برنامه نویسی بازی های کامپیوتری (نسخه 3.0)

بهترین اندازه دید : 1024 * 768

نویسنده : حسن مهدی اصل (مهدی 2190)
mehdi_2190@yahoo.com

WWW.Persian-Designers.COM

دوستان سلام امیدوارم که آمادگی لازم برای یادگیری قسمت سوم را داشته باشید .
پیش از ادامه بحث چند نکته را یادآور می شوم . اول اگر با دقت به حلقه رویداد نگاه کنید به نظر نمی رسد که رویداد بلادرنگ باشد . یعنی زمانی که برنامه منتظر پیامی از طریق GetMessage() می ماند ، حلقه رویداد اصلی بلوکه خواهد شد . برای حل این مشکل باید فکری کنید ، چون پردازش بازی باید به صورت پیوسته انجام گیرد و در عین حال رویدادهای ویندوز را هر جا و هر زمان که اتفاق بیفتد ، مدیریت نمایید.

ایجاد حلقه رویداد زمان واقعی (بلادرنگ)

ایجاد حلقه رویداد زمان واقعی بدون انتظار بسیار ساده است . فقط باید به روشی صف پیام را بررسی کنید که آیا شامل پیام است یا نه . اگر پیامی وجود دارد ، آن را پردازش می کنید ، در غیر این صورت پردازش منطق و تکرار بازی را ادامه می دهید . تابعی که این وظیفه را انجام می دهد ، ( )PeekMessage نام دارد. الگوی عمومی آن همواره شبیه تابع ( )GetMessage و به صورت زیر است :

```
BOOL PeekMessage(
LPMSG lpMsg,
HWND hwnd,
UINT wMsgFilterMin,
UINT wMsgFilterMax,
UINT wRemoveMsg) ;
```

اگر پیامی در دسترس باشد ، این تابع مقداری غیر صفر را برمی گرداند .

تفاوت اصلی در پارامتر آخر است که شیوه بازیابی پیامها از صف پیام را تعیین می کند . پرچم های معتبر برای wRemoveMsg عبارتند از :

PM_NOREMOVE – پیامها پس از پایان پردازش توسط ( ) PeekMessage از صف حذف نمی شوند .
PM_REMOVE – پیامها پس از پایان پردازش توسط ( ) PeekMessage از صف حذف می شوند .

با در نظر گرفتن این دواحتمال ، یکی از دو کار را انجام می دهید : استفاده از PeekMessage() با PM_NOREMOVE و اگر پیامی وجود داشته باشد ، فراخوانی GetMessage() ، و یا استفاده از PM_REMOVE و استفاده از PeekMessage() برای بازیابی پیام موجود . از روش اخیر استفاده نمایید . در اینجا منطق اصلی طوری تغیر یافته تا عملکرد این تکنیک جدید در حلقه رویداد اصلی را نشان دهد :

```
-whlie(TRUE)
{
If(PeekMessage(&msg,NULL,0,0,PM_REMOVE))
{
If(msg.message == WM_QUIT)
Break;
TranslateMessage(&msg);
DispatchMessage(&msg);
Game_Main( );
}
```

در خط پنجم کد (;if(msg.message ==WM_QUIT)break) روش ضمانت کافی در برابر حلقه نامتناهی while(TRUE) است . بیاد داشته باشید که هنگام پردازش پیام WM_DESTROY در winproc ، وظیفه شماست تا یک پیام WM_QUIT از طریق postQuitMessage( ) ارسال نمایید . سپس پیام WM_QUIT به تدریج در صف رویداد پیش آمده و می توانید آن را شناسایی کنید تا مانع حلقه اصلی گردید.

در قسمت آخر یک تابع جدید به چشم می خورد . این تابع نشان می دهد که کجا باید فراخوانی به حلقه کد بازی اصلی را انجام دهید . اما بیاد داشته باشید که فراخوانی Game_Main( ) یا هر فراخوانی دیگر ، باید مقداری را بعد از یک فریم از متحرک سازی یا منطق بازی برگرداند. در غیر این صورت پیام ها توسط حلقه رویداد اصلی ویندوز مورد پردازش قرار نمی گیرند .

