Win32 API (Application Programming Interface).

Programación 2

MGTI Alejandro Villarreal Mojica

Modal Dialog Boxes

- Un **modal dialog box** es una ventana de diálogo que no se puede cerrar hasta que el usuario la atienda. (Es decir, el usuario no puede cambiarse a otra ventana de la aplicación hasta que el "modal dialog box" haya finalizado).
- Típicamente, una aplicación usará un "modal dialog box", cuando se necesite inmediatamente que el usuario ingrese algún dato, y no pueda continuar hasta que se reciba ese dato.

Modal Dialog Boxes

- Un dialog box es un tipo de ventana, y como tal tiene su propia función CALLBACK Procedure.
- Tal Procedimiento luce muy similara una ventana regular, excepto porque regresa **BOOL**, la cual está predefinida como **int** para Win32 API.
- Declaración típica de un CALLBACK Procedure para un diálogo :

```
BOOL CALLBACK AboutDlgProc(
HWND hDlg, UINT msg, WPARAM wParam,
LPARAM lParam);
```

Modal Dialog Boxes

- A special message that is specific to dialog boxes is the WM_INITDIALOG message.
  - This message is an analog to the WM_CREATE message we handle for non-dialog windows.
- Typically, we handle the WM_INITDIALOG message so that we can initialize the controls of the dialog to default values.
- In addition, the WM_INITDIALOG is a good place to obtain window handles to each control on the dialog (remember, controls are child windows and the dialog is the parent window).

DialogBox function

- After we have implemented a dialog window procedure, we can display a modal dialog with the DialogBox function:

```
DialogBox(
      ghAppInst,  // Application instance.
      MAKEINTRESOURCE(IDD_ABOUTBOX),
                  // Dialog resource ID.
      hWnd,       // Parent window of dialog box.
      AboutDlgProc); // Ptr to dialog box window
procedure.
```

MAKEINTRESOURCE

- Converts an integer value to a resource type compatible with the resource-management functions. This macro is used in place of a string containing the name of the resource.

- Sintaxis:

```
LPTSTR MAKEINTRESOURCE(
  WORD wInteger
);
```

DialogBox function

Note that DialogBox is **used only** for modal dialog boxes.

To close a modal dialog box, we use the EndDialog function.

- This function is used only for modal dialog boxes—modeless dialog boxes are destroyed via another approach.
- By convention, a modal dialog box is typically ended when the user presses the OK or CANCEL buttons of the dialog box.

Modeless Dialog Boxes

- A **modeless dialog box** is a dialog box that behaves like a normal window, but is still a dialog box.
- Unlike modal dialog boxes, the user can switch to other windows as desired.
- Therefore, the modeless dialog box acts like an independent freestanding window. In fact, you can make a modeless dialog box your primary application window.

CreateDialog function

- First, as with modal dialog boxes, modeless dialog boxes are also a kind of window and as such, they too have a window procedure.
- The window procedure is implemented in the same way a modal dialog box procedure is, so we will not discuss it further.
- Let us now examine how modeless dialog boxes are created. To create a modeless dialog box, we use the CreateDialog function instead of the DialogBox function.

ShowWindow function

```
HWND ghDlg = CreateDialog(
      hInstance, // Application instance.
      MAKEINTRESOURCE(IDD_MSGDLG), // Dialog
resource ID.
      0, // Parent window--null for no parent.
      EditDlgProc); // Dialog window procedure.
```

The parameters are similar to that of DialogBox. We note, however, that CreateDialog returns a window handle to the created dialog box window.

After a dialog is created, we need to show it with ShowWindow:

```
// Show the dialog.
ShowWindow(ghDlg, showCmd);
```

Modeless Dialog Boxes

- **Note:** There is a dialog property you can set in the dialog resource editor called “Visible.” If the property “Visible” is set to true then the dialog will be shown by default and a call to ShowWindow is unnecessary. Try it.

