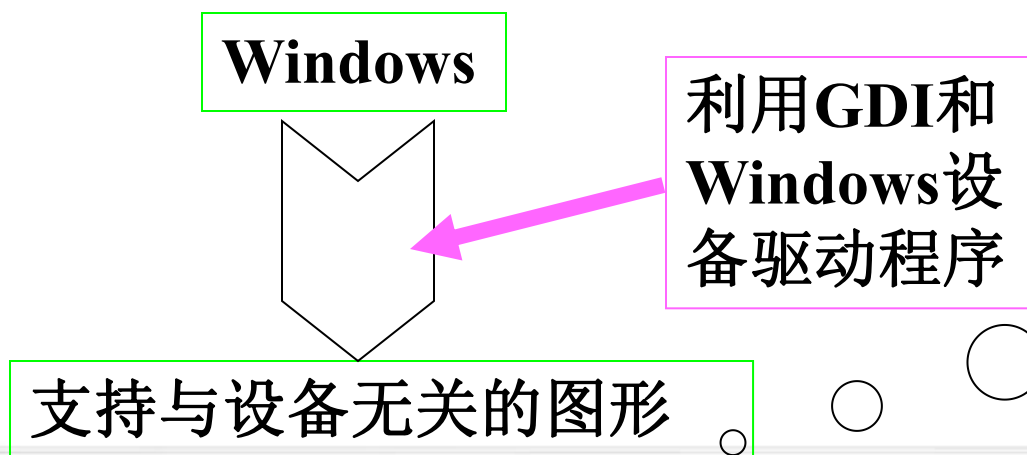


第4章 Windows的图形设备接口及 Windows绘图

- Windows图形设备接口(GDI) 是为与**设备无关**的图形设计的。所谓设备的无关性，就是操作系统屏蔽了硬件设备的差异，因而设备无关性能使用户编程时无需考虑特殊的硬件设置。

一、图形设备接口(GDI)

GDI负责系统与用户或绘图程序之间的信息交换，并控制在输出设备上**显示图形或文字**，是**Windows**系统的重要组成部分。



开发人员只要建立与输出设备的关联，让系统加载相应的设备驱动程序即可

1. GDI的一些基本概念

设备描述表即为设备环境的属性的集合。



应用程序

通过设备描述表的句柄来间接地存取

设备描述表及其属性

应用程序每一次图形
操作均参照设备描述
表中的属性执行

2. 图形刷新

图形刷新是绘图过程中必须考虑的重要问题

包括

刷新请求
对刷新请求的响应
刷新方法

(1) 刷新请求

窗口大小的调整
窗口移动
被覆盖后的恢复

应用程序在窗口中
绘制了一个椭圆，
颜色列表框覆盖了
椭圆的一部分

关闭颜色选框后，
应用程序需要恢
复被覆盖部分的
颜色和形状



(2) 系统对刷新请求的响应

当用户区的内容需要刷新时，系统向应用程序消息队列发送**WM_PAINT**消息，系统在应用程序的消息队列中加入该消息，以通知窗口函数执行刷新处理

三种刷新

窗口移动后的刷新

用户区移动或显示
用户窗口大小改变
程序通过滚动条滚动窗口

被覆盖区域的刷新

窗口被另一个窗口覆盖的
恢复如下拉式菜单关闭等

对象穿越后的刷新
(系统自动完成)

光标穿过用户区
图标拖过用户区

窗口被另一个窗口覆盖的区域称为**无效区域**。 CHINA UNIVERSITY PRESS

Windows系统为每个窗口建立了一个**PAINTSTRUCT**结构，该结构中包含了包围**无效区域**的一个最小矩形的结构**RECT**，应用程序可以根据这个无效矩形执行刷新操作。

Typedef struct tagPAINTSTRUCT

{

HDC hdc; //设备环境句柄

BOOL fErase; //一般取真值，表示擦除无效矩形的背景

RECT rcPaint; //无效矩形标识

BOOL fRestore; //系统保留

BOOL fIncUpdate; //系统保留

BYTE rgbReserved[16]; //系统保留

} PAINTSTRUCT;

rcPaint 为标准的**RECT**数据结构，其作用是标识无效矩形，它包含了无效矩形的左上角和右下角的坐标

(3) 有效的刷新方法

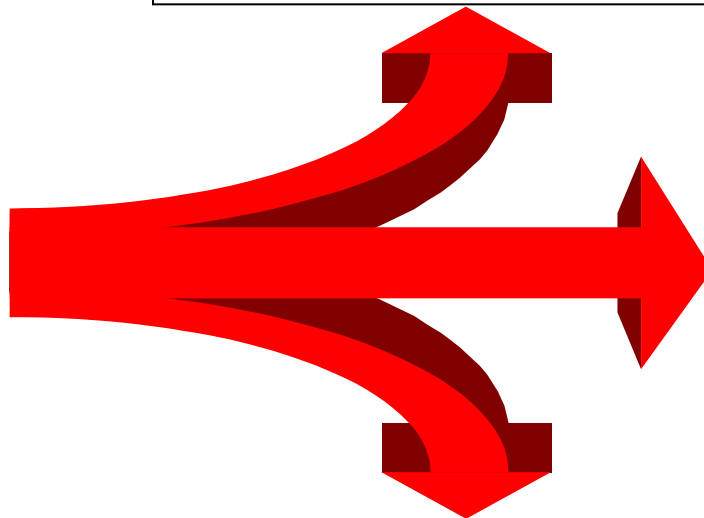
TSINGHUA UNIVERSITY PRESS

常用的Windows
应用程序刷新窗
口的的方法

记录事件。刷新
时重新执行这个
曾经发生的事件

保存副本。刷
新时将副本拷
贝到相应的窗
口中

重新绘制。将图形
绘制处理程序放在
消息WM_PAINT
响应模块中，刷新
时重绘图形



3. 获取设备环境的方法

获取设备环境是应用程序输出图形的先决条件，常用的两种方法是调用函数 **BeginPaint** 或 **GetDC**

(1) 调用 **BeginPaint** 函数

应用程序响应 **WM_PAINT** 消息进行图形刷新时，主要通过调用 **BeginPaint** 函数获取设备环境

hdc=BeginPaint (hwnd, &ps); // **ps** 为 **PAINTSTRUCT** 类型结构

定义方式为: **PAINTSTRUCT ps;**

系统获取设备环境的同时填写 **ps** 结构，以标识无效矩形区

由 **BeginPaint** 函数获取的设备环境要用 **EndPaint** 函数释放

```
void EndPaint(HWND hwnd, PAINTSTRUCT &ps)
```

(2) 调用GetDC函数

如果绘图工作并非由WM_PAINT消息驱动，则调用GetDC函数获取设备环境。

hdc=GetDC(hwnd);

由**GetDC**函数获取的设备环境必须用**ReleaseDC**函数释放

void ReleaseDC(HWND hwnd);

