这一节我们来学习gdi编程

# 学习大纲

|  |
| --- |
|  |

## GetDC函数的用法

|  |
| --- |
| **GetDC** 函数检索指定窗口的工作区或整个屏幕的设备上下文 (DC) 的句柄。 可以在后续 GDI 函数中使用返回的句柄在 DC 中绘制。 设备上下文是一种不透明的数据结构，其值由 GDI 在内部使用。  [GetDCEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getdcex) 函数是 **GetDC** 的扩展，它使应用程序能够更好地控制在工作区中发生剪裁的方式和是否发生。 语法 C++复制  HDC GetDC(  [in] HWND hWnd  ); 参数 [in] hWnd  要检索其 DC 的窗口的句柄。 如果此值为 **NULL**， **则 GetDC** 将检索整个屏幕的 DC。 返回值 如果函数成功，则返回值是指定窗口工作区的 DC 的句柄。  如果函数失败，则返回值为 NULL。 注解 **GetDC** 函数根据指定窗口的类样式检索公共、类或专用 DC。 对于类和专用 **DC，GetDC** 保留以前分配的属性不变。 但是，对于常见的 **DC，GetDC** 在每次检索 DC 时都会将默认属性分配给 DC。 例如，默认字体为 System，即位图字体。 因此， **GetDC** 返回的通用 DC 的句柄不会告诉你在绘制窗口时使用了哪种字体、颜色或画笔。 若要确定字体，请调用 [GetTextFace](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-gettextfacea)。  请注意，DC 的句柄一次只能由单个线程使用。  使用通用 DC 进行绘制后，必须调用 [ReleaseDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-releasedc) 函数来释放 DC。 类和专用 DC 不必释放。 **ReleaseDC** 必须从调用 **GetDC** 的同一线程调用。 DC 的数量仅受可用内存的限制。 示例 有关示例，请参阅 [使用鼠标绘图](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/drawing-with-the-mouse)。  BOOL fDraw = FALSE;  POINT ptPrevious;    .  .  .    case WM\_LBUTTONDOWN:  fDraw = TRUE;  ptPrevious.x = LOWORD(lParam);  ptPrevious.y = HIWORD(lParam);  return 0L;    case WM\_LBUTTONUP:  if (fDraw)  {  hdc = GetDC(hwnd);  MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);  LineTo(hdc, LOWORD(lParam), HIWORD(lParam));  ReleaseDC(hwnd, hdc);  }  fDraw = FALSE;  return 0L;    case WM\_MOUSEMOVE:  if (fDraw)  {  hdc = GetDC(hwnd);  MoveToEx(hdc, ptPrevious.x, ptPrevious.y, NULL);  LineTo(hdc, ptPrevious.x = LOWORD(lParam),  ptPrevious.y = HIWORD(lParam));  ReleaseDC(hwnd, hdc);  }  return 0L;  启用绘图的应用程序（如本示例所示）通常记录点或线，以便每当更新窗口时都可以重新绘制线条。 绘图应用程序通常使用内存设备上下文和关联的位图来存储使用鼠标绘制的线条。 |