برای دسترسی به کد اصلی این مثال می توانید از فایل همراه جزوه test3-1.cpp استفاده نمائید .

باز نمودن پینجره های بیشتر

در این قسمت بازکردن بیش از یک پنجره را می آموزید . کافی است دو یا چند فراخوانی به CreateWindowEX( ) انجام داده تا پنجره ها ایجاد گردند . اما برخی نکات را باید در نظر بگیرید . نکته اول اینکه هنگامی که پنجره ای ایجاد می کنید ، این پنجره به یک کلاس متکی است . این کلاس مشخص کننده winproc یا مدیر رویداد برای کل کلاس است . این نکته بسیار مهم است . می توانید با یک کلاس به تعداد دلخواه پنجره ایجاد نمایید ، اما همه پیامها به یک winproc ارسال می شود که توسط مدیر رویداد مورد اشاره توسط فیلد lpfnwndproc در ساختار WINCLASSEX تعریف شده است .

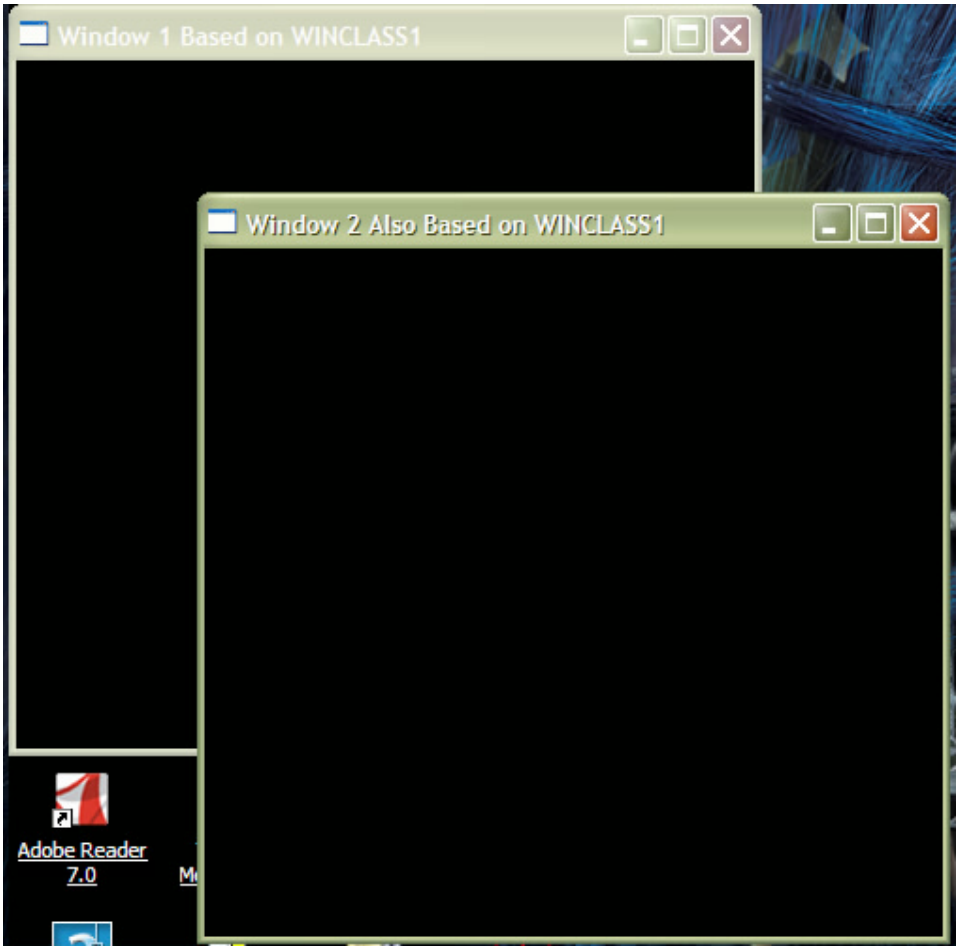
اگر برای هر پنجره به یک winproc متفاوت نیاز دارید ، باید بیش از یک کلاس ویندوز ایجاد کنید و هر پنجره را با یک کلاس متفاوت ایجاد کنید . بدین ترتیب پیامهای مربوط به هر کلاس ویندوز به یک winproc متفاوت ارسال خواهد شد .
در اینجا کد لازم برای ایجاد دو پنجره بر اساس یک کلاس را مشاهده می فرمائید :

```
If(!(hwnd = CreateWindowEx(NULL,WINDOW_CLASS_NAME,
" window 1 bassed on winclass 1" ,
WS_OVERLAPPEDWINDOW | WS_VISIBLE ,
0 , 0 ,
400 , 400 ,
NULL ,
NULL ,
Hinstance ,
NULL )))
Return (0);
If(!(hwnd = CreateWindowEx(NULL,WINDOW_CLASS_NAME,
" window 2 also on winclass 1" ,
WS_OVERLAPPEDWINDOW | WS_VISIBLE ,
100 , 100 ,
400 , 400 ,
NULL ,
NULL ,
Hinstance ,
NULL )))
Return (0);
```

