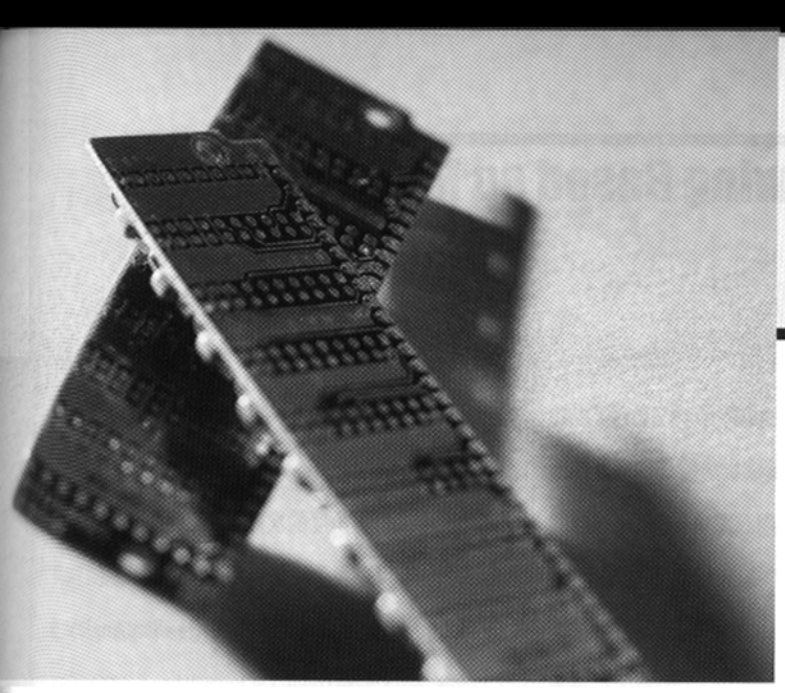


基于内存设备环境的图形快速显示技术

范宝德　刘惊雷　王培进

(山东省烟台市莱山区烟台大学计算机系　264635)

摘要：本文描述了一般图形的显示原理，并指出传统的动画显示方式的缺点——速度慢和屏幕闪烁。提出了采用GDI位图与内存设备环境来显示图形的方法。该方法大大减少了I/O的次数，缩短了显示图形的时间，并且不必擦除图形原先的背景，实现了图形的快速显示，达到了真正意义上的图形无闪烁的连续显示过程。

关键词：GDI位图　内存设备环境　无闪烁

1　引言

Visual C++在处理图形方面提供了强大的支持，在MFC类库中有一个很重要的类CDC类，在CDC类中定义了设备描述表对象，并且提供了在显示器、打印机或Windows用户区上绘图的方法，它封装使用设备环境的GDI对象。所有的绘图函数都是直接或间接地运用了CDC类的成员函数，这些函数中有的是进行设备描述表操作，有的是由来画图的，有的是用来获取或设置绘图属性，为映像和视点服务的。在MFC中还提供了画笔、画刷、字体、位图、调色板类来为绘图服务。在本文所举的例子中，就是使用CDC类的成员函数来画图形的。

2　与绘图有关的两个消息

由于Windows是基于消息驱动的操作系统，很多与Windows编程的工作都包含消息的处理，因此掌握Windows的消息处理机制是很重要的，在Windows中有两个与绘图有关的消息，大多数的动画程序都是利用它们来实现的。

2.1 WM-TIMER消息

WM－TIMER是一个很重要的系统消息，当系统所设置的时间到达以后，系统就会自动发送该消息。与该消息联系密切的一个函数是SetTimer()，它设置一个系统时钟，当设置的时间到时，系统产生WM-TIMER消息，通过对SetTimer()函数的参数进行设置，告诉用户哪一个时钟的时间到了，因此，可以将一些周期性的工作放入WM-TIMER的消息处理函数中，在本文所举的无闪烁的"蓝宝石旋转"程序，就是利用了该消息，当时间到后，显示下一幅图形，由于人的视觉暂留原因，人感觉屏幕上的图形在旋转，当应用程序结束后，可以通过调用KillTimer()函数来清除定时器。

2.2 WM-PAINT消息

在窗口输出中，最重要的消息是WM-PAINT消息，简单地说，该消息的作用是请求窗口重新绘制其内容。在很多情况下，例如其他窗口覆盖了当前窗口，用户复原已缩为图标的窗口，或者是用户的应用程序刚刚启动等等，应用程序都需要获取该消息。

