Un premier programme

Dans ce chapitre

2.1 – Ecrire un premier programme

Note

En VBA, les projets sont souvent appelés *macros*. Dans ce livre j'utilise aussi facilement le mot *programme* qui, à mon sens, parfois convient mieux. Nous allons considérer que ces deux termes sont synonymes (au moins dans ce livre).

Il y a beaucoup de choses à apprendre avant de pouvoir écrire un programme. Pourtant je me doute que vous êtes impatient de vous lancer dans la programmation. C'est pourquoi nous allons tenter un premier exemple. Ne soyez pas effrayé par tous ces nouveaux termes. Tout cela vous deviendra vite familier, au fur et à mesure de votre progression dans ce livre.

1. Lancer AutoCAD R14, R2000 ou ultérieur, si ce n'est déjà fait. Ceci est indispensable : VBA pour AutoCAD ne fonctionne qu'avec AutoCAD !

2. Dans le menu déroulant *Outils* ⇒ *Macro VBA*, cliquez sur *Editeur Visual Basic* ou sur l'icône correspondante de la barre d'outils VBA si vous l'avez installée.

Figure 2-1 Un moyen simple de lancer l'*Editeur Visual Basic*.

3. La fenêtre 'EDI' (Environnement de développement intégré) de Microsoft Visual Basic s'ouvre alors.

4. Dans le menu *Insertion (ou `Insert`)*, cliquez sur ***UserForm***. Une *feuille* s'affiche alors, ainsi que la boîte à outils (`toolbox`) ***Contrôles***.

5. Dans la fenêtre ***Propriétés***, sélectionnez la propriété ***(Name)*** et remplacez le nom par défaut *UserForm1* par ***frmFeuille1***, un nom plus conforme aux normes (voir le chapitre 3). C'est ce nom qui servira dans le programme pour désigner cette feuille. Sélectionnez ensuite la propriété ***Caption*** et remplacez le nom par défaut *UserForm1* par ***Exercice 1 - Ajout d'un texte***. C'est ce qui sera affiché, et qui déjà s'affiche,comme titre de la feuille.

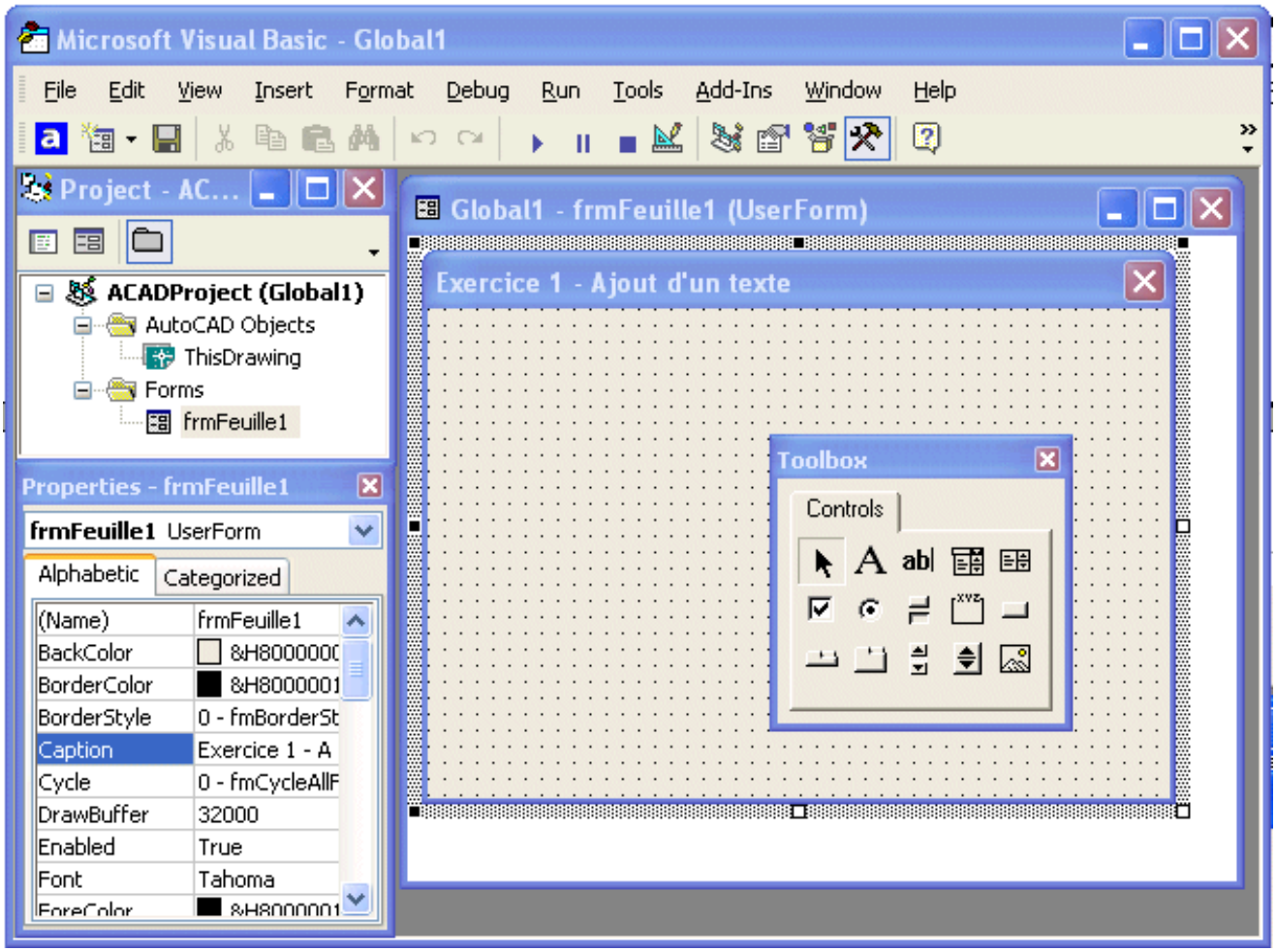

Figure 2-2 La feuille UserForm1 est rebaptisée.