## ReleaseDC函数的语法

|  |  |
| --- | --- |
| **ReleaseDC** 函数 (DC) 释放设备上下文，释放它供其他应用程序使用。 **ReleaseDC** 函数的效果取决于 DC 的类型。 它仅释放公用 DC 和窗口 DC。 它对类或专用 DC 没有影响。 语法 C++复制  int ReleaseDC(  [in] HWND hWnd,  [in] HDC hDC  ); 参数 [in] hWnd  要释放其 DC 的窗口的句柄。  [in] hDC  要释放的 DC 的句柄。 返回值 返回值指示是否释放了 DC。 如果释放 DC，则返回值为 1。  如果未释放 DC，则返回值为零。 注解 对于每次调用 [GetWindowDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getwindowdc) 函数和每次调用检索公用 DC 的 [GetDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getdc) 函数，应用程序都必须调用 **ReleaseDC** 函数。  应用程序不能使用 **ReleaseDC** 函数释放通过调用 [CreateDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdca) 函数创建的 DC;相反，它必须使用 [DeleteDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deletedc) 函数。 **ReleaseDC** 必须从调用 [GetDC](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getdc) 的同一线程调用。 示例 有关示例，请参阅 [缩放图像](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/scaling-an-image)。   |  | | --- | | #include "stdafx.h"  #include "GDIBitmapScaling.h"  #include <commctrl.h>  #include <CommDlg.h>  #define MAX\_LOADSTRING 100  // Global Variables:  HINSTANCE hInst; // current instance  TCHAR szTitle[MAX\_LOADSTRING]; // The title bar text  TCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name  // Forward declarations of functions included in this code module:  ATOM MyRegisterClass(HINSTANCE hInstance);  BOOL InitInstance(HINSTANCE, int);  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);  int APIENTRY \_tWinMain(HINSTANCE hInstance,  HINSTANCE hPrevInstance,  LPTSTR lpCmdLine,  int nCmdShow)  {  UNREFERENCED\_PARAMETER(hPrevInstance);  UNREFERENCED\_PARAMETER(lpCmdLine);  // TODO: Place code here.  MSG msg;  HACCEL hAccelTable;  // Initialize global strings  LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);  LoadString(hInstance, IDC\_GDIBITMAPSCALING, szWindowClass, MAX\_LOADSTRING);  MyRegisterClass(hInstance);  // Perform application initialization:  if (!InitInstance (hInstance, nCmdShow))  {  return FALSE;  }  hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_GDIBITMAPSCALING));  // Main message loop:  while (GetMessage(&msg, NULL, 0, 0))  {  if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  }  return (int) msg.wParam;  }  //  // FUNCTION: MyRegisterClass()  //  // PURPOSE: Registers the window class.  //  // COMMENTS:  //  // This function and its usage are only necessary if you want this code  // to be compatible with Win32 systems prior to the 'RegisterClassEx'  // function that was added to Windows 95. It is important to call this function  // so that the application will get 'well formed' small icons associated  // with it.  //  ATOM MyRegisterClass(HINSTANCE hInstance)  {  WNDCLASSEX wcex;  wcex.cbSize = sizeof(WNDCLASSEX);  wcex.style = CS\_HREDRAW | CS\_VREDRAW;  wcex.lpfnWndProc = WndProc;  wcex.cbClsExtra = 0;  wcex.cbWndExtra = 0;  wcex.hInstance = hInstance;  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_GDIBITMAPSCALING));  wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);  wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_GDIBITMAPSCALING);  wcex.lpszClassName = szWindowClass;  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));  return RegisterClassEx(&wcex);  }  //  // FUNCTION: InitInstance(HINSTANCE, int)  //  // PURPOSE: Saves instance handle and creates main window  //  // COMMENTS:  //  // In this function, we save the instance handle in a global variable and  // create and display the main program window.  //  #define NEW\_DIB\_FORMAT(lpbih) (lpbih->biSize != sizeof(BITMAPCOREHEADER))  BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  {  HWND hWnd;  hInst = hInstance; // Store instance handle in our global variable  hWnd = CreateWindow(szWindowClass, szTitle, WS\_SYSMENU,  CW\_USEDEFAULT, 0, 1024, 768, NULL, NULL, hInstance, NULL);  if (!hWnd)  {  return FALSE;  }  ShowWindow(hWnd, nCmdShow);  UpdateWindow(hWnd);  return TRUE;  }  static HCURSOR hcurSave;  WORD DIBNumColors (LPVOID lpv)  {  INT bits;  LPBITMAPINFOHEADER lpbih = (LPBITMAPINFOHEADER)lpv;  LPBITMAPCOREHEADER lpbch = (LPBITMAPCOREHEADER)lpv;  /\* With the BITMAPINFO format headers, the size of the palette  \* is in biClrUsed, whereas in the BITMAPCORE - style headers, it  \* is dependent on the bits per pixel ( = 2 raised to the power of  \* bits/pixel).  \*/  if (NEW\_DIB\_FORMAT(lpbih)) {  if (lpbih->biClrUsed != 0)  return (WORD)lpbih->biClrUsed;  bits = lpbih->biBitCount;  }  else  bits = lpbch->bcBitCount;  if (bits > 8)  return 0; /\* Since biClrUsed is 0, we dont have a an optimal palette \*/  else  return (1 << bits);  }  /\* Macro to determine to round off the given value to the closest byte \*/  #define WIDTHBYTES(i) ((((i)+31) >> 5) << 2)  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* FUNCTION : DIBInfo(HANDLE hbi, LPBITMAPINFOHEADER lpbih) \*  \* \*  \* PURPOSE : Retrieves the DIB info associated with a CF\_DIB \*  \* format memory block. \*  \* \*  \* RETURNS : TRUE - if successful. \*  \* FALSE - otherwise \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  BOOL DIBInfo (HANDLE hbi, LPBITMAPINFOHEADER lpbih)  {  if (hbi){  \*lpbih = \*(LPBITMAPINFOHEADER)hbi;  /\* fill in the default fields \*/  if (NEW\_DIB\_FORMAT(lpbih)) {  if (lpbih->biSizeImage == 0L)  lpbih->biSizeImage = WIDTHBYTES(lpbih->biWidth\*lpbih->biBitCount) \* lpbih->biHeight;  if (lpbih->biClrUsed == 0L)  lpbih->biClrUsed = DIBNumColors (lpbih);  }  return TRUE;  }  return FALSE;  }  /\* flags for mmioSeek() \*/  #ifndef SEEK\_SET  #define SEEK\_SET 0 /\* seek to an absolute position \*/  #define SEEK\_CUR 1 /\* seek relative to current position \*/  #define SEEK\_END 2 /\* seek relative to end of file \*/  #endif /\* ifndef SEEK\_SET \*/  VOID ReadPackedFileHeader(HFILE hFile, LPBITMAPFILEHEADER lpbmfhdr, LPDWORD lpdwOffset)  {  \*lpdwOffset = \_llseek(hFile, 0L, (UINT) SEEK\_CUR);  \_hread(hFile, (LPSTR) &lpbmfhdr->bfType, sizeof(WORD)); /\* read in bfType\*/  \_hread(hFile, (LPSTR) &lpbmfhdr->bfSize, sizeof(DWORD) \* 3); /\* read in last 3 dwords\*/  }  /\* macro to determine if resource is a DIB \*/  #define ISDIB(bft) ((bft) == BFT\_BITMAP)  /\* Header signatutes for various resources \*/  #define BFT\_ICON 0x4349 /\* 'IC' \*/  #define BFT\_BITMAP 0x4d42 /\* 'BM' \*/  #define BFT\_CURSOR 0x5450 /\* 'PT' \*/  HANDLE ReadDIBBitmapInfo (INT hFile)  {  DWORD dwOffset;  HANDLE hbi = NULL;  INT size;  INT i;  WORD nNumColors;  LPRGBQUAD lprgbq;  BITMAPINFOHEADER bih;  BITMAPCOREHEADER bch;  LPBITMAPINFOHEADER lpbih;  BITMAPFILEHEADER bf;  DWORD dwDWMasks= 0;  DWORD dwWidth = 0;  DWORD dwHeight = 0;  WORD wPlanes, wBitCount;  if (hFile == HFILE\_ERROR)  return NULL;  /\* Read the bitmap file header \*/  ReadPackedFileHeader(hFile, &bf, &dwOffset);  /\* Do we have a RC HEADER? \*/  if (!ISDIB (bf.bfType)) {  bf.bfOffBits = 0L;  \_llseek(hFile, dwOffset, (UINT)SEEK\_SET); /\* seek back to beginning of file \*/  }  if (sizeof(bih) != \_hread(hFile, (LPSTR)&bih, (UINT)sizeof(bih)))  return FALSE;  nNumColors = DIBNumColors (&bih);  /\* Check the nature (BITMAPINFO or BITMAPCORE) of the info. block  \* and extract the field information accordingly. If a BITMAPCOREHEADER,  \* transfer it's field information to a BITMAPINFOHEADER-style block  \*/  switch (size = (INT)bih.biSize){  case sizeof (BITMAPINFOHEADER):  break;  case sizeof (BITMAPCOREHEADER):  bch = \*(LPBITMAPCOREHEADER)&bih;  dwWidth = (DWORD)bch.bcWidth;  dwHeight = (DWORD)bch.bcHeight;  wPlanes = bch.bcPlanes;  wBitCount = bch.bcBitCount;  bih.biSize = sizeof(BITMAPINFOHEADER);  bih.biWidth = dwWidth;  bih.biHeight = dwHeight;  bih.biPlanes = wPlanes;  bih.biBitCount = wBitCount;  bih.biCompression = BI\_RGB;  bih.biSizeImage = 0;  bih.biXPelsPerMeter = 0;  bih.biYPelsPerMeter = 0;  bih.biClrUsed = nNumColors;  bih.biClrImportant = nNumColors;  \_llseek(hFile, (LONG)sizeof (BITMAPCOREHEADER) - sizeof (BITMAPINFOHEADER), (UINT)SEEK\_CUR);  break;  default:  /\* Not a DIB! \*/  return NULL;  }  /\* Fill in some default values if they are zero \*/  if (bih.biSizeImage == 0){  bih.biSizeImage = WIDTHBYTES((DWORD)bih.biWidth \* bih.biBitCount) \* bih.biHeight;  }  if (bih.biClrUsed == 0)  bih.biClrUsed = DIBNumColors(&bih);  /\* Allocate for the BITMAPINFO structure and the color table. \*/  if ((bih.biBitCount == 16) || (bih.biBitCount == 32))  dwDWMasks = sizeof(DWORD) \* 3;  hbi = GlobalAlloc (GPTR, (LONG)bih.biSize + nNumColors \* sizeof(RGBQUAD) + dwDWMasks);  if (!hbi)  return NULL;  lpbih = (LPBITMAPINFOHEADER)hbi;  \*lpbih = bih;  /\* Get a pointer to the color table \*/  lprgbq = (LPRGBQUAD)((LPSTR)lpbih + bih.biSize);  if (nNumColors){  if (size == sizeof(BITMAPCOREHEADER)){  /\* Convert a old color table (3 byte RGBTRIPLEs) to a new  \* color table (4 byte RGBQUADs)  \*/  \_hread(hFile, (LPSTR)lprgbq, (UINT)nNumColors \* sizeof(RGBTRIPLE));  for (i = nNumColors - 1; i >= 0; i--){  RGBQUAD rgbq;  rgbq.rgbRed = ((RGBTRIPLE\*)lprgbq)[i].rgbtRed;  rgbq.rgbBlue = ((RGBTRIPLE\*)lprgbq)[i].rgbtBlue;  rgbq.rgbGreen = ((RGBTRIPLE\*)lprgbq)[i].rgbtGreen;  rgbq.rgbReserved = (BYTE)0;  lprgbq[i] = rgbq;  }  }  else  \_hread(hFile, (LPSTR)lprgbq, (UINT)nNumColors \* sizeof(RGBQUAD));  } else  if (dwDWMasks)  \_hread(hFile, (LPSTR)lprgbq, dwDWMasks);  if (bf.bfOffBits != 0L){  \_llseek(hFile, dwOffset + bf.bfOffBits, (UINT)SEEK\_SET);  }    return hbi;  }  HGLOBAL GlobalFreeDIB(HGLOBAL hDIB)  {  LPBITMAPINFOHEADER lpbi = (LPBITMAPINFOHEADER)hDIB;  if (!lpbi->biClrImportant)  return GlobalFree(hDIB);  if (GlobalFlags((HGLOBAL)lpbi->biClrImportant) == GMEM\_INVALID\_HANDLE) {  SetLastError(0);  return GlobalFree(hDIB);  } else  return GlobalFree((HANDLE)lpbi->biClrImportant);  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* FUNCTION : ColorTableSize(LPVOID lpv) \*  \* \*  \* PURPOSE : Calculates the palette size in bytes. If the info. block \*  \* is of