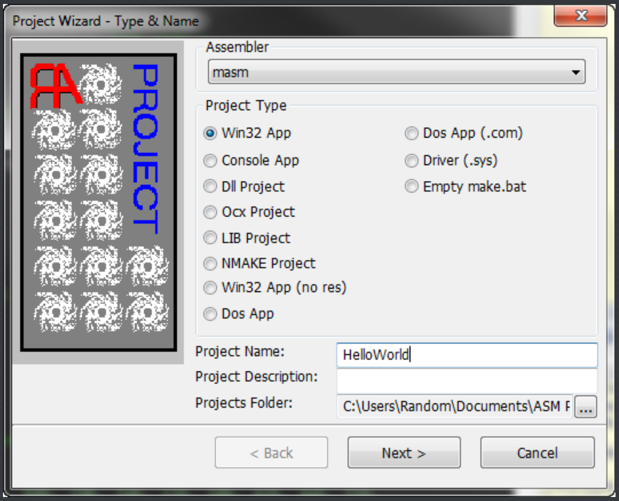
# R4ndom's Guide to RadASM: Our First Project

by R4ndom on Sep.24, 2012, under Beginner, Tools, Tutorials

## Creating Our First Project

Let's create our first Windows program. Select "New Project" from the File menu. This brings up the New Project Wizard:



Assembler: masm

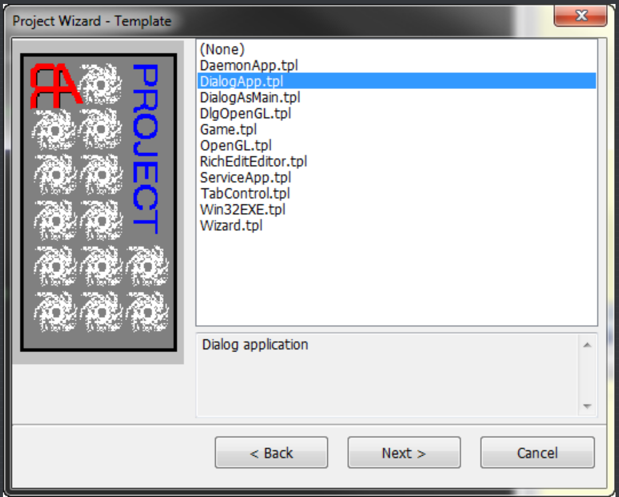
Project Type:
- [✓ Win32 App]
- Console App
- Dll Project
- Ocx Project
- LIB Project
- NMAKE Project
- Win32 App (no res)
- Dos App
- Dos App (.com)
- Driver (.sys)
- Empty make.bat

Project Name: HelloWorld
Project Description:
Projects Folder: C:\Users\Random\Documents\ASM F

The assembler drop down lists all of the various languages we installed at the beginning. In my case, I only have one: Masm.

The project type allows you to designate what type of project we will be making. Setting the different types allows RadASM to pre-make some files for you, along with setting certain settings that go along with your program type. For example, if you select "Dll Project", files will be created to help set up DLL exports and such. For our first program, let's keep it set to Win32 App. Type in a name in the Project Name field. I entered "HelloWorld". The other fields can be left blank. Click "Next".

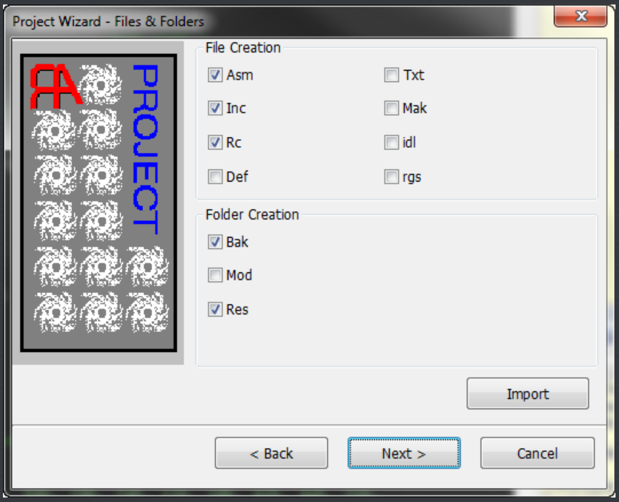
Next is the template selection screen:



(None)
DaemonApp.tpl
[✓ DialogApp.tpl]
DialogAsMain.tpl
DlgOpenGL.tpl
Game.tpl
OpenGL.tpl
RichEditEditor.tpl
ServiceApp.tpl
TabControl.tpl
Win32EXE.tpl
Wizard.tpl

Dialog application

Templates allow you to pre-populate files with code templates. RadASM comes with several templates for Win32 applications, or you can write your own and save them. The “DialogasMain sets up a project using a dialog as the main screen. The Win32exe sets up a project with a single window but more callback message handlers. For now, select the “DialogApp.tpl” template and click “Next” This will set up  a simple window project.