Note

Ne confondez pas les propriétés *Name* et *Caption* :

Name : Nom du contrôle, de la feuille ou de tout objet qui identifie l'élément dans le programme.

Caption : Texte qui apparaît sur un objet, en général pour l'identifier.

6. Cliquez sur le contrôle ***Bouton de commande (`CommandButton`)*** et glissez le jusqu'au milieu de la feuille en maintenant le bouton gauche appuyé.

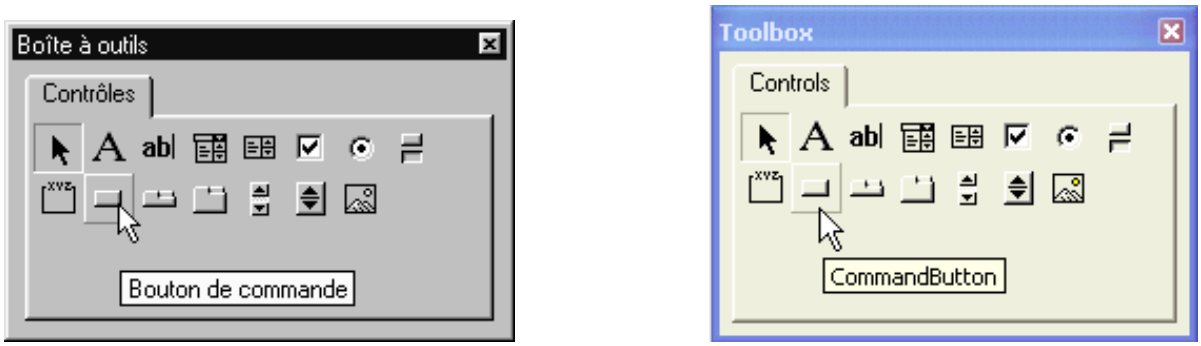

Figure 2-3 Choisissez le contrôle *Bouton de commande*.

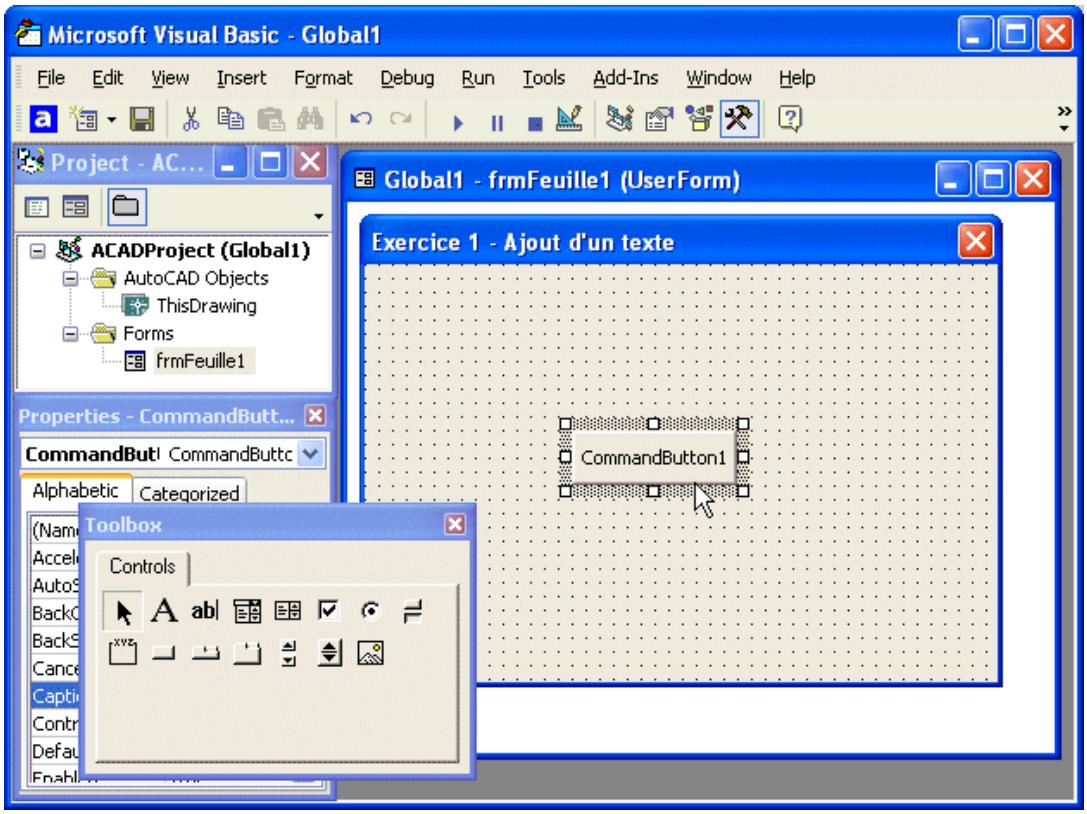

Figure 2-4 Placez le contrôle *Bouton de commande* au milieu de la feuille.