the BITMAPCOREHEADER type, the number of colors is \*  \* multiplied by 3 to give the palette size, otherwise the \*  \* number of colors is multiplied by 4. \*  \* \*  \* RETURNS : Color table size in number of bytes. \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  WORD ColorTableSize (LPVOID lpv)  {  LPBITMAPINFOHEADER lpbih = (LPBITMAPINFOHEADER)lpv;    if (NEW\_DIB\_FORMAT(lpbih))  {  if (((LPBITMAPINFOHEADER)(lpbih))->biCompression == BI\_BITFIELDS)  /\* Remember that 16/32bpp dibs can still have a color table \*/  return (sizeof(DWORD) \* 3) + (DIBNumColors (lpbih) \* sizeof (RGBQUAD));  else  return (DIBNumColors (lpbih) \* sizeof (RGBQUAD));  }  else  return (DIBNumColors (lpbih) \* sizeof (RGBTRIPLE));  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* FUNCTION :OpenDIB(LPSTR szFilename) \*  \* \*  \* PURPOSE :Open a DIB file and create a memory DIB -- a memory handle \*  \* containing BITMAPINFO, palette data and the bits. \*  \* \*  \* RETURNS :A handle to the DIB. \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  HANDLE OpenDIB (LPSTR szFilename)  {  HFILE hFile;  BITMAPINFOHEADER bih;  LPBITMAPINFOHEADER lpbih;  DWORD dwLen = 0;  DWORD dwBits;  HANDLE hDIB;  HANDLE hMem;  OFSTRUCT of;  /\* Open the file and read the DIB information \*/  hFile = OpenFile(szFilename, &of, (UINT)OF\_READ);  if (hFile == HFILE\_ERROR)  return NULL;  hDIB = ReadDIBBitmapInfo(hFile);  if (!hDIB)  return NULL;  DIBInfo(hDIB, &bih);  /\* Calculate the memory needed to hold the DIB \*/  dwBits = bih.biSizeImage;  dwLen = bih.biSize + (DWORD)ColorTableSize (&bih) + dwBits;  /\* Try to increase the size of the bitmap info. buffer to hold the DIB \*/  hMem = GlobalReAlloc(hDIB, dwLen, GMEM\_MOVEABLE);  if (!hMem){  GlobalFreeDIB(hDIB);  hDIB = NULL;  }  else  hDIB = hMem;  /\* Read in the bits \*/  if (hDIB){  lpbih = (LPBITMAPINFOHEADER)hDIB;  \_hread(hFile, (LPSTR)lpbih + (WORD)lpbih->biSize + ColorTableSize(lpbih), dwBits);  }  \_lclose(hFile);  return hDIB;  }/\* Macros to display/remove hourglass cursor for lengthy operations \*/  #define StartWait() hcurSave = SetCursor(LoadCursor(NULL, IDC\_WAIT))  #define EndWait() SetCursor(hcurSave)  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* FUNCTION : BitmapFromDIB(HANDLE hDIB, HPALETTE hPal) \*  \* \*  \* PURPOSE : Will create a DDB (Device Dependent Bitmap) given a global \*  \* handle to a memory block in CF\_DIB format \*  \* \*  \* RETURNS : A handle to the DDB. \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  HBITMAP BitmapFromDIB (HANDLE hDIB, HPALETTE hPal)  {  LPBITMAPINFOHEADER lpbih;  HPALETTE hPalOld;  HDC hDC;  HBITMAP hBitmap;  StartWait();  if (!hDIB)  return NULL;  lpbih = (LPBITMAPINFOHEADER)hDIB;  if (!lpbih)  return NULL;  hDC = GetDC(NULL);  if (hPal){  hPalOld = SelectPalette(hDC, hPal, FALSE);  RealizePalette(hDC);  }    hBitmap = CreateDIBitmap(hDC,  lpbih,  CBM\_INIT,  (LPSTR)lpbih + lpbih->biSize + ColorTableSize(lpbih),  (LPBITMAPINFO)lpbih,  DIB\_RGB\_COLORS );  if (hPal)  SelectPalette(hDC, hPalOld, FALSE);  ReleaseDC(NULL, hDC);  EndWait();  return hBitmap;  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* \*  \* FUNCTION : DrawBitmap(HDC hDC, int x, int y, \*  \* HBITMAP hBitmap, DWORD dwROP) \*  \* \*  \* PURPOSE : Draws bitmap <hBitmap> at the specified position in DC <hDC> \*  \* \*  \* RETURNS : Return value of BitBlt() \*  \* \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  BOOL DrawBitmap (HDC hDC, INT x, INT y, HBITMAP hBitmap, DWORD dwROP)  {  HDC hDCBits;  BITMAP Bitmap;  BOOL bResult;  if (!hDC || !hBitmap)  return FALSE;  hDCBits = CreateCompatibleDC(hDC);  GetObject(hBitmap, sizeof(BITMAP), (LPSTR)&Bitmap);  SelectObject(hDCBits, hBitmap);  bResult = BitBlt(hDC, x, y, Bitmap.bmWidth, Bitmap.bmHeight, hDCBits, 0, 0, dwROP);  DeleteDC(hDCBits);  return bResult;  }  /\*FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)  PURPOSE: Processes messages for the main window.  WM\_COMMAND - process the application menu  WM\_PAINT - Paint the main window  WM\_DESTROY - post a quit message and return\*/  #define ID\_LOADBITMAP 1  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  //Handle to a GDI device context  HDC hDC;  //Handle to a DDB(device-dependent bitmap)  HBITMAP hBitmap;    HFONT hFont;  NONCLIENTMETRICS ncm={0};  ncm.cbSize= sizeof(NONCLIENTMETRICS);      static HWND hwndButton;  static HWND hwndButtonExit;  static HANDLE hDIB = NULL;    char szDirName[MAX\_PATH];  char szFilename[MAX\_PATH]="\0";  char szBitmapName[MAX\_PATH]="\\Waterfall.bmp";  //char szBitmapName[MAX\_PATH]="\\tulips256.bmp";  OPENFILENAMEA ofn;      switch (message)  {    case WM\_CREATE:    //Creates a font from the current theme's caption font  SystemParametersInfo(SPI\_GETNONCLIENTMETRICS, NULL, &ncm, NULL);  hFont = CreateFontIndirect(&ncm.lfCaptionFont);    //Gets the device context for the current window  hDC = GetDC(hWnd);    //Gets the directory of the current project and loads Waterfall.bmp  GetCurrentDirectoryA(MAX\_PATH, szDirName);  strcat\_s(szDirName, szBitmapName);  strcat\_s(szFilename,szDirName);  hDIB = OpenDIB(szFilename);  hBitmap = BitmapFromDIB(hDIB, NULL);  //Draws Waterfall.bmp as a device dependent bitmap  DrawBitmap(hDC,0,0,hBitmap,SRCCOPY);  InvalidateRect(hWnd, NULL, FALSE);  ReleaseDC(hWnd,hDC);    //Draws the "Load Bitmap" button  hwndButton = CreateWindowW(TEXT("button"),TEXT("Load Bitmap"),  WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 600, 200, 150,50,  hWnd,  (HMENU)ID\_LOADBITMAP,  ((LPCREATESTRUCT) lParam)-> hInstance, NULL);  //Set the font of the button to the theme's caption font  SendMessage(hwndButton, WM\_SETFONT, (WPARAM)hFont, TRUE );      return 0;  case WM\_COMMAND:  wmId = LOWORD(wParam);  wmEvent = HIWORD(wParam);      //if Load Bitmap button is pressed  if (wmId == ID\_LOADBITMAP && wmEvent == BN\_CLICKED)  {    //Get the current directory name, and store in szDirName  GetCurrentDirectoryA(MAX\_PATH, szDirName);    //Set all structure members to zero.  ZeroMemory(&ofn, sizeof(OPENFILENAMEA));    //Initializing the OPENFILENAMEA structure  ofn.lStructSize = sizeof(OPENFILENAMEA);  ofn.hwndOwner = hWnd;  ofn.lpstrFilter ="BMP Files (\*.BMP)\0 \*.BMP\0\0";  ofn.lpstrFile = szFilename;  ofn.nMaxFile = sizeof(szFilename);  ofn.lpstrDefExt = "BMP";  ofn.lpstrInitialDir = szDirName;  ofn.Flags = OFN\_EXPLORER | OFN\_SHOWHELP | OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;    if (GetOpenFileNameA(&ofn)) {  hDIB = OpenDIB((LPSTR)szFilename);  if (!hDIB)  MessageBox(hWnd, TEXT("Unable to load file!"), TEXT("Oops"), MB\_ICONSTOP);  } else {  if (strlen((const char \*)szFilename) != 0)  MessageBox(hWnd, TEXT("Unable to load file!"), TEXT("Oops"), MB\_ICONSTOP);    return 0;  }        InvalidateRect(hWnd, NULL, FALSE);      }      break;  case WM\_PAINT:  {  //Initializing arrays for boxes  POINT pRect1[5] = {{0,0},{400,0},{400,400},{0,400},{0,0}};  POINT pRect2[5] = {{0,500}, {200, 500}, {200, 700}, {0,700},{0,500}};  POINT pRect3[5] = {{210,500}, {410, 500}, {410, 700}, {210,700},{210,500}};  POINT pRect4[5] = {{420,500}, {620, 500}, {620, 700}, {420,700},{420,500}};  POINT pRect5[5] = {{630,500}, {830, 500}, {830, 700}, {630,700},{630,500}};    //For the white background  RECT clientRect;  HRGN hRegion1;  HRGN hRegion2;  HRGN hRegion3;  HBRUSH hBGBrush;    //Handle to a logical font  HFONT hFont;  //Get the caption font that is currently in use  SystemParametersInfo(SPI\_GETNONCLIENTMETRICS, NULL, &ncm, NULL);  hFont = CreateFontIndirect(&ncm.lfCaptionFont);    //Begin drawing  hDC = BeginPaint(hWnd, &ps);  //Draw and fill rectangles for the background  GetClientRect(hWnd, &clientRect);  hRegion1 = CreateRectRgn(clientRect.left,clientRect.top,clientRect.right,clientRect.bottom);  hBGBrush = CreateSolidBrush(RGB(255,255,255));  FillRgn(hDC, hRegion1, hBGBrush);    //Create an HBITMAP(device dependent bitmap) to be drawn  hBitmap = BitmapFromDIB(hDIB,NULL);  //Draw the DDB  DrawBitmap(hDC,0,0,hBitmap,SRCCOPY);      if(hDIB)  {  hRegion2 = CreateRectRgn(401,0,clientRect.right,401);  hRegion3 = CreateRectRgn(0,401,clientRect.right,clientRect.bottom);  FillRgn(hDC,hRegion2,hBGBrush);  FillRgn(hDC,hRegion3,hBGBrush);  //Set stretch mode as BLACKONWHITE and then copy from the  //source rectangle into the smaller rectangle  SetStretchBltMode(hDC,BLACKONWHITE);  StretchBlt(hDC,0,500,200,200,hDC, 0,0,400,400,SRCCOPY);    //Set stretch mode as WHITEONBLACK and then copy from the  //source rectangle into the smaller rectangle  SetStretchBltMode(hDC,WHITEONBLACK);  StretchBlt(hDC,210,500,200,200, hDC, 0,0,400,400, SRCCOPY);    //Set stretch mode as COLORONCOLOR and then copy from the  //source rectangle into the smaller rectangle  SetStretchBltMode(hDC,COLORONCOLOR);  StretchBlt(hDC,420,500,200,200, hDC, 0,0,400,400, SRCCOPY);  //Set stretch mode as HALFTONE and then copy from the  //source rectangle into the smaller rectangle  SetStretchBltMode(hDC,HALFTONE);  StretchBlt(hDC,630,500,200,200, hDC, 0,0,400,400, SRCCOPY);  }  //Select the caption font created earlier  SelectObject(hDC,hFont);  //Create captions for each demonstration of color loss modes  TextOut(hDC,50,480,TEXT("BLACKONWHITE"),12);  TextOut(hDC,250,480,TEXT("WHITEONBLACK"),12);  TextOut(hDC,460,480,TEXT("COLORONCOLOR"),12);  TextOut(hDC,680,480,TEXT("HALFTONE"),8);  DeleteObject(hFont);  //Selecting the stock object pen to draw with  SelectObject(hDC, GetStockObject(DC\_PEN));  //The pen is gray  SetDCPenColor(hDC, RGB(80,80,80));    //Polylines are drawn from arrays of POINTs  Polyline(hDC, pRect1, 5);  Polyline(hDC, pRect2, 5);  Polyline(hDC, pRect3, 5);  Polyline(hDC, pRect4, 5);  Polyline(hDC, pRect5, 5);    FillRgn(hDC,hRegion2,hBGBrush);  EndPaint(hWnd, &ps);    break;  }  case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  } | |