BeginPaint 与 GetDC 的区别

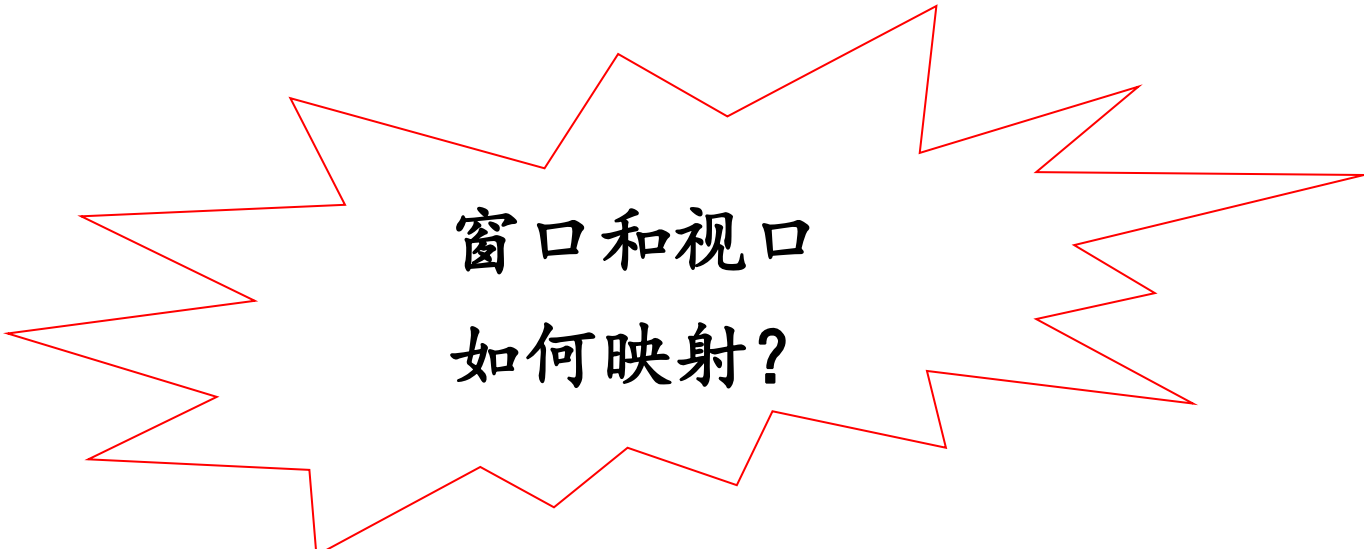
项目 \ 函数	BeginPaint 函数	GetDc 函数
使用环境	只用于图形刷新时获取设备环境	使用较为广泛
操作区域	使用 BeginPaint 函数获取设备环境后，操作区域为无效区域	使用 GetDC 函数获取设备环境后，操作区域为整个用户区
释放设备环境所用函数	由 EndPaint 函数释放	由 ReleaseDC 函数释放

4. 映像模式

映像模式定义了将逻辑单位转化为设备的度量单位以及设备的x方向和y方向，程序员可在一个统一的逻辑坐标系中操作而不必考虑输出设备的坐标系情况。

窗口： 对应逻辑坐标系上程序员设定的区域

视口： 对应实际输出设备上程序员设定的区域



窗口和视口
如何映射？

坐标
系统

逻辑坐标系

设备坐标系

屏幕坐标系
窗口坐标系
用户区坐标系

按照窗口和视口的
坐标比例进行映射

将窗口中的对称
图形映射到视口
时仍为对称图形

默认的映射模式

映像模式	将一个逻辑 单位映射为	坐标系设定
MM_ANISOTROPIC	系统确定	Optional
MM_HIENGLISH	0.001 英寸	Y 上, X 右
MM_HIMETRIC	0.01 毫米	Y 上, X 右
MM_ISOTROPIC	系统确定	Optional, 但 x 轴和 y 轴的单元 位此例为 1: 1
MM_LOENGLISH	0.01 英寸	Y 上, X 右
MM_LOMETRIC	0.1 毫米	Y 上, X 右
MM_TEXT	一个像素	Y 下, X 右
MM_TWIPS	1/1440 英寸	Y 上, X 右

应用程序可获取设备环境的当前映像模式，并根据需要设置映像模式。

相关
函数

设置设备环境的映像模式

SetMapMode(hdc, nMapMode);

nMapMode为映像模式的整型标识符

获取当前设备环境的映像模式

nMapMode=GetMapMode(hdc);

窗口区域的定义由**SetWindowExtEx**函数完成，其函数原型为

```
BOOL SetWindowExtEx  
( HDC hdc,  
  int nHeight, nWidth, //以逻辑单位表示的窗口区域高宽  
度  
  LPSIZE lpSize, //函数调用前窗口区域尺寸的SIZE结构地  
址  
);
```

视口区域的定义由**SetViewportExtEx**函数完成，函数原型为：

```
BOOL Set ViewportExtEx  
(  
  HDC hdc,  
  int nHeight, nWidth, //以物理设备单位表示的新视口区域高宽  
度  
  LPSIZE lpSize  
);
```

只有在映射模式为
MM_ANISOTROPIC
和**MM_ISOTROPIC**
时才有意义

视口的默认原点和窗口的默认原点均为(0, 0)。可通过调用函数SetViewportOrgEx和SetWindowOrgEx设定窗口与视口的原点。

只有在映射模式为
MM_ANISOTROPIC
和MM_ISOTROPIC
时才有意义

SetWindowOrgEx函数的原型为:

BOOL SetWindowOrgEx

(

HDC hdc,

int X, Y,

LPPPOINT lpPoint

);

//以逻辑单位表示的窗口原点坐标

//函数调用前原点坐标的POINT结构的地址

二、绘图工具与颜色

1. 画笔

画笔的操作

创建画笔

将画笔选入设备环境

删除画笔

(1) 画笔的创建

使用画笔之前必须事先定义一个画笔句柄。形式如下：

HPEN hP;

然后调用函数 **GetStockObject** 获取 Windows 系统定义的 **四种** 画笔。
例如获取画笔 **BLACK_PEN** 的形式如下：

hP=GetStockObject (BLACK_PEN)

WHITE_PEN

BLACK_PEN

DC_PEN

NULL_PEN

(2) 创建新画笔，形式如下：

```
hP=CreatePen
```

```
(
```

```
int nPenStyle,      //确定画笔样式
```

```
int nWidth,         //画笔宽度
```

```
COLORREF rgbColor //画笔颜色
```

```
);
```

PS_DASH:

PS_DASHDOT:

PS_DASHDOTDOT:

PS_DOT:

PS_INSIDEFRAME:

PS_NULL:

PS_SOLID:

虚线

点划线

双点划线

点线

实线

无

实线

创建画笔后，必须调用SelectObject函数将其选入设备环境。

SelectObject(hdc, hP); //hP为所创建或获取的画笔句柄

不再使用当前画笔时，需删除画笔，以免占内存

DeleteObject(hP);

2. 画刷

画刷的创建与应用与画笔很相似，**操作画刷也包括创建、选入设备环境和删除。**

(1) 画刷的创建

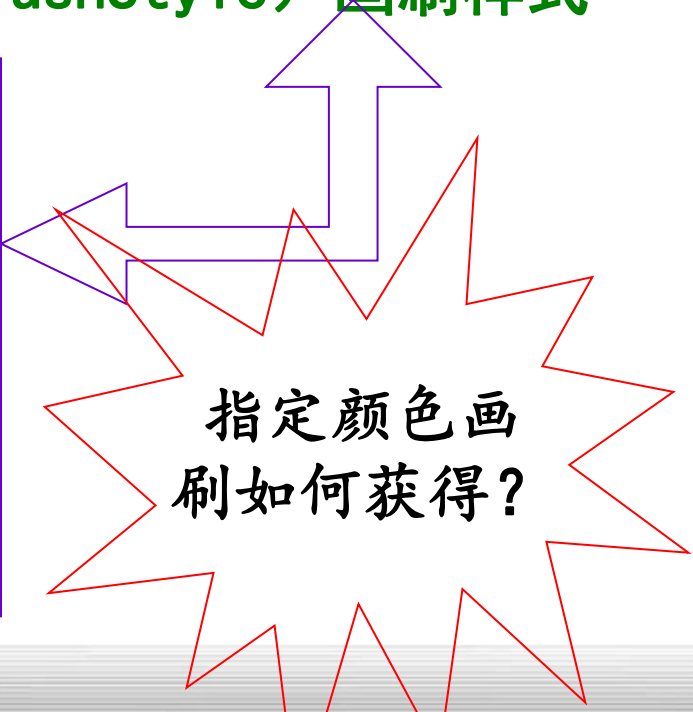
使用画刷需事先定义一个画刷句柄。形式如下：

HBRUSH hBr; //hBr为画刷句柄

然后调用函数**GetStockObject**获取Windows系统提供的7种画刷

hBr = (HBRUSH) GetStockObject(nBrushStyle) 画刷样式

BLACK_BRUSH	黑色画刷
DKGRAY_BRUSH	深灰色画刷
GRAY_BRUSH	灰色画刷
HOLLOW_BRUSH	虚画刷
LTGRAY_BRUSH	亮灰色画刷
NULL_BRUSH	空画刷
WHITE_BRUSH	白色画刷



指定颜色画
刷如何获得?