DestroyWindow

- To destroy a modeless dialog, we use the DestroyWindow function instead of the EndDialog function:

```
// If the dialog was closed (user pressed 'X' button)
// then terminate the dialog.
case WM_CLOSE:
   DestroyWindow(hDlg);
   return true;
```

IsDialogMessage function

- The final difference between a modeless dialog box and a modal dialog box has to do with the way messages are processed; we must do a little more work with a modeless dialog.

IsDialogMessage function

- In particular, we need to intercept messages aimed for the dialog box from the message loop and forward them to the dialog box procedure. This is done for us with modal dialog boxes. To do this interception we use the IsDialogMessage function.

IsDialogMessage function

- This function returns true if the message is indeed a dialog message, and if it is a dialog message it will also dispatch the message to the dialog's window procedure. Here is the rewritten message loop:

```
while( GetMessage( &msg, 0, 0, 0 ) )
{
    // Is the message a dialog message? If so the function
    // IsDialogMessage will return true and then dispatch
    // the message to the dialog window procedure.
    // Otherwise, we process as the message as normal.
    if( ghDlg == 0 || !IsDialogMessage(ghDlg, &msg ) )
    {
        // Process message as normal.
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
```

IsDialogMessage function

- Essentially, the line

```
IsDialogMessage(ghDlg, &msg )
```

- can be read as “is the message stored in msg aimed for the dialog box referred to by ghDlg?”

- Also note that if the dialog handle is null (ghDlg == 0, which means the dialog was destroyed or has not been created yet) then the ‘if’ statement will be true and the message will be processed normally.

Windows types

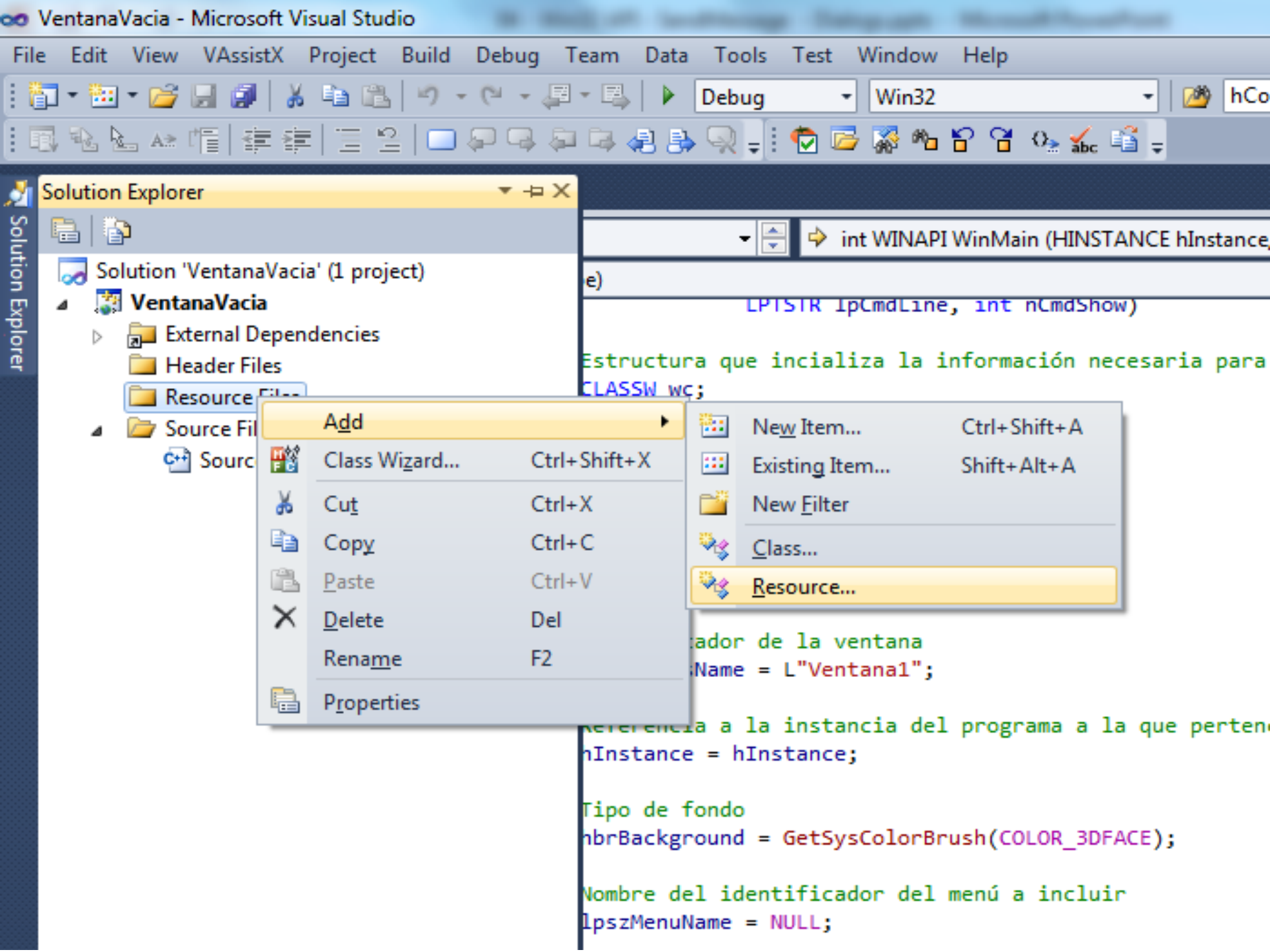
Tipo de Ventana:	Con que se muestra:	Con que se cierra:
Client Window	1. Define a window class 2. Register a window class 3. Create the window 4. Show the window	`DestroyWindow`
Modal Dialog, Modal FormView	DialogBox	EndDialog
Modeless Dialog	`CreateDialog` `ShowWindow`	`DestroyWindow`