البته ممکن است بخواهید هر دستگیره پنجره را در متغیرهای متفاوت به صورت جداگانه ردیابی نمایید ، مانند حالت hwnd . به منظور بازکردن دو پنجره همزمان فایل test3-2.cpp همراه جزوه را می توانید مشاهده نمایید . دقت کنید اگر هر کدام از پنجره ها را ببندید هر دو پنجره بسته خواهد شد .

آیا می توانید روشی برای بستن یک پنجره پیدا کنید ؟

کمی کمکتان می کنم ، دو کلاس ویندوز ایجاد نموده و تا زمانی که هر دو پنجره بسته نشده اند پیام WM_QUIT را ارسال نکنید .
در زیر می توانید برنامه کامپایل شده را مشاهده کنید :



در اینجا شما مطالب لازم برای برنامه نویسی پیشرفته را در اختیار دارید . اکنون درباره معماری ویندوز اطلاعات کافی دارید . شیوه ایجاد کلاسهای ویندوز ، کلاسهای رجیستری ، ایجاد پنجره ها و نوشتن جلقه های رویداد و مدیران رویداد را آموختید . در قسمتهای بعد برخی مطالب مربوط به ویندوز مانند استفاده از منابع ، ایجاد منوها ، کار با کادرهای تبادلی و کسب اطلاعات از کاربر را می آموزید .

برنامه نویسی پیشرفته ویندوز

برنامه نویسی ویندوز واقعا مشکل است . اما جالب است که برای برنامه نویسی ویندوز به معلومات زیادی نیاز ندارید . در این فصل از سری مقالات برنامه نویسی بازهای کامپیوتری ، مطالب تکمیلی برای نوشتن برنامه های کامل ویندوز را می آموزید :

- استفاده از منابع طبیعی همچون آیکونها ، مکان نما ها ، صدا ها .
- منو ها
- GDI اصلی و سیستم ویدیوئی
- دستگاههای ورودی
- ارسال پیامها

استفاده از منابع

یکی از مهمترین نکاتی که مورد نظر سازندگان ویندوز قرار داشته ، حل مشکل ذخیره سازی اطلاعاتی فراتر از کد برنامه در یک قالب استاندارد ویندوز است . راه حل آنها این است که داده های یک برنامه اجرایی باید داخل EXE همان برنامه قرار گیرند . این ایده به چند دلیل بسیار خوب است .

- توزیع یک فایل exe شامل کد و اطلاعات برنامه بسیار ساده تر است .
- اگر فایل داده های خارجی نداشته باشد ، هرگز منابع خود را گم نمی کند .

- نیروهای بیرونی و خراب کاران بسادگی به فایلهای داده ای شما مانند bmp و wav دسترسی ندارند (البته هم اکنون شاهد برنامه هایی مثل resource hacker هستیم که تمام منابع یک فایل exe که به این طریق ساخته می شود را به کاربران دیگر ارائه می دهد)

برنامه های ویندوز به منظور تسهیل این تکنولوژی بانک اطلاعاتی از چیزی با نام منابع (resources) پشتیبانی می کنند. این تابع در حقیقت قطعاتی از داده های ترکیب شده با کد برنامه هستند که امکان دارد توسط خود برنامه و بعدا طی زمان اجرا بارگذاری شوند .