7. Cliquez sur ce bouton de commande que vous avez positionné et qui porte l'indication ***CommandButton1***. Nous allons en profiter pour changer quelques-unes des propriétés :

- Sélectionnez la propriété ***(Name)*** et remplacez le nom par défaut *CommandButton1* par ***CmdAction***, un nom plus conforme aux normes (voir plus loin). C'est ce nom qui servira dans le programme pour désigner ce contrôle.

- Sélectionnez la propriété ***Caption*** et remplacez le nom par défaut *CommandButton1* par ***Action***. C'est le mot qui sera affiché sur le bouton

- Et pour faire plus joli, nous allons changer la police. Sélectionnez la ligne ***Font*** et cliquez sur la case contenant 3 points près de *Tahoma*, la police par défaut. La fenêtre ***Police*** s'ouvre. Choisissez, par exemple, *Swis721 Ex Bt*, une des polices amenées par AutoCAD. Tant que nous y sommes, pourquoi ne pas mettre un style gras (bold) et une taille de 10 ?

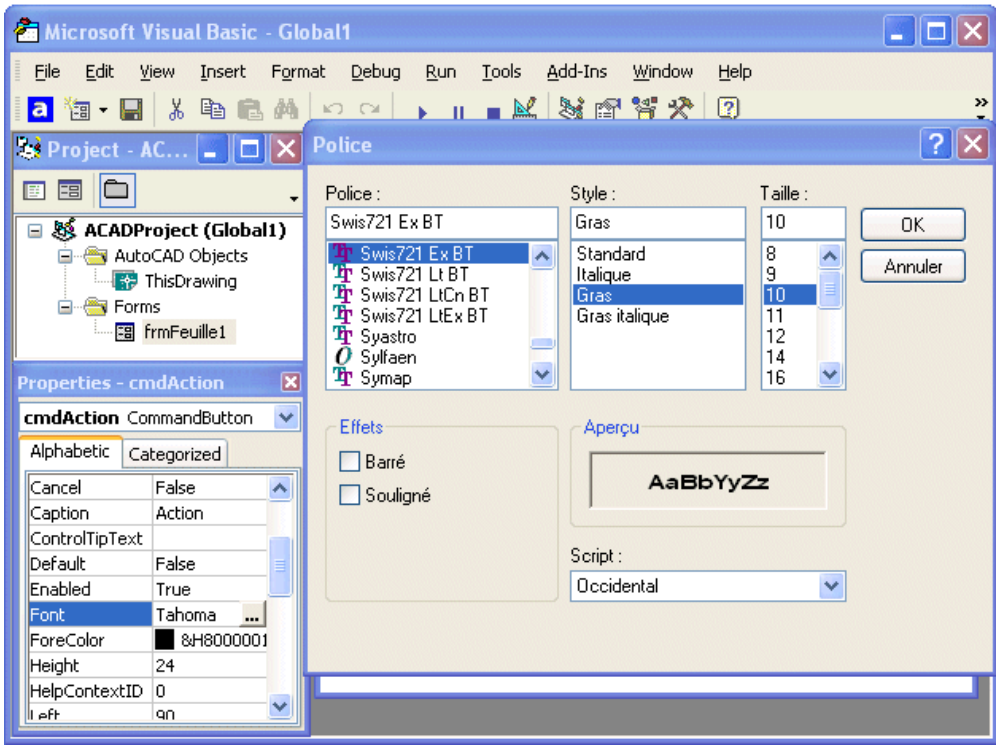

Figure 2-5 Choix d'une nouvelle police pour l'étiquette du bouton.

Note

Le style de police que nous indiquons ici ne concerne que l'écriture sur ce contrôle (c'est à dire le bouton *Action*). Cela ne concerne pas le texte qui sera écrit dans le dessin.

8. Maintenant double-cliquez sur ce bouton de commande *Action*. La fenêtre de code ***CmdAction*** s'affiche alors.

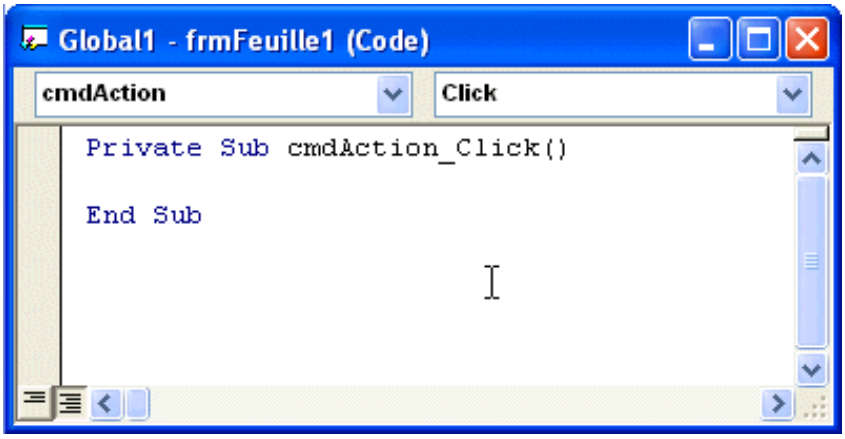

Figure 2-6 La fenêtre de code est prête pour écrire le code de la routine pour le bouton Action.

