第五课　资源

一、图标资源

1. 添加图标资源

一个图标文件中可以包含多个不同大小的图标。

2. 加载图标资源

1）注册窗口类时加载图标资源

```
HICON LoadIcon (
    HINSTANCE hInstance, // 应用程序实例句柄
    LPCTSTR   lpIconName // 图标资源名
);
```

成功返回图标句柄，失败返回NULL。

```
typedef struct {
    ...
    HICON hIcon;   // 大图标句柄(Alt+Tab)
    HICON hIconSm; // 小图标句柄(标题栏左上角)
    ...
}   WNDCLASSEX, *PWNDCLASSEX;

#include "resource.h"

WNDCLASSEX wcex = {0};
...
wcex.hIcon = LoadIcon (g_hInstance,
    MAKEINTRESOURCE (IDI_WINICON));
wcex.hIconSm = LoadIcon (g_hInstance,
    MAKEINTRESOURCE (IDI_WINICON_SM));
...
```

根据此类创建的所有的窗口都使用此图标。

2）动态设置窗口图标

```
WM_SETICON - 向窗口发送此消息以设置其图标
    wParam - 图标类型，ICON_BIG/ICON_SMALL
    lParam - 图标句柄

在代码中发送此消息可令窗口的大/小图标(根据wParam)
被替换为lParam所标识的图标。

HICON hIcon = LoadIcon (g_hInstance,
    MAKEINTRESOURCE (IDI_SETICON_BIG));
SendMessage (hWnd, WM_SETICON, ICON_BIG, (LPARAM)hIcon);
```

3. 绘制图标

```
BOOL DrawIcon (
    HDC   hDC,  // 设备上下文句柄
    int   X,    // 左上角水平坐标
    int   Y,    // 左上角垂直坐标
    HICON hIcon // 图标句柄
);
```

成功返回TRUE，失败返回FALSE。

```
BOOL DrawIconEx (
    HDC    hdc,                // 设备上下文句柄
    int    xLeft,              // 左上角水平坐标
    int    yTop,               // 左上角垂直坐标
    HICON  hIcon,              // 图标句柄
    int    cxWidth,            // 图标宽度
    int    cyHeight,           // 图标高度
    UINT   istepIfAniCur,      // 不支持动画图标，置0
    HBRUSH hbrFlickerFreeDraw, // 由系统直接绘制，置NULL
    UINT   diFlags             // 绘制标志
                               // DI_NORMAL/DI_IMAGE/DI_MASK
                               // 正常/图像/掩码
);
```

成功返回TRUE，失败返回FALSE。

范例：WinIcon

```
0 & X = 0 -> B & C = B
1 & X = X -> W & C = C
0 | X = X -> B | C = C
1 | X = 1 -> W | C = W

DST &= MASK
DST |= IMAGE
```

二、光标资源

---

1. 添加光标资源

光标的默认大小32X32像素，每个光标有一个热点(HotSpot)，
即鼠标的位置点。

2. 加载光标资源

1) 注册窗口类时加载光标资源

```
HCURSOR LoadCursor (
    HINSTANCE hInstance,   // 应用程序实例句柄
    LPCTSTR   lpCursorName // 光标资源名
);
```

成功返回光标句柄，失败返回NULL。

```
typedef struct {
    ...
    HCURSOR hCursor; // 光标句柄
    ...
}   WNDCLASSEX, *PWNDCLASSEX;

#include "resource.h"

WNDCLASSEX wcex = {0};
...
wcex.hCursor = LoadCursor (g_hInstance,
    MAKEINTRESOURCE (...));
...
```

根据此类创建的所有的窗口都使用此光标。

2) 动态设置窗口光标

```
WM_SETCURSOR - 有鼠标动作且其未被捕获(SetCapture/ReleaseCapture)时发送此消息
      wParam - 当前光标句柄
      lParam - LOWORD当前区域代码,
               可据此判断鼠标是在客户区还是在标题栏或者其它区域,
               HIWORD当前鼠标消息ID
```

处理此消息时调用以下函数：

```
HCURSOR SetCursor (
    HCURSOR hCursor // 光标句柄
);
```

返回原光标句柄，无原光标返回NULL。

3. 系统自带光标

静态光标：C:\WINDOWS\Cursors*.cur

动画光标：C:\WINDOWS\Cursors*.ani

4. 从文件加载光标

```
HCURSOR LoadCursorFromFile (
    LPCTSTR lpFileName // 光标文件路径
);
```

范例：WinCursor

三、字符串资源
---

1. 添加字符串表资源

添加字符串表，在表中添加字符串。

2. 加载字符串资源

```
int LoadString (
    HINSTANCE hInstance,     // 应用程序实例句柄
    UINT      uID,           // 字符串ID
    LPTSTR    lpBuffer,      // 字符串缓冲区
    int       cchBufferMax   // 字符串缓冲区大小(以字符为单位)
);
```

成功返回实际载入lpBuffer缓冲区中的字符个数(不含结尾空字符)，
失败返回0。

范例：WinString

四、加速键资源
---

1. 添加加速键表资源

添加加速键列表，增加与命令ID相对应的加速键。

2. 加载加速键表资源

```
HACCEL LoadAccelerators (
    HINSTANCE hInstance,  // 应用程序实例句柄
    LPCTSTR   lpTableName // 加速键表资源名
);
```

成功返回加速键表句柄，失败返回NULL。

3. 翻译加速键消息

```
int TranslateAccelerator (
    HWND   hWnd,       // 处理消息的窗口句柄
    HACCEL hAccTable, // 加速键表句柄
    LPMSG  lpMsg       // 消息结构体
);
```

成功返回非零，失败返回零。

```
int TranslateAccelerator (HWND hWnd, HACCEL hAccTable, LPMSG lpMsg)
{
    if (lpMsg -> message == WM_KEYDOWN || lpMsg -> message == WM_SYSKEYDOWN)
    {
        从lpMsg中提取按键信息;

        if (hAccTable加速键表中存在与所按键相对应的命令ID)
        {
            PostMessage (hWnd, WM_COMMAND, MAKELONG (命令ID, 1), NULL);
```

```
            return 1;
        }
    }

    return 0;
}
```

加入消息循环：

```
MSG msg = {0};
while (GetMessage (&msg, NULL, 0, 0))
    if (! TranslateAccelerator (msg.hwnd, hAccTable, &msg))
    {
        TranslateMessage (&msg);
        DispatchMessage (&msg);
    }
```

4. 独立于菜单的加速键消息

```
WM_COMMAND - 命令消息
    wParam - HIWORD菜单0/加速键1/控件通知码，
             LOWORD菜单ID/加速键ID/控件ID
    lParam - 若发自控件则为其句柄，否则为NULL
```

5. Alt热键

```
&File
  E&xit
```

范例：WinAcc

五、版本资源

范例：WinVersion