در واقعیت هیچگونه محدودیتی برای نوع داده های کامپایل شده در یک برنامه وجود ندارد ، چون برنامه های ویندوز از انواع منابع تعریف شده توسط کاربر پشتیبانی می کنند . اما انواع از پیش تعریف شده ای وجود دارند که بسیاری از نیازهای شما را مرتفع می سازد .

آیکون ها (ICONS)

تصاویر نقش بیتی کوچکی که در مکانهای مختلفی استفاده می شوند . مانند تصویری که کلیک می کنید تا یک برنامه اجرا شود . آیکون ها از پسوند ico. استفاده می کنند .

مکان نما ها (CURSORS)

یک نقش بیتی که معرف اشاره گر ماوس است . در ویندوز به روشهای گوناگون می توانید مکان نما ها را تغییر دهید . مثلا هنگام انتقال مکان نما از یک پنجره به پنجره دیگر می توانید شکل آن را تغییر دهید . مکان نما ها از پسوند cur. استفاده می کنند .

رشته ها (STRINGS)

رشته ها ممکن است به عنوان یک منبع چندان آشکار نباشند . ویندوز به شما این امکان را می دهد تا جدولی از رشته ها را داخل برنامه ها و به عنوان منبع قرار داده و به آنها از طریق ID دسترسی پیدا کنید .

صدا ها (SOUNDS)

بیشتر برنامه های ویندوز حداقل استفاده از صدا ها را توسط فایلهای wav انجام می دهند . فایلهای wav. را می توانید به منابع اضافه کنید .

نقشه های بیتی (BITMAPS)

شامل نقشه های بیتی استاندارد است : یک ماترس چهار ضلعی برای پیکسلها در حالت مونوکروم یا با فرمت 4 ، 8 ، 24 ، 32 بیتی . اشیا بسیار رایجی در سیستم عامل های گرافیکی مانند ویندوز وجود دارند که می توانید آنها را به عنوان منابع اضافه نمایید . نقشه های بیتی از پسوند BMP. استفاده می کنند .

کادرها (DIALOGS)

کادرهای تبادلی در ویندوز آنچنان رواج دارند که طراحان ویندوز تصمیم گرفتند آنها را به عنوان منابع قرار دهند . می توانید کادرهای تبادلی را بر روی هوا و از طریق کدها ایجاد نمایید ، و یا آنها را توسط یک ویراستار طراحی کرده و به عنوان یک منبع ذخیره نمایید .

فایلهای متا (META FILES)

اینها اندکی پیشرفته ترند . این نوع فایلها اجازه می دهند تا یک توالی از عملیات گرافیکی را ضبط کرده و در یک فایل قرار داده و سپس آنها را اجرا نمایید .

اکنون درباره منابع و انواع آنها اطلاعاتی کافی را دارید ، اما چگونه آنها را در کنار یکدیگر قرار می دهید ؟ برنامه ای با نام resource compiler وجود دارد . این برنامه به عنوان ورودی یک فایل منبع متنی اسکی با پسوند RC. می پذیرد. این فایل یک توصیف از کلیه منابع است که می خواهید در یک فایل داده ای کامپایل کنید . این برنامه سپس کلیه منابع را بارگذاری کرده و آنها را در یک فایل داده ای بزرگ با پسوند RES. قرار می دهد .
فایل RES. شامل داده های باینری است که سازنده آیکونها ، مکان نما ها ، نقشه های بیتی و صدا ها و سایر مواردی است که امکان دارد در فایل منبع RC. تعریف کرده باشید . سپس فایل RES. به همراه H-LB-OBJ- cpp – و غیره در یک فایل exe کامپایل خواهد شد .