9. Le volet de gauche liste tous les objets du projet, mais comme tout est objet (nous en reparlerons plus loin), c'est la liste de tous les éléments ; tandis que le volet de droite indique les ***procédures***, c'est à dire les actions qui seront exécutées en liaison aux événements rattachés aux objets de la liste des objets. En effet, les objets Visual Basic sont capables de reconnaître les actions de l'utilisateur : clic, double-clic de souris, appui d'une touche, etc. Ces actions, appelées ***événements***, permettent de contrôler le déroulement du programme. Si nous restons sur la procédure ***Click***, qui est proposée d'office, tout le code que nous allons ajouter entre les lignes

```
Private Sub CmdAction_Click()
   et
End Sub
```

sera automatiquement exécuté à chaque clic sur le bouton de commande ***Action.***

Vous pouvez également ajouter du code pour les autres événements : *KeyDown, MouseUp...*
Mais en général, c'est surtout la procédure *Click* qui sera utilisée. On peut également créer d'autres événements.

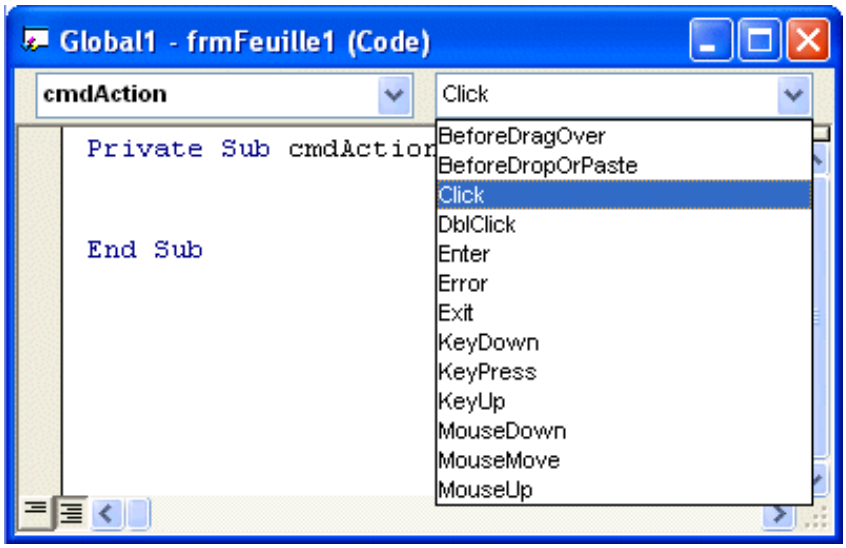

Figure 2-7 La liste déroulante des événements par défaut pour un bouton de commande.

10. Ecrivez maintenant votre code comme ci-dessous ou faites un copier-coller à partir du fichier texte *vbaa0201.txt* qui se trouve dans le répertoire *Exercices\Chap02* du package de ce livre. Inutile, bien sûr de taper les indications de numéro de ligne

```
Private Sub cmdAction_Click()
    ' Déclaration des variables                              Ligne 2
    Dim Pt1 As Variant
    Dim txtTexte As String

    ' On cache la feuille pour pouvoir choisir une entité  Ligne 6
    frmFeuille1.Hide
    ' Demande du point d'insertion du texte                  Ligne 8
    Pt1 = ThisDrawing.Utility.GetPoint(, _
        "Sélectionnez la position du texte : ")
    ' Définition du texte à insérer dans le dessin          Ligne 11
    txtTexte = "Premier texte écrit le " & Now
    ' Ajout du texte                                         Ligne 13
    ThisDrawing.ModelSpace.AddText txtTexte, Pt1, 5
End Sub
```

2.1.1 - Quelques remarques concernant l'éditeur de texte

Quand vous tapez du code dans l'éditeur de VBA, vous pouvez vous apercevoir que Visual Basic codifie les couleurs de votre texte pendant la frappe. Les couleurs les plus habituelles sont:

Noir	Texte normal
Rouge	Erreur de syntaxe du texte
Vert	Commentaires
Bleu	Mots clés

Vous pouvez personnaliser ces couleurs en choisissant l'onglet *Format de l'éditeur* à partir du menu déroulant *Outils* ⇒ *Options* de l'IDE. Sélectionnez un type de texte et ensuite choisissez la couleur désirée. Cliquez OK.

Quand vous commencez à taper un mot clé connu de VBA, vous verrez parfois une boîte se dérouler qui vous permet de choisir ou de compléter un mot en le choisissant dans une liste. L'éditeur ajoute ou enlève également des espaces et met des majuscules à certains mots pour faciliter la lecture. La plupart du temps un petit message d'erreur apparaît si vous faites une erreur de syntaxe. De cette manière, l'éditeur vous aide à taper un code correct.

2.1.2 - Explications de texte

Ligne 1 : `Private Sub CmdAction_Click()`

Private sera expliqué dans le prochain chapitre

Sub : Le mot clé *Sub* est une abréviation de subroutine, c'est-à-dire sous-programme. C'est une *instruction*.

CmdAction_Click : Nom de la procédure constitué du nom de l'objet (ici CmdAction) suivi du caractère de soulignement <_> et du nom de l'événement (ici click).

