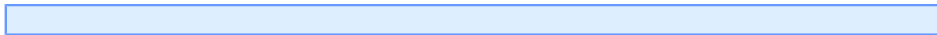


Tutoriel 20 : Convertir des programmes VB3 pour VB6.0

par Gilbert Miralles (gilmir.developpez.com)

Date de publication : Lundi 18 mars 2003

Dernière mise à jour : Lundi 4 février 2008



I - C'est possible !

II - L'indispensable, c'est quoi ?

Prochain tutoriel :

I - C'est possible !

Il est en effet possible de convertir des programmes écrits sous une version inférieure de VB en programme pouvant être lancés avec la version de VB 6.0

Il est vrai que si vous ne prenez pas certaines précautions au préalable la conversion risque de ne pas fonctionner et laisser perplexe l'utilisateur néophyte en la matière.

Pour pouvoir réaliser la conversion, il est indispensable de posséder la version dans laquelle a été écrit le programme.

Si nous devons convertir un programme écrit avec la version de VB 3.0, il est indispensable d'en posséder la version.

Partons du principe que VB6.0 ne veut pas récupérer les fichiers des versions inférieures tout simplement parce que écrits bien souvent sous une plate-forme 16 bits la plupart des fichiers ne peuvent être lus par VB6 car les applications générées sont exclusivement écrites pour du 32 bits, donc incompatibles avec les précédentes versions.

Certains programmes font également appels à des fichiers avec extensions "*.vbX", ou bien des fichiers avec extensions "*.dll" et ne pourront pas être lus par VB6.0

Je vous donne comme exemple, un programme qui a été écrit avec un fichier "Threed.vbx" ne pourra pas être lus par VB6

Un programme faisant appel à une "API" de Windows du genre "Kernel.dll" ne pourra pas fonctionner sous VB6.

Vous allez me dire que tout cela est bien compliqué ?

Et bien non il suffit d'avoir le réflexe logique et partir du principe que nous allons éliminer tout ce qui nous gêne et ne garder que l'indispensable.

II - L'indispensable, c'est quoi ?

L'indispensable c'est tout simplement le code qui génère les événements nécessaires au fonctionnement de l'application.

Je crée un nouveau projet sous VB6.0 et je retire tous les fichiers qui composent ce projet, je ne peux pas me tromper il n'y en a qu'un, c'est le fichier "Form1", je le retire en sélectionnant la "Form" avec le bouton droit de ma souris, et en cliquant sur l'étiquette "**Supprimer Form1**".

Je lance le projet VB3.0 et je regarde dans la fenêtre de projets de l'application à transformer les contrôles qui sont affichés.

Si je constate que celui-ci affiche par exemple le contrôle "**Threed.vbx**" je sais par avance que je vais générer un problème, aussi pour éviter de générer une erreur si je le transfère, je prends au préalable la précaution d'éliminer dans la feuille "Form" tous les objets qui font référence à ce contrôle.

Par exemple si les boutons sont des boutons 3D, donc qui ont une relation avec ce contrôle, j'insère dans la feuille le nombre de boutons standards nécessaire au fonctionnement de l'application, et une fois positionnés sur la feuille, je transfère les lignes de codes, procédure par procédure, en utilisant le "**Couper/Coller**" de mon clavier.

J'ai bien dit "**Couper/Coller**" : ceci pour éviter d'engendrer une erreur de VB qui pourrait se rendre compte qu'il y a deux procédures identiques, et qui vous le signalerait automatiquement.

Lorsque le transfert est terminé, j'efface les boutons 3D.

Si notre application comprend des "Frames" 3D, VB6.0 ne les digérant pas, il faut également les éliminer en prenant la précaution de ne pas éliminer les éléments positionnés et qui sont indispensables.

Ce n'est pas facile, mais on y arrive avec un peu de patience et de dextérité.

En ce qui concerne les modules, le système d'écriture étant différant d'une version à une autre il est impératif de récupérer les codes de chaque procédures et de les transférer directement dans le module de VB6.0

On n'insère pas le module d'une version inférieure à VB6.0, on ne fait que transférer les lignes de code, **donc, les procédures contenues dans le module.**

Si votre application fait appel à une DLL, contrôler s'il existe une DLL équivalent écrite pour des applications 32 bits. Si elle n'existe pas, alors chercher une autre application à convertir.

Attention, il arrive que parfois les instructions afférentes à une version différent dans une autre version. Dans ce cas, essayer de récupérer le fichier des "API" Windows qui est un fichier Texte donc à extension *.txt et rechercher la procédure qui vous intéresse afin d'appliquer les nouvelles instructions

Lorsque Windows95 est passé dans un environnement 32 bits, les noms de fichiers des DLL standards ont été modifiés. Pour distinguer les nouveaux fichiers des anciens, Microsoft a ajouté 32 dans leur nom (par exemple: GDI32.DLL, SPIN32, etc...).

Si votre programme fait appel sous VB3.0 à une DLL qui s'appelle pour l'exemple, "**Kernel.dll**", regardez dans le répertoire de "Windows/System" si vous n'avez pas la "DLL" du nom de "**Kernel32.dll**" qui est la version identique, mais, pour des applications 32 bits.

Lorsque vous avez appliqué les directives indiquées, vous pouvez transférer les fichiers "**Form**", c'est à dire les feuilles qui composent votre application.

