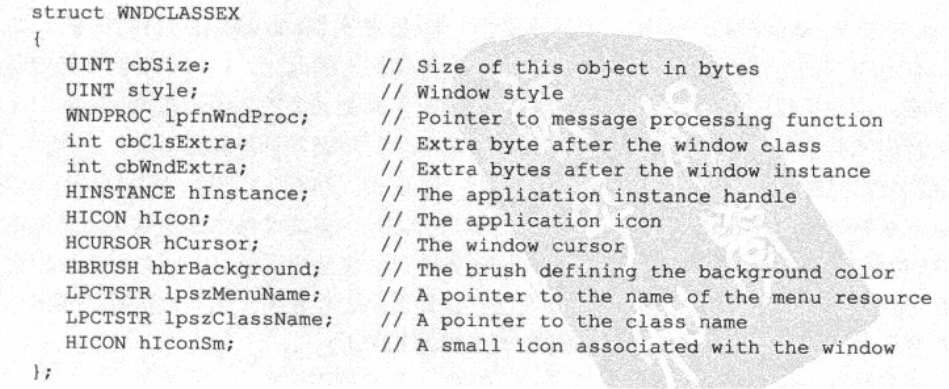
# Windows API程序快速入门

## 基本流程

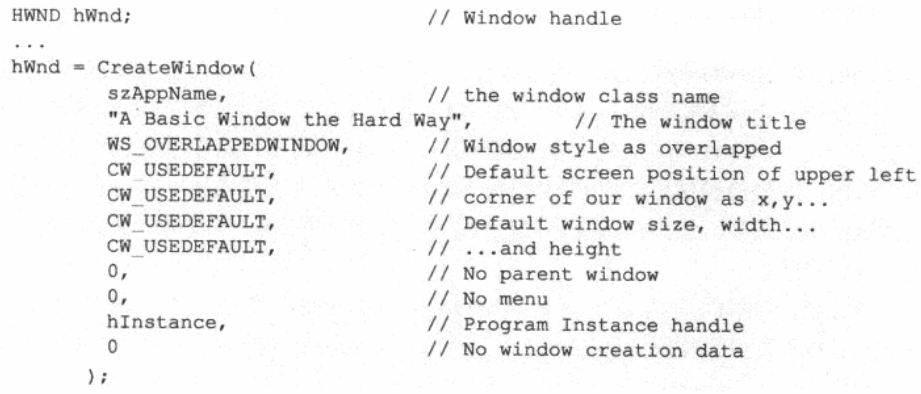
### 注册窗口RegisterClassExW

需要指定的WNDCLASSEX内容：



然后执行RegisterClassExW(&wcex);//wce为此结构类型的实际对象

### 构建窗口CreateWindow（典型用法）：



### 移动窗口MoveWindow

MoveWindow(

\_In\_ HWND hWnd,

\_In\_ int X,

\_In\_ int Y,

\_In\_ int nWidth,

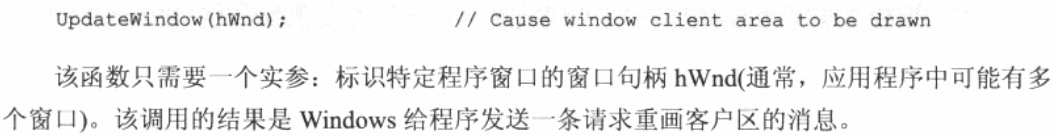
\_In\_ int nHeight,

\_In\_ BOOL bRepaint);