Les 2 parenthèses () qui suivent le nom de la procédure servent à ajouter les arguments éventuels. Ici, il n'y en a pas.

Ligne 2 : `' Déclaration des variables`

L'apostrophe indique que ce qui suit dans la ligne est un commentaire et que le programme ne doit pas en tenir compte.

Lignes 3 et 4 : Ce sont des déclarations de variables. Les variables servent à emmagasiner des données et sont expliquées en détail dans le chapitre 3

Ligne 7 : `frmFeuille1.Hide`

est Nom de la feuille suivi par sa méthode *Hide*, séparé par un point. Une *méthode* une action qu'un objet peut exécuter. Par exemple, *hide* est une méthode de l'objet *UserForm* qui cache la feuille pour avoir accès au dessin.

Note

Vous avez remarquez que lorsque vous avez tapé le point après frmFeuille1, l'éditeur vous a proposé une liste déroulante. C'est la liste des propriétés et méthodes applicables à cet objet. En cliquant sur *Hide*, cette méthode est ajoutée à son objet.

Lignes 9 et 10 :
```
Pt1 = ThisDrawing.Utility.GetPoint(, _
      "Sélectionnez la position du texte : ")
```

On assigne à la variable Pt1 le résultat de la méthode ***GetPoint*** appliqué à l'objet *Utility* de l'objet *ThisDrawing*

ThisDrawing : objet représentant le dessin en cours.

Utility : objet contenant une série de méthodes utilitaires.

GetPoint : méthode analogue à *GetPoint* en AutoLISP. Cette méthode sera expliquée plus en détail ultérieurement.

Le signe souligné précédé d'un espace < _ > en fin de ligne 9 est le *caractère de continuation de ligne*, ce qui signifie que la ligne de code se prolonge sur la ligne suivante et qu'elle est interrompue pour une raison de mise en page.
Ne fonctionne pas à l'intérieur d'une chaîne de caractères.

Ligne 11 :
```
txtTexte = "Premier texte écrit le " & Now
txtTexte = "Premier texte écrit le "  >>
```
On assigne à la variable txtTexte la chaîne placée entre les guillemets y compris les espaces.

& : Opérateur de concaténation qui relie deux chaînes ensemble. Ici, ajoute le résultat de la fonction Now à la phrase "Premier texte écrit le "

Now : Fonction Visual Basic qui renvoie la date et l'heure système de votre ordinateur. *Une fonction est une procédure, c'est à dire une série d'instructions, qui renvoie une valeur.*

Ligne 14 :
```
ThisDrawing.ModelSpace.AddText txtTexte, Pt1, 5
```

On utilise la méthode AddText pour ajouter l'objet txtTexte dans la collection ModelSpace de ThisDrawing (le dessin en cours).

`ModelSpace` : Objet spécial, appelé *collection* qui contient toutes les entités

contenues dans l'espace objet.
La méthode AddText nécessite trois arguments :
- TxtTexte, le texte à insérer est le premier argument.
- Pt1 : est le point d'insertion du texte
- Le 3ème argument détermine la hauteur du texte. Ici 5 unités.

Ligne 15 : `End Sub` **:** Nécessaire pour terminer la procédure.

2.2 – Exécuter un programme

Vous êtes impatient de voir l'effet du programme que vous avez tapé. Pour exécuter (ou lancer) un programme il y a trois manières :

1. Tapez sur la touche de fonction **F5**,
2. Cliquez sur le bouton  de la barre d'outils Standard de VBA,
3. Choisissez dans le menu déroulant *Exécution* ⇒ *Exécuter Sub/UserForm* (`Run ⇒ Run Sub/UserForm`) Choisissez l'une quelconque de ces manières. La feuille *frmFeuille1* s'affiche au-dessus du dessin comme montré dans la figure 2-8.

Attention : La feuille doit être active. Sinon, cliquez sur la feuille puis lancer l'exécution.

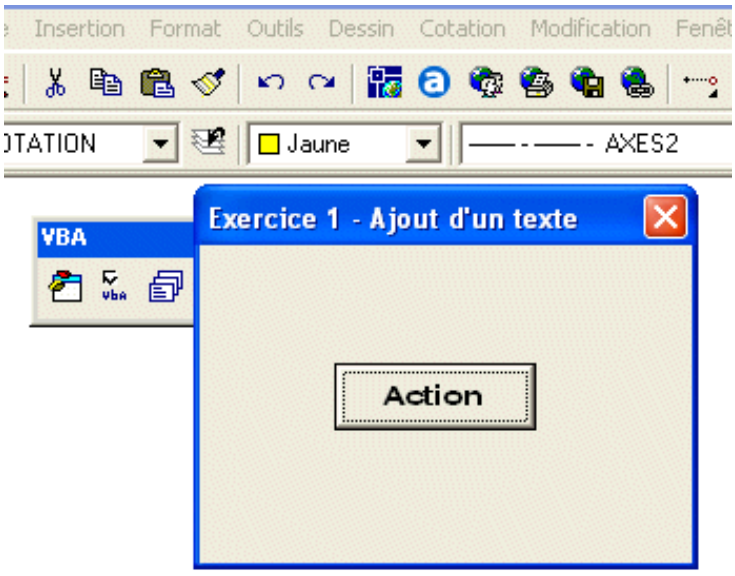

Figure 2-8 La feuille frmFeuille1 s'affiche au-dessus du dessin, attendant que vous appuyez sur l'unique bouton.