可调用函数 **CreateSolidBrush** 和 **CreateHatchBrush** 创建画刷,

创建具有
指定颜色
的单色画刷

创建指定阴
影图案和颜
色的画刷

`hBr=CreateSolidBrush(rgbColor);`

`hBr=CreateHatchBrush
(int nHatchStyle,
COLORREF rgbColor
);`

HS_BDIAGONAL	45度从左上到右下
HS_DIAGCROSS	45度叉线
HS_FDIAGONAL	45度从左下到右上
HS_CROSS	垂直相交的阴影线
HS_HORIZONTAL	水平阴影线
HS_VERTICAL	垂直阴影线

(2) 选入设备环境

创建画刷后, 通过 **SelectObject(hdc,hBr);** 将其选入设备环境

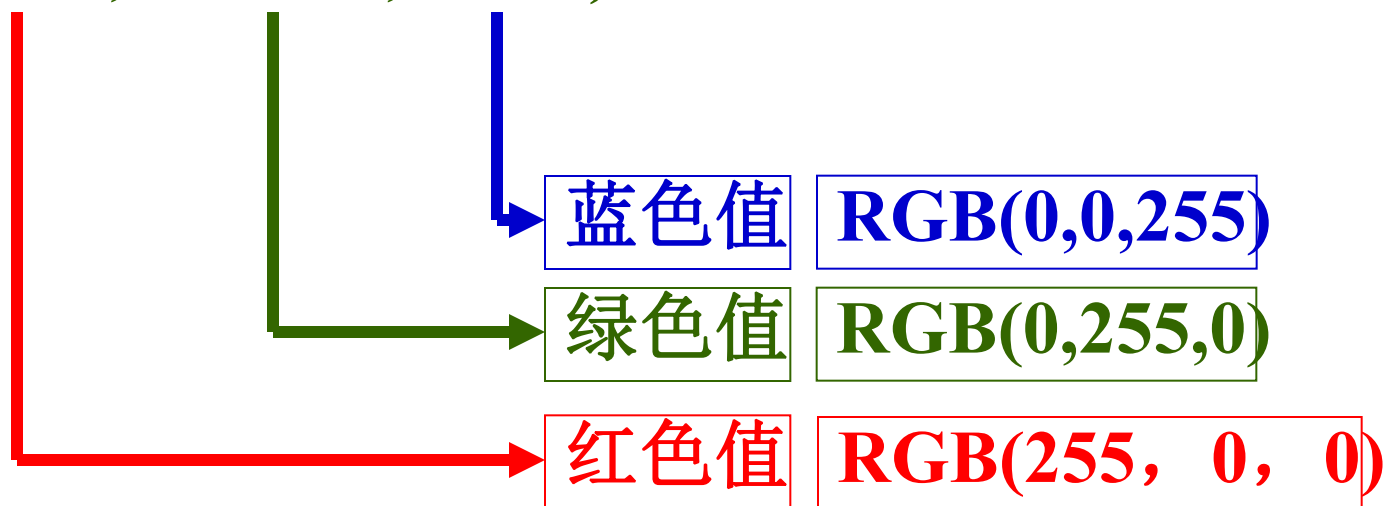
(3) 删除画刷

不使用画刷时, 可用 **DeleteObject(hBr);** 删除画刷, 释放内存

3. 颜色

Windows使用宏RGB定义绘图的颜色，其形式为：

RGB(nRed, nGreen, nBlue)



三、常用绘图函数

(1) 设置画笔当前位置的函数**MoveToEx**,

BOOL MoveToEx

(HDC hdc,

int X,Y,

// X、Y分别为新位置的逻辑坐标

LPPPOINT lpPoint //存放原画笔位置的POINT结构地址

)

(2) 从当前位置向指定坐标点画直线的函数**LineToEx**,

BOOL LineToEx (HDC hdc, int X, int Y) //X和Y为线段的终点坐标

(3) 从当前位置开始, 依次用线段连接lpPoints中指定的各点

BOOL Polyline

(HDC hdc,

LPPPOINT lpPoints, //指向包含各点坐标的POINT结构数组的指针

int nCount // nCount为POINT数组中点的个数

)

(4) 绘制椭圆弧线的函数Arc

BOOL Arc

(

HDC hdc,

int X1,intY1,

int X2,int Y2,

int X3,int Y3,

int X4,int Y4

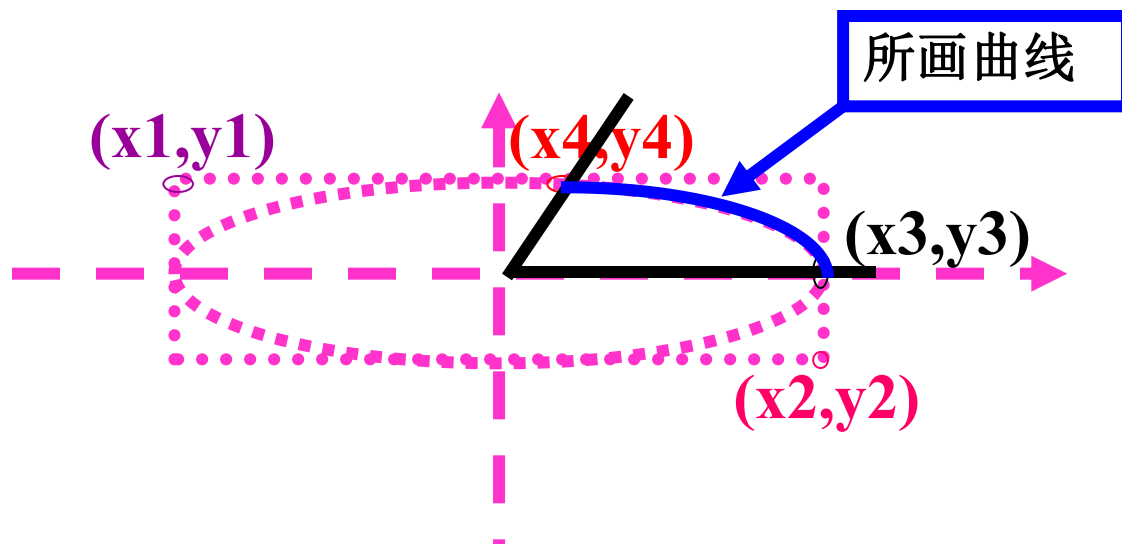
)

