \Delta y-\Delta x \quad (2-13')$$

递推式：

```
当d≥0时：{ d=d+2 • (Δy－Δx)；
y++；                                   (2－14')
x++；
}
```

```
否则： { d=d+2 • Δy；
x++；
}
```

四、直线Bresenham算法实现：

条件：0≤m≤1且$x_1<x_2$

1、输入线段的两个端点坐标和画线颜色：x1，y1，x2，y2，color；

2、设置象素坐标初值：x=x1，y=y1；

3、设置初始误差判别值：p=2 • Δy-Δx；

4、分别计算：Δx=x2-x1、Δy=y2-y1；

5、循环实现直线的生成：

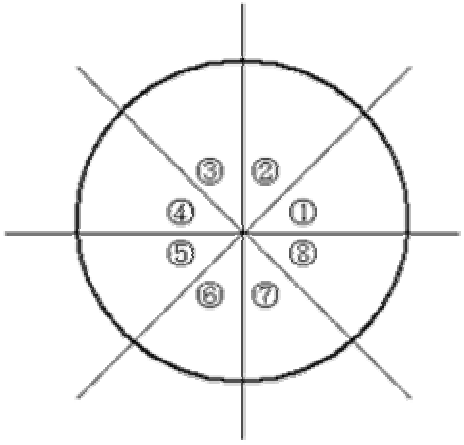
```
for(x=x1；x<=x2；x++)
{ putpixel(x,y,color) ；
if(p>=0)
{ y=y+1；
p=p+2 • (Δy-Δx)；
}
else
{ p=p+2 • Δy；
}

}
```

五、直线Bresenham算法完善：

现在我们修正(2-16)公式，以适应对任何方向及任何斜率线段的绘制。如下图所示，线段的方向可分为八种，从原点出发射向八个区。由线段按图中所示的区域位置可决定x_{i+1}和y_{i+1}的变换规律。

容易证明：当线段处于①、④、⑧、⑤区时，以|△x|和|△y|代替前面公式中的△x和△y，当线段处于②、③、⑥、⑦区时，将公式中的|△x|和|△y|对换，则上述两公式仍有效。



①④：$x_{i+1}=x_i+1$、x_i-1
$y_{i+1}=y_i$ 或 y_i+1

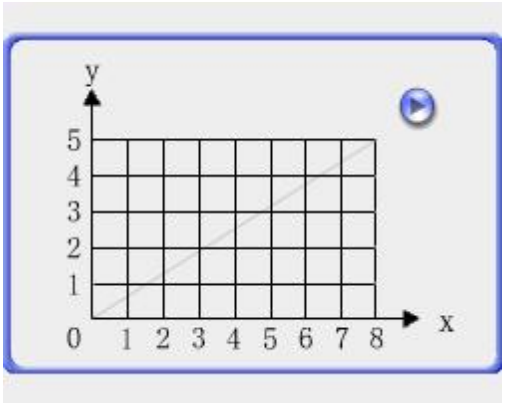
⑧⑤：$x_{i+1}=x_i+1$、x_i-1
$y_{i+1}=y_i$ 或 y_i-1

②⑦：$y_{i+1}=y_i+1$、y_i-1
$x_{i+1}=x_i$ 或 x_i+1

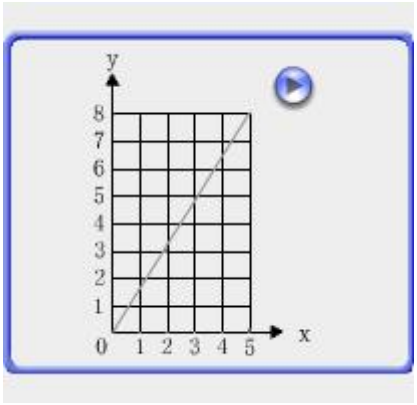
③⑥：$y_{i+1}=y_i+1$、y_i-1
$x_{i+1}=x_i$ 或 x_i-1

在线段起点区分线段方向

六、直线Bresenham算法演示：



斜率小于1



斜率大于1

七、直线Bresenham算法特点：

由于程序中不含实型数运算，因此速度快、效率高，是一种有效的画线算法。

八、直线Bresenham算法程序：

```
void Bresenhamline (int x1,int y1,int x2,int y2,int color)
{
  int x, y, dx, dy, s1, s2, p, temp, interchange, i;
  x=x1;
  y=y1;
  dx=abs(x2-x1);
  dy=abs(y2-y1);

  if(x2>x1)
    s1=1;
  else
    s1=-1;

  if(y2>y1)
    s2=1;
  else
    s2=-1;

  if(dy>dx)
  {
    temp=dx;
    dx=dy;
    dy=temp;
    interchange=1;
  }
  else
    interchange=0;

  p=2*dy-dx;
  for(i=1;i<=dx;i++)
```

```
  {
    putpixel(x,y,color);
    if(p>=0)
    {
      if(interchange= =0)
        y=y+s2;
      else
        x=x+s1;
      p=p-2*dx;
    }
    if(interchange= =0)
      x=x+s1;
    else
      y=y+s2;
    p=p+2*dy;
  }
}
```

联系我们 | 制作小组 | 友情连接 © 2009 洛阳理工学院. All rights reserved