## CreatePen函数的语法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CreatePen** 函数创建具有指定样式、宽度和颜色的逻辑笔。 笔随后可以选择到设备上下文中，并用于绘制线条和曲线。 语法 C++复制  HPEN CreatePen(  [in] int iStyle,  [in] int cWidth,  [in] COLORREF color  ); 参数 [in] iStyle  笔样式。 它可以是以下任意一个值。  展开表   |  |  | | --- | --- | | **值** | **含义** | | **PS\_SOLID** | 触控笔是实心的。 | | **PS\_DASH** | 触控笔虚线。 仅当笔宽度为 1 或更少（以设备单位为单位）时，此样式才有效。 | | **PS\_DOT** | 笔被点点。 仅当笔宽度为 1 或更少（以设备单位为单位）时，此样式才有效。 | | **PS\_DASHDOT** | 笔具有交替的短划线和点。 仅当笔宽度为 1 或更少（以设备单位为单位）时，此样式才有效。 | | **PS\_DASHDOTDOT** | 笔具有交替的短划线和双点。 仅当笔宽度为 1 或更少（以设备单位为单位）时，此样式才有效。 | | **PS\_NULL** | 笔不可见。 | | **PS\_INSIDEFRAME** | 触控笔是实心的。 在采用边框的任何 GDI 绘图函数中使用此笔时，图形的尺寸会缩小，使其完全适合边界矩形，同时考虑笔的宽度。 这仅适用于几何笔。 |   [in] cWidth  笔的宽度（以逻辑单位为单位）。 如果 *nWidth* 为零，则无论当前转换如何，笔都是一个像素宽。  如果为以下样式指定宽度大于 1 的宽度，**CreatePen** 将返回具有指定宽度但具有PS\_SOLID样式的笔：PS\_DASH、PS\_DOT、PS\_DASHDOT PS\_DASHDOTDOT。  [in] color  笔颜色的颜色参考。 若要生成 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 结构，请使用 [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb) 宏。 返回值 如果函数成功，则返回值是标识逻辑笔的句柄。  如果函数失败，则返回值为 NULL。 注解 应用程序创建逻辑笔后，可以通过调用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数将该笔选择到设备上下文中。 将笔选入设备上下文后，可用于绘制线条和曲线。  如果 *nWidth* 参数指定的值为零，则无论当前转换如何，使用创建的笔绘制的线条始终为单个像素宽。  如果 *nWidth* 指定的值大于 1，则 *fnPenStyle* 参数必须是PS\_NULL、PS\_SOLID或PS\_INSIDEFRAME。  如果 *nWidth* 指定的值大于 1 且 *fnPenStyle* PS\_INSIDEFRAME，则与笔关联的线条将绘制在所有基元（多边形和折线除外）的框架内。  如果 *nWidth* 指定的值大于 1，则 *fnPenStyle* 为PS\_INSIDEFRAME，并且 *crColor* 参数指定的颜色与逻辑调色板中的某个条目不匹配，则系统会使用抖动的颜色绘制线条。 抖变颜色不适用于纯色笔。  使用PS\_DASH、PS\_DOT、PS\_DASHDOT或PS\_DASHDOTDOT *的 iStyle* 参数时，为了使短划线或点之间的间隔透明，请使用 [SetBkMode](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/nf-wingdi-setbkmode) 将模式设置为 TRANSPARENT。  如果不再需要触控笔，请调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数将其删除。  **Icm：** 创建时不执行颜色管理。 但是，当将笔选入启用了 ICM 的设备上下文时，将执行颜色管理。 示例 有关示例，请参阅 [创建彩色笔和画笔](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/creating-colored-pens-and-brushes)。   |  | | --- | | LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)  {  int wmId, wmEvent;  PAINTSTRUCT ps;  HDC hdc;  RECT clientRect;  RECT textRect;  HRGN bgRgn;  HBRUSH hBrush;  HPEN hPen;  switch (message)  {    case WM\_PAINT:  {  hdc = BeginPaint(hWnd, &ps);    // Fill the client area with a brush  GetClientRect(hWnd, &clientRect);  bgRgn = CreateRectRgnIndirect(&clientRect);  hBrush = CreateSolidBrush(RGB(200,200,200));  FillRgn(hdc, bgRgn, hBrush);    hPen = CreatePen(PS\_DOT,1,RGB(0,255,0));  SelectObject(hdc, hPen);  SetBkColor(hdc, RGB(0,0,0));  Rectangle(hdc, 10,10,200,200);    // Text caption  SetBkColor(hdc, RGB(255,255,255));  SetRect(&textRect, 10, 210, 200,200);  DrawText(hdc,TEXT("PS\_DOT"),-1,&textRect, DT\_CENTER | DT\_NOCLIP);    hPen = CreatePen(PS\_DASHDOTDOT,1,RGB(0,255,255));  SelectObject(hdc, hPen);  SetBkColor(hdc, RGB(255,0,0));  SelectObject(hdc,CreateSolidBrush(RGB(0,0,0)));  Rectangle(hdc, 210,10,400,200);    // Text caption  SetBkColor(hdc, RGB(255,200,200));  SetRect(&textRect, 210, 210, 400,200);  DrawText(hdc,TEXT("PS\_DASHDOTDOT"),-1,&textRect, DT\_CENTER | DT\_NOCLIP);    hPen = CreatePen(PS\_DASHDOT,1,RGB(255,0,0));  SelectObject(hdc, hPen);  SetBkColor(hdc, RGB(255,255,0));  SelectObject(hdc,CreateSolidBrush(RGB(100,200,255)));  Rectangle(hdc, 410,10,600,200);    // Text caption  SetBkColor(hdc, RGB(200,255,200));  SetRect(&textRect, 410, 210, 600,200);  DrawText(hdc,TEXT("PS\_DASHDOT"),-1,&textRect, DT\_CENTER | DT\_NOCLIP);    // When fnPenStyle is PS\_SOLID, nWidth may be more than 1.  // Also, if you set the width of any pen to be greater than 1,  // then it will draw a solid line, even if you try to select another style.  hPen = CreatePen(PS\_SOLID,5,RGB(255,0,0));  SelectObject(hdc, hPen);  // Setting the background color doesn't matter  // when the style is PS\_SOLID  SetBkColor(hdc, RGB(255,255,255));  SelectObject(hdc,CreateSolidBrush(RGB(200,100,50)));  Rectangle(hdc, 10,300,200,500);    // Text caption  SetBkColor(hdc, RGB(200,200,255));  SetRect(&textRect, 10, 510, 200,500);  DrawText(hdc,TEXT("PS\_SOLID"),-1,&textRect, DT\_CENTER | DT\_NOCLIP);    hPen = CreatePen(PS\_DASH,1,RGB(0,255,0));  SelectObject(hdc, hPen);  SetBkColor(hdc, RGB(0,0,0));  SelectObject(hdc,CreateSolidBrush(RGB(200,200,255)));  Rectangle(hdc, 210,300,400,500);    // Text caption  SetBkColor(hdc, RGB(255,255,200));  SetRect(&textRect, 210, 510, 400,200);  DrawText(hdc,TEXT("PS\_DASH"),-1,&textRect, DT\_CENTER | DT\_NOCLIP);  hPen = CreatePen(PS\_NULL,1,RGB(0,255,0));  SelectObject(hdc, hPen);  // Setting the background color doesn't matter  // when the style is PS\_NULL  SetBkColor(hdc, RGB(0,0,0));  SelectObject(hdc,CreateSolidBrush(RGB(255,255,255)));  Rectangle(hdc, 410,300,600,500);    // Text caption  SetBkColor(hdc, RGB(200,255,255));  SetRect(&textRect, 410, 510, 600,500);  DrawText(hdc,TEXT("PS\_NULL"),-1,&textRect, DT\_CENTER | DT\_NOCLIP);      // Clean up  DeleteObject(bgRgn);  DeleteObject(hBrush);  DeleteObject(hPen);  GetStockObject(WHITE\_BRUSH);  GetStockObject(DC\_PEN);  EndPaint(hWnd, &ps);  break;  }    case WM\_DESTROY:  PostQuitMessage(0);  break;  default:  return DefWindowProc(hWnd, message, wParam, lParam);  }  return 0;  } | |