在有些情况下，产生绘图消息不是外部的因素(如用户改变了窗口的大小)，而是只有程序员自己才能识别的内部因素，例如，用户在文本输入框中添加了一些文字，当窗口中显示的数据发生了变化，便需要产生一个绘图消息。为了人工产生一个绘图消息，需要将窗口内容声明为无效，CWnd类中有几个成员函数能够完成此功能，一个是Invalidate (BOOL bErase=TRUE)函数，它的作用是使整个客户区无效，由此引发WM-PAINT消息，其中参数bErase用于指定当更新区被处理时，更新区的背景是否被清除，如果bErase的值为TRUE，更新区的背景被擦除；如果bErase的值为FALSE，更新区的背景保持不变。另一个是void InvalidateRect(LPCRECT lpRect,BOOL bErase=TRUE)函数，它的作用是把客户区的一个矩形区域添加到CWnd的更新区中，从而使客户在这个矩形区内无效，这个矩形和更新区的其他部分一样，当WM-PAINT消息被送到时，被标记以作图。无效区在更新区内堆积，直到下一个WM-PAINT消息传来，区域被处理。其中参数bErase用于指定当更新区被处理时，更新矩形区的背景是否被清除。如果bErase的值为TRUE，更新矩形区的背景被擦除；如果bErase的值为FALSE，更新矩形区的背景保持不变。

上述方法虽然能够实现绘图，但在动画的显示过程中，往往使用定时器，当时间到了后调用

The Technique of Quick Graph Showing Based on Memory Device

Invalidate()函数，使得客户区无效，间接引发WM-PAINT消息，使得客户区重新绘制，这样屏幕常常由一个场景变化到另一个场景。这时我们首先要将原先的背景擦除，然后再将新绘制的图形输出到屏幕上，由于显示背景不断的擦除，计算机频繁的进行I/O操作都使绘制所需的图形时间变长，这样不但屏幕闪烁频繁，而且显示的图形不连续，速度也急剧下降。

3 利用内存设备环境实现图形的快速显示的原理与过程

要想消除屏幕的闪烁，使得图形显示连续且速度较快，我们就必须不擦除背景并减少图形绘制到屏幕上的时间。在VC中，我们可以采用GDI位图和内存设备环境的方法来消除屏幕的闪烁,并提高显示的速度。这种方法的基本思想是：首先利用绘图函数在内存设备环境以及与显示兼容的位图中开始绘制，等整个所需要显示的图形绘制完后，再将图形快速地复制到屏幕上。这样就不必擦除屏幕，并且图形在显示之前已经在内存中的位图中保存好，然后直接复制到屏幕，消除了在屏幕上直接绘图的时间和背景的擦除，从而消除了屏幕的闪烁，也加快图形的显示速度。以下为使用绘图函数在内存中建立位图的步骤：

(1) 利用CBitmap类的成员函数CreateCompatibleBitmap创建空白位图。

(2) 利用CDC的成员函数CreateCompatible DC创建内存设备环境对象。

(3) 利用CDC的成员函数SelectObject()将位图选入内存设备环境对象中

(4) 利用CDC的绘图函数在内存中绘图(如调用Ellipse (3,3,40,40) 函数画一个圆)