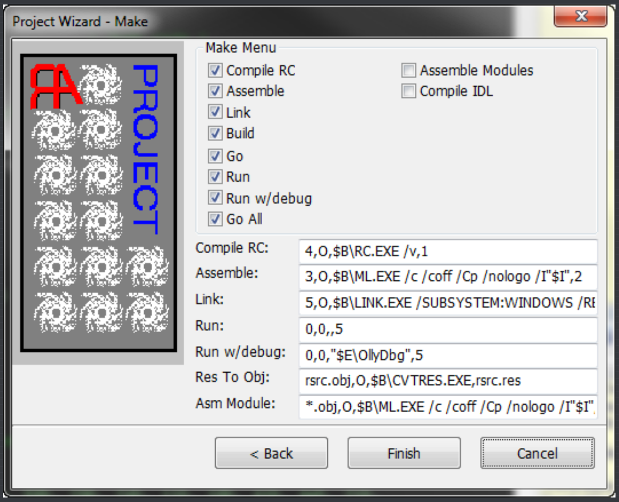
File Creation

[✓] Asm [ ] Txt
[✓] Inc [ ] Mak
[✓] Rc [ ] idl
[ ] Def [ ] rgs

Folder Creation

[✓] Bak
[ ] Mod
[✓] Res

The next screen is the Files and Folders screen. this screen allows you to choose some files and folders you would like the program wizard to create for you. For example, if programming a Win32 application, we will probably want a resource file (.rc) as well as an include file (.inc). The Bak, Mod and Res options are for creating folders, so if you like to keep all resource objects in a Res folder, keep this highlighted. The Bak folder is a place to store backups of your project. Just leave everything as it is and select “Next”:



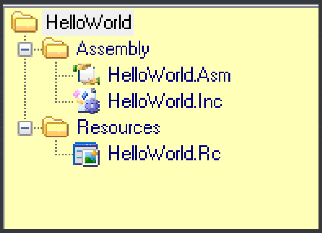
Make Menu

[✓] Compile RC [ ] Assemble Modules
[✓] Assemble [ ] Compile IDL
[✓] Link
[✓] Build
[✓] Go
[✓] Run
[✓] Run w/debug
[✓] Go All

| | |
|---|---|
| Compile RC: | 4,O,$B\RC.EXE /v,1 |
| Assemble: | 3,O,$B\ML.EXE /c /coff /Cp /nologo /I"$I",2 |
| Link: | 5,O,$B\LINK.EXE /SUBSYSTEM:WINDOWS /R |
| Run: | 0,0,,5 |
| Run w/debug: | 0,0,"$E\OllyDbg",5 |
| Res To Obj: | rsrc.obj,O,$B\CVTRES.EXE,rsrc.res |
| Asm Module: | *.obj,O,$B\ML.EXE /c /coff /Cp /nologo /I"$I" |

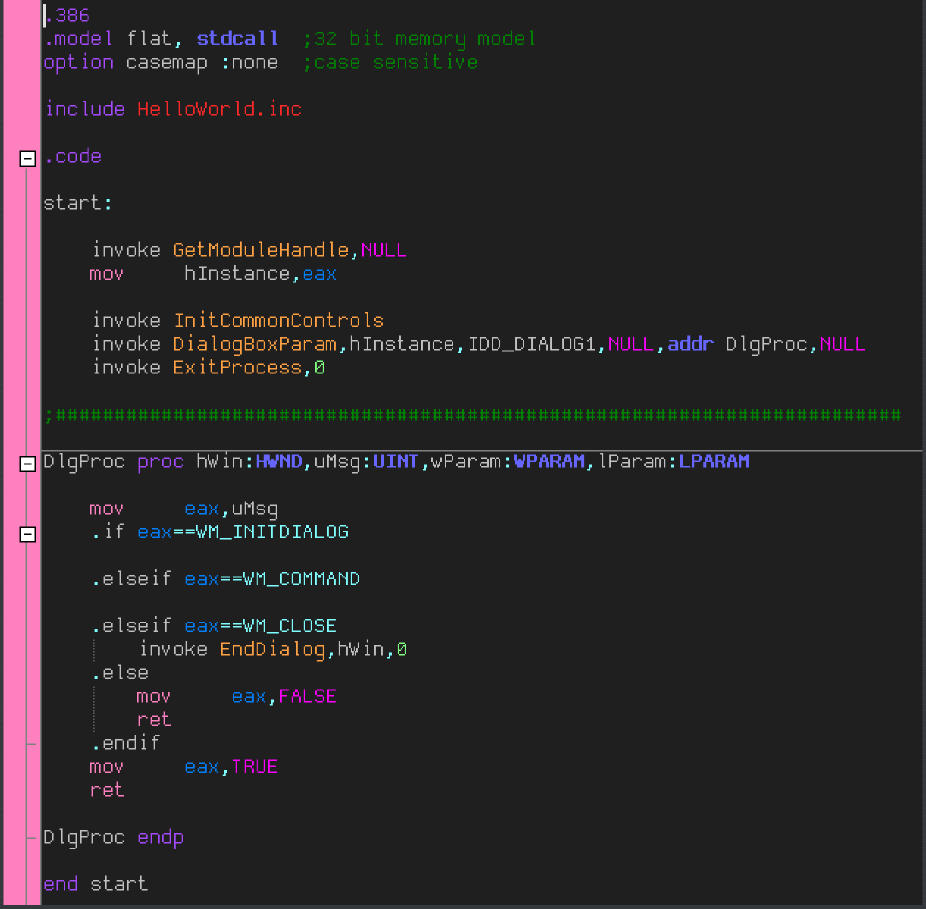
Finally, we reach the Make screen. This allows us to change the command line options, debugger, and the options we would like displayed on the “Make” menu drop-down. A lot of this is more advanced, so just leave everything as it is and click “Finish”.