FormView (Modeless)	`CreateDialog`
	`ShowWindow`
	UpdateWindow
		`DestroyWindow`

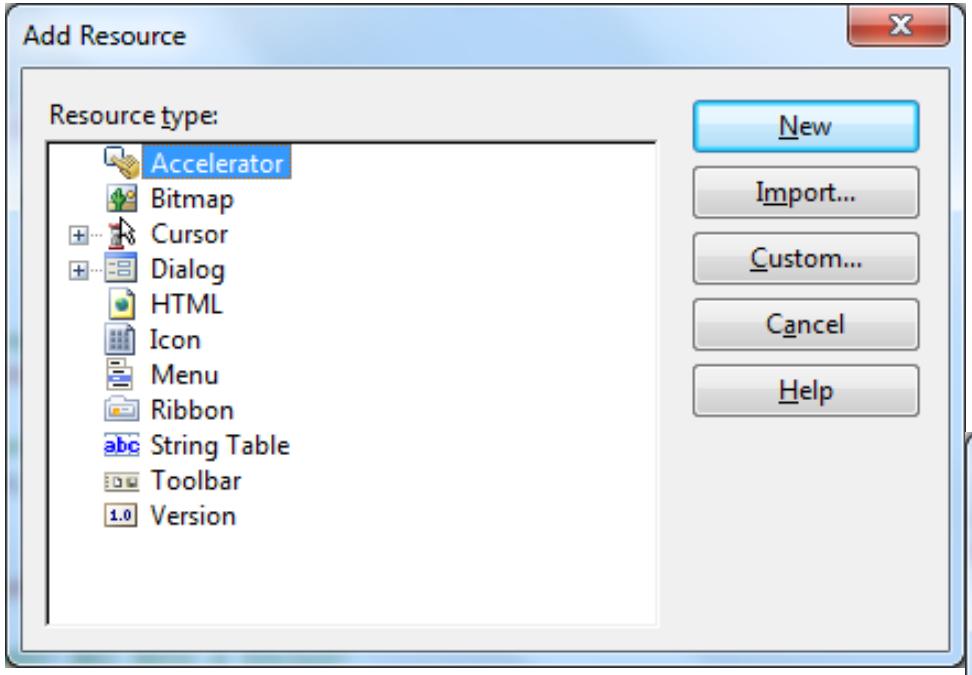
Resources

- Para agregar, crear y diseñar algunos tipos de recursos, como DialogBox, FormView, Menu, Icon, Cursores, Toolbars, etc., se puede hacer en Visual Studio, en la solución, dando clic derecho en la carpeta Resource Files, luego clic en Add, luego en Resource:

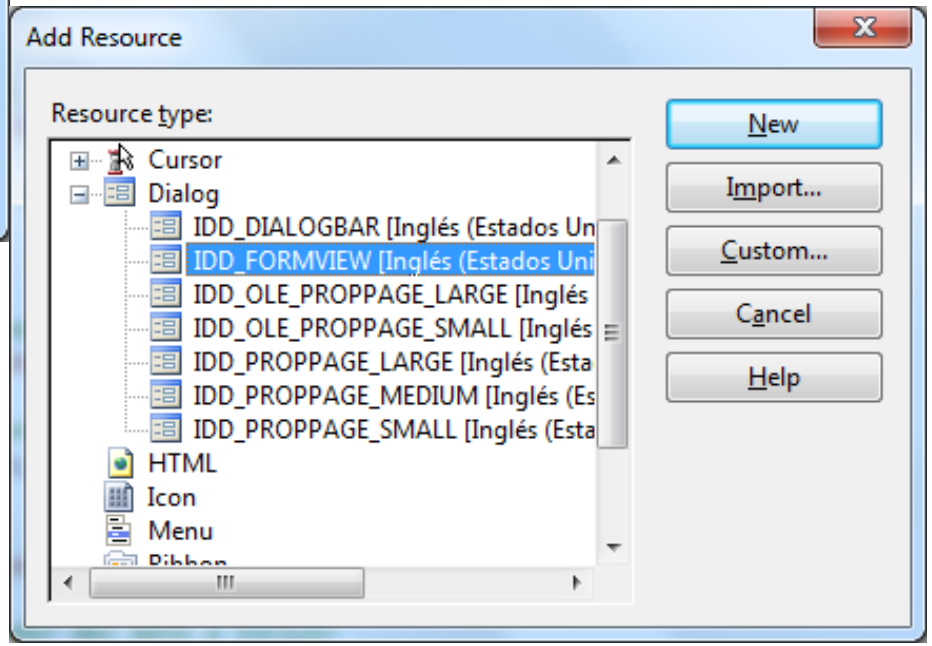
**\ Solution \ Resource Files \ Add \ Resource **


VentanaVacia - Microsoft Visual Studio
File Edit View VAssistX Project Build Debug Team Data Tools Test Window Help
Debug
Win32
Solution Explorer
Solution 'VentanaVacia' (1 project)
VentanaVacia
External Dependencies
Header Files
Resource Files
Source Files
Add
Class Wizard... Ctrl+Shift+X
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Del
Rename F2
Properties
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Filter
Class...
Resource...
int WINAPI WinMain (HINSTANCE hInstance,
LPSTR lpCmdLine, int nCmdShow)
Estructura que incializa la información necesaria para
CLASSW wc;
ador de la ventana
Name = L"Ventana1";
Referencia a la instancia del programa a la que perten
hInstance = hInstance;
Tipo de fondo
hbrBackground = GetSysColorBrush(COLOR_3DFACE);
Nombre del identificador del menú a incluir
lpszMenuName = NULL;

