使用Windows API进行GDI窗口绘图

1.概述
在Windows上绘图方式，跟美术大师绘图差不多。美术绘画，首先要具备以下工具：画板，画布，画笔，画刷。同样，Windows上也有相关的概念。绘图设备DeviceContext(DC)，位图Bitmap，画笔Pen，画刷brush。他们一一对应。

2.画板
在Windows中被称作设备上下文（Device Context，DC），我习惯称之为绘图设备。但是Windows的“画板”与美术大师手中的“画板”不一样，Windows中的“画板”实质上是一个工具的集合体，将画布、画笔、画刷、绘画方式全部综合管理起来，然后，所有的绘画操作都将在这上面进行。

1）系统绘图设备。系统有默认的绘图，我们可以直接使用它进行一系列的绘图操作。

获取系统绘图设备：

```
1. HDC GetDC(
2. HWND hWnd // 窗口句柄。即指定获得那个窗口的绘图设备。
3. );
```

如：

```
1. HDC hDC = GetDC(g_hWnd);
2. Rectangle(hDC,0,0,100,100);//绘制矩形
3. ....
```

使用完毕后要归还给窗口：

```
1. int ReleaseDC(
2. HWND hWnd, // handle to window
3. HDC hDC // handle to DC
4. );
```

2）创建画板。通常我们不直接使用系统画板，因为系统画板直接和显示器关联，在上面一边画，就会一边显示到窗口，由于在绘制的时候，中间有时间差（我们可能经过若干次绘制，才绘制完一个地图），而游戏程序需要反复的擦除、重绘，所以常常会造成画面剧烈闪烁。为了解决这个问题，我们会创建一个辅助画板，在幕后绘制完毕后，一次性的将它显示到屏幕上，因为显示的时间非常快，所以就看不到闪烁。

创建辅助绘图设备：

```
1. HDC CreateCompatibleDC(
2. HDC hdc // 指向一个已经存在的DC，如果该值传入0，则系统会创建一个存贮DC。
3. );
```

存贮DC（辅助绘图设备），在创建完毕后，它的大小只有一个像素，我们没办法直接往他上面直接绘图，我们还需要往它上面贴一张画布。

使用完毕后记得要释放资源：

```
1. BOOL DeleteDC(
2. HDC hdc // handle to DC
3. );
```

例：

```
1. HDC memDC = CreateCompatibleDC(0); //创建辅助绘图设备
2. SelectObject(memDC,hBitmap); //将画布贴到绘图设备上
3. Rectangle(memDC,x1,y1,x2,y2); //绘制矩形
4. HDC hDC = GetDC(g_hWnd); //获得系统绘图设备
5. copy memDC to hDC //复制到系统设备上显示
6. ReleaseDC(g_hWnd,hDC); //归还系统绘图设备
7. DeleteDC(memDC); //释放辅助绘图设备
```

3.画布
画布其实就是位图。位图的作用非常强大，在此我只介绍它的普通用法——充当画布。其他作用，后面再介绍。

创建掩码位图（画布）：

```
1. HBITMAP CreateCompatibleBitmap(
2. HDC hdc, //指向绘图设备，最好是系统的绘图设备
3. int nWidth, // 位图宽度
4. int nHeight // 位图高度
5. );
```

释放资源：

```
1. BOOL DeleteObject(
2. HGDIOBJ hObject // handle to graphic object
3. );
```

4.画笔
画笔就不用解释了，Windows的画笔和美术大师的差不多。

创建画笔：

```
1. HPEN CreatePen(
2. int fnPenStyle, // 画笔风格。直线：PS_SOLID，点线：PS_DOT
3. int nWidth, // 画笔宽度。决定了画出来的线条粗细
4. COLORREF crColor // 画笔颜色
5. );
```

释放资源：

```
1. BOOL DeleteObject(
2. HGDIOBJ hObject // handle to graphic object
3. );
```

颜色：可由红绿蓝三种颜色按程度混合起来构成。

如COLORREF cr = RGB(255,0,255)，此时cr是粉红色（红色+蓝色）。COLORREF实际上是无符号长整型的别名，RGB将红绿蓝三个数值整合成一个无符号长整型数。

```
1. COLORREF RGB(
2. BYTE byRed, //红色分量0~255，值越大表面程度越深。
3. BYTE byGreen, // 绿色分量0~255
4. BYTE byBlue // 蓝色分量0~255
5. );
```

5.画刷

创建画刷：

```
1. HBRUSH CreateSolidBrush(
2. COLORREF crColor // brush color value
3. );
```

释放资源：

```
1. BOOL DeleteObject(
2. HGDIOBJ hObject // handle to graphic object
3. );
```

6.绘画
画布、画笔、画刷创建好之后，要是分别使用API，将它们选入绘图设备。当然，绘图设备，有自己的默认画刷和画笔。

```
1. HGDIOBJ SelectObject(
2.   HDC hdc,          // 直线绘图设备
3.   HGDIOBJ hgdiobj   // 指向GDI对象。就是画刷、画笔、位图等。
4. );
```

返回值是旧有的相关内容。

一些常用绘图API：
1）绘制矩形。

```
1. BOOL Rectangle(
2.   HDC hdc,         // 绘图设备
3.   int nLeftRect,    // 矩形区域左上角x坐标
4.   int nTopRect,    // 矩形区域左上角y坐标
5.   int nRightRect,  //矩形区域右下角x坐标
6.   int nBottomRect  // 矩形区域右下角y坐标
7. );
```

2）填充区域。**用画刷来填充区域，经常用来擦除整个窗口。**

```
1. int FillRect(
2. HDC hDC, // 绘图设备
```

```
3. CONST RECT *lprc, // 矩形区域
4. HBRUSH hbr // 填充画刷
5. );
```