You will see that RadASM has created several files for us already:



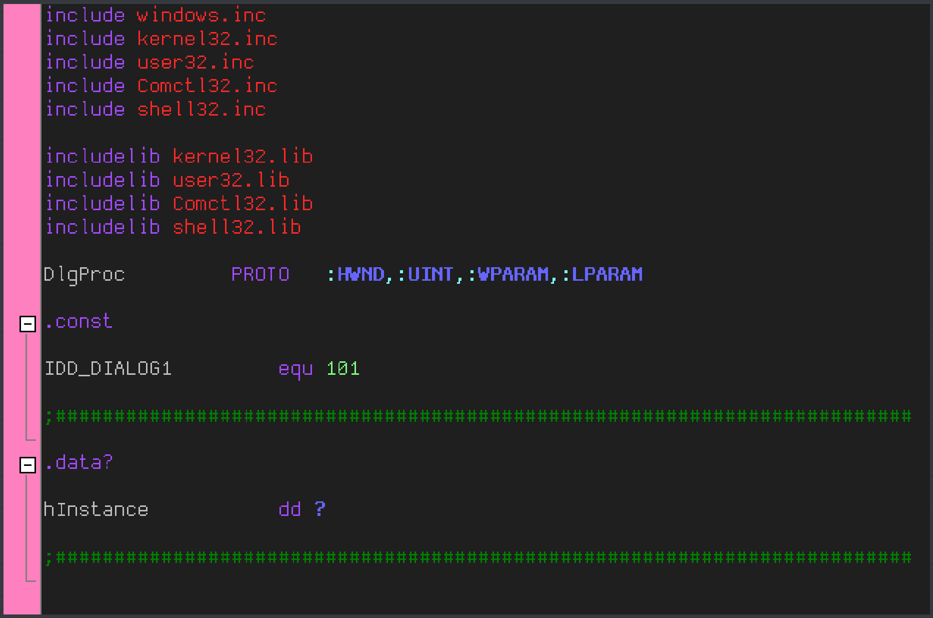
```
HelloWorld
  Assembly
    HelloWorld.Asm
    HelloWorld.Inc
  Resources
    HelloWorld.Rc
```

Let’s take a look at what files RadASM has created. First, double-click the HelloWorld.Asm file. This opens the code in the main window:



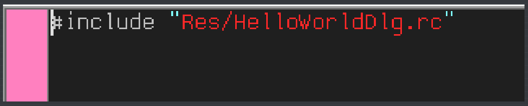
```
.386
.model flat, stdcall  ;32 bit memory model
option casemap :none  ;case sensitive

include HelloWorld.inc

.code

start:

    invoke GetModuleHandle,NULL
    mov     hInstance,eax

    invoke InitCommonControls
    invoke DialogBoxParam,hInstance,IDD_DIALOG1,NULL,addr DlgProc,NULL
    invoke ExitProcess,0

;########################################################################

DlgProc proc hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

    mov     eax,uMsg
    .if eax==WM_INITDIALOG

    .elseif eax==WM_COMMAND

    .elseif eax==WM_CLOSE
        invoke EndDialog,hWin,0
    .else
        mov     eax,FALSE
        ret
    .endif
    mov     eax,TRUE
    ret

DlgProc endp

end start
```

