Petit guide pour la création d'une application Windows simple sous l'environnement Visual C++.

Les bases d'une application Windows

1. Création d'une boîte de dialogue

Partie théorique :

Avant de commencer à programmer des applications Windows, il est essentiel de bien comprendre le principe de leurs fonctionnements. Ce sera le but de ce petit guide que d'exposer le principe global de fonctionnement d'une application créant une boite de dialogue simple avec deux boutons.

Tout d'abord, il est important de bien garder à l'esprit qu'une application fonctionnant sous Windows et gérant une fenêtre doit rester en dialogue constant avec Windows. La dite application ne connaît (à priori) rien sur son environnement, c'est Windows qui lui signale si elle doit redessiner le contenu d'une de ses fenêtres, ou encore si l'utilisateur essaie de les fermer.

Cette communication se fait au travers de messages que Windows envoie à chaque fenêtre concernée. C'est à l'application d'effectuer la réception de ces messages et de les transmettre aux fonctions gérant les différentes fenêtres. La réception de ces messages ne doit donc pas prendre de retard, et le traitement de chacun des messages doit être bref.

En cas de retard, le redessinement des fenêtres n'est plus assuré, ce qui a pour conséquence des fenêtres blanches, indéplaçables, similaires à celles des programmes "plantés".

Chaque fenêtre est associée à une fonction ou procédure de fenêtre (Window Proc). Parfois plusieurs fenêtres peuvent être associées à une même procédure. Chaque message reçu est transmis à la procédure correspondante qui se chargera de traiter ce message (redessiner la fenêtre, la redimensionner, afficher un caractère entré par l'utilisateur...etc.).

Une partie du travail de rafraîchissement de la fenêtre est prise en charge par Windows. L'application n'a à redessiner que la zone client de sa fenêtre (et non pas la barre de titre, les menus éventuels...).

Partie pratique :

- **But :** créer une boîte de dialogue et une procédure simple chargée de la gérer.

- **Création :** la boîte de dialogue sera créée grâce à l'éditeur de ressources de VC++. Le programme sera chargé de la réception des messages et du traitement des plus simples.

Tout d'abord, il s'agit de créer le projet dans VC++ : pour cela, on ouvre Visual Studio 6 C++ et on clique sur **File/New/Projects/Win32 Application** (voir Fig. 1).