//边框矩形左上角的逻辑坐标

//边框矩形右下角的逻辑坐标

//椭圆弧起始点坐标

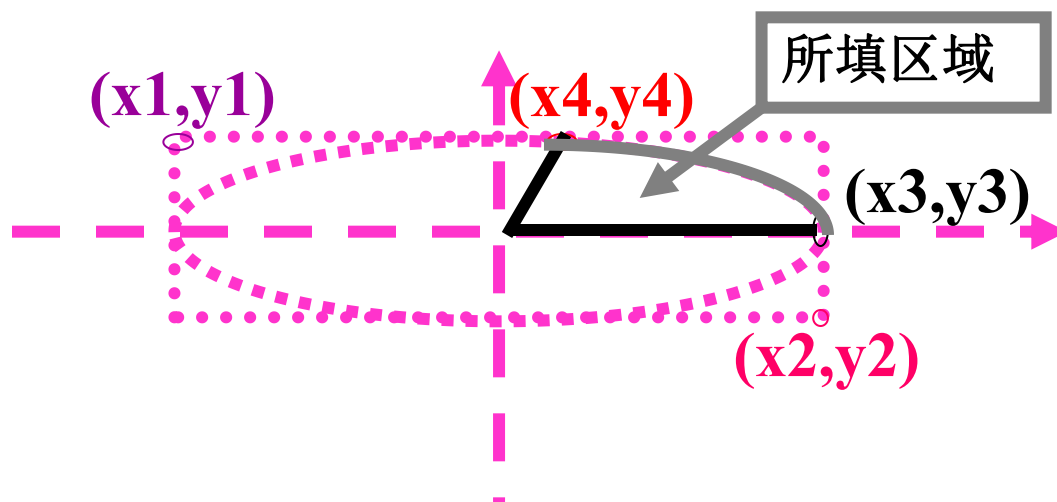
//椭圆弧终止点坐标



(5) 绘制饼图，并用当前画刷进行填充

BOOL Pie

```
(  
    HDC hdc,  
    int X1,intY1,           //边框矩形左上角的逻辑坐标  
    int X2,int Y2,         //边框矩形右下角的逻辑坐标  
    int X3,int Y3,         //椭圆弧起始经线的确定点坐标  
    int X4,int Y4         //椭圆弧终止经线的确定点坐标  
)
```



(6) 绘制矩形，并用当前画刷进行填充

BOOL Rectangle(HDC hdc, int X1, int Y1, int X2, int Y2)

(X1, Y1) 和 (X2, Y2)
分别为矩形的左上角和
右下角的逻辑坐标

(7) 绘制圆角矩形，并用当前画刷填充

BOOL RoundRect (HDC hdc, int X1, int Y1, int X2, int Y2,
int nHeight, int
nWidth)

圆角的高度和宽
度

(8) 绘制椭圆，并用当前画刷填充

BOOL Ellipse(HDC hdc, int X1, int Y1, int X2, int Y2)

(9) 绘制多边形，并用当前画刷填充

BOOL Polygon(HDC hdc, LPPPOINT lpPoints, int nCount)

包含各点坐标的
POINT数组的地址

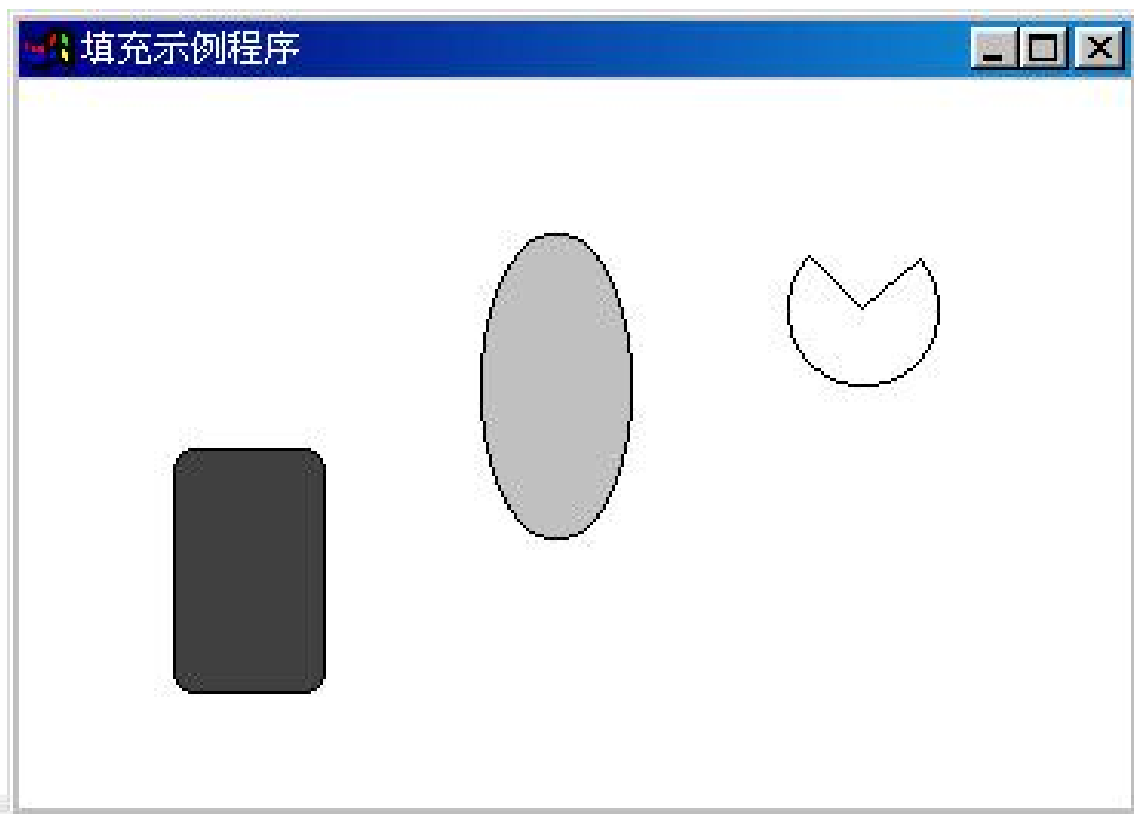
多边形点的个数

四、应用实例

【例4-1】利用绘图函数创建填充区。共有三个填充图形，第一个是用深灰色画刷填充带圆角的矩形，第二个是采用亮灰色画刷填充一个椭圆形图，第三个是用虚画刷填充饼形图。

使用虚画刷填充时，看不出填充效果!!!

```
#include<windows.h>
#include<stdlib.h>
#include<string.h>
long WINAPI WndProc
(
    HWND hWnd,
    UINT iMessage,
    UINT wParam,
    LONG lParam
);
```



BOOL InitWindowsClass(HINSTANCE hInstance);

BOOL InitWindows(HINSTANCE hInstance,int nCmdShow);

HWND hWndMain

int WINAPI WinMain

//主函数

**(HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,
int nCmdShow)**

{

MSG Message;

if(!InitWindowsClass(hInstance))

return FALSE;

if(!InitWindows(hInstance,nCmdShow))

return FALSE;

while(GetMessage(&Message,0,0,0))

//消息循环

{

TranslateMessage(&Message);

DispatchMessage(&Message);

}

return Message.wParam;

}

清华大学出版社
long WINAPI **WndProc** (HWND hWnd, UINT iMessage, SS
消息处理函数
UINT wParam, LONG lParam) {

HDC hDC; //定义指向设备的句柄
HBRUSH hBrush; //定义指向画刷的句柄
HPEN hPen; //定义指向画笔的句柄
PAINTSTRUCT PtStr; //定义指向包含绘图信息的结构体变量

switch(iMessage) //处理消息
{case WM_PAINT: //处理绘图消息

hDC=BeginPaint(hWnd,&PtStr);
SetMapMode(hDC,MM_ANISOTROPIC); //设置映像模式
hPen=(HPEN)GetStockObject(BLACK_PEN); //黑色画笔
hBrush=(HBRUSH)GetStockObject(DKGRAY_BRUSH); //画刷
SelectObject(hDC,hBrush); //选择画刷
SelectObject(hDC,hPen); //选择画笔
RoundRect(hDC,50,120,100,200,15,15); //绘制圆角矩形
hBrush=(HBRUSH)GetStockObject(LTGRAY_BRUSH); //采用
亮灰色画刷