## CreateSolidBrush函数的语法

|  |
| --- |
| **CreateSolidBrush** 函数创建具有指定纯色的逻辑画笔。 语法 C++复制  HBRUSH CreateSolidBrush(  [in] COLORREF color  ); 参数 [in] color  画笔的颜色。 若要创建 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 颜色值，请使用 [RGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-rgb) 宏。 返回值 如果函数成功，则返回值将标识逻辑画笔。  如果函数失败，则返回值为 NULL。 注解 如果不再需要 **HBRUSH** 对象，请调用 [DeleteObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-deleteobject) 函数将其删除。  实心画笔是系统用于绘制填充形状内部的位图。  应用程序通过调用 **CreateSolidBrush** 创建画笔后，可以通过调用 [SelectObject](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectobject) 函数将画笔选择到任何设备上下文中。  若要使用系统颜色画笔进行绘制，应用程序应使用 GetSysColorBrush (nIndex) 而不是 CreateSolidBrush(GetSysColor(nIndex))，因为 [GetSysColorBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-getsyscolorbrush) 返回缓存的画笔，而不是分配新的画笔。  **Icm：** 创建画笔时不执行颜色管理。 但是，将画笔选入启用了 ICM 的设备上下文时，将执行颜色管理。 示例 有关示例，请参阅 [创建彩色笔和画笔](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/creating-colored-pens-and-brushes)。 |

## RGB宏的语法

|  |
| --- |
| **RGB** 宏根据提供的参数和输出设备的颜色功能，选择红色、绿色、蓝色 (RGB) 颜色。 语法 C++复制  void RGB(  r,  g,  b  ); 参数 r  红色的强度。  g  绿色的强度。  b  蓝色的强度。 返回值 无 备注 每个参数的强度范围为 0 到 255。 如果所有三个强度均为零，则结果为黑色。 如果所有三个强度均为 255，则结果为白色。  若要提取 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref) 颜色值的红色、绿色和蓝色分量的各个值，请分别使用 [GetRValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getrvalue)、 [GetGValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getgvalue) 和 [GetBValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbvalue) 宏。  创建或检查逻辑调色板时，请使用 [RGBQUAD](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-rgbquad) 结构定义颜色值并检查各个分量值。 有关在调色板中使用颜色值的详细信息，请参阅 [PALETTEINDEX](https://learn.microsoft.com/zh-cn/previous-versions/dd162770(v=vs.85)) 和 [PALETTERGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-palettergb) 宏的说明。 另请参阅 [COLORREF](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colorref)  [颜色宏](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/color-macros)  [颜色概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/colors)  [GetBValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getbvalue)  [GetGValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getgvalue)  [GetRValue](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-getrvalue)  [PALETTEINDEX](https://learn.microsoft.com/zh-cn/previous-versions/dd162770(v=vs.85))  [PALETTERGB](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-palettergb)  [RGBQUAD](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/ns-wingdi-rgbquad) |

## SelectObject函数的语法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SelectObject** 函数在 DC) (指定设备上下文中选择对象。 新 对象替换同一类型的上一个对象。 语法 C++复制  HGDIOBJ SelectObject(  [in] HDC hdc,  [in] HGDIOBJ h  ); 参数 [in] hdc  DC 的句柄。  [in] h  要选择的对象的句柄。 指定的对象必须已使用以下函数之一创建。  展开表   |  |  | | --- | --- | | **Object** | **函数** | | **Bitmap** | [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)、 [CreateBitmapIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmapindirect)、 [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)、 [CreateDIBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibitmap)、 [CreateDIBSection](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibsection)  位图只能选择到内存 DC 中。 不能将单个位图同时选入多个 DC。 | | **Brush** | [CreateBrushIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbrushindirect)、 [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)、 [CreateDIBPatternBrushPt](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrushpt)、 [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)、 [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush)、 [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush) | | **字体** | [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)、 [CreateFontIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirecta) | | **笔** | [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen)、 [CreatePenIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpenindirect) | | **区域** | [CombineRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-combinergn)、 [CreateEllipticRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgn)、 [CreateEllipticRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgnindirect)、 [CreatePolygonRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpolygonrgn)、 [CreateRectRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgn)、 [CreateRectRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgnindirect) |  返回值 如果所选对象不是区域且函数成功，则返回值是所替换对象的句柄。 如果所选对象是区域且函数成功，则返回值是以下值之一。  展开表   |  |  | | --- | --- | | **值** | **含义** | | SIMPLEREGION | 区域由单个矩形组成。 | | COMPLEXREGION | 区域由多个矩形组成。 | | NULLREGION | 区域为空。 |     如果发生错误，并且所选对象不是区域，则返回值为 **NULL**。 否则，它将HGDI\_ERROR。 注解 此函数返回指定类型的以前选择的对象。 应用程序在用新对象完成绘图后，应始终将新对象替换为原始的默认对象。  应用程序不能一次选择一个位图进入多个 DC。  **Icm：** 如果要选择的对象是画笔或笔，则执行颜色管理。 示例 有关示例，请参阅 [设置笔或画笔颜色](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/setting-the-pen-or-brush-color)。   |  |  | | --- | --- | | **目标平台** | Windows | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll |  另请参阅 [CombineRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-combinergn)  [CreateBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmap)  [CreateBitmapIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbitmapindirect)  [CreateBrushIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createbrushindirect)  [CreateCompatibleBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createcompatiblebitmap)  [CreateDIBPatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibpatternbrush)  [CreateDIBitmap](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createdibitmap)  [CreateEllipticRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgn)  [CreateEllipticRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createellipticrgnindirect)  [CreateFont](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfonta)  [CreateFontIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createfontindirecta)  [CreateHatchBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createhatchbrush)  [CreatePatternBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpatternbrush)  [CreatePen](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpen)  [CreatePenIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpenindirect)  [CreatePolygonRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createpolygonrgn)  [CreateRectRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgn)  [CreateRectRgnIndirect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createrectrgnindirect)  [CreateSolidBrush](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-createsolidbrush)  [设备上下文函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-context-functions)  [设备上下文概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/device-contexts)  [SelectClipRgn](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectcliprgn)  [SelectPalette](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-selectpalette) |