### 显示窗口ShowWindow：

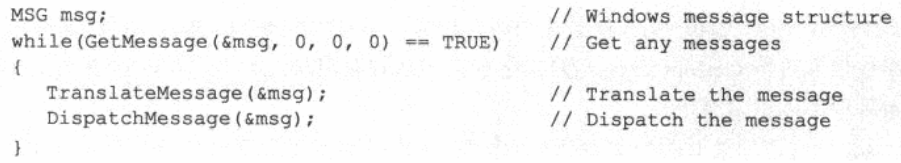


### 更新窗口UpdateWindow:

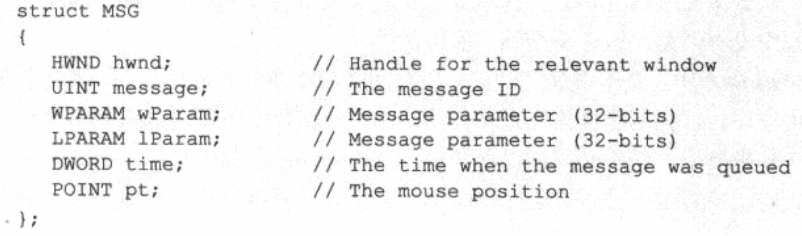


### 消息获取GetMessage

基本处理方式：



其中：



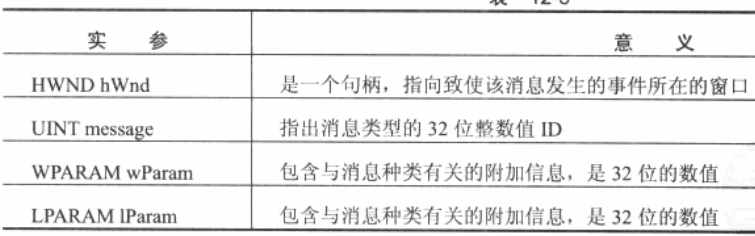
1. GetMessage（）：取消息。如果无消息，会将控制权交给其他应用程序，相当于挂起。返回时，若是VM\_QUIT消息，返回FALSE；若函数发生错误，返回-1；其余情况返回TRUE。
2. TranslateMessage（）：请求windows为与键盘有关的消息做一些转换。
3. DispatchMessage（）：调用消息处理函数来处理该函数。在处理完成之前，该函数不会返回。

### 消息处理WndProc：

消息处理函数由“注册窗口”中的WNDCLASSEX结构体的lpfnWndProc成员决定。

默认为WndProc函数。以下以WndProc为例说明。

#### WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)



#### DefWindowProc(hWnd, message, wParam, lParam)

处理default情况的函数。实际上是将消息回传给windows。

#### BeginPaint



典型用法：

case WM\_PAINT:

{

PAINTSTRUCT ps;

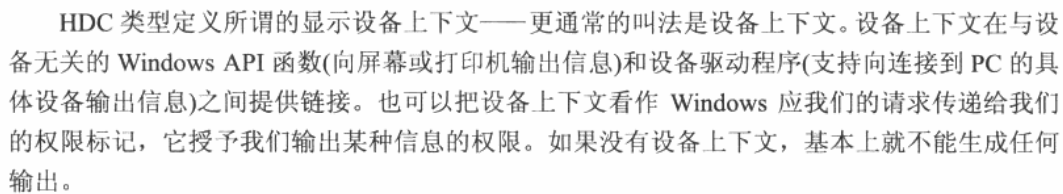
HDC hdc = BeginPaint(hWnd, &ps);

// TODO: 在此处添加使用 hdc 的任何绘图代码...

EndPaint(hWnd, &ps);

}

break;



实际上HDC相当于一个权限证书

#### GetClinentRect:获取窗口坐标

典型用法



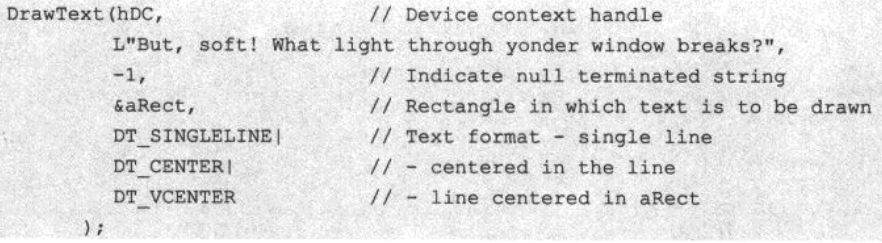
#### SetBkMode设置背景色

eg.



#### DrawText 输出文本

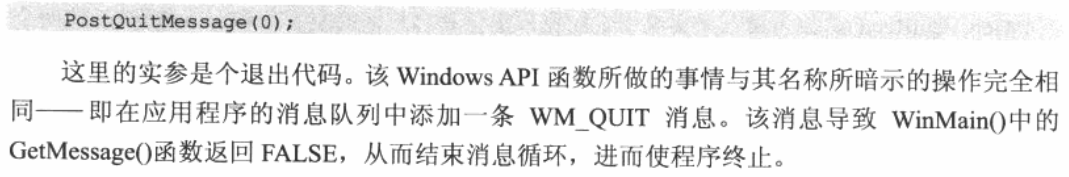
Eg.



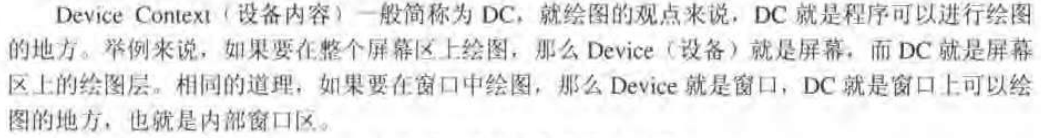
#### PostQuitMessage

关闭窗口后，系统会向程序发出WM\_DESTROY消息。此函数需要一般添加在WM\_DESTROY消息之后，作用是再让系统发一条WM\_QUIT消息，以便程序退出循环。

Eg.