下面只改变
画刷，不改
变笔

```
SelectObject(hDC,hBrush);           //选择画刷
Ellipse(hDC,150,50,200,150);         //绘制椭圆
hBrush=(HBRUSH)GetStockObject(HOLLOW_BRUSH); //虚画刷
SelectObject(hDC,hBrush);           //选择画刷
Pie(hDC,250,50,300,100,250,50,300,50); //绘制饼形
EndPaint(hWnd,&PtStr);               //结束绘图
return 0;

case WM_DESTROY:                     //结束应用程序
    PostQuitMessage(0);              return 0;

default:                             //其他消息处理程序
    return (DefWindowProc (hWnd, iMessage, wParam, lParam))

;

}
```

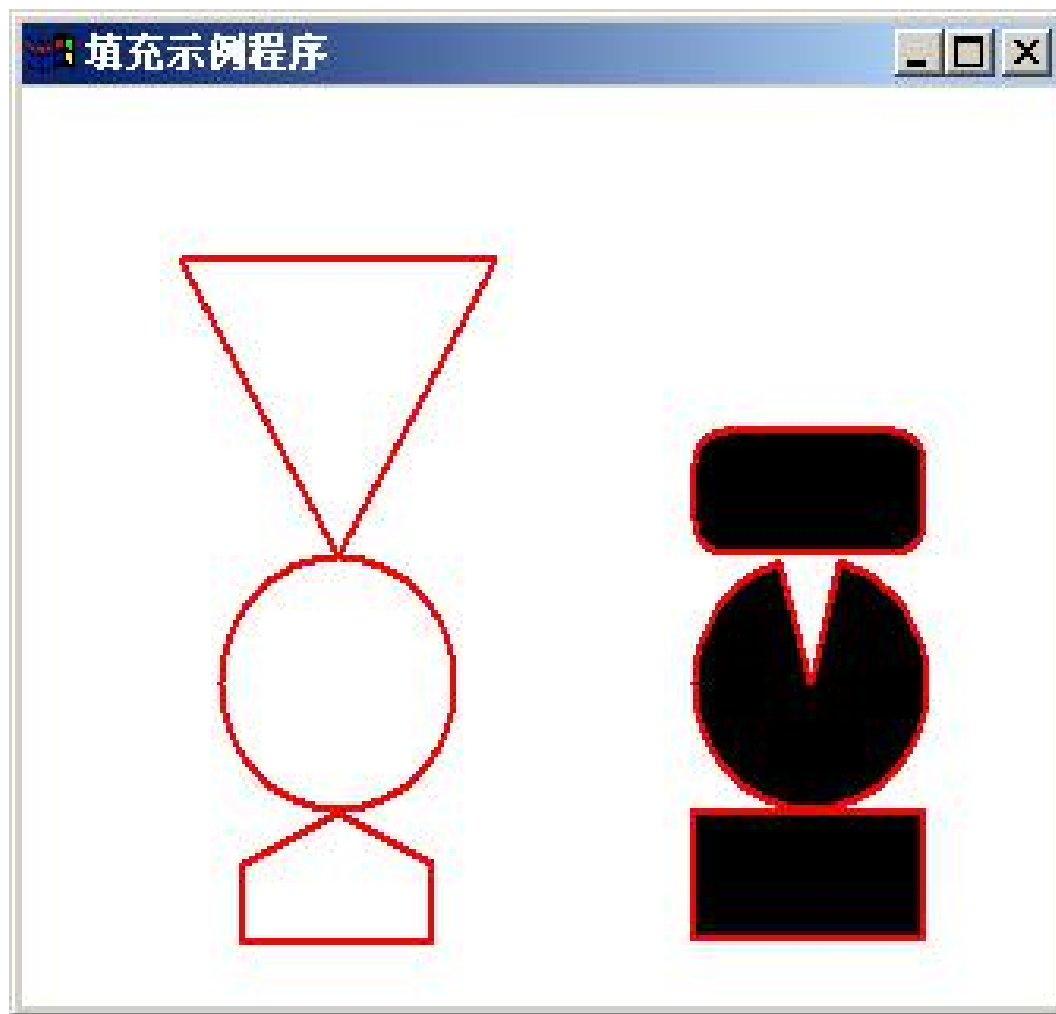
BOOL InitWindows(HINSTANCE hInstance, int nCmdShow) //初始化窗口

```
{    HWND hWnd;
    hWnd=CreateWindow("WinFill",    //生成窗口
                      "填充示例程序",
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      0,
                      CW_USEDEFAULT,
                      0,
                      NULL,
                      NULL,
                      hInstance,
                      NULL);

    if(!hWnd)    return FALSE;
    hWndMain=hWnd;
    ShowWindow(hWnd, nCmdShow);    //显示窗口
    UpdateWindow(hWnd);
    return TRUE;
}
```

```
BOOL InitWindowsClass (HINSTANCE hInstance) //定义窗口类
{
    WNDCLASS WndClass;
    WndClass.cbClsExtra=0;
    WndClass.cbWndExtra=0;
    WndClass.hbrBackground=(HBRUSH) (GetStockObject(WHITE_BRUSH));
    WndClass.hCursor=LoadCursor(NULL, IDC_ARROW);
    WndClass.hIcon=LoadIcon(NULL, "END");
    WndClass.hInstance=hInstance;
    WndClass.lpfnWndProc=WndProc;
    WndClass.lpszClassName="WinFill";
    WndClass.lpszMenuName=NULL;
    WndClass.style=CS_HREDRAW|CS_VREDRAW;
    return RegisterClass(&WndClass);
}
```

例： 应用画笔和画刷等工具绘制如图所示的图形。



1... **#include<windows.h>**

2... **#include<stdlib.h>**

3... **#include<string.h>**

4... **long WINAPI WndProc(HWND hWnd,UINT iMessage,UINT wParam,LPARAM lParam);**

5... **BOOL InitWindowsClass(HINSTANCE hInstance);**

6... **BOOL InitWindows(HINSTANCE hInstance,int nCmdShow);**

7... **HWND hWndMain;**

//主函数

8... **int WINAPI WinMain(HINSTANCE hInstance,**

HINSTANCE hPrevInstance,LPSTR lpCmdLine,int nCmdSh

9... **{ MSG Message;**

10... **if(!InitWindowsClass(hInstance)) return FALSE;**

11... **if(!InitWindows(hInstance,nCmdShow)) return FALSE;**

12... **while(GetMessage(&Message,0,0,0)) //消息循环**

13... **{ TranslateMessage(&Message);**

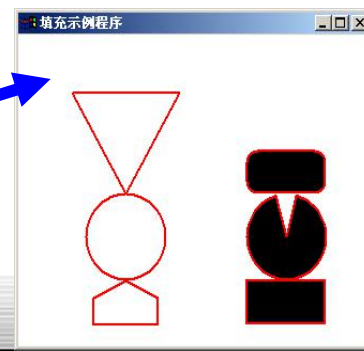
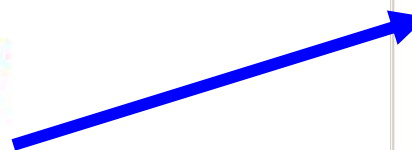
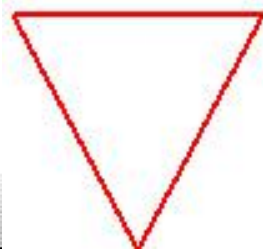
14... **DispatchMessage(&Message); }**