## MoveToEx函数的语法

|  |  |  |
| --- | --- | --- |
| **MoveToEx** 函数将当前位置更新为指定点，并根据需要返回上一个位置。 语法 C++复制  BOOL MoveToEx(  [in] HDC hdc,  [in] int x,  [in] int y,  [out] LPPOINT lppt  ); 参数 [in] hdc  设备上下文的句柄。  [in] x  以逻辑单元形式指定新位置的 x 坐标（以逻辑单位为单位）。  [in] y  以逻辑单元形式指定新位置的 y 坐标（以逻辑单位为单位）。  [out] lppt  指向接收上一个当前位置的 [POINT](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point) 结构的指针。 如果此参数是 **NULL** 指针，则不返回上一个位置。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 **MoveToEx** 函数影响所有绘图函数。 示例 有关示例，请参阅 [绘图标记](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/drawing-markers)。  可以使用线条函数绘制标记。 标记是一个以点为中心的符号。 绘图应用程序使用标记来指定起点、终点和控制点。 电子表格应用程序使用标记来指定图表或图形上的兴趣点。  在以下代码示例中，应用程序定义的 Marker 函数使用 [**MoveToEx**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Wingdi/nf-wingdi-movetoex) 和 [**LineTo**](https://learn.microsoft.com/zh-cn/windows/desktop/api/Wingdi/nf-wingdi-lineto) 函数创建标记。 这些函数绘制两条相交线，长度为 20 像素，居中游标坐标。   |  | | --- | | void Marker(LONG x, LONG y, HWND hwnd)  {  HDC hdc;    hdc = GetDC(hwnd);  MoveToEx(hdc, (int) x - 10, (int) y, (LPPOINT) NULL);  LineTo(hdc, (int) x + 10, (int) y);  MoveToEx(hdc, (int) x, (int) y - 10, (LPPOINT) NULL);  LineTo(hdc, (int) x, (int) y + 10);  ReleaseDC(hwnd, hdc);  } |   当用户按下鼠标左键时，系统会将光标的坐标存储在[**WM\_LBUTTONDOWN**](https://learn.microsoft.com/zh-cn/windows/win32/inputdev/wm-lbuttondown)消息的 lParam 参数中。 以下代码演示应用程序如何获取这些坐标，确定它们是否位于其工作区内，并将其传递给 Marker 函数以绘制标记。   |  | | --- | | // Line- and arc-drawing variables    static BOOL bCollectPoints;  static POINT ptMouseDown[32];  static int index;  POINTS ptTmp;  RECT rc;    case WM\_LBUTTONDOWN:      if (bCollectPoints && index < 32)  {  // Create the region from the client area.    GetClientRect(hwnd, &rc);  hrgn = CreateRectRgn(rc.left, rc.top,  rc.right, rc.bottom);    ptTmp = MAKEPOINTS((POINTS FAR \*) lParam);  ptMouseDown[index].x = (LONG) ptTmp.x;  ptMouseDown[index].y = (LONG) ptTmp.y;    // Test for a hit in the client rectangle.    if (PtInRegion(hrgn, ptMouseDown[index].x,  ptMouseDown[index].y))  {  // If a hit occurs, record the mouse coords.    Marker(ptMouseDown[index].x, ptMouseDown[index].y,  hwnd);  index++;  }  }  break; |  另请参阅 [AngleArc](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-anglearc)  [线条和曲线函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/line-and-curve-functions)  [LineTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-lineto)  [线条和曲线概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/lines-and-curves)  [点](https://learn.microsoft.com/zh-cn/windows/win32/api/windef/ns-windef-point)  [PolyBezierTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polybezierto)  [PolylineTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polylineto) |

## LineTo函数的语法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ineTo** 函数从当前位置到指定点（但不包括）绘制一条线。 语法 C++复制  BOOL LineTo(  [in] HDC hdc,  [in] int x,  [in] int y  ); 参数 [in] hdc  设备上下文的句柄。  [in] x  指定线条终点的 x 坐标（以逻辑单位为单位）。  [in] y  指定直线终点的 y 坐标（以逻辑单位为单位）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 线条是使用当前笔绘制的，如果笔是几何笔，则使用当前画笔绘制。  如果 **LineTo** 成功，则当前位置设置为指定的终点。 示例 有关示例，请参阅 [绘图标记](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/drawing-markers)。 要求  | **目标平台** | Windows | | --- | --- | | **标头** | wingdi.h (包括 Windows.h) | | **Library** | Gdi32.lib | | **DLL** | Gdi32.dll | |  |  | |  |  | |  |  |  另请参阅 [直线和曲线函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/line-and-curve-functions)  [线条和曲线概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/lines-and-curves)  [MoveToEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-movetoex)  [折线](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polyline)  [PolylineTo](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-polylineto) |

## Rectangle函数的语法

|  |  |  |  |
| --- | --- | --- | --- |
| **Rectangle** 函数绘制一个矩形。 矩形使用当前笔轮廓，并使用当前画笔填充。 语法 C++复制  BOOL Rectangle(  [in] HDC hdc,  [in] int left,  [in] int top,  [in] int right,  [in] int bottom  ); 参数 [in] hdc  设备上下文的句柄。  [in] left  矩形左上角的 x 坐标（以逻辑坐标为单位）。  [in] top  矩形左上角的 y 坐标（以逻辑坐标为单位）。  [in] right  矩形右下角的 x 坐标（以逻辑坐标为单位）。  [in] bottom  矩形右下角的 y 坐标（以逻辑坐标为单位）。 返回值 如果该函数成功，则返回值为非零值。  如果函数失败，则返回值为零。 注解 **矩形**既不使用当前位置，也不更新当前位置。  绘制的矩形不包括下边缘和右边缘。  如果使用PS\_NULL笔，矩形的尺寸高度减少 1 像素，宽度减少 1 像素。 示例 有关示例，请参阅 [使用填充形状](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/using-filled-shapes)。  用户通过以下方式绘制填充形状：从菜单中选择特定形状，将光标定位在形状的左上角 (，或者在椭圆) 的情况下将形状的边框定位，然后拖动鼠标，直到获得所需的尺寸。  下图显示了使用本部分中的示例代码绘制的三个填充形状。  显示一个圆圈和两个矩形的屏幕截图 -- 一个带有方角，一个带有圆角  若要使用户能够绘制填充的形状，请在应用程序中包含以下弹出菜单   |  | | --- | | POPUP "Filled &Shapes"  {  MENUITEM "&Ellipse", IDM\_ELLIPSE  MENUITEM "&Rectangle", IDM\_RECTANGLE  MENUITEM "R&oundRect", IDM\_ROUNDRECT  } |   菜单模板中的菜单项值是必须在应用程序的头文件中按如下所示定义的常量   |  | | --- | | #define IDM\_ELLIPSE 1100  #define IDM\_RECTANGLE 1200  #define IDM\_ROUNDRECT 1300 |   最后，在应用程序中包含以下窗口过程。   |  | | --- | | BOOL CALLBACK MainWndProc(HWND hwnd, UINT uMsg, WPARAM wParam,  LPARAM lParam)  {  HDC hdc; // handle to device context (DC)  PAINTSTRUCT ps; // paint data for Begin/EndPaint  POINT ptClientUL; // client area upper left corner  POINT ptClientLR; // client area lower right corner  static HDC hdcCompat; // handle to DC for bitmap  static POINT pt; // x- and y-coordinates of cursor  static RECT rcTarget; // rect to receive filled shape  static RECT rcClient; // client area rectangle  static BOOL fSizeEllipse; // TRUE if ellipse is sized  static BOOL fDrawEllipse; // TRUE if ellipse is drawn  static BOOL fDrawRectangle; // TRUE if rectangle is drawn  static BOOL fSizeRectangle; // TRUE if rectangle is sized  static BOOL fSizeRoundRect; // TRUE if rounded rect is sized  static BOOL fDrawRoundRect; // TRUE if rounded rect is drawn  static int nEllipseWidth; // width for round corners  static int nEllipseHeight; // height for round corners    switch (uMsg)  {  case WM\_COMMAND:  switch (wParam)  {  // Set the appropriate flag to indicate which  // filled shape the user is drawing.  case IDM\_ELLIPSE:  fSizeEllipse = TRUE;  break;    case IDM\_RECTANGLE:  fSizeRectangle = TRUE;  break;    case IDM\_ROUNDRECT:  fSizeRoundRect = TRUE;  break;    default:  return DefWindowProc(hwnd, uMsg, wParam,  lParam);  }  break;      case WM\_CREATE:  nEllipseWidth = 20;  nEllipseHeight = 20;    return 0;    case WM\_PAINT:      BeginPaint(hwnd, &ps);    // Because the default brush is white, select  // a different brush into the device context  // to demonstrate the painting of filled shapes.    SelectObject(ps.hdc, GetStockObject(GRAY\_BRUSH));    // If one of the filled shape "draw" flags is TRUE,  // draw the corresponding shape.    if (fDrawEllipse)  {  Ellipse(ps.hdc, rcTarget.left, rcTarget.top,  rcTarget.right, rcTarget.bottom);  fDrawEllipse = FALSE;  rcTarget.left = rcTarget.right = 0;  rcTarget.top = rcTarget.bottom = 0;  }    if (fDrawRectangle)  {  Rectangle(ps.hdc, rcTarget.left, rcTarget.top,  rcTarget.right, rcTarget.bottom);  fDrawRectangle = FALSE;  rcTarget.left = rcTarget.right = 0;  rcTarget.top = rcTarget.bottom = 0;  }    if (fDrawRoundRect)  {  RoundRect(ps.hdc, rcTarget.left, rcTarget.top,  rcTarget.right, rcTarget.bottom,  nEllipseWidth, nEllipseHeight);  fDrawRectangle = FALSE;  rcTarget.left = rcTarget.right = 0;  rcTarget.top = rcTarget.bottom = 0;  }    EndPaint(hwnd, &ps);  break;    case WM\_SIZE:    // Convert the client coordinates of the client area  // rectangle to screen coordinates and save them in a  // rectangle. The rectangle is passed to the ClipCursor  // function during WM\_LBUTTONDOWN processing.    GetClientRect(hwnd, &rcClient);  ptClientUL.x = rcClient.left;  ptClientUL.y = rcClient.top;  ptClientLR.x = rcClient.right;  ptClientLR.y = rcClient.bottom;  ClientToScreen(hwnd, &ptClientUL);  ClientToScreen(hwnd, &ptClientLR);  SetRect(&rcClient, ptClientUL.x, ptClientUL.y,  ptClientLR.x, ptClientLR.y);  return 0;    case WM\_LBUTTONDOWN:    // Restrict the cursor to the client area.  // This ensures that the window receives a matching  // WM\_LBUTTONUP message.    ClipCursor(&rcClient);    // Save the coordinates of the cursor.    pt.x = (LONG) LOWORD(lParam);  pt.y = (LONG) HIWORD(lParam);    // If the user chooses one of the filled shapes,  // set the appropriate flag to indicate that the  // shape is being sized.    if (fDrawEllipse)  fSizeEllipse = TRUE;    return 0;    case WM\_MOUSEMOVE:    // If one of the "size" flags is set, draw  // the target rectangle as the user drags  // the mouse.    if ((wParam && MK\_LBUTTON)  && (fSizeEllipse || fSizeRectangle  || fSizeRoundRect))  {    // Set the mixing mode so that the pen color is the  // inverse of the background color. The previous  // rectangle can then be erased by drawing  // another rectangle on top of it.    hdc = GetDC(hwnd);  SetROP2(hdc, R2\_NOTXORPEN);    // If a previous target rectangle exists, erase  // it by drawing another rectangle on top.    if (!IsRectEmpty(&rcTarget))  {  Rectangle(hdc, rcTarget.left, rcTarget.top,  rcTarget.right, rcTarget.bottom);  }  // Save the coordinates of the target rectangle.  // Avoid invalid rectangles by ensuring that the  // value of the left coordinate is greater than  // that of the right, and that the value of the  // bottom coordinate is greater than that of  // the top.    if ((pt.x < (LONG) LOWORD(lParam)) &&  (pt.y > (LONG) HIWORD(lParam)))  {  SetRect(&rcTarget, pt.x, HIWORD(lParam),  LOWORD(lParam), pt.y);  }    else if ((pt.x > (LONG) LOWORD(lParam)) &&  (pt.y > (LONG) HIWORD(lParam)))  {  SetRect(&rcTarget, LOWORD(lParam),  HIWORD(lParam), pt.x, pt.y);  }    else if ((pt.x > (LONG) LOWORD(lParam)) &&  (pt.y < (LONG) HIWORD(lParam)))  {  SetRect(&rcTarget, LOWORD(lParam), pt.y,  pt.x, HIWORD(lParam));  }  else  {  SetRect(&rcTarget, pt.x, pt.y, LOWORD(lParam),  HIWORD(lParam));  }    // Draw the new target rectangle.    Rectangle(hdc, rcTarget.left, rcTarget.top,  rcTarget.right, rcTarget.bottom);  ReleaseDC(hwnd, hdc);  }  return 0;    case WM\_LBUTTONUP:    // If one of the "size" flags is TRUE, reset it to FALSE,  // and then set the corresponding "draw" flag. Invalidate  // the appropriate rectangle and issue a WM\_PAINT message.    if (fSizeEllipse)  {  fSizeEllipse = FALSE;  fDrawEllipse = TRUE;  }    if (fSizeRectangle)  {  fSizeRectangle = FALSE;  fDrawRectangle = TRUE;  }    if (fSizeRoundRect)  {  fSizeRoundRect = FALSE;  fDrawRoundRect = TRUE;  }    if (fDrawEllipse || fDrawRectangle || fDrawRoundRect)  {  InvalidateRect(hwnd, &rcTarget, TRUE);  UpdateWindow(hwnd);  }    // Release the cursor.    ClipCursor((LPRECT) NULL);  return 0;    case WM\_DESTROY:    // Destroy the background brush, compatible bitmap,  // and bitmap.    DeleteDC(hdcCompat);  PostQuitMessage(0);  break;    default:  return DefWindowProc(hwnd, uMsg, wParam, lParam);  }  return (LRESULT) NULL;  } |  另请参阅 [填充形状函数](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shape-functions)  [填充形状概述](https://learn.microsoft.com/zh-cn/windows/desktop/gdi/filled-shapes)  [RoundRect](https://learn.microsoft.com/zh-cn/windows/desktop/api/wingdi/nf-wingdi-roundrect) |