Cliquez ensuite sur le bouton ***Action*** de la feuille. La feuille disparaît et un message sur la ligne de commande d'AutoCAD vous demande de désigner le point d'insertion du texte.

Cliquez le point désiré. Vous revenez brusquement à votre éditeur VBA. C'est normal, le programme s'est terminé. Revenez à AutoCAD en cliquant sur le bouton  de la barre d'outils Standard ou par le menu déroulant *Affichage* ⇒ *AutoCAD* et vous verrez votre texte. Il est possible, toutefois que vous soyez obligé de zoomer, soit plus grand, soit plus petit.

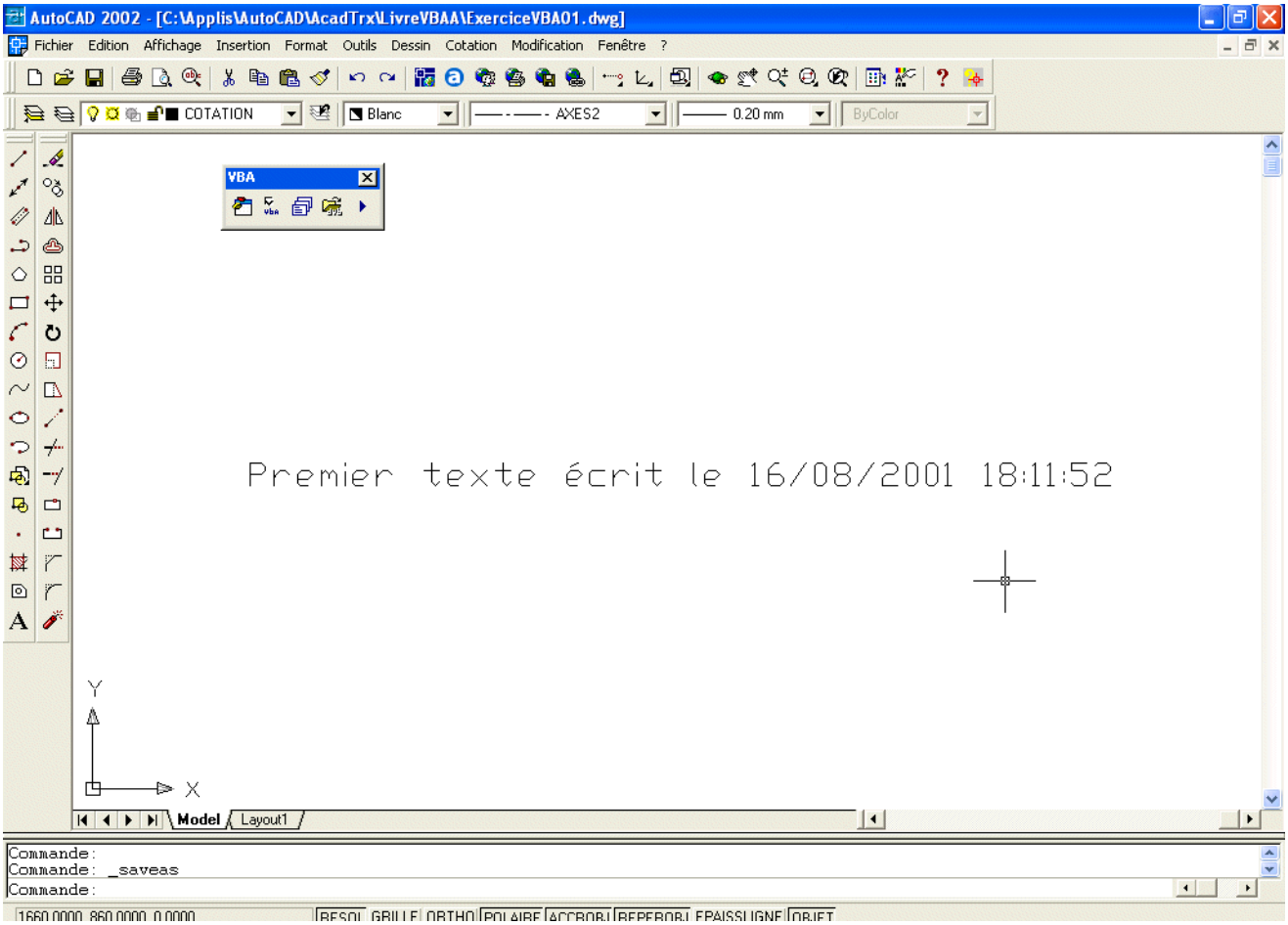

Figure 2-9 Voilà .le résultat de la routine. Si vous avez la même date qu'ici, lancez-vous dans la magie !

Il est possible que votre application ne s'exécute pas : Vérifiez que votre feuille est actuellement active ; si oui, il doit exister une erreur dans votre code ! Pour obtenir de l'aide, cliquez sur le

bouton d'aide, le point d'interrogation (?) ou appuyez sur F1 quand vous obtenez un message d'erreur. Essayez les corrections proposées pour corriger l'erreur avant de tenter un nouveau lancement de votre code.

Si cela ne fonctionne toujours pas, faites un copier-coller à partir du fichier texte *vbaa0201.txt* qui se trouve dans le répertoire *Exercices\Chap02* accompagnant ce livre.