Add Resource



- Se elige el tipo de recurso a agregar

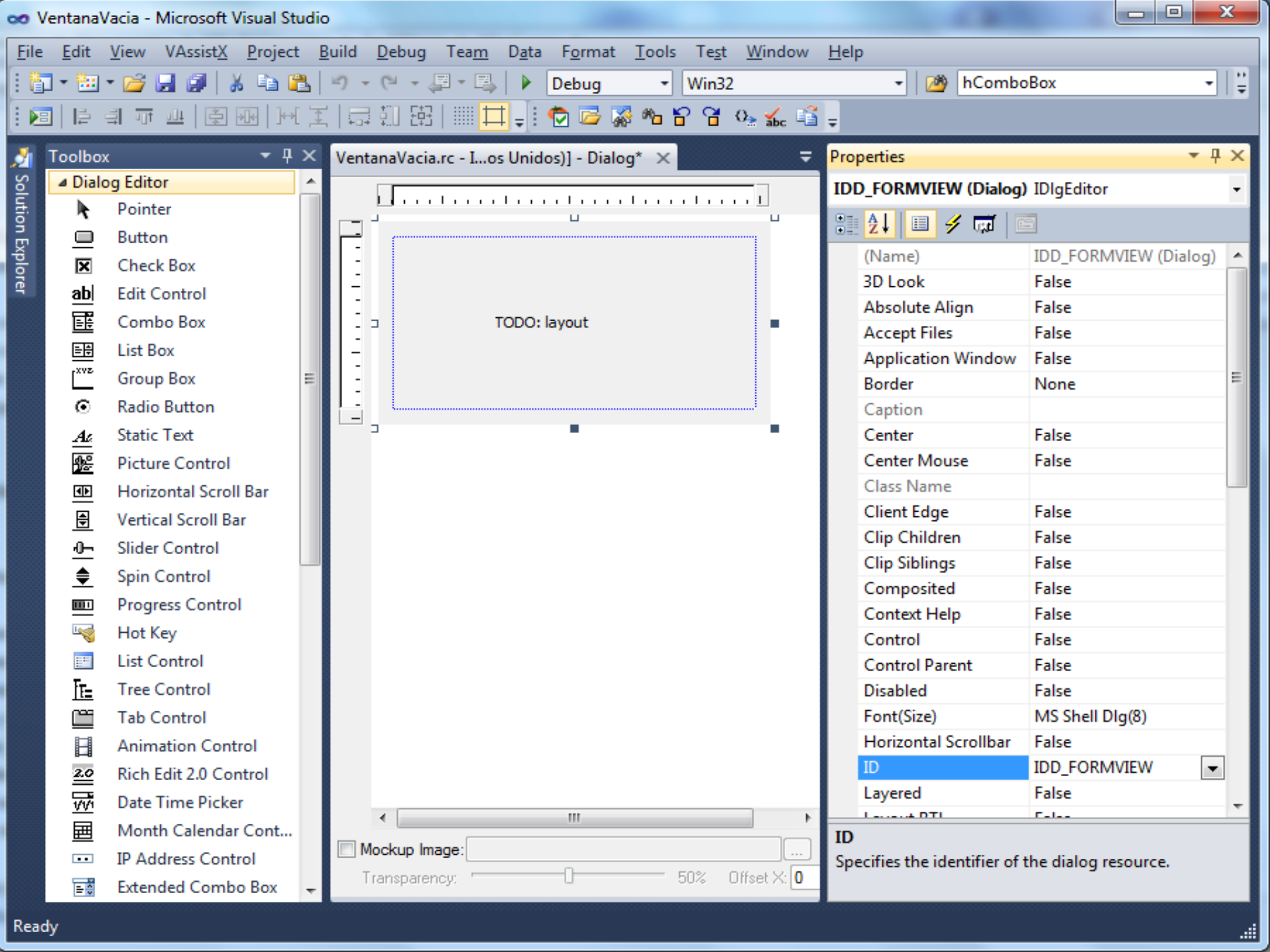


- Clic al botón New, y a diseñar el recurso elegido.

ToolBox - Properties

- En la ventana de ToolBox se eligen los controles a incluir en la forma.