# 演练

## 演练准备：新建一个文件夹，取名Lesson33-gdi-demos

## 演练项目1.

## 1.新建一个基于对话框的mfc程序，取名gdidemo1，把取消按钮删除，然后把确定按钮的文本改为退出，双击它进入点击事件，添加如下的代码：

|  |
| --- |
|  |

## 2.给对话框添加一个按钮，把文本改为：对话框上面绘图 双击按钮进入点击事件处理函数

|  |
| --- |
|  |
|  |

## 3.然后我们添加一个按钮，文本为：桌面绘图，点击进入事件处理函数添加代码

|  |
| --- |
|  |
|  |

### 运行程序，点击桌面绘图按钮效果如下

|  |
| --- |
|  |

### 上面是用默认的画笔和画刷，我们创建自己的画笔和画刷等等

## 4.在我们修改一下GdiOut函数在里面创建一个新画笔和一个新的画刷并且选进随便描述表，绘制完毕后把恢复为原来的画笔和画刷并且删除新创建的画笔和画刷，然后2个按钮的代码不变

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## 演练项目2.在命令行中使用GDI函数

## 1.新建一个cpp常规空项目，取名：gdidemo-cmd，然后新建一个cpp源文件，我们gdicmd.cpp

|  |
| --- |
|  |

## 2.然后在gdicmd.cpp里面添加如下内容

|  |
| --- |
| #include<Windows.h>  int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE prev,LPTSTR cmdline,int nshow)  {  HDC hdc = GetDC(NULL);  HPEN pen,penOld;  HBRUSH brush,brushOld;  pen = CreatePen(PS\_SOLID,2,RGB(255,0,255));  brush = CreateSolidBrush(RGB(0,255,255));  penOld = (HPEN)SelectObject(hdc,pen);  brushOld = (HBRUSH)SelectObject(hdc,brush);  LineTo(hdc,500,500);  Rectangle(hdc,200,200,500,500);  SelectObject(hdc,penOld);  DeleteObject(pen);  SelectObject(hdc,brush);  DeleteObject(brush);  ReleaseDC(NULL,hdc);  return 0;  } |

### 运行程序效果如下

|  |
| --- |
|  |

### 可见，在win32平台的API中凡是不需要窗口句柄或者窗口句柄可以传递NULL的API都可以在命令行中使用，如文件操作，目录操作，绘图，获取电脑信息，操作注册表等等。

# 这一节的学习到此为止，相关的项目的完整代码如下

## gdidemo1.cpp

|  |
| --- |
| // gdidemo1.cpp : 定义应用程序的类行为。  //  #include "stdafx.h"  #include "gdidemo1.h"  #include "gdidemo1Dlg.h"  #ifdef \_DEBUG  #define new DEBUG\_NEW  #endif  // Cgdidemo1App  BEGIN\_MESSAGE\_MAP(Cgdidemo1App, CWinApp)  ON\_COMMAND(ID\_HELP, &CWinApp::OnHelp)  END\_MESSAGE\_MAP()  // Cgdidemo1App 构造  Cgdidemo1App::Cgdidemo1App()  {  // 支持重新启动管理器  m\_dwRestartManagerSupportFlags = AFX\_RESTART\_MANAGER\_SUPPORT\_RESTART;  // TODO: 在此处添加构造代码，  // 将所有重要的初始化放置在 InitInstance 中  }  // 唯一的一个 Cgdidemo1App 对象  Cgdidemo1App theApp;  // Cgdidemo1App 初始化  BOOL Cgdidemo1App::InitInstance()  {  // 如果一个运行在 Windows XP 上的应用程序清单指定要  // 使用 ComCtl32.dll 版本 6 或更高版本来启用可视化方式，  //则需要 InitCommonControlsEx()。否则，将无法创建窗口。  INITCOMMONCONTROLSEX InitCtrls;  InitCtrls.dwSize = sizeof(InitCtrls);  // 将它设置为包括所有要在应用程序中使用的  // 公共控件类。  InitCtrls.dwICC = ICC\_WIN95\_CLASSES;  InitCommonControlsEx(&InitCtrls);  CWinApp::InitInstance();  AfxEnableControlContainer();  // 创建 shell 管理器，以防对话框包含  // 任何 shell 树视图控件或 shell 列表视图控件。  CShellManager \*pShellManager = new CShellManager;  // 标准初始化  // 如果未使用这些功能并希望减小  // 最终可执行文件的大小，则应移除下列  // 不需要的特定初始化例程  // 更改用于存储设置的注册表项  // TODO: 应适当修改该字符串，  // 例如修改为公司或组织名  SetRegistryKey(\_T("应用程序向导生成的本地应用程序"));  Cgdidemo1Dlg dlg;  m\_pMainWnd = &dlg;  INT\_PTR nResponse = dlg.DoModal();  if (nResponse == IDOK)  {  // TODO: 在此放置处理何时用  // “确定”来关闭对话框的代码  }  else if (nResponse == IDCANCEL)  {  // TODO: 在此放置处理何时用  // “取消”来关闭对话框的代码  }  // 删除上面创建的 shell 管理器。  if (pShellManager != NULL)  {  delete pShellManager;  }  // 由于对话框已关闭，所以将返回 FALSE 以便退出应用程序，  // 而不是启动应用程序的消息泵。  return FALSE;  } |