As you can see, RadASM has set up a skeleton app that displays a dialog, and has areas we can enter code for some basic message handlers. Next, let's look at the HelloWorld.Inc file:



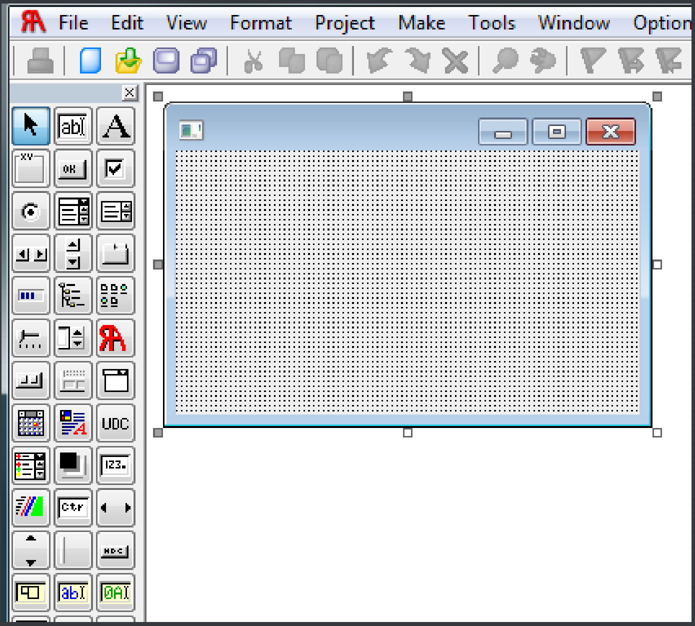
```
include windows.inc
include kernel32.inc
include user32.inc
include Comctl32.inc
include shell32.inc

includelib kernel32.lib
includelib user32.lib
includelib Comctl32.lib
includelib shell32.lib

DlgProc         PROTO   :HWND,:UINT,:WPARAM,:LPARAM

.const

IDD_DIALOG1         equ 101

;#########################################################################

.data?

hInstance           dd ?

;#########################################################################
```

Here we have some basic includes, an ID for the dialog and a dword space set up for the instance handle. Now let's look at HelloWorld.rc (skipping the .dlg file for a moment):



```
#include "Res/HelloWorldDlg.rc"
```

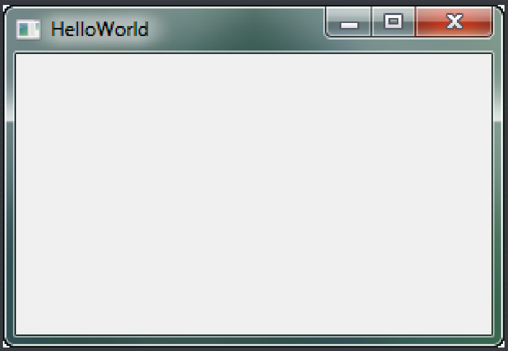
This just contains an include so we can have our .rc file in a folder (Res). Lastly, double-click the HelloWorld.dlg file. This opens the Dialog Editor:



Notice we have a toolbar along the left-hand side. This is for adding elements to our dialog. This screen is also for changing attributes and IDs of our various dialog controls.

Now let's run our new program. You can either build the target in steps (ie. assemble, link, run) or you can do everything in one step. These options are on the "Make" menu drop-down. Clicking "Go" assembles, links and runs the program. "Run" can be selected if you are re-running the app (after it has already been built). Also remember, any time you change the resource file you must rebuild it by selecting "Compile Resource", though clicking "Go" will do this automatically.

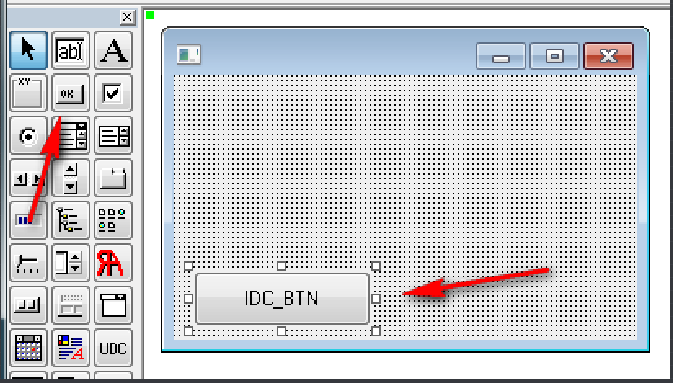
Go ahead and select "Go" or click the appropriate toolbar button. If a dialog pops up asking if you want to accept the command, select the check mark "Don't ask again" and click OK. Now our app's dialog should appear:



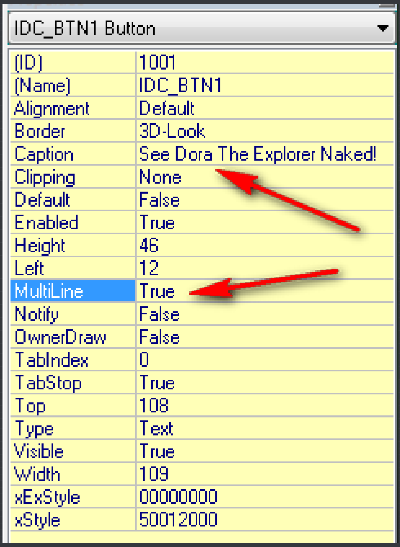
Not a lot to it, though this *is* only a template. Let's spice it up a little…

## Creating Some Controls

Let's spiff up our dialog a little by adding some buttons and bitmaps. First we'll add the buttons. Double-click on the HelloWorld.dlg file to open the dialog editor. Next click on the button tool in the dialog editor toolbar (it says "OK" on it). If for some reason the toolbar is not displayed, click on the little hammer and wrench icon on the main toolbar to display it. Then, click and drag a button in the bottom left-hand corner of the dialog:



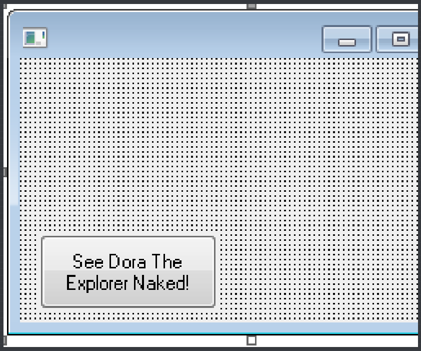
RadASM will automatically name this button IDC_BTN. Now we want to change the attributes of this button, so while the button is highlighted, you will see the attributes in the bottom right of the RadASM window:



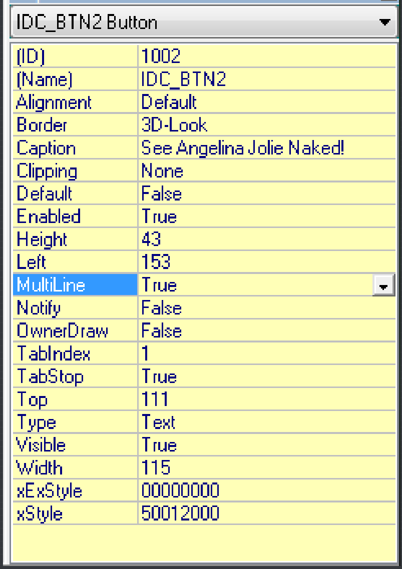
IDC_BTN1 Button

| | |
|---|---|
| (ID) | 1001 |
| (Name) | IDC_BTN1 |
| Alignment | Default |
| Border | 3D-Look |
| Caption | See Dora The Explorer Naked! |
| Clipping | None |
| Default | False |
| Enabled | True |
| Height | 46 |
| Left | 12 |
| MultiLine | True |
| Notify | False |
| OwnerDraw | False |
| TabIndex | 0 |
| TabStop | True |
| Top | 108 |
| Type | Text |
| Visible | True |
| Width | 109 |
| xExStyle | 00000000 |
| xStyle | 50012000 |

I have changed the caption (the text in the button) and have changed the button to display multiple lines. You will see the new caption show up in the button:



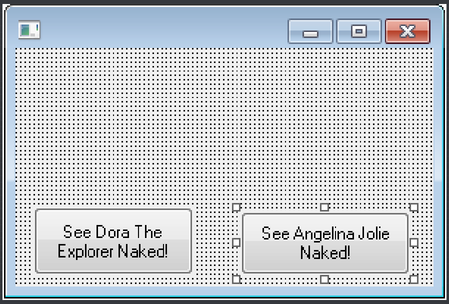
Now add another button on the right-hand side in the same way, setting it’s attributes to the following:



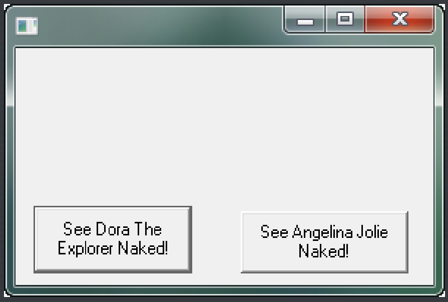
IDC_BTN2 Button

| | |
|---|---|
| (ID) | 1002 |
| (Name) | IDC_BTN2 |
| Alignment | Default |
| Border | 3D-Look |
| Caption | See Angelina Jolie Naked! |
| Clipping | None |
| Default | False |
| Enabled | True |
| Height | 43 |
| Left | 153 |
| MultiLine | True |
| Notify | False |
| OwnerDraw | False |
| TabIndex | 1 |
| TabStop | True |
| Top | 111 |
| Type | Text |
| Visible | True |
| Width | 115 |
| xExStyle | 00000000 |
| xStyle | 50012000 |

and the buttons will update in the dialog editor:

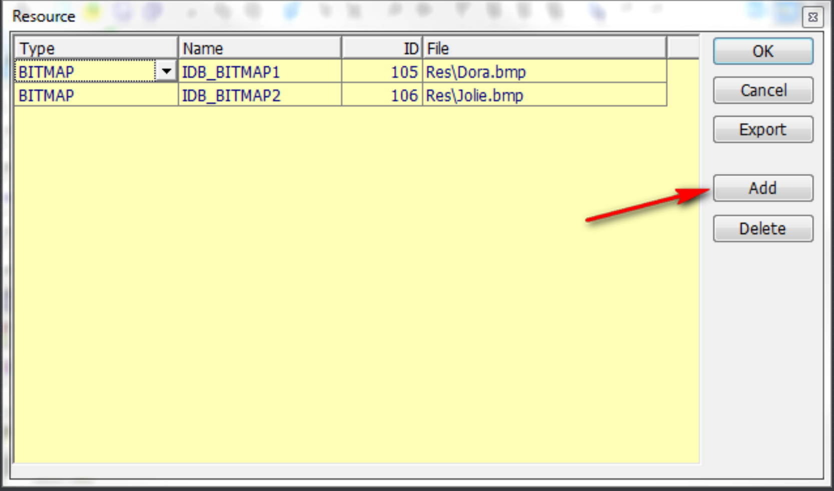


Now if you build and run the app, you will see our new dialog. Of course the buttons won't do anything yet:



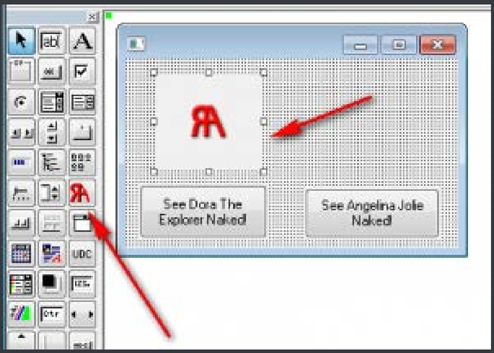
Next let's add some pictures. Like any Windows application, we must load in the bitmaps as resources. Clicking the Resources option in the Project drop-down menu, we will be presented with the resource import screen. To add a new resource, simply click the Add button and fill in the information. When you get to the "file" column, click the three dots and drill to the resource folder and select the appropriate bitmap.

**** You should copy the bitmaps into the Res folder of your project. The bitmaps are available in the download of this tutorial. ****

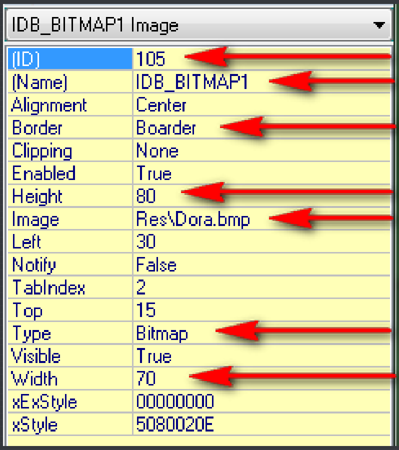


| Type | Name | ID | File |
|---|---|---|---|
| BITMAP | IDB_BITMAP1 | 105 | Res\Dora.bmp |
| BITMAP | IDB_BITMAP2 | 106 | Res\Jolie.bmp |

Notice I named the bitmaps IDB_BITMAP1 and IDB_BITMAP2. We will refer to these IDs later. Now, select the Image tool (the icon with the red “RA” in it in the dialog editor toolbar), then click and drag an image like I have done here:

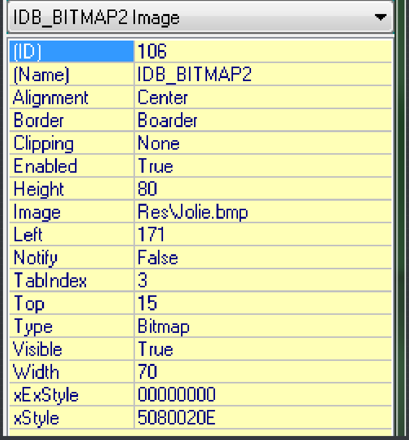


Now we need to change the attributes of this image, pointing it at our resources we loaded earlier. For the ID number, I entered 105. We will refer to this ID in the HelloWorld.Asm file. I changed the name to match the name we gave our first resource in the resource import screen, IDB_BITMAP1. I added a border, changed the width and height to match the dimensions of the actual bitmap, and changed the type from “Icon” to “Bitmap”:


IDB_BITMAP1 Image

| | |
|---|---|
| (ID) | 105 |
| (Name) | IDB_BITMAP1 |
| Alignment | Center |
| Border | Boarder |
| Clipping | None |
| Enabled | True |
| Height | 80 |
| Image | Res\Dora.bmp |
| Left | 30 |
| Notify | False |
| TabIndex | 2 |
| Top | 15 |
| Type | Bitmap |
| Visible | True |
| Width | 70 |
| xExStyle | 00000000 |
| xStyle | 5080020E |

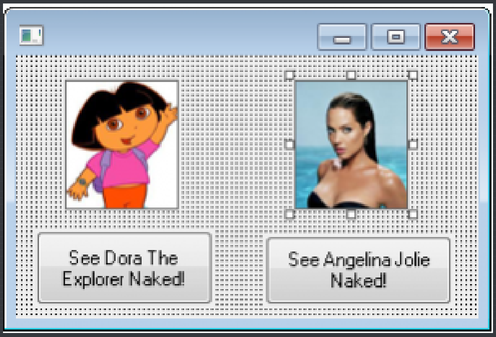
The image should appear in the dialog editor. Now go ahead and add another exactly as we did the first but this time on the above the Angelina Jolie button, naming this one IDB_BITMAP2 and changing the other attributes to match these:



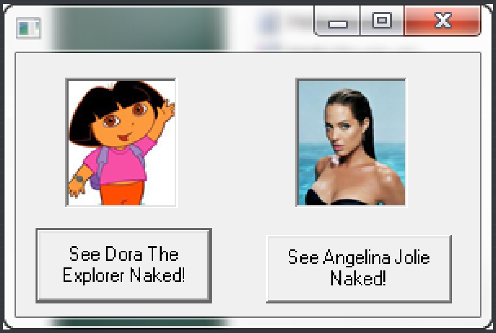
IDB_BITMAP2 Image

| | |
|---|---|
| (ID) | 106 |
| (Name) | IDB_BITMAP2 |
| Alignment | Center |
| Border | Boarder |
| Clipping | None |
| Enabled | True |
| Height | 80 |
| Image | Res\Jolie.bmp |
| Left | 171 |
| Notify | False |
| TabIndex | 3 |
| Top | 15 |
| Type | Bitmap |
| Visible | True |
| Width | 70 |
| xExStyle | 00000000 |
| xStyle | 5080020E |

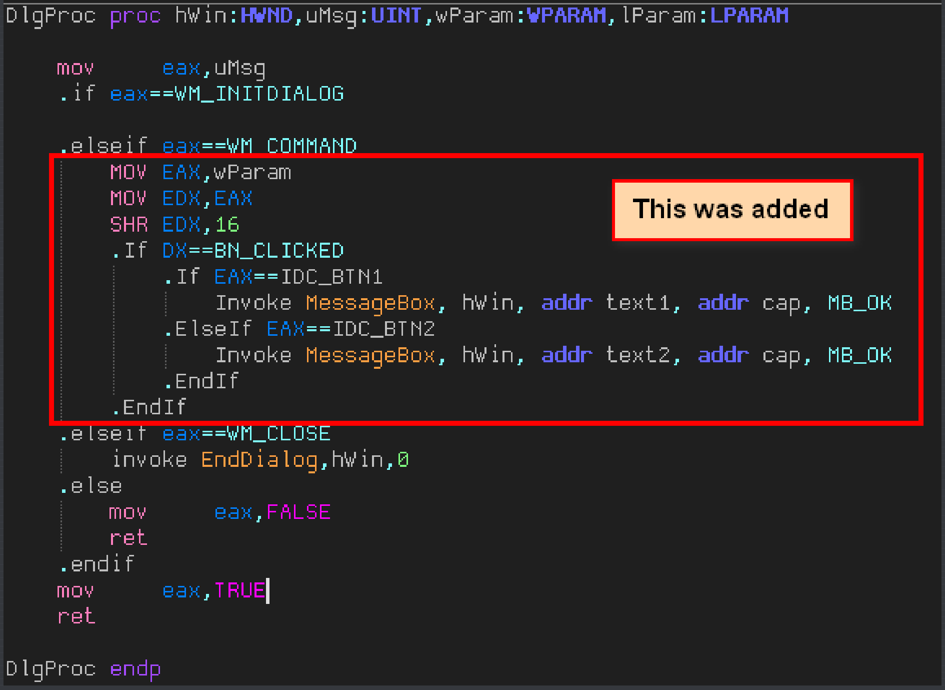
You should see both images, along with our buttons in the dialog editor now:



Building and running the app shows us the fruits of our labors:



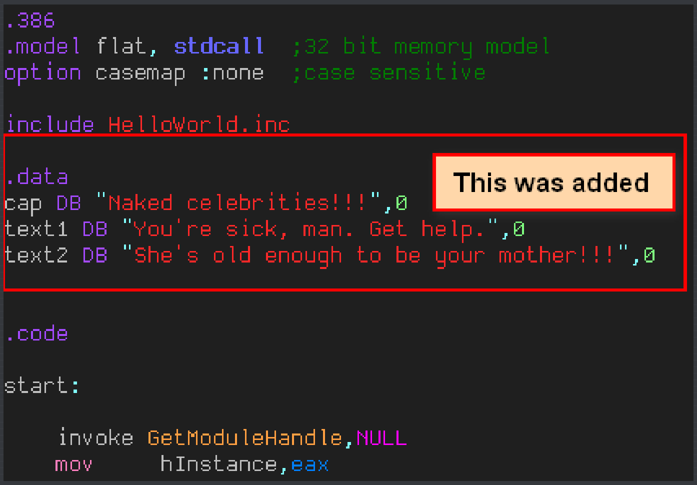
Now we must add the code to handle the events of clicking on the buttons. First, we will add the message handler for the two buttons. Load the HelloWorld.Asm file into the main window. We will make it so clicking on a button will display a message box. I will define the text strings for the message box in the next step:



```
DlgProc proc hWin:HWND,uMsg:UINT,wParam:WPARAM,lParam:LPARAM

    mov     eax,uMsg
    .if eax==WM_INITDIALOG

    .elseif eax==WM_COMMAND
        MOV EAX,wParam
        MOV EDX,EAX
        SHR EDX,16
        .If DX==BN_CLICKED
            .If EAX==IDC_BTN1
                Invoke MessageBox, hWin, addr text1, addr cap, MB_OK
            .ElseIf EAX==IDC_BTN2
                Invoke MessageBox, hWin, addr text2, addr cap, MB_OK
            .EndIf
        .EndIf
    .elseif eax==WM_CLOSE
        invoke EndDialog,hWin,0
    .else
        mov     eax,FALSE
        ret
    .endif
    mov     eax,TRUE
    ret

DlgProc endp
```

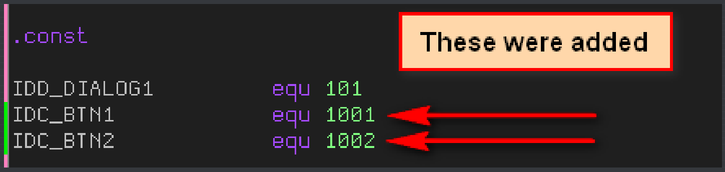
Notice that eax is compared with the ID we gave the buttons, IDC_BTN1 and IDC_BTN2.

Now we can add the text strings to the top of HelloWorld.Asm:



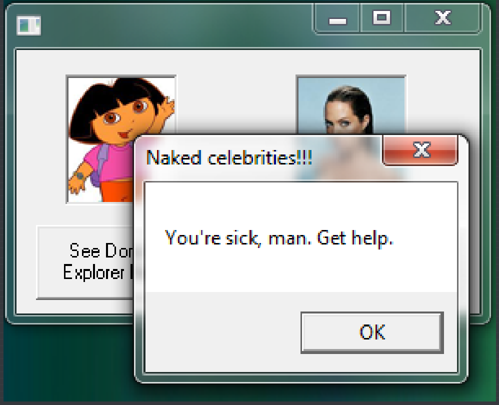
```
.386
.model flat, stdcall  ;32 bit memory model
option casemap :none  ;case sensitive

include HelloWorld.inc

.data
cap DB "Naked celebrities!!!",0
text1 DB "You're sick, man. Get help.",0
text2 DB "She's old enough to be your mother!!!",0

.code

start:

    invoke GetModuleHandle,NULL
    mov     hInstance,eax
```

Finally, we have to add the IDs we created for the buttons in the HelloWorld.Inc file:



```
.const

IDD_DIALOG1         equ 101
IDC_BTN1            equ 1001
IDC_BTN2            equ 1002
```

Running the app now displays our wonderfully compelling and useful app:



Naked celebrities!!!

You're sick, man. Get help.