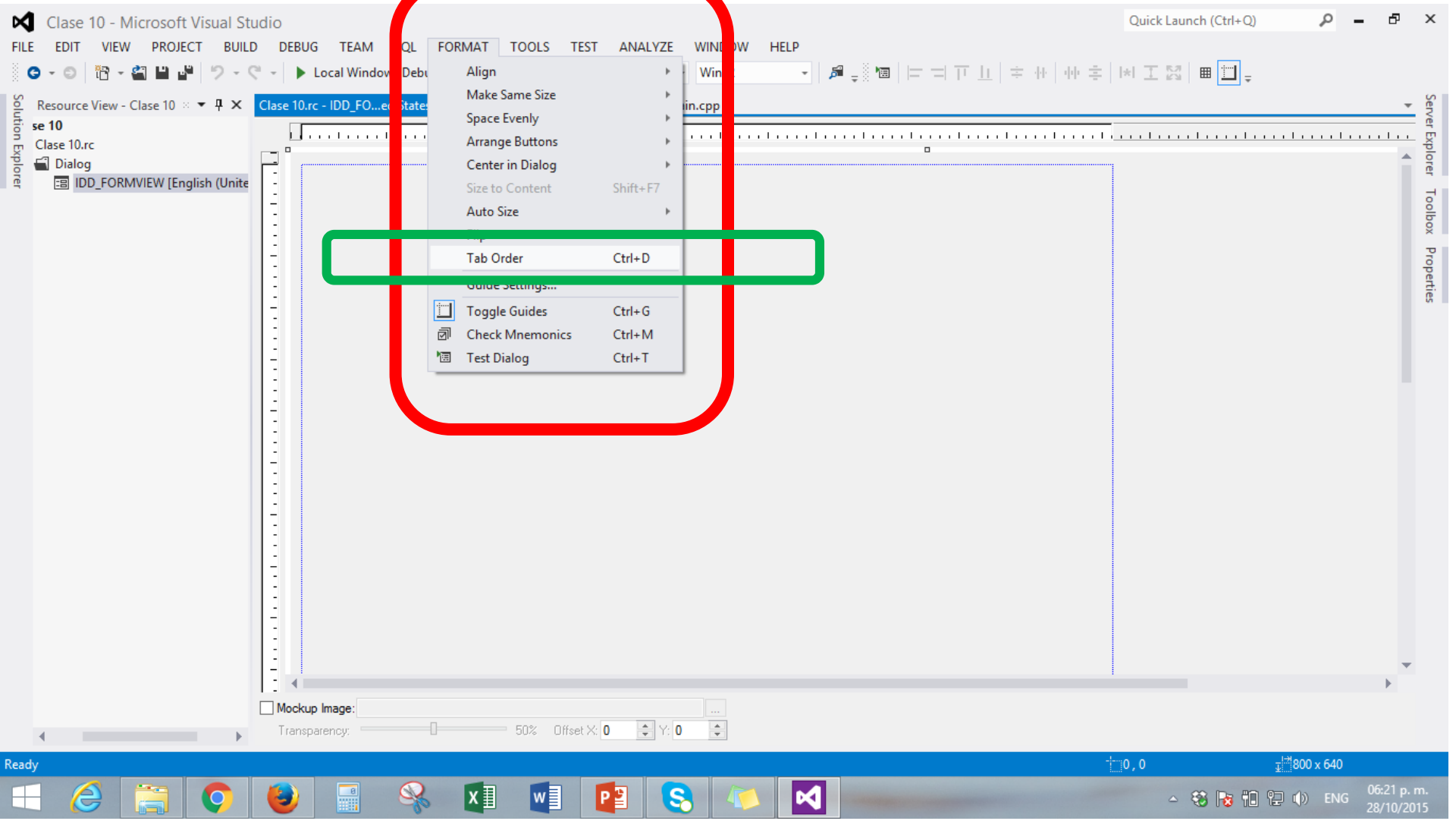
- Y en la ventana de Properties, se pueden cambiar sus propiedades.