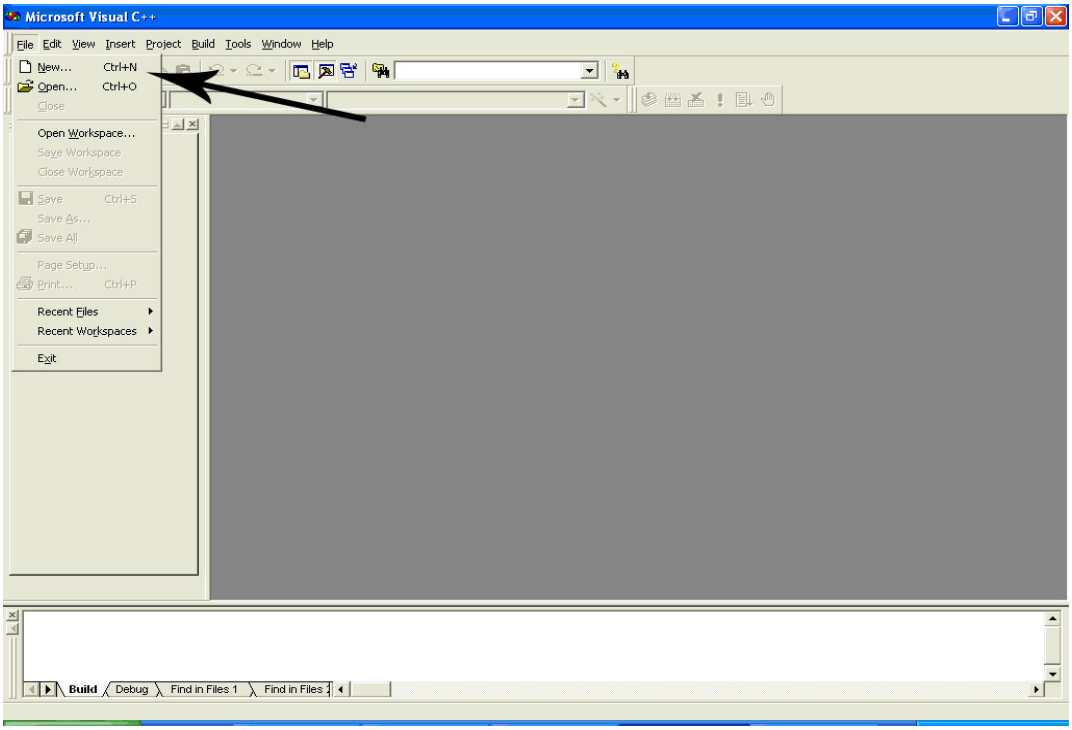

Fig. 1

On donne ensuite un nom au projet et on clique sur 'Ok'. Vous remarquerez qu'un répertoire du même nom que votre projet est créé dans votre répertoire courant de travail.

Ensuite, il faut sélectionner 'An empty project' puis 'Finish' et valider par 'Ok'.

Nous venons de créer un projet vide, il ne contient aucun fichier de code (Fig. 2, 3 et 4).