区域结构：

```
1. typedef struct _RECT {
2. LONG left; //矩形左边（左上角x坐标）
3. LONG top; //矩形顶边（左上角y坐标）
4. LONG right; //矩形右边（右下角x坐标）
5. LONG bottom; //矩形底边（右下角y坐标）
6. } RECT, *PRECT;
```

获得窗口客户区区域：

```
1. BOOL GetClientRect(
2. HWND hWnd, // handle to window
3. LPRECT lpRect // client coordinates
4. );
```

3）绘图设备之间的拷贝。

```
1. BOOL BitBlt(
2. HDC hdcDest,//目标绘图设备
3. int nXDest, // 目标区域左上角x坐标
4. int nYDest, //目标区域左上角y坐标
5. int nWidth, // 目标区域宽度
6. int nHeight,// 目标区域高度
7. HDC hdcSrc, // 源绘图设备
8. int nXSrc, // 源区域左上角x坐标
9. int nYSrc, // 源区域左上角y坐标
10. DWORD dwRop // 操作码。常用SRCCOPY（复制）。
11. );
```

可以简单的理解这个API：将源绘图设备拷贝给目标绘图设备。从幕后画布拷贝给前景画布。从幕后绘图设备拷贝给显示器。拷贝的区域，就是上面指定的。
如：BitBlt(hDC,0,0,g_nWidth,g_nHeight,memDC,0,0,SRCCOPY);

4）其他常用API

文字处理
创建字体:CreateFont，CreatePointFont，SelectObject(.,font)。
显示文本：Textout，DrawText
设置文本颜色：SetTextColor ,GetTextColor
设置背景：SetBkColor，SetBkMode(设置背景模式，TRANSPARENT为透明)

绘制
点：SetPixel,SetPixelV,GetPixel。
线：MoveToEx-LinTo。
圆：Ellipse
多边形：Polygon
弧：Arc

位图处理
从文件加载位图/图标/光标：LoadImage
从资源加载位图：LoadBitmap
获得位图信息：GetObject

绘图设备间的拷贝复制
直接模式：BitBlt
拉伸拷贝：StretchBlt
透明处理：TranparentBlt，需要加 Msimg32.lib库。
半透明处理(Alpha混合)：AlpaBlend

其他
获得系统属性：GetSystemMetrics，一些很重要的信息。
获得鼠标屏幕坐标：GetCursorPos，需使用坐标转换转换为窗口坐标
坐标转换：ClientToScreen,ScreenToClient
获得键盘键状态：GetAsyncKeyState
获得整个键盘信息：GetKeyboardState

消息框：MessageBox

一个完整的代码示例：

```
#include "app.h"

int g_nWidth = 600;//窗口宽度
int g_nHeight = 480;//窗口高度

void init()//游戏初始化
{
}

void update()//逻辑更新
{
}

void render()//画面渲染
{
    HDC hDC = GetDC(getHWnd());          //获得系统绘图设备

    HDC memDC = CreateCompatibleDC(0);  //创建辅助绘图设备

    HBITMAP bmpBack = CreateCompatibleBitmap(hDC,g_nWidth,g_nHeight);//创建掩码位图（画布）
    SelectObject(memDC,bmpBack);         //将画布贴到绘图设备上

    HPEN penBack = CreatePen(PS_SOLID,1,RGB(255,0,255));//创建画笔
    SelectObject(memDC,penBack);     //将画笔选到绘图设备上

    HBRUSH brushBack = CreateSolidBrush(RGB(255,255,255));//创建画刷
    SelectObject(memDC,brushBack);  //将画刷选到绘图设备上

    //擦除背景
    RECT rcClient;//区域结构
    GetClientRect(getHWnd(),&rcClient);//获得客户区域
    HBRUSH brushTemp = (HBRUSH)GetStockObject(WHITE_BRUSH);//获得库存物体，白色画刷。
    FillRect(memDC,&rcClient,brushTemp);//填充客户区域。
    //
        HBRUSH brushObj = CreateSolidBrush(RGB(0,255,0));//创建物体画刷
    //绘制维网格，矩形画法。
    int dw = 30;
    int rows = g_nHeight/dw;
    int cols = g_nWidth/dw;
    for (int r=0; r<rows; ++ r)
    {
        for (int c=0; c<cols; ++c)
        {
            if (r == c)
```

```
45.             {
46.                 SelectObject(memDC,brushObj);
47.             }
48.             else
49.             {
50.                 SelectObject(memDC,brushBack);
51.             }
52.             Rectangle(memDC,c*dw,r*dw,(c+1)*dw,(r+1)*dw);
53.         }
54.     }
55.
56.     DeleteObject(brushObj);
57.     //
58.     BitBlt(hDC,0,0,g_nWidth,g_nHeight,memDC,0,0,SRCCOPY);//复制到系统设备上显示
59.     DeleteObject(penBack);  //释放画笔资源
60.     DeleteObject(brushBack);//释放画刷资源
61.     DeleteObject(bmpBack);  //释放位图资源
62.     DeleteDC(memDC);        //释放辅助绘图设备
63.     ReleaseDC(getHWnd(),hDC);   //归还系统绘图设备
64.     Sleep(1);
65. }
66.
67. void clear()//资源释放
68. {
69. }
70.
71.
72. //主函数
73. int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE , LPSTR , int  )
74. {
75.     if(!initApp(hInstance,L"贪吃蛇游戏",g_nWidth,g_nHeight))
76.     {
77.         return 0;
78.     }
79.
80.     //初始化游戏
81.     init();
82.
83.     //游戏循环
84.     mainLoop();
85.
86.     //释放资源
87.     clear();
88.
89.     return 0;
90. }
```

原文地址：http://blog.csdn.net/you_lan_hai/article/details/6911497