15... **return Message.wParam;**

}

清华大学出版社
long WINAPI WndProc(HWND hWnd,UINT iMessage,UINT wParam,LPARAM lParam)

```
1... {HDC hDC; //定义设备环境句柄
2... HBRUSH hBrush; //定义画刷的句柄
3... HPEN hPen; //定义画笔的句柄
4... PAINTSTRUCT PtStr; //定义指向包含绘图信息的结构体变量
//定义一个POINT数组，包括6个点
5... POINT points[6]={100,212},{70,227},{70,250},{130,250},{130,227},{100,212}}
6... switch(iMessage) //处理消息
7... { case WM_PAINT: //处理绘图消息
8...     hDC=BeginPaint(hWnd,&PtStr);
9...     hPen=(HPEN)GetStockObject(NULL_PEN); //获取系统定义的空画笔
10...     SelectObject(hDC,hPen); //选择画笔
11...     hBrush=(HBRUSH)GetStockObject(BLACK_BRUSH); //获取系统定义的
12...     SelectObject(hDC,hBrush); //选择画刷
13...     LineTo(hDC,50,50); //画线
14...     DeleteObject(hPen); //删除画笔
15...     hPen=CreatePen(PS_SOLID,2,RGB(255,0,0)); //创建画笔
16...     SelectObject(hDC,hPen); //选择画笔
//画一个三角形
17...     LineTo(hDC,150,50);
18...     LineTo(hDC,100,137);
19...     LineTo(hDC,50,50);
```



```
1... Polyline(hDC,points,6);           //画一个五 
2... Arc(hDC,63,137,138,212,100,137,100,137); //画一个圆 
3... Pie(hDC,213,137,288,212,240,137,260,137); //画一个圆饼 
4... Rectangle(hDC,213,212,287,250);      //画一个长方形 
5... RoundRect(hDC,213,100,287,137,20,20); //画一个圆角长方形 
6... DeleteObject(hPen); //删除画笔
7... DeleteObject(hBrush); //删除画刷
8... EndPaint(hWnd,&PtStr); //结束绘图
9... return 0;
10...case WM_DESTROY: //结束应用程序
11... PostQuitMessage(0);
12... return 0;
13...default://其他消息处理程序
14... return(DefWindowProc(hWnd,iMessage,wParam,lParam)) ;
    }
}
```

```
BOOL InitWindows(HINSTANCE hInstance,int nCmdShow) //初始化窗
{
    HWND hWnd;
    hWnd=CreateWindow("WinFill",                                //生成窗口
                      "填充示例程序",
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      0,
                      CW_USEDEFAULT,
                      0,
                      NULL,
                      NULL,
                      hInstance,
                      NULL);

    if(!hWnd)
        return FALSE;
    hWndMain=hWnd;
    ShowWindow(hWnd,nCmdShow);                                //显示窗口
    UpdateWindow(hWnd);
    return TRUE;
}
```

BOOL InitWindowsClass(HINSTANCE hInstance)

//定义窗口类

{WNDCLASS WndClass;

WndClass.cbClsExtra=0;

WndClass.cbWndExtra=0;

WndClass.hbrBackground=(HBRUSH)(GetStockObject(WHITE_BRUSH));

WndClass.hCursor=LoadCursor(NULL, IDC_ARROW);

WndClass.hIcon=LoadIcon(NULL, "END");

WndClass.hInstance=hInstance;

WndClass.lpfnWndProc=WndProc;

WndClass.lpszClassName="WinFill";

WndClass.lpszMenuName=NULL;

WndClass.style=CS_HREDRAW|CS_VREDRAW;

return RegisterClass(&WndClass);

}

清华大学出版社

【例4-2】编写一个程序，在屏幕上出现一个圆心沿正弦曲线轨迹移动的实心圆，而且，每隔四分之一周期，圆的填充色和圆的周边颜色都发生变化，同时，圆的半径在四分之一周期之内由正弦曲线幅值的0.2倍至0.6倍线性增长。 [双击此处运行程序](#)

(1) 正弦曲线是此题的基础。在WndMain()函数消息循环前，生成正弦曲线各点的坐标。把正弦曲线一个周期的横坐标分成100个等分点，存储在数组lpSin[100]中，100个点的坐标计算如下：

```
for(int j=0;j<100;j++)    //生成正弦曲线的点坐标
{
    lpSin[j].x=(long)(j*2*Pi/100*60);
    lpSin[j].y=(long)(dfRange*sin(j*2*Pi/100));
}
```

(2) 动态显示圆在正弦曲线上移动

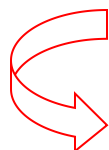
数组lpSin[100]的长度为100



设定圆在正弦曲线移动时共有100个位置



数组中每一个值是圆移动时圆心的坐标



每四分之一周期有25个位置

$i \leq 25$	处于第1个1/4周期, 创建红色画笔和画刷;
$25 < i < 50$	处于第2个1/4周期, 创建绿色画笔和画刷;
$50 < i < 75$	处于第3个1/4周期, 创建蓝色画笔和画刷;
$75 < i < 100$	处于第4个1/4周期, 创建黄色画笔和画刷;

在消息WM_PAINT处理程序中，调用函数BeginPaint() 获得设备环境句柄。由此经过线性差分计算圆半径的大小lRadiums，第1个1/4周期的程序代码如下：

```
if (i<=25)                //第一个1/4周期
{
    hPen=CreatePen (PS_DASH, 1, RGB (255, 0, 0)) ;

    hBrush=CreateHatchBrush (HS_BDIAGONAL, RGB (255, 0, 0)) ;
    lRadiums=(long) (dfRange*0. 2+i%25*dfRange*0. 4/25) ;//计算
半径
}
```


创建的画笔和画刷选入设备环境后，调用函数Ellipse(...)绘制圆形。下面这段代码是动态显示的关键：

```
Sleep(100);           //停0.1秒
```

```
if(i<100) InvalidateRect(hWnd, NULL, 1); //刷新用户区
```

代表清除用户区中所有的显示内容

i<100时调用函数刷新用户区发送WM_PAINT消息

消息发到的窗口的句柄

代表刷新整个用户区

调用Sleep(100)函数使程序暂停0.1秒。所含参数100代表暂停的时间，使用毫秒作单位。


```
#include <windows.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <math.h>
```

```
#define Pi 3.1415926
```

```
long WINAPI WndProc(HWND hWnd,UINT iMessage,  
                    UINT wParam,LONG lParam);
```

```
double dfTheta=0,dfRange=100.0;           //正弦曲线的角度变量
```

```
long i=0,j=0;
```

```
long lCentreX=0,lCentreY=0,lRadius=(long)(0.2*dfRange);  
                                           //定义圆心坐标和圆半径
```

```
POINT lpSin[100]; //定义正弦曲线的点坐标
```

```
int WINAPI WinMain(...,....,....,....)
```

```
{      ..... // 填写窗口类属性
```

```
    if(!RegisterClass(&WndClass))           //注册窗口
```

```
    {      MessageBeep(0);      return FALSE;}
```

hWnd=CreateWindow

```
("SIN", //窗口类名
"4_6", //标题名
WS_OVERLAPPEDWINDOW, //带标题栏,最大/小按钮的窗口
CW_USEDEFAULT, //窗口左上角坐标
0,
CW_USEDEFAULT, //采用默认的宽度和高度
0,
NULL, //无父窗口
NULL, //无主菜单
hInstance, //当前实例句柄
NULL);.
```