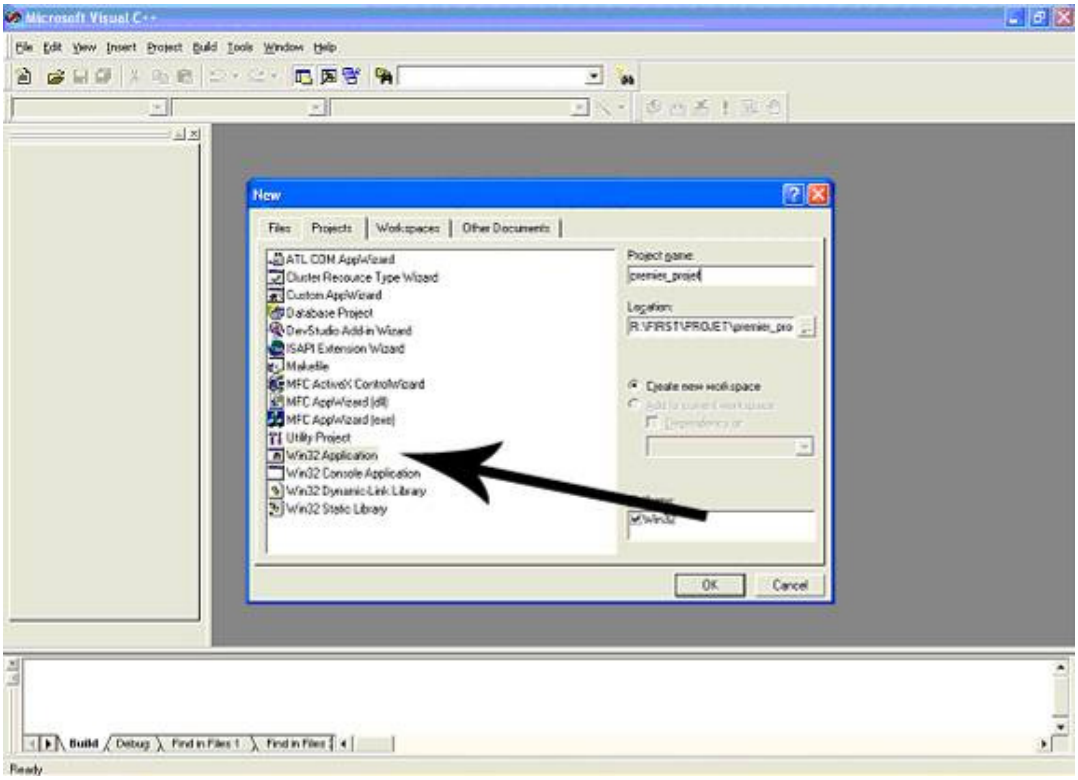

Fig. 2

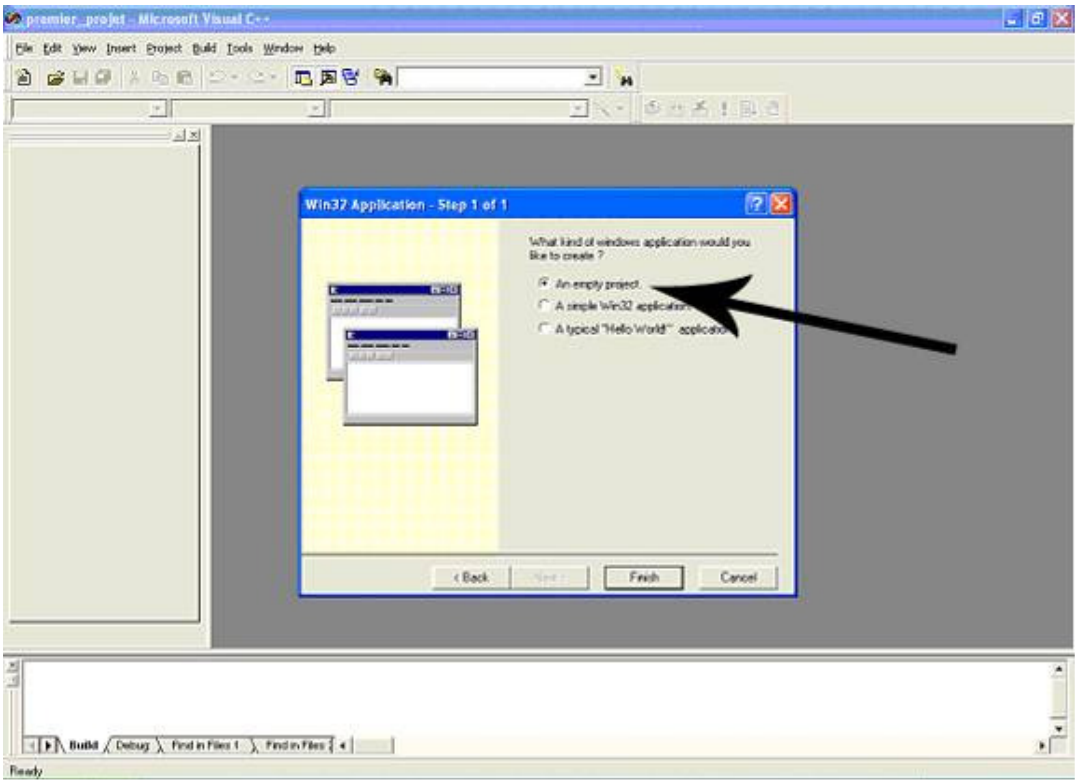

Fig.3

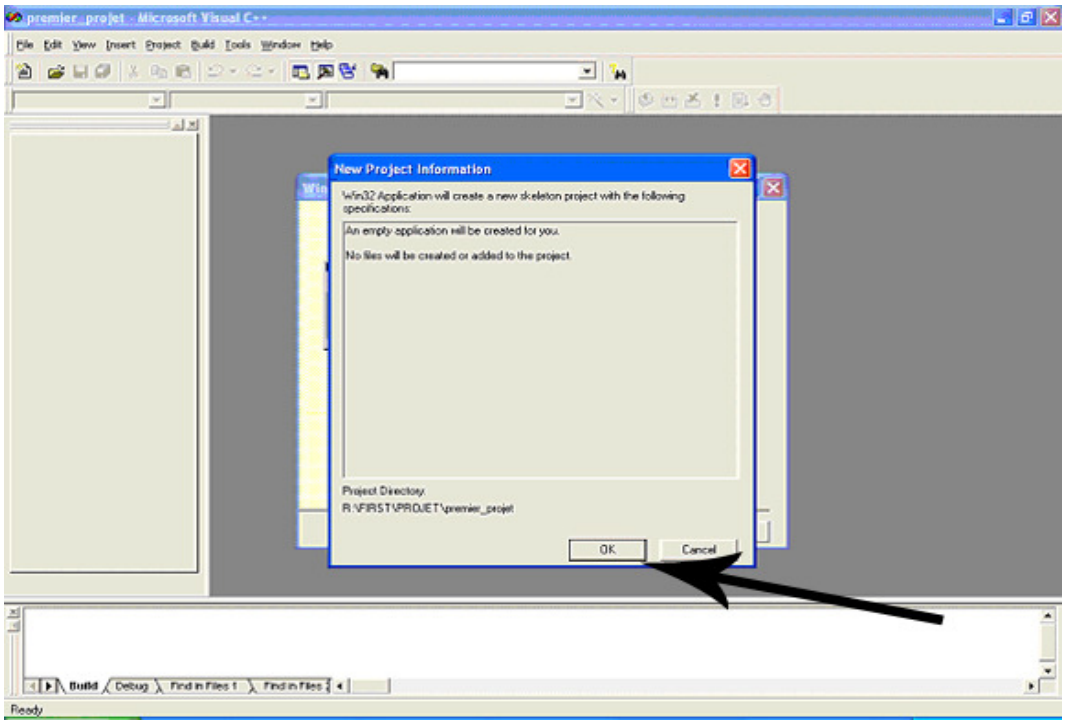

Fig.4

Ajoutons maintenant les fichiers dont on aura besoin, à savoir le fichier source du code, et celui de ressources. Pour cela, dans le feuillet **FileView** du panneau de gauche, avec le bouton droit de la souris on clique sur **Source Files** et on sélectionne **Add Files to Folder** (Fig.5), on donne un nom au nouveau fichier et on valide. De la même façon on ajoute un fichier de type **Ressource Script**, fichier qui sera ouvert dans l'éditeur correspondant. Ce script de ressources est associé à un fichier d'en-tête créé et mis à jour automatiquement, mais qu'il convient d'ajouter au projet. Aussi, dans le feuillet **FileView** du panneau de gauche, il faut faire un clic droit sur **Header Files** et sélectionner **Add Files to Folder**, et on sélectionne le fichier 'resource.h' qui doit se trouver dans le même répertoire que le projet.