VentanaVacia - Microsoft Visual Studio
File Edit View VAssistX Project Build Debug Team Data Format Tools Test Window Help
Debug
Win32
hComboBox
Toolbox
Dialog Editor
Pointer
Button
Check Box
Edit Control
Combo Box
List Box
Group Box
Radio Button
Static Text
Picture Control
Horizontal Scroll Bar
Vertical Scroll Bar
Slider Control
Spin Control
Progress Control
Hot Key
List Control
Tree Control
Tab Control
Animation Control
Rich Edit 2.0 Control
Date Time Picker
Month Calendar Cont...
IP Address Control
Extended Combo Box
Solution Explorer
VentanaVacia.rc - I...os Unidos)] - Dialog*
TODO: layout
Mockup Image:
Transparency:
50%
Offset X:
0
Properties
IDD_FORMVIEW (Dialog) IDlgEditor
(Name) IDD_FORMVIEW (Dialog)
3D Look False
Absolute Align False
Accept Files False
Application Window False
Border None
Caption
Center False
Center Mouse False
Class Name
Client Edge False
Clip Children False
Clip Siblings False
Composited False
Context Help False
Control False
Control Parent False
Disabled False
Font(Size) MS Shell Dlg(8)
Horizontal Scrollbar False
ID IDD_FORMVIEW
Layered False
ID
Specifies the identifier of the dialog resource.
Ready

Orden de objetos - Tab

- En la pantalla de diseño del Diálogo o FormView
- **Para ver el orden actual del tab, para todos los controles en un dialog box:**
  - En el menu **Format**, clic **Tab Order**.
- **Para cambiar el orden de los tab, para todos los controles en un dialog box:**
  - En el menu **Format**, clic **Tab Order**.
  - Un número en la esquina superior izquierda de cada control muestra que lugar corresponde en el orden actual del tab.
  - Para establecer el orden a seguir por el tab, de clic en cada control, siguiendo el orden que desea que la Tecla TAB tenga al mostrar el diálogo.
  - Presione **ENTER** para salir del modo **Tab Order**.