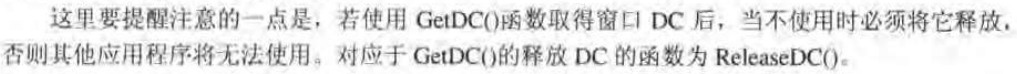


## 关于Device Context

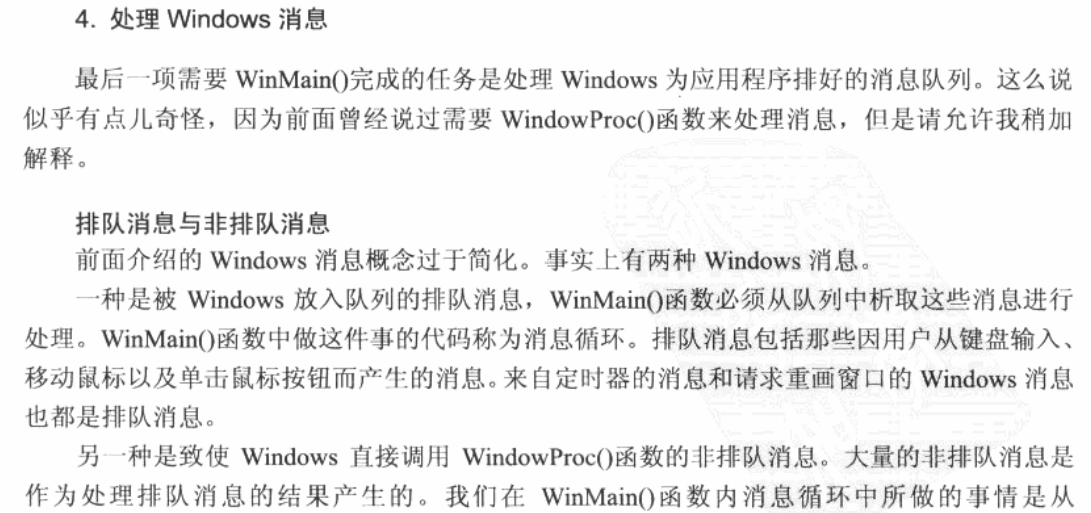


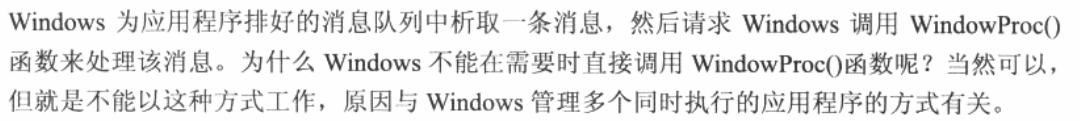
### HDC GetDC(HWND hWnd)：获取DC

### Int ReleaseDC(HWND hWnd,HDC hdc)：释放DC



## 关于消息





### VM\_MOUSEMOVE:鼠标在窗口移动时产生此消息。

Lparam中包含鼠标状态的相关信息

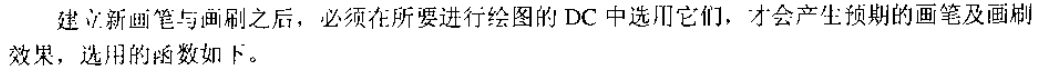
# 游戏画面绘图

## 准备画图工具

### HPEN CreatePen(int样式,int 宽度,COLORREF顔色); //建立画笔

### HBRUSH CreateHatchBrush(int样式,COLORREF颜色); //建立阴影画刷

### HBRUSH CreateSolidBrush(COLORREF 颜色)； //建立单色画刷



### HCDIOBJ SelectObject(HDC hdc, HGDIOBJ GDI对象); //选用GDI对象



### BOOL DeleteObject(HGDIOBJ GDI对象)； //删除GDI对象

## 各种图形

### 画线（1）MoveToEx()//移到线条起点

＊注意：是相对窗口（０,０）。

### 画线（2）LineTo( HDC hdc, int x, int y); 画线终点

＊注意：也是相对窗口（０,０）

### 多边形Polygon(),polyline(),polylineto(),polypolygon(),polypolyline()

### 矩形 rectangle()

### 椭圆 ellipse()

### 圆角矩形 roundrect()

### 扇形弓形 pie()

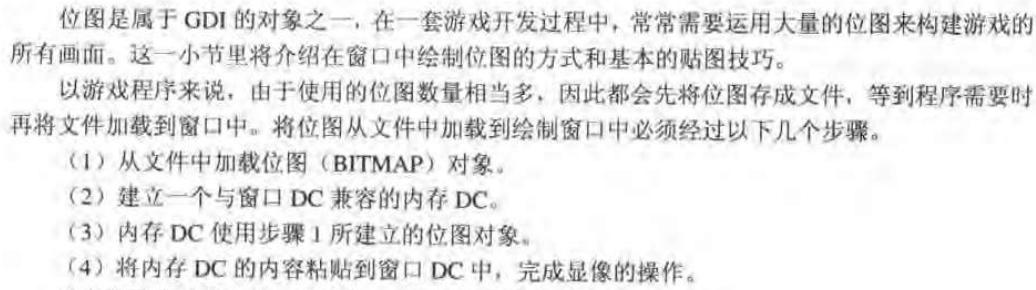
## 文字

### 设置文字颜色 COLORREF SetTextColor( HDC hdc, COLORREF color)

Eg. 