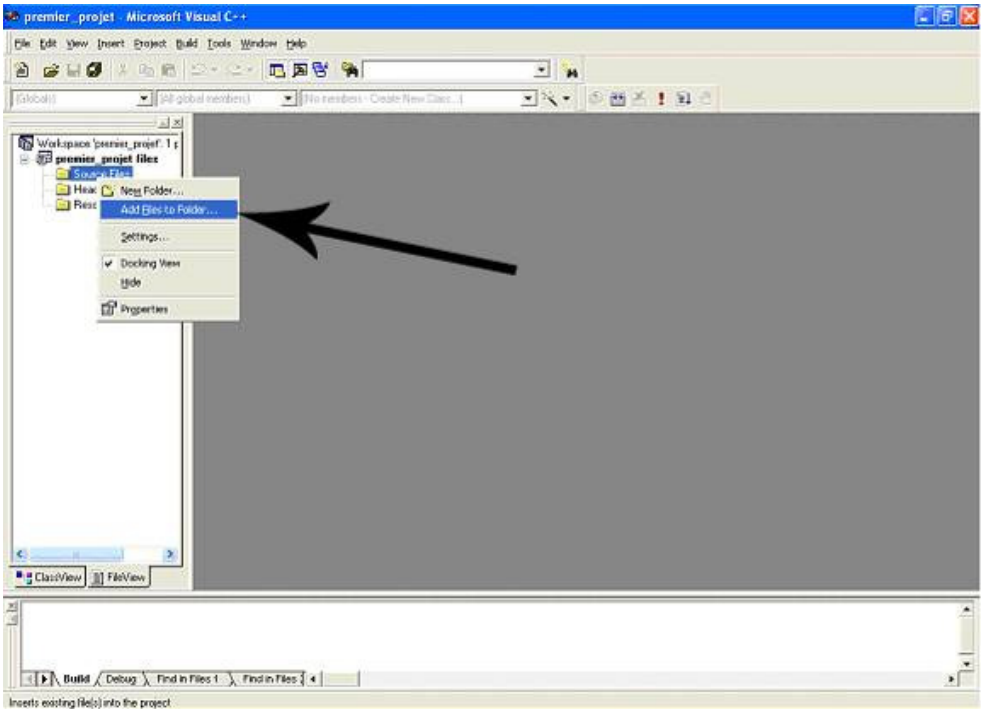

Fig. 5

Créons à présent le modèle de notre boîte de dialogue. Dans le menu du VC++ du panneau de travail (en haut), on clique sur **Insert** et on sélectionne **Resources.** Une boite de choix (**Insert Resource**) s'ouvre qui nous propose toutes les ressources disponibles, dans notre cas, on choisit **Dialog.** Boite de dialogue que nous ne modifierons pas dans ce projet (Fig. 6 et 7).

Apres la validation par **New,** la nouvelle boite de dialogue est créée avec l'interface suivant par défaut (Fig. 8).

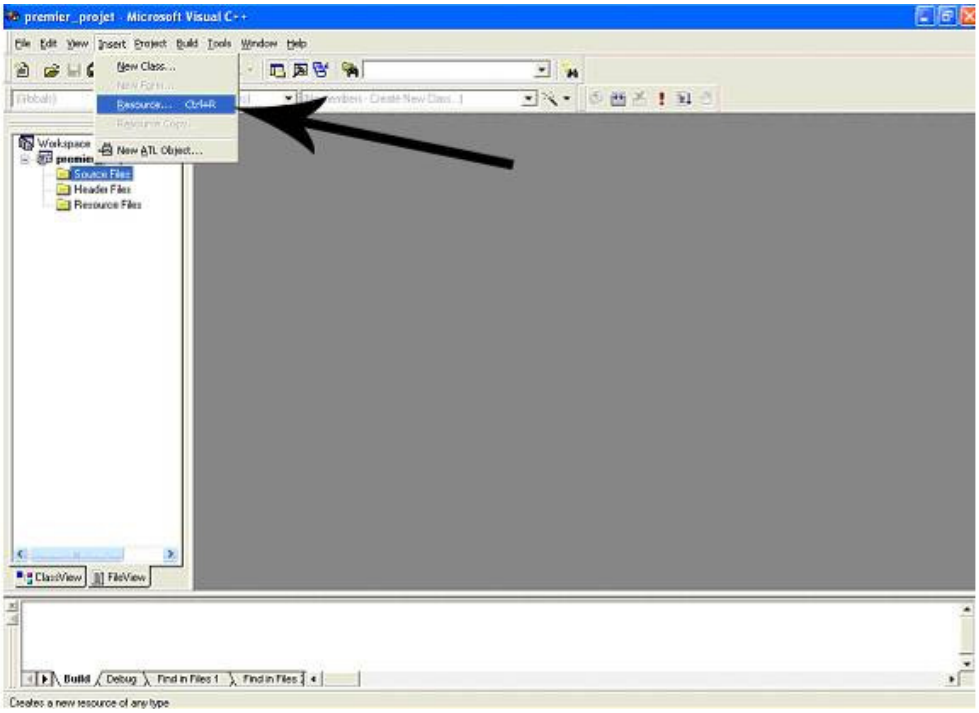
premier_projet - Microsoft Visual C++
File Edit View Insert Project Build Tools Window Help
New Class...
Resource... Ctrl+R
New ATL Object...
Source Files
Header Files
Resource Files
ClassView
FileView
Build
Debug
Find in Files 1
Find in Files 2
Creates a new resource of any type