Téléchargez l'application **Econom16.zip** qui est un programme écrit sous Visual Basic 3.0, et procédez aux modifications indiquées pour le transformer en une application 32 bits.

Dans le projet "Economis.mak" Vous devez avoir deux fichiers, la feuille "Economis.frm" et la feuille frmParams.

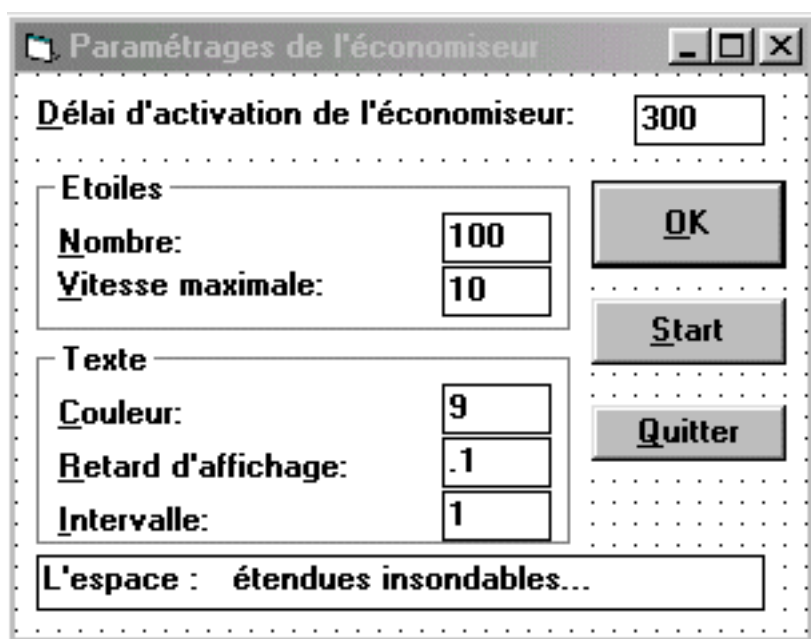
Lorsque tous les fichiers ont été transférés selon les directives indiquée, et que l'application ne génère plus d'erreur, transformez la en fichier exécutable par la méthode habituelle.

Dans cette application nous générons une erreur au niveau de la procédure "GetPrivate ProfileString", alors que la procédure "WritePrivateProfileString" est reconnue par VB6.0

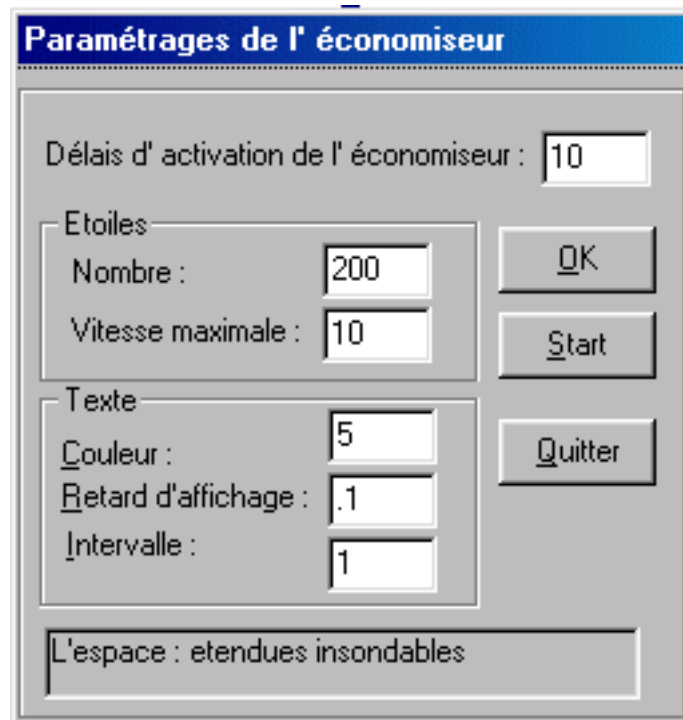
Cette procédure devant servir à dialoguer avec un fichier "*.ini", nous pouvons nous en passer en créant un fichier à accès direct pour la mise en mémoire des informations nécessaires au fonctionnement de cette application.

je n'ai pas étudié la procédure défaillante, mais je pense qu'en faisant des recherches plus approfondies, on doit pouvoir modifier le code selon les nouvelles fonctions mises en place par Microsoft.

Amusez-vous et découvrez le cobaye mis à votre disposition, disséquez le et si vous trouvez la solution, faites-moi le savoir à titre d'information, d'autant que cela pourra servir à d'autres.



le fichier Params de l'application 16 bits



le fichier Params issu de la transformation en application 32 bits

Si vous rencontrez des erreurs lors de la compilation, rectifiez en conséquence jusqu'à ce que VB génère le fichier exécutable.

Un des avantages de VB6.0 est qu'il est plus *pointu* que les versions précédentes au niveau de la procédure de compilation des applications générés.

Dans l'exemple cité précédemment, et en règle générale, nous constatons que le concepteur des fichiers "API" utilisent parfois des procédures différentes entre un fichier 16 bits et un fichier 32 bits.

Rechercher la procédure adéquate dans les "API" correspondants pour rétablir (si possible) le fonctionnement de la procédure.

Prochain tutoriel :

 ***Utilisation des contrôles documentées - Le contrôle "ANIBTN32.OCX"***