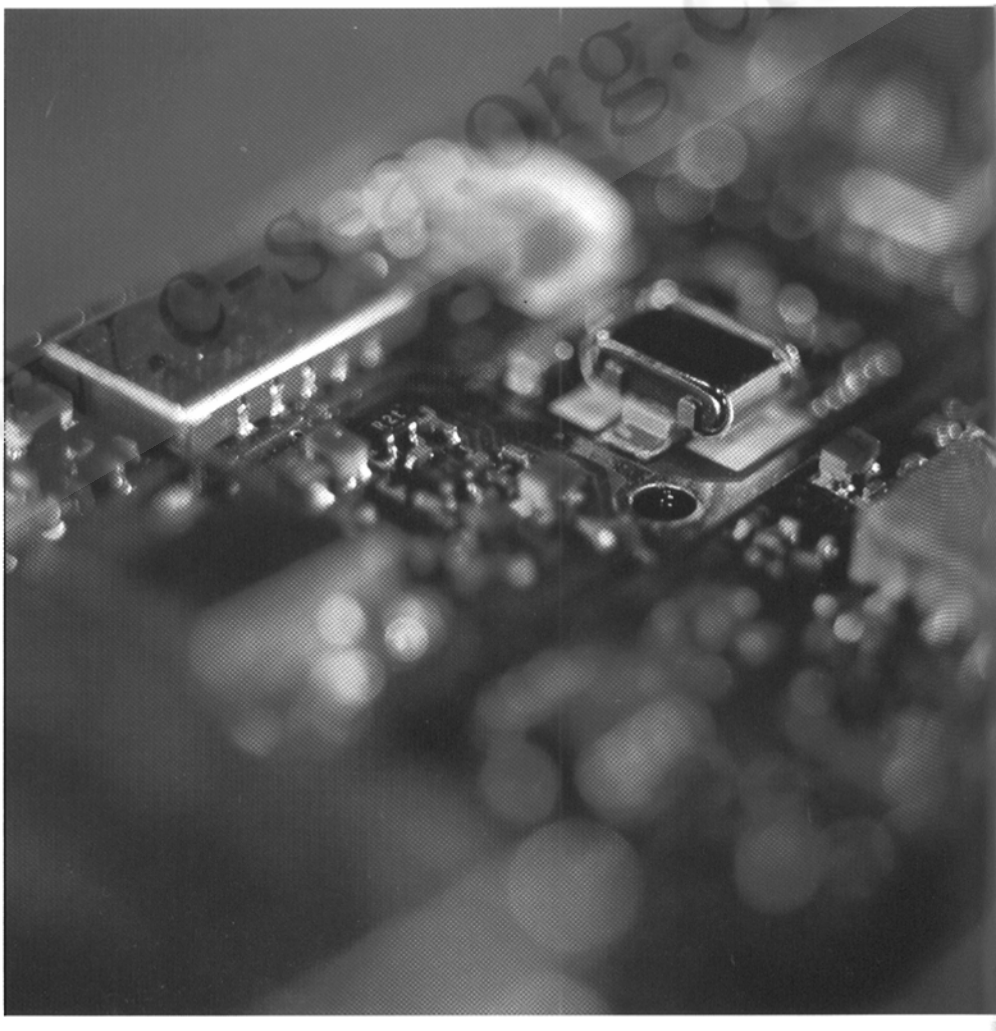
(5) 利用CDC的位传输函数BitBlt()将位图从内存中传输到屏幕上;

4 实例

利用上述的原理，我们编写了以下显示“旋转的金刚石”的动画程序，其编程思路是利用画线函数在内存中绘出静态的金刚石图案，然后将该幅图形从内存中复制到屏幕上，等定时时间到后，再用画线函数在内存中绘出另一幅金刚石图案(方法类似，只是画线的点的坐标偏转了一个小角度)，再将该幅图形从内存中复制到屏幕上。周而复始的做上述的工作，人就感觉所绘制的金刚石(由线所组成的图案)在屏幕上旋转。

下面为程序的主要代码。

```
void CDiamondView::RotatePoints()
{ const double StepAngle=2*PI/MaxPoints; // 角点之间的圆心角
rotation+=PI/32; // 设定每次旋转的角度
if(rotation>StepAngle)
rotation-=StepAngle; // 保持每次旋转角度小于两点之间的圆心角
int i; //i 用来表示角度的次序
double j; //j 用来表示旋转后每个角点的角度
for(i=0,j=rotation;i<MaxPoints;i++,j+=StepAngle)
{ Points [i].x=cos(j);
Points [i].y=sin(j); }
}
int CDiamondView::OnCreate(LPCREATES TRUCT lpCreateStruct)
{
if (CView::OnCreate(lpCreateStruct) == -1)
return -1;
SetTimer(1,10,NULL); // 设置 10 毫秒的时间间隔
```



```
rotation=0; // 初始旋转角度为 0
return 0; }
void CDiamondView::OnDestroy()
{
KillTimer(1); // 清除定时器
CView::OnDestroy(); }
void CDiamondView::OnTimer(UINT nIDEvent)
{ RotatePoints();
DrawDiamond();
CView::OnTimer(nIDEvent); }
void CDiamondView::DrawDiamond()
{ CClientDC dc(this); CDC memDC; CRect rect;
CPen pen ,*pOldPen;
CBitmap pBitmap;
pen.CreatePen(PS-SOLID,1,RGB(0,0,255)); //建立一个蓝色的笔
GetClientRect(rect); // 求出客户区的大下
int CenterX=rect.Width()/2;
int CenterY=rect.Height()/2;
int radius=min(CenterX,CenterY); //设置金刚石的半径
pBitmap.CreateCompatibleBitmap(&dc,2*radius,2*radius);// 初始化空白位图 memDC.CreateCompatibleDC(&dc); //创建内存设备文本对象
CBitmap *pOldBitmap=memDC.SelectObject(&pBitmap);
// 将位图选入内存设备文本对象
pOldPen=memDC.SelectObject(&pen);// 将一个蓝色的笔选入内存设备文本对象
memDC.PatBlt(0,0,2*radius,2*radius,WHITENESS); // 画一个白背景
for(int i=0;i<MaxPoints;i++)
{ int x=radius+Points [i].x*radius;
int y=radius+Points [i].y*radius;
for(int j=i+1;j<MaxPoints;j++)
{ memDC.MoveTo(x,y);
int m=radius+Points [j].x*radius;
int n=radius+Points [j].y*radius;
memDC.LineTo(m,n); }
}
dc.BitBlt(0,0,2*radius,2*radius,&memDC,0,0,SRCCOPY);
// 将图形从内存中复制到屏幕上
memDC.SelectObject(pOldPen); // 恢复旧笔
memDC.SelectObject(pOldBitmap); // 恢复旧位图
memDC.DeleteDC(); // 删除内存设备环境
}
```