ShowWindow(hWnd,nCmdShow); //显示窗口

UpdateWindow(hWnd); //更新并绘制用户区

```
1.      for(int j=0;j<100;j++)                //生成正弦曲线的点坐标
2.      {
3.          lpSin[j].x=(long)(j*2*Pi/100*60);
4.          lpSin[j].y=(long)(dfRange*sin(j*2*Pi/100));
5.      }

6.      while(GetMessage(&Message,0,0,0))
7.      {
8.          TranslateMessage(&Message);        //消息循环
9.          DispatchMessage(&Message);
10.     }
11.     return Message.wParam;
}
```

```
long WINAPI WndProc(HWND hWnd,UINT iMessage,  
                    UINT wParam,LPARAM lParam)
```

```
{  
1.   HDC hDC;                //定义设备环境句柄  
2.   HBRUSH hBrush;          //定义画刷句柄  
3.   HPEN hPen;              //定义画笔句柄  
4.   PAINTSTRUCT PtStr;      //定义包含绘图信息的结构体变量  
  
5.   switch(iMessage)  
{  
6.   case WM_PAINT:          //处理绘图消息  
7.       hDC=BeginPaint(hWnd,&PtStr);    //获得设备环境指针  
8.       SetWindowOrgEx(hDC,-200,-200,NULL); //设置原点坐标  
9.       hPen=CreatePen(PS_DASH,1,RGB(255,0,0)); //建新画笔  
10.      SelectObject(hDC,hPen);          //选入画笔  
11.      Polyline(hDC,lpSin,100);         //绘制正弦曲线
```

```
if(i<=25) //第一个1/4周期
{
    hPen=CreatePen(PS_DASH,1,RGB(255,0,0));
    hBrush=CreateHatchBrush(HS_BDIAGONAL,RGB(255,0,0));
    lRadius=(long)(dfRange*0.2+i%25*dfRange*0.4/25); //计算半径
}

else if(i<=50) //第二个1/4周期
{
    hPen=CreatePen(PS_DASH,1,RGB(0,255,0));
    hBrush=CreateHatchBrush(HS_DIAGCROSS,RGB(0,255,0));
    lRadius=(long)(dfRange*0.2+i%25*dfRange*0.4/25);
}

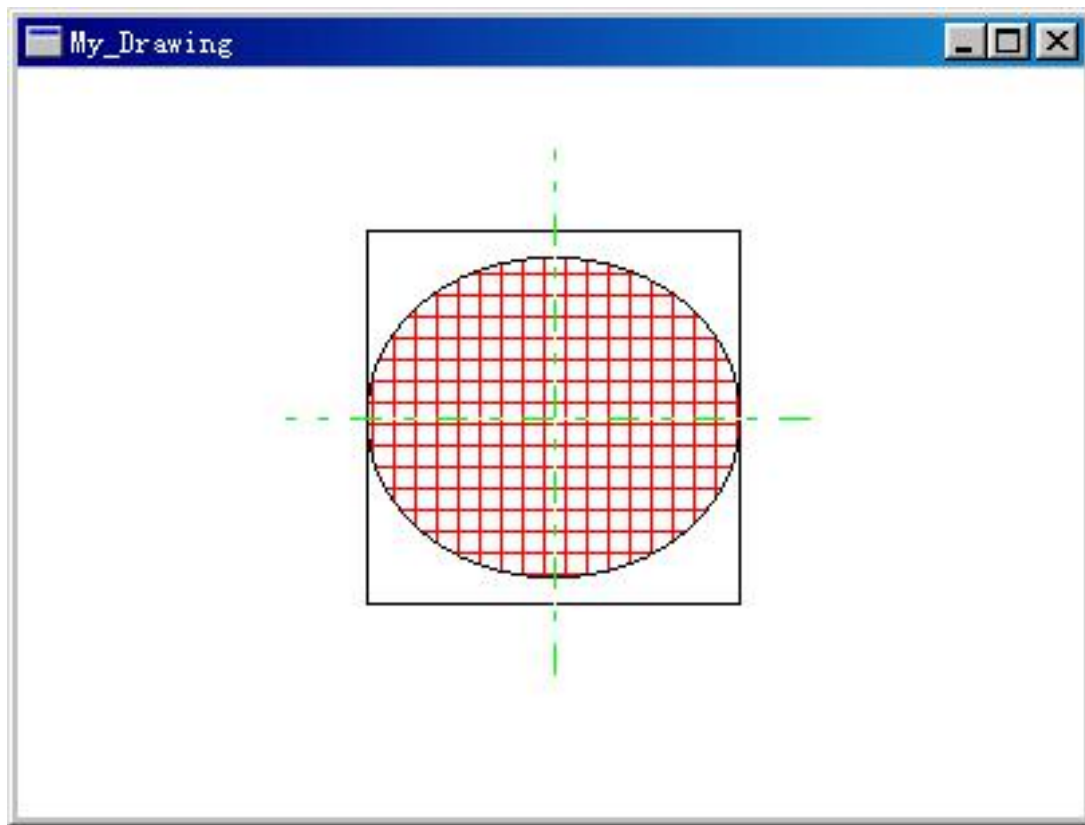
else if(i<=75) //第三个周期
{
    hPen=CreatePen(PS_DASH,1,RGB(0,0,255));
    hBrush=CreateHatchBrush(HS_CROSS,RGB(0,0,255));
    lRadius=(long)(dfRange*0.2+i%25*dfRange*0.4/25);
}

else //第四个周期
{
    hPen=CreatePen(PS_DASH,1,RGB(255,255,0));
    hBrush=CreateHatchBrush(HS_VERTICAL,RGB(255,255,0));
    lRadius=(long)(dfRange*0.2+i%25*dfRange*0.4/25);
}
```

```
1.      SelectObject(hDC,hBrush);           //选入画刷
2.      SelectObject(hDC,hPen);             //选入画笔
3.      lCentreX=lpSin[i].x;                //圆心x坐标
4.      lCentreY=lpSin[i].y;                //圆心y坐标
5.      Ellipse(hDC,lCentreX-lRadious,lCentreY-lRadious,
           lCentreX+lRadious,lCentreY+lRadious); //画圆

6.      i++;
7.      DeleteObject(hPen);                 //删除画笔
8.      DeleteObject(hBrush);              //删除画刷
9.      EndPaint(hWnd,&PtStr);              //删除设备环境指针
10.     Sleep(100);                         //停0.1秒
11.     if(i<100) InvalidateRect(hWnd,NULL,1); //刷新用户区
12.     return 0;
13. case WM_DESTROY:                        //关闭窗口
14.     PostQuitMessage(0);                 return 0;
15. default:
16.     return(DefWindowProc(hWnd,iMessage,wParam,lParam));
} }
```

【例4-3】绘图与刷新。制订一种重新绘制图形的刷新方式，将图形绘制模块放在消息WM_PAINT的处理过程中，当窗口需要刷新时，通知窗口函数重新绘制图形以完成刷新工作。本例要求先使用画笔和画刷绘制一个矩形，然后使用红色网格绘制一个椭圆，再使用绿色点划线绘制椭圆的轴线。




```
#include <windows.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst,
                  LPSTR lpszCmdLine,          int
nCmdShow)
{ HWND hwnd; MSG Msg;      WNDCLASS wndclass;
  char lpszClassName[] = "基本绘图";
  char lpszTitle[] = "My_Drawing";
  wndclass.style = 0;
      .....                //填写属性
  wndclass.lpszClassName = lpszClassName ;
  if(!RegisterClass(&wndclass))
      { MessageBeep(0); return FALSE; }

  hwnd = CreateWindow(..., ....., ...);
  ShowWindow(hwnd, nCmdShow) ;   UpdateWindow(hwnd);
  while( GetMessage(&Msg, NULL, 0, 0))
      { TranslateMessage(&Msg); DispatchMessage(&Msg); }
  return Msg.wParam;
}
```


LRESULT CALLBACK WndProc(HWND hwnd, UINT message,
WPARAM wParam, LPARAM lParam)

```
{ HDC hdc;  
  PAINTSTRUCT ps;  
  HPEN hP;           //定义画笔句柄  
  HBRUSH hB;         //定义画刷句柄  
  switch(message)  
  {  
    case WM_PAINT:    //通过响应WM_PAINT消息完成绘图工作  
      hP=CreatePen(PS_DASHDOT, 1, RGB(0, 255, 0)); //自定义绿笔  
                                              //所画线条为点划线，宽度为1  
      hB=CreateHatchBrush(HS_CROSS, RGB(255, 0, 0)); //红色网状  
      hdc=BeginPaint(hwnd, &ps); //取得设备环境句柄  
      SetMapMode(hdc, MM_TEXT); //设置映射模式，用默认模式  
                               //使用当前默认画笔、画刷进行绘图  
      Rectangle(hdc, 130, 60, 270, 200); //绘制矩形，并填充  
      SelectObject(hdc, hB); //更新画刷，用“红色网状”  
      Ellipse(hdc, 130, 70, 270, 190); //绘制椭圆，并填充
```