Fig. 6

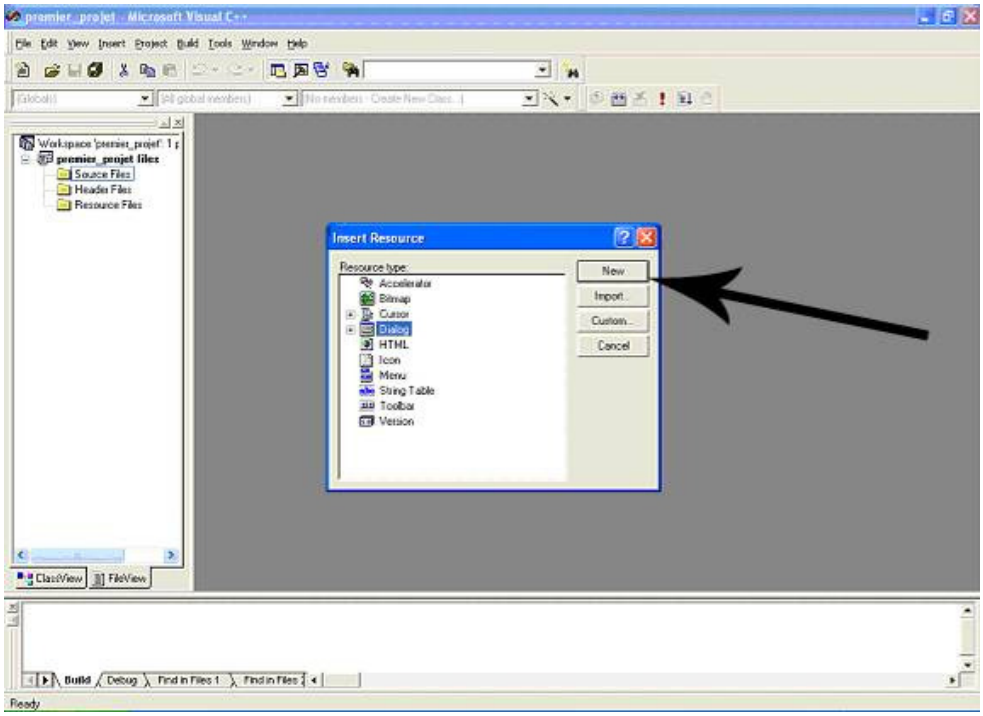
premier_projet - Microsoft Visual C++
File Edit View Insert Project Build Tools Window Help
premier_projet files
Source Files
Header Files
Resource Files
Insert Resource
Resource type:
Accelerator
Bitmap
Cursor
Dialog
HTML
Icon
Menu
String Table
Toolbar
Version
New
Import...
Custom...
Cancel
ClassView
FileView
Build
Debug
Find in Files 1
Find in Files 2
Ready