کنار یکدیگر قرار دادن منابع

در گذشته از یک کامپایلر منابع خارجی مانند RC.exe برای کامپایل کلیه منابع با یکدیگر استفاده می شد . اما در حال حاضر کامپایلر IDE همه این وظایف را انجام می دهد . بنابراین اگر مایلید تا منبعی را به برنامه خود بیفزایید ، کافی است از منوی file و گزینه new را انتخاب کنید و سپس نوع منبع را از زبانه files انتخاب نمایید . می توانید تعداد زیادی از انواع داده ها و اشیا را به برنامه خود اضافه نمایید . آنها به عنوان منابع داخل exe و در کنار کد اصلی برنامه قرار می گیرند . سپس طی اجرای برنامه ، می توانید به بانک اطلاعاتی این منابع دسترسی یافته و اطلاعات منبع را از طریق خود برنامه بارگذاری نمایید . به علاوه برای ایجاد فایل منبع ، باید یک فایل توصیف منبع به صورت متن اسکی و با پسوند RC. داشته باشید (در نسخه vc++6 شما نیاز ندارید که این فایل را به صورت اسکی کد ببینید ، چون کامپایلر کل عملیات کد نویسی را خود انجام می دهد و شما فقط به صورت بصری با آن کار می کنید) . این فایل سپس به کامپایلر منتقل شده و یک فایل با پسوند RES تولید خواهد شد . این فایل RES سپس با کلیه اشیا دیگر برنامه پیوند یافته تا یک فایل exe ایجاد گردد . اکنون تعدادی از اشیا منبع و شیوه ایجاد آنها و بارگذاری آنها در کنار یکدیگر در یک برنامه را بررسی می کنیم . عملکرد آنها یکسان است .

استفاده از منابع آیکون

برای کار با منابع فقط دو فایل باید ایجاد کنید ، یک فایل RC. و احتمالا در صورت لزوم یک فایل H.
اگر بخواهید ارجاعاتی به شناسه های سمبولیک در فایل RC. داشته باشید . البته در نهایت باید یک فایل RES. ایجاد کنید که ما اجازه می دهیم تا کامپایلر IDE این کار را انجام دهد .
به عنوان مثالی از ایجاد یک منبع icon ، شیوه تغییر آیکون قرار گرفته بر روی نوار وظیفه برنامه و آیکون نزدیک منوی سیستم در خود ویندوز را بررسی می کنیم . اگر بیاد داشته باشید ، این آیکونها را طی ایجاد کلاس ویندوز با کدهای زیر پر کردیم :

```
Winclass.hicon          =LoadIcon(NULL,IDI_APPLICATION);
Winclass.hiconsm        =LoadIcon(NULL,IDI_APPLICATION);
```

این کدها ، آیکون پیش فرض برنامه و هم آیکون عادی و هم نسخه کوچکتر آیکون را بارگذاری می کنند اما با استفاده از آیکون های کامپایل شده در یک فایل منبع ، می توانید هر آیکون دلخواه را به جای آیکون های پیش فرض قرار دهید .

ابتدا به یک آیکون نیاز دارید . همراه این جزوه برای شما آیکونی قرار داده ام (ID_ICON1.ico) . اما شما می توانید آیکون ها ، مکان نما ها ، نقشه های بیتی و سایر منابع را با هر برنامه دلخواه دیگر ایجاد نمایید.
این آیکون با ابعاد 32*32 پیکسل و دارای 256 رنگ است . اندازه آیکونها از 16*16 تا 64*64 پیکسل و رنگ آنها تا 256 رنگ مجاز است . اما بیشتر به اندازه 32*32 و دارای 16 رنگ یا بیشتر هستند .

هرگاه بخواهید آیکونی را در یک فایل منبع قرار دهید ، باید ابتدا یک فایل منبع rc ایجاد کنید به ترتیب زیر این کار را انجام دهید .
از منوی فایل گزینه new را بزنید و از پنجره باز شده زبانه files ، سپس resource script را انتخاب کنید و سپس نامی برای آن در نظر بگیرید و ok را بزنید . فایل rc شامل تعریف همه منابع است ، پس ممکن است بیش از یک منبع در برنامه داشته باشید .

تذکر مهم :

پیش از بررسی کدها بهتر است توجه شما را به یک نکته بسیار مهم جلب نمایم . ویندوز از رشته های متنی اسکی یا ID های عدد صحیح برای ارجاع به منابع استفاده می کند . در اکثر شرایط می توانید از هر دو در فایلهای rc استفاده نمایید . اما برای برخی منابع فقط از یک روش می توان استفاده نمود . در هر یک از شرایط فوق ، منابع با کمی تفاوت بارگذاری می شوند و اگر از ID ها استفاده کنید ، یک فایل اضافی H شامل ارجاعات نمادی را باید داخل پروژه بگنجانید .