Si vous avez toujours un problème, charger le programme comme expliqué plus loin dans ce chapitre. En désespoir de cause faites une prière ou envoyez-moi un E-mail. Non, je plaisante, ce n'est pas si compliqué, si vous respectez la syntaxe. Try again !

Nous verrons par la suite que le code ne peut pas s'exécuter indépendamment lorsque la procédure ou la fonction nécessite des arguments. Ce qui n'est pas le cas pour ce premier exemple.

2.3 – Ajouter un module de départ

Nous avons vu que notre programme pouvait fonctionner directement. Cependant, il est plus facile et même recommandé d'insérer un module de départ qui appellera la procédure déjà créée.

- Retournez dans la fenêtre 'EDI' de Microsoft Visual Basic.
- Dans le menu *Insertion*, cliquez sur ***Module***. Voyez le paragraphe 1.3.2.3 si vous ne vous souvenez plus comment faire.
- Choisissez *Insertion* ⇒ *Procédure*. Dans la zone de texte *Nom*, tapez **InsertTexte.** Le type doit être *Sub* et la portée(`Scope`) doit être *Public*. Cliquez OK.

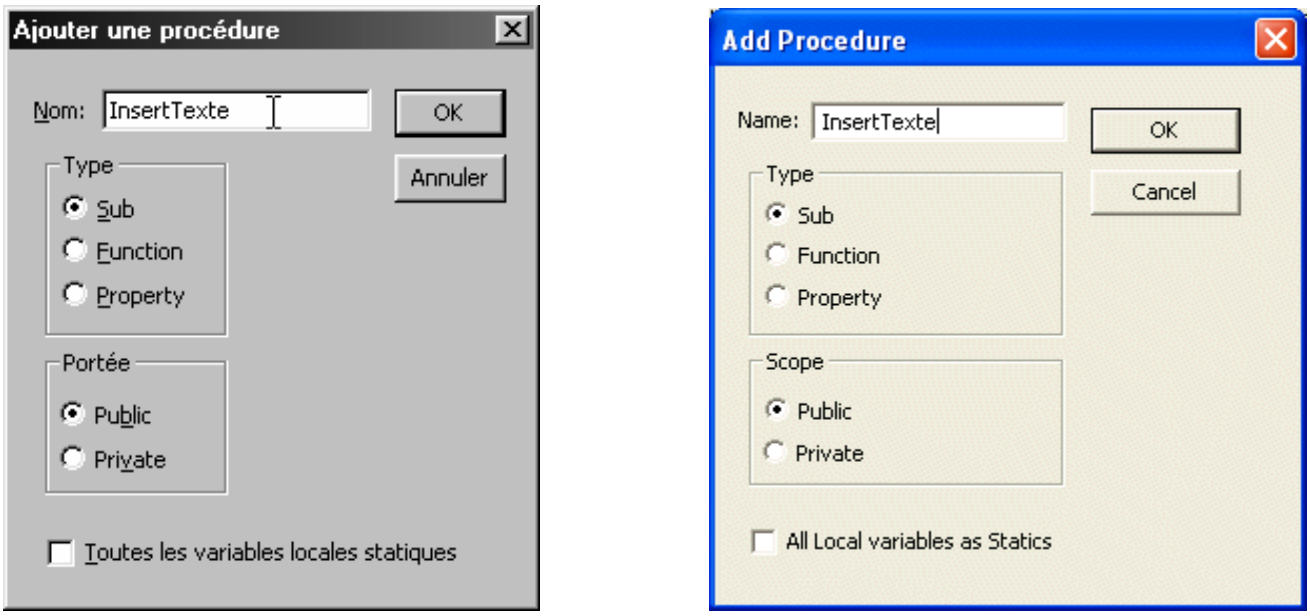

Figure 2-10 La fenêtre ajouter une procédure

- Il vous suffit d'ajouter

```
FrmFeuille1.Show
```

entre les deux lignes

```
Sub InsertTexte()
```

et

```
End Sub
```

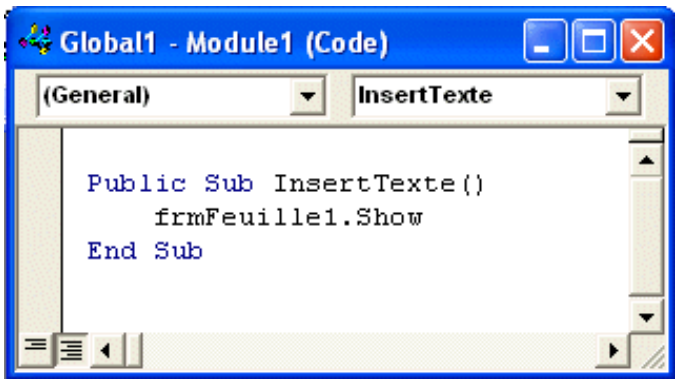

Figure 2-11 Le module ne comporte que ces trois lignes

Ligne 1 : **InsertTexte** est le nom qui a été choisi, tout à fait arbitrairement, pour le programme (ou la macro).

Ligne 2 : On appelle la méthode **Show** (montrer) pour afficher la feuille. C'est le contraire de la méthode **Hide** (cacher) que l'on utilise dans la procédure cmdAction_Click pour accéder au dessin.