### 输出固定字符串BOOL TextOut(HDC hdc, int x, int y, LPCWSTR lpString, int c)

## 位图



### 加载位图 HANDLE LoadImage(

\_In\_opt\_ HINSTANCE hInst,

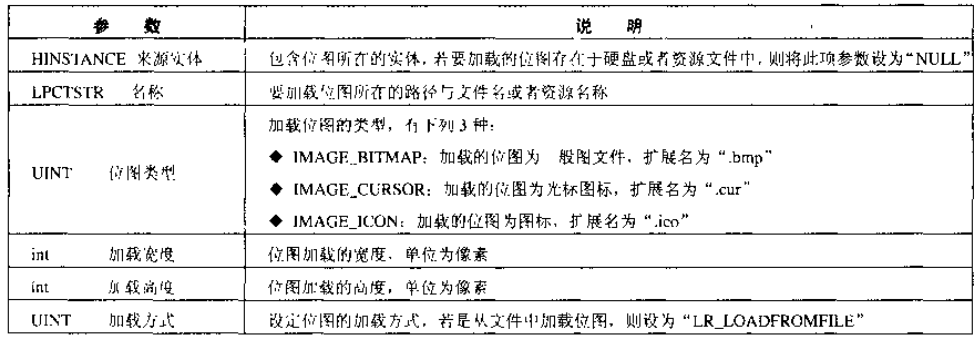
\_In\_ LPCWSTR name,

\_In\_ UINT type,

\_In\_ int cx,

\_In\_ int cy,

\_In\_ UINT fuLoad)



### 建立与窗口兼容的位图HBITMAP CreateCompatibleBitmap(

\_In\_ HDC hdc, \_In\_ int cx, \_In\_ int cy);

\*利用此建立的位图不同于loadimage建立的位图。前者随窗口变化而变化，后者并不变。

### 建立位图内存DC：HDC CreateCompatibleDC( \_In\_opt\_ HDC hdc);

### 释放内存DC： BOOL DeleteDC( \_In\_ HDC hdc);

### 贴图BOOL BitBlt(HDC hdc, int x, int y, int cx, int cy, HDC hdcSrc int x1, int y1, DWORD rop);

Rop:



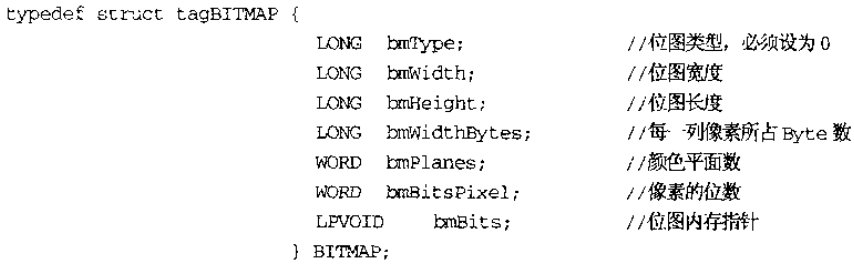
### 获取位图信息 int GetObject( HANDLE h, int c, LPVOID pv)

第三个变量是位图信息指针。常用法：

BITMAP a；

GetObject(handle, sizeof(BITMAP), &a);

其中，BITMAP是个结构体



### 获取位图的二进制值 GetBitmapBits(

\_In\_ HBITMAP hbit,

\_In\_ LONG cb,

\_Out\_writes\_bytes\_(cb) LPVOID lpvBits

);

### 将二进制值写入位图中LONG SetBitmapBits(

\_In\_ HBITMAP hbm,

\_In\_ DWORD cb,

\_In\_reads\_bytes\_(cb) CONST VOID \*pvBits);

# 其它函数

## LOWORD / HIWORD：提取32位数据的低/高16位。

## Sprintf(..)按照printf的格式将字符串存进指定的变量中。

Eg. Char str[20];

Char kc[20]=”kangchi”;

Sprintf(str,”hello word,%s\n”,kc);

## MessageBox：弹出消息窗口

# 需要注意

## Unicode编程

* 在“项目”-“属性”-“常规”-“字符集”中选择



Eg. \_T(“hello world!”)

* 许多操作字符的ASCII函数有其对应的UNICODE函数。需要自行查找帮助。例如spintf函数对应的通用型函数形式：