با دوبار کلیک بر روی فایل rc ایجاد شده ، پنجره کوچکی باز می شود . با راست کلیک کردن درون این پنجره منویی ظاهر می شود که شامل تعدادی فرمان است . برای افزودن منابع آماده از گزینه import و برای ایجاد منابع جدید توسط خود برنامه از گزینه insert استفاده می کنید .

فعلا قصد داریم از فایلهای آماده استفاده کنیم با زدن import و انتخاب آیکون مورد نظر در لیست پنجره فایل rc پوشه ای به نام icon ایجاد می شود و فایل لود شده نیز در آنجا قرار میگیرد .
نرم افزار به طور خود کار یک id به آن اختصاص می دهد که شما می توانید آن را با گزینه properties به نام یا شماره دلخواه خود تغییر دهید .

به همان ترتیب که یک فایل rc ساختید ، می توانید با انتخاب گزینه C/C++ Header file و اختصاص یک نام به آن یک فایل H. بسازید .
بعد از ساختن فایل H. بر حسب فایلهای آیکونی که وارد فایل rc کردید (هر چند تا که هستند) باید ارجاعاتی در فایل H. ایجاد نمایید . به کدهای نمونه زیر توجه فرمایید که برای سه آیکون موجود در فایل rc که ارجاع شده اند توسط یک فایل نمونه H.

```
#define ID_ICON1            100
```

```
#define ID_ICON2        101
#define ID_ICON3        102
```

البته اگر کدی برای آنها تعریف نکنید خود کامپایلر برای آنها کدی اختصاص می دهد . ولی بهتر است این کار را خودتان انجام دهید .

سپس باید این دو فایل را به یکدیگر لینک کنیم (اتصال دهیم)
در فایل rc قسمتی که خالی باشد راست کلیک کنید و از منوی ظاهر شده گزینه resource includes را انتخاب کنید و در پنجره ظاهر شده Read_Only symbol directives را که به صورت پیش فرض کدی مثل این در آن قرار دارد #include"afxres.h" را به نام فایل سرآیند (H) خود تغییر دهید . یعنی به جای کلمه afxres در این خط کد نام اختصاص داده شده برای فایل سرآیند خود را وارد نمایید . به طور مثال : # include "test.h"

اگر نمادها را توسط #define برای آیکونها تعریف نکنید و یک فایل H. را بگنجانید ، آنگاه کامپایلر فرض می کند نمادهای ID_ICON1 و ID_ICON2 و ID_ICON3 رشته های عادی هستند و به صورت زیر به آنها ارجاع می کند .
"ID_ICON1" "ID_ICON2" "ID_ICON3"

به منظور بارگذاری یک آیکون توسط نام رشته ای عملیات زیر انجام دهید :

```
Winclass.hicon   = LoadIcon(hinstance, "your_icon_name");
Winclass.hiconsm=LoadIcon(hinstance, "your_icon_name");
```

به جای your icon name نام آیکون خود را قرار دهید .
و به منظور بارگذاری توسط ارجاع نمادی باید بعد از افزودن ارجاعات لازم به rc و H کد زیر را برای دست رسی به منابع نوشت :

```
Winclass.hicon =LoadIcon(hinstance,MAKEINTRESOURCE(ID_ICON1));
Winclass.hiconsm=LoadIcon(hinstance,MAKEINTRESOURCE(ID_ICON1));
```

به کاربرد ماکروی MAKEINTRESOURE( ) دقت نمایید . این ماکرو عدد صحیح را به یک اشاره گر رشته تبدیل می کند . از این ماکرو فقط هنگام استفاده از تعریف ثابت های نمادی استفاده می کنید .