程序运行画面如图 1 所示。

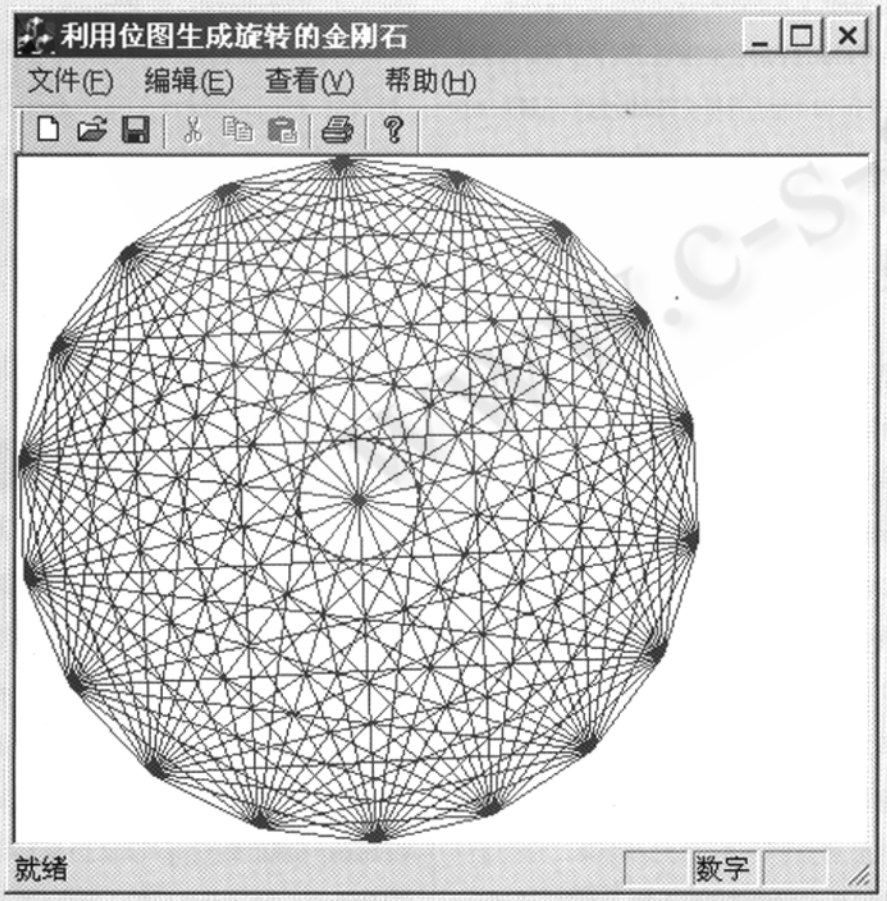


图 1 无闪烁的旋转的金刚石图案

5 结束语

以上绘图方法已经成功的运用在中国光大银行的《触摸屏查询系统》中的动态图形的显示过程中，实践证明，与传统的图形显示方式相比，它彻底地消除了在图形显示画面切换过程中的屏幕闪烁现象，并大大提高了图形的显示速度，实现了图形切换画面的平滑过度。

1 范辉、刘惊雷，Visual C++6.0 程序设计简明教程，高等教育出版社，2001.7。

2 李力、欧阳等，Visual C++6.0 实用数据库编程，中国科技大学出版社，1999.7。