Fig. 7

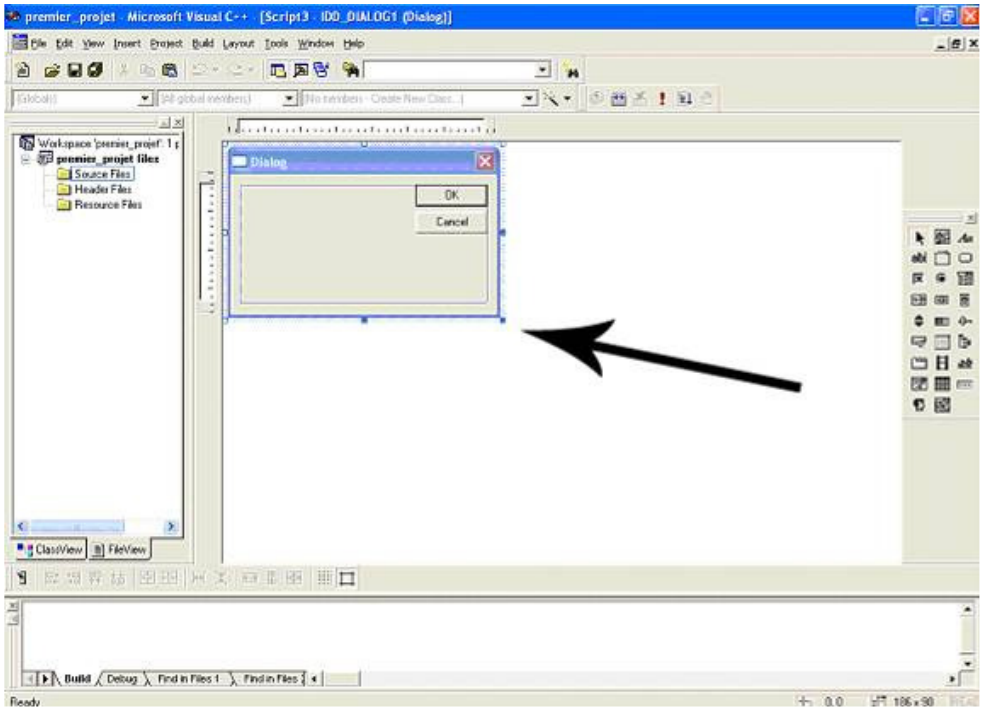

Fig. 8

Pour éditer les propriétés des champs créés, il suffit de double-cliquer dessus, pour qu'une fenêtre s'ouvre (Fig.9).Vous pouvez alors modifier ces propriétés pour les adapter à vos besoins.

Il est important de noter qu'à chaque champ de notre boite de dialogue, incluant la boite elle-même est identifiée par un identificateur unique. L'application accèdera à chaque composant de notre interface par le biais de cet identificateur.Nous vous conseillons de modifier les identificateurs générés par défaut par Visual C++, et leur assigner des noms significatifs.

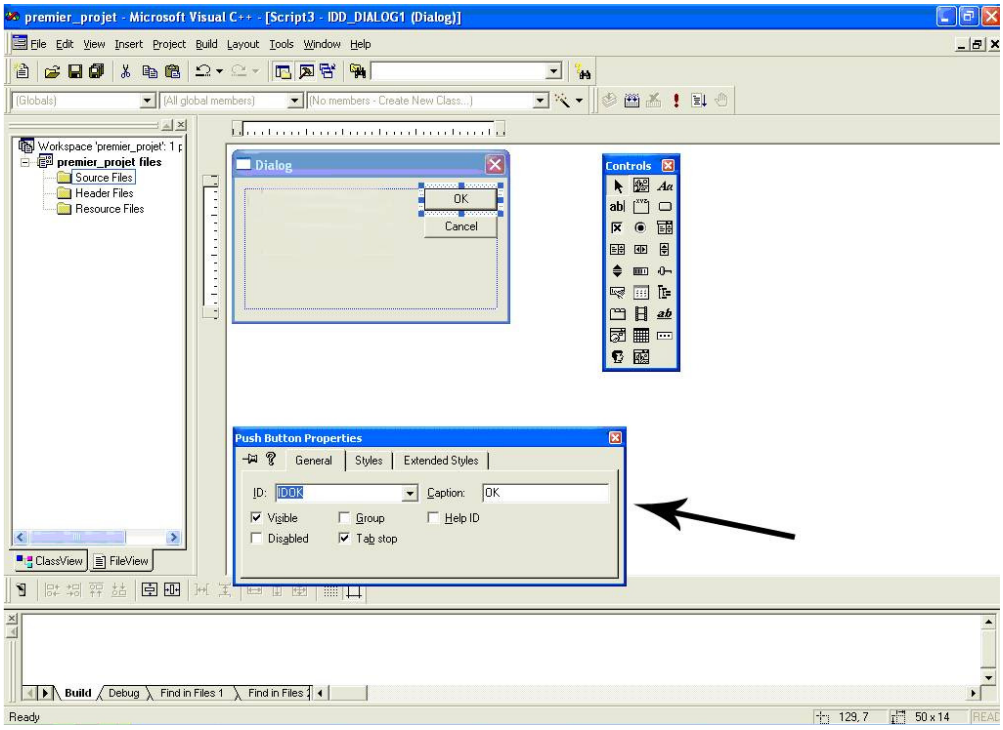

Fig. 9

Apres avoir crée la partie visuelle, il faut s'occuper maintenant du code qui va constituer le programme.

Tout d'abord, les fichiers d'en-tête, deux sont nécessaires, ***Windows.h*** inclus automatiquement dans ***resource.h*** qui lui, par contre, doit se faire par le programmeur.