استفاده از منابع مکان نما

منابع مکان نما مشابه منابع آیکونها هستند (در طرز استفاده) . فایلهای Cursor در حقیقت نقشهای بیتی کوچک با پسوند CUR. هستند که در بیشتر IDE کامپایلرها و یا توسط سایر برنامه های تصویر پردازی می توان ایجاد نمود . مکان نماها اغلب با ابعاد 32*32 پیکسل و دارای 16 رنگ هستند اما تا ابعاد 64*64 پیسکل و تا 256 رنگ را پذیرفته و حتی امکان متحرک سازی آنها وجود دارد .

برای افزودن فایل مکان نما نیز مانند مراحل افزودن آیکون عمل نمایید .
کدهای زیر را برای ارجاع به فایل H. درون آن می نویسیم :

```
#define ID_CURSOR1        200
#define ID_CURSOR2        201
```

خوب حالا کد مربوط به بارگذاری مکان نما به صورت رشته را در زیر می بینید :

```
Winclass.hCursor =LoadCursor(hinstance ,"ID_CURSOR1");
```

البته کد بالا در شرایطی درست است که #define را در اراجاعات به کار نبریم .
خوب حالا کد مربوط به دسترسی با ID را مشاهده می نمائید (این روش بیشتر پیشنهاد می شود)

```
Winclass.hCursor =LoadCursor(hinstance,MAKEINTRESOURCE(ID_CURSOR1));
```

دوباره از ماکروی MAKEINTRESOURCE( ) استفاده می کنید تا ID عدد صحیح را به فرم دلخواه ویندوز تبدیل نمایید .
در اینجا می توانید بعد از بارگذاری یک آیکون و یک مکان نما در فایل rc و نوشتن ارجاعات لازم در فایل H و لینک کردن فایل H به rc و از همه مهمتر لینک کردن فایل H به محیط اصلی فایل cpp. با کمک کد زیر می توانید یک پنجره به همراه مکان نما در خروجی داشته باشید :

```
#include <windows.h>
...
#include <test3-3.h>
```

```
...
...
Winclass.hicon=LoadIcon(hinstance,MAKEINTRESOURCE (ICON_T3DX));
Winclass.hCursor=LoadCursor(hinstance,MAKEINTRESOURCE (CURSOR1));
Winclass.hiconsm=LoadIcon(hinstance,MAKEINTRESOURCE(ICON_T3DX));
```

به منظور دست رسی به کد کامل و صحیح به پوشه compile همراه جزوه و فایل (test3-3.cpp) مراجعه نمائید . همراه این فایل دو فایل دیگر برای استفاده موجود می باشد که در صورت لزوم می توانید از آنها استفاده نمائید .
(test3-3.h test3-3.rc)

در پوشه exe همراه جزوه فایل کامپایل شده در اختیار شماست .
منتظر جزوات بعدی باشید ...

سایت طراحان ایرانی با هدف آموزش ساخت بازیهای کامپیوتری به زبان فارسی طراحی شده است و تا کنون مقالات متعددی در زمینه های مختلف برنامه نویسی و ساخت بازی در آن قرار گرفته است. مدیریت سایت از تمامی عزیزان علاقمند به بازی های کامپیوتری ، برنامه نویسان ، طراحان و سایر کسانی که به نحوی با بازی ها در ارتباطند ، دعوت به همکاری به عمل می آورد تا بدینوسیله یک پایگاه علمی و موثق در زمینه صنعت ساخت بازیهای کامپیوتری در ایران ایجاد گردد.

در ضمن بسیاری از نرم افزار های ساخت بازی های کامپیوتری که امروزه در سطح وسیع مورد استفاده قرار میگیرند ، در سایت جمع آوری شده است و با مبلغ بسیار ناچیزی در اختیار علاقمندان به طراحی بازی های کامپیوتری قرار داده شده است. استفاده از این نرم افزار ها در آغاز کار و به منظور آشنا شدن با اصول اولیه در طراحی بازیها بسیار موثر و مفید بوده و شما میتوانید تا با چند جستجوی ساده در این زمینه ، به صحت موضوع پی ببرید.

برخی از نرم افزار هایی که توسط فروشگاه الکترونیکی سایت به مشتاقان عرضه میشوند :

آدرس فروشگاه الکترونیکی : http://www.persian-designers.com/index.php?pid=1