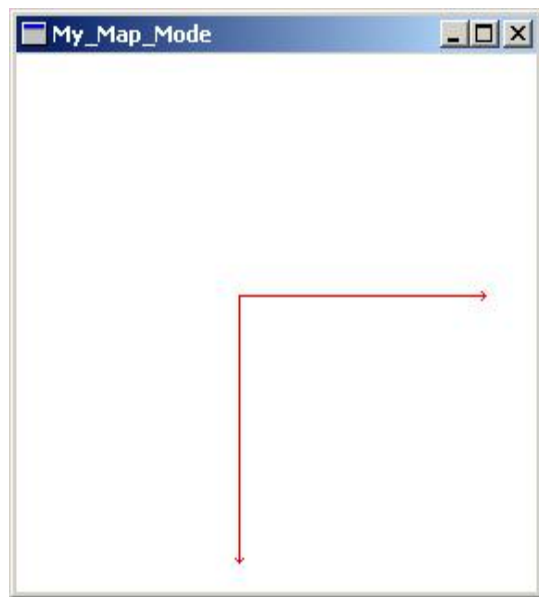
```
SelectObject(hdc, hP);    //更新画笔, 选“自定义绿笔”
MoveToEx(hdc, 100, 130, NULL); //使用当前画笔绘制轴线
LineTo(hdc, 300, 130);
MoveToEx(hdc, 200, 30, NULL);
LineTo(hdc, 200, 230);
EndPaint(hwnd, &ps);      //释放设备环境句柄
break;

case WM_DESTROY:
    DeleteObject(hP);        //退出窗口时删除画笔
    DeleteObject(hB);        //退出窗口时删除黑色画刷
    PostQuitMessage(0);
    break;

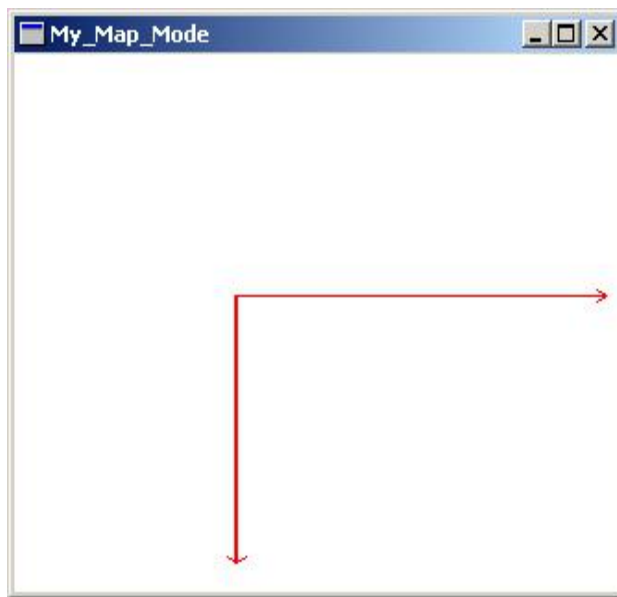
default:
    return DefWindowProc(hwnd, message, wParam, lParam);
}

return 0;
}
```

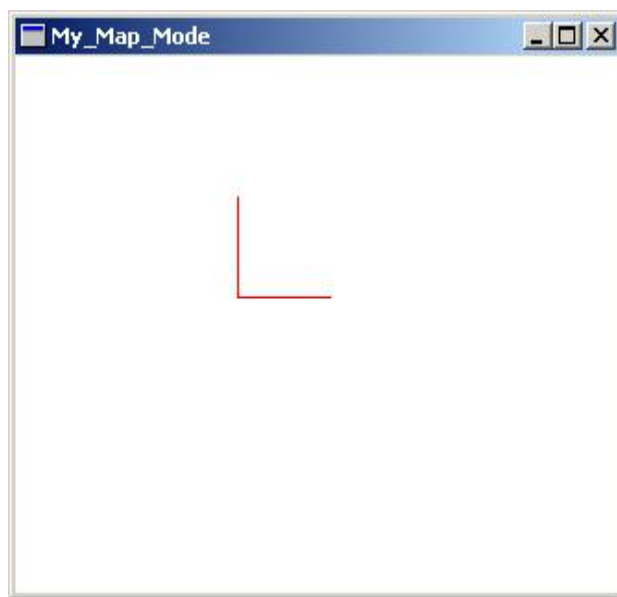
【例4-4】设置映像与使用映像模式实例。本例中的程序运行时，初始阶段按模式MM_TEXT绘图，图形为一个坐标系，以逻辑坐标系的原点为原点，X、Y轴分别是逻辑坐标系的X、Y轴。当用户按下A键、B键或C键时，产生WM_CHAR消息，将映像模式分别设置为ISOTROPIC、ANISOTROPIC或LOMETRIC，同时调用InvalidateRect函数刷新用户区。



按A键



按B键



按C键

```
1. #include <windows.h>
2. #include <string.h>
3. #include <stdlib.h>
4. #include <stdio.h>
5. LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
6. int nMode=MM_TEXT;           //设置映像模式的初始值
7. ....
8. LRESULT CALLBACK WndProc (HWND hwnd, UINT message,
                               WPARAM wParam, LPARAM lParam)
9. {   HDC hdc;
10.    PAINTSTRUCT ps;
11.    HPEN hPen;
12.    switch (message)
13.    {
14.    case WM_CHAR:           //按下不同的键时，设置不同的映像模式
15.        if (wParam=='a' || wParam=='A')
16.            nMode=MM_ISOTROPIC;
17.        else if (wParam=='b' || wParam=='B')   nMode=MM_ANISOTROPIC;
18.        else if (wParam=='c' || wParam=='C')   nMode=MM_LOMETRIC;
19.        else ;
20.        InvalidateRect (hwnd, NULL, 1);        //刷新用户区
        break;
```

```
1. case WM_PAINT:
2.     hdc=BeginPaint(hwnd, &ps); //取得设备环境句柄
3.     SetMapMode(hdc, nMode); //设置映像模式
4.     SetWindowExtEx(hdc, 15, 15, NULL); //设置窗口区域
5.     SetViewportExtEx(hdc, 15, 10, NULL); //设置视口区域
6.     SetViewportOrgEx(hdc, 120, 120, NULL); //设置视口原点
7.     hPen=CreatePen(PS_SOLID, 2, RGB(255, 0, 0)); //创建红色画笔
8.     SelectObject(hdc, hPen); //将画笔选入设备环境
9.     //画坐标系, 原点在视口原点
10.    LineTo(hdc, 200, 0);
11.    LineTo(hdc, 195, -5);
12.    MoveToEx(hdc, 200, 0, NULL);
13.    LineTo(hdc, 195, 5);
14.    MoveToEx(hdc, 0, 0, NULL);
15.    LineTo(hdc, 0, 200);
16.    LineTo(hdc, -5, 195);
17.    MoveToEx(hdc, 0, 200, NULL);
18.    LineTo(hdc, 5, 195);
19.    DeleteObject(hPen);
20.    EndPaint(hwnd, &ps);
    break;

    case WM_DESTROY: PostQuitMessage(0); break;
default: return DefWindowProc(hwnd, message, wParam, lParam);
}

return 0; }
```