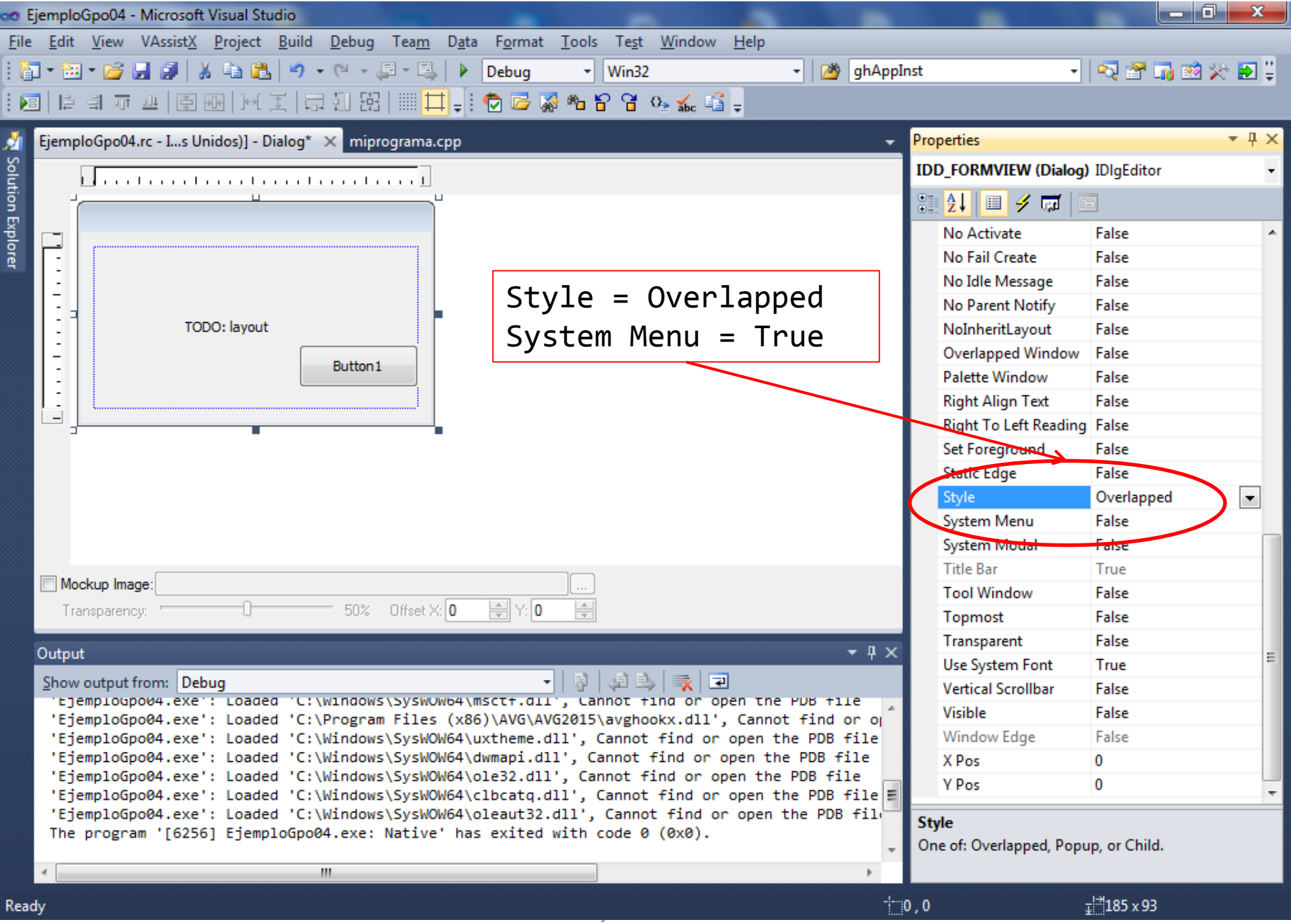
https://msdn.microsoft.com/en-us/library/csz6b8x8.aspx



MGTI Alejandro Villarreal --> alvillarreal@gmail.com

SET FOCUS

- La función **SetFocus** asigna el foco del teclado a la ventana especificada.
- Todas las entradas desde el teclado a partir de ese momento se dirigen a esa ventana.
- La ventana que tenía el foco del teclado previamente, si es que había alguna, lo pierde.



alvillarreal@gmail.com

Poner un ícono en el Diálogo

- Es una tarea simple, solo necesitamos enviar a nuestro diálogo el mensaje WM_SETICON.
- Sin embargo, debido a que Windows utiliza dos íconos, necesitamos enviar el mensaje dos veces:
  - una vez para el ícono pequeño mostrado en la esquina de la ventana
  - y otra vez para el ícono grande mostrado cuando presionamos Alt+Tab.
- Podemos enviar el mismo handle ambas veces, a menos que tengamos iconos de diferentes tamaños.

Poner un ícono en el Diálogo

- Para poner el icono de aplicación por default, podemos usar el siguiente código:

```
SendMessage(hwnd, WM_SETICON, ICON_SMALL,
(LPARAM)LoadIcon(NULL,
MAKEINTRESOURCE(IDI_APPLICATION)));
SendMessage(hwnd, WM_SETICON, ICON_BIG,
(LPARAM)LoadIcon(NULL,
MAKEINTRESOURCE(IDI_APPLICATION)));
```

- Cuando reemplaces tu propio ícono por uno por default, recuerda cambiar el parámetro HINSTANCE de LoadIcon( ) por el de la instancia de tu aplicación (si no lo tienes almacenado en WinMain( ), puedes obtener éste llamando a GetModuleHandle( )).

Íconos Predefinidos

- ✓IDI_APPLICATION
- ✓IDI_HAND
- ✓IDI_QUESTION
- ✓IDI_EXCLAMATION
- ✓IDI_ASTERISK
- ✓IDI_WINLOGO
- ✓IDI_SHIELD