Les informations sur les composants ajoutés à l'interface graphique sont stockées dans un fichier de ressources créé automatiquement nommé ***resource.rc.***

Maintenant, il ne reste plus qu'à écrire le code source en C++ pour donner à nos composants le comportement voulu.

Le point d'entrée de toute application windows est la fonction WinMain(). Elle est exécutée automatiquement par Windows lorsque l'application est démarrée. Pour gérer le comportement des boites de dialogue et des fenêtres créées, celle-ci a recours à des fonctions de rappel (Mot clé CALLBACK).

Dans l'exemple suivant, on définit la fonction principale WinMain(), et une seule fonction de rappel MainProc() .

```
#include <Windows.h>
#include "resource.h"

LRESULT CALLBACK MainProc(HWND Dlg,UINT message,WPARAM wParam,LPARAM lParam);

/*On va, maintenant, créer la boîte de dialogue grâce à la fonction CreateDialog(). Le paramètre
HINSTANCE est un identifiant de notre application, on passe l'identifiant fourni par Windows en
paramètre de la fonction WinMain(). On donne ensuite l'identifiant ressource de notre boîte de dialogue.
Elle n'a pas de fenêtre parent et la procédure est la fonction MainProc(). La fonction retourne une variable
```

de type HWND qui identifie notre fenêtre. Puis, on affiche cette fenêtre, toute fenêtre créée est invisible par défaut.*/

```
int APIENTRY WinMain(HINSTANCE hInstance,
                                HINSTANCE  hPrevInstance,
                               LPSTR lpCmdLine,
                               intnCmdShow)
{
  HWND hDlg =CreateDialog(hInstance,(LPCTSTR)IDD_DIALOG1,NULL,(DLGPROC)MainProc);
 ShowWindow(hDlg,SW_SHOW);
```

/*Occupons-nous maintenant de la réception des messages. La réception des messages est prise en charge par la fonction GetMessage(). Cette fonction retourne FALSE dès que le message WM_QUIT est reçu, ce qui indique que l'application doit être fermée.*/

```
MSG msg;
while(GetMessage(&msg,NULL,0,0)==TRUE)
{
        TranslateMessage(&msg);
       DispatchMessage(&msg);
}
return 0;
```

/*Il ne reste plus qu'à définir le comportement de notre fenêtre dans quelques cas simples: elle doit être fermée et l'application quittée si l'utilisateur presse 'ok' ou essaie de quitter. La fermeture de la fenêtre ne signifie pas forcément l'arrêt de l'application. Ces deux comportements correspondent à deux messages distincts: WM_QUIT indique que l'application doit se terminer, WM_CLOSE indique la fermeture de la fenêtre. Mais dans ce cas, on a une fenêtre un peu particulière, puisqu'il s'agit d'une boîte de dialogue. Sa procédure est définie de la même manière que celle d'une fenêtre.*/

```
LRESULT CALLBACK MainProc(HWND Dlg,UINT message,WPARAM wParam,LPARAM lParam)
  {
int Select;
   switch(message)
{
            case WM_COMMAND:
                   Select=LOWORD(wParam);
                    switch(Select)
{
            case IDOK:
                  EndDialog(Dlg,0);
                  PostQuitMessage(0);
                 return TRUE;

           case IDCANCEL:
                  EndDialog(Dlg,Select);
             PostQuitMessage(0);
            return TRUE;
}
default:
return FALSE;
}
}
```

/*Cette procédure est très simple, elle ne réagit qu'à la pression des boutons 'Ok' et 'Cancel' (le bouton ou la "croix"). En réponse à l'un ou l'autre de ces événements, la procédure demande la fermeture de la boîte de dialogue et termine l'application en envoyant un message WM_QUIT grâce à la fonction PostQuitMessage().

Ainsi, nous avons vu comment créer une boite de dialogue simple, cette création, obtenue grâce à la fonction CreateDialog(), est relativement simple puisqu'une grande partie du travail est effectué par la dite méthode CreateDialog().*/