Retourner maintenant dans AutoCAD. Plusieurs moyens disponibles ici aussi :

- Dans le menu déroulant *Outils* ⇒ *Macro VBA*, cliquez sur *Macros*…
- Tapez sur les touches *Alt* et *F8* en même temps.
- Tapez **vbarun** sur la ligne de commande d'AutoCAD.
- Cliquez sur l'icône la plus à droite sur la barre d'outils VBAA.

Dans la fenêtre *Macros* qui s'ouvre alors, vous devriez voir votre macro listée. Et pour le moment, c'est la seule.

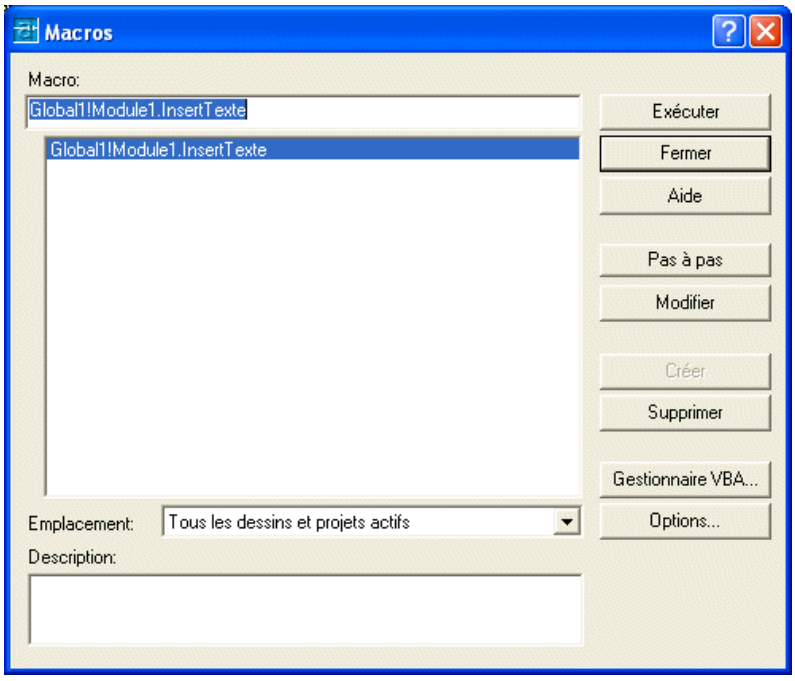

Figure 2-12 La fenêtre Macros liste les programmes VBA disponibles dans le ou les dessins actifs.

Cliquez sur le bouton *Exécuter* pour voir si elle fonctionne toujours.

2.4 – Enregistrer un projet

Il est plus que temps de sauvegarder votre projet.

La version VBA d'AutoCAD enregistre les projets VBA comme fichiers séparés avec une extension *.dvb*.

Tant que vous n'avez pas sauvegardé votre projet, il s'appelle Global1. La première option du menu *Fichier* indique *Enregistrer Global1* (`Save Global1`). Vous pouvez aussi cliquer sur l'icône  *Enregistrer Global1* sur la barre d'outils standard de VBA.

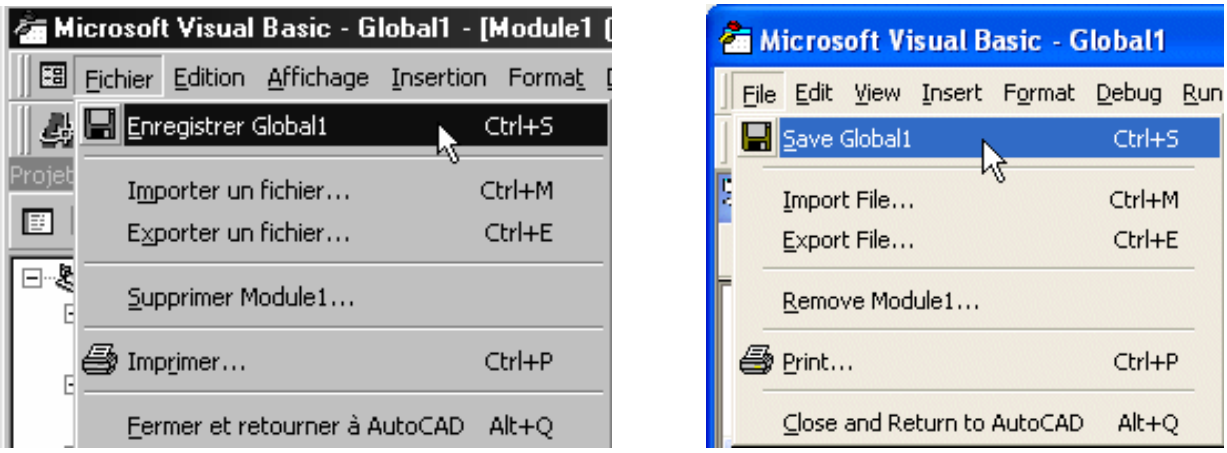

Figure 2-13 L'option *Enregistrer Global1* du menu Fichier.

VBA vous retourne à AutoCAD et ouvre la boîte de dialogue *Enregistrer sous*. Tapez un nom pour votre projet, choisissez un emplacement et cliquez *Enregistrer*.

> **Note**
>
> Le nom du projet (aussi bien que le nom des modules, feuilles et contrôles) doit commencer avec une lettre et peut avoir jusqu'à 40 caractères. Seuls les lettres, les nombres et le caractère souligné sont permis.