## gdidemo1Dlg.cpp

|  |
| --- |
| **// gdidemo1Dlg.cpp : 实现文件**  **//**  **#include "stdafx.h"**  **#include "gdidemo1.h"**  **#include "gdidemo1Dlg.h"**  **#include "afxdialogex.h"**  **#ifdef \_DEBUG**  **#define new DEBUG\_NEW**  **#endif**  **// 用于应用程序“关于”菜单项的 CAboutDlg 对话框**  **class CAboutDlg : public CDialogEx**  **{**  **public:**  **CAboutDlg();**  **// 对话框数据**  **enum { IDD = IDD\_ABOUTBOX };**  **protected:**  **virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持**  **// 实现**  **protected:**  **DECLARE\_MESSAGE\_MAP()**  **};**  **CAboutDlg::CAboutDlg() : CDialogEx(CAboutDlg::IDD)**  **{**  **}**  **void CAboutDlg::DoDataExchange(CDataExchange\* pDX)**  **{**  **CDialogEx::DoDataExchange(pDX);**  **}**  **BEGIN\_MESSAGE\_MAP(CAboutDlg, CDialogEx)**  **END\_MESSAGE\_MAP()**  **// Cgdidemo1Dlg 对话框**  **Cgdidemo1Dlg::Cgdidemo1Dlg(CWnd\* pParent /\*=NULL\*/)**  **: CDialogEx(Cgdidemo1Dlg::IDD, pParent)**  **{**  **m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);**  **}**  **void Cgdidemo1Dlg::DoDataExchange(CDataExchange\* pDX)**  **{**  **CDialogEx::DoDataExchange(pDX);**  **}**  **BEGIN\_MESSAGE\_MAP(Cgdidemo1Dlg, CDialogEx)**  **ON\_WM\_SYSCOMMAND()**  **ON\_WM\_PAINT()**  **ON\_WM\_QUERYDRAGICON()**  **ON\_BN\_CLICKED(IDOK, &Cgdidemo1Dlg::OnBnClickedOk)**  **ON\_BN\_CLICKED(IDC\_BUTTON1, &Cgdidemo1Dlg::OnBnClickedButton1)**  **ON\_BN\_CLICKED(IDC\_BUTTON2, &Cgdidemo1Dlg::OnBnClickedButton2)**  **END\_MESSAGE\_MAP()**  **// Cgdidemo1Dlg 消息处理程序**  **BOOL Cgdidemo1Dlg::OnInitDialog()**  **{**  **CDialogEx::OnInitDialog();**  **// 将“关于...”菜单项添加到系统菜单中。**  **// IDM\_ABOUTBOX 必须在系统命令范围内。**  **ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);**  **ASSERT(IDM\_ABOUTBOX < 0xF000);**  **CMenu\* pSysMenu = GetSystemMenu(FALSE);**  **if (pSysMenu != NULL)**  **{**  **BOOL bNameValid;**  **CString strAboutMenu;**  **bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);**  **ASSERT(bNameValid);**  **if (!strAboutMenu.IsEmpty())**  **{**  **pSysMenu->AppendMenu(MF\_SEPARATOR);**  **pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);**  **}**  **}**  **// 设置此对话框的图标。当应用程序主窗口不是对话框时，框架将自动**  **// 执行此操作**  **SetIcon(m\_hIcon, TRUE); // 设置大图标**  **SetIcon(m\_hIcon, FALSE); // 设置小图标**  **// TODO: 在此添加额外的初始化代码**  **return TRUE; // 除非将焦点设置到控件，否则返回 TRUE**  **}**  **void Cgdidemo1Dlg::OnSysCommand(UINT nID, LPARAM lParam)**  **{**  **if ((nID & 0xFFF0) == IDM\_ABOUTBOX)**  **{**  **CAboutDlg dlgAbout;**  **dlgAbout.DoModal();**  **}**  **else**  **{**  **CDialogEx::OnSysCommand(nID, lParam);**  **}**  **}**  **// 如果向对话框添加最小化按钮，则需要下面的代码**  **// 来绘制该图标。对于使用文档/视图模型的 MFC 应用程序，**  **// 这将由框架自动完成。**  **void Cgdidemo1Dlg::OnPaint()**  **{**  **if (IsIconic())**  **{**  **CPaintDC dc(this); // 用于绘制的设备上下文**  **SendMessage(WM\_ICONERASEBKGND, reinterpret\_cast<WPARAM>(dc.GetSafeHdc()), 0);**  **// 使图标在工作区矩形中居中**  **int cxIcon = GetSystemMetrics(SM\_CXICON);**  **int cyIcon = GetSystemMetrics(SM\_CYICON);**  **CRect rect;**  **GetClientRect(&rect);**  **int x = (rect.Width() - cxIcon + 1) / 2;**  **int y = (rect.Height() - cyIcon + 1) / 2;**  **// 绘制图标**  **dc.DrawIcon(x, y, m\_hIcon);**  **}**  **else**  **{**  **CDialogEx::OnPaint();**  **}**  **}**  **//当用户拖动最小化窗口时系统调用此函数取得光标**  **//显示。**  **HCURSOR Cgdidemo1Dlg::OnQueryDragIcon()**  **{**  **return static\_cast<HCURSOR>(m\_hIcon);**  **}**  **void Cgdidemo1Dlg::OnBnClickedOk()**  **{**  **// TODO: 在此添加控件通知处理程序代码**  **//防止用户按下回车键盘后退出程序，注意需要使用平台sdk版本的消息框而且需要2个按钮**  **if(IDOK == ::MessageBox(this->m\_hWnd,\_T("退出程序?"),\_T("退出确认"),MB\_OKCANCEL))**  **{**  **CDialogEx::OnOK();**  **}**  **}**  **//自定义函数**  **void GdiOut(HDC hdc)**  **{**  **HPEN pen,OldPen;**  **HBRUSH brush,OldBrush;**  **pen = CreatePen(PS\_SOLID,2,RGB(255,0,255));**  **brush = CreateSolidBrush(RGB(0,255,255));**  **OldPen = (HPEN)SelectObject(hdc,pen);**  **OldBrush = (HBRUSH)SelectObject(hdc,brush);**  **::LineTo(hdc,400,400);**  **::Rectangle(hdc,200,200,400,400);**  **DeleteObject(SelectObject(hdc,OldPen));**  **DeleteObject(SelectObject(hdc,OldBrush));**  **}**  **void Cgdidemo1Dlg::OnBnClickedButton1()**  **{**  **// TODO: 在此添加控件通知处理程序代码**    **HWND hwnd =GetSafeHwnd();**  **HDC hdc = ::GetDC(hwnd);**  **GdiOut(hdc);**    **::ReleaseDC(hwnd,hdc);**  **}**  **void Cgdidemo1Dlg::OnBnClickedButton2()**  **{**  **// TODO: 在此添加控件通知处理程序代码**  **HDC hdc = ::GetDC(NULL);//在GetDC中传递NULL，就可以获取桌面DC**  **GdiOut(hdc);**  **::ReleaseDC(NULL,hdc);**  **}** |

## gdidcmd.cpp

|  |
| --- |
| #include<Windows.h>  int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE prev,LPTSTR cmdline,int nshow)  {  HDC hdc = GetDC(NULL);  HPEN pen,penOld;  HBRUSH brush,brushOld;  pen = CreatePen(PS\_SOLID,2,RGB(255,0,255));  brush = CreateSolidBrush(RGB(0,255,255));  penOld = (HPEN)SelectObject(hdc,pen);  brushOld = (HBRUSH)SelectObject(hdc,brush);  LineTo(hdc,500,500);  Rectangle(hdc,200,200,500,500);  SelectObject(hdc,penOld);  DeleteObject(pen);  SelectObject(hdc,brush);  DeleteObject(brush);  ReleaseDC(NULL,hdc);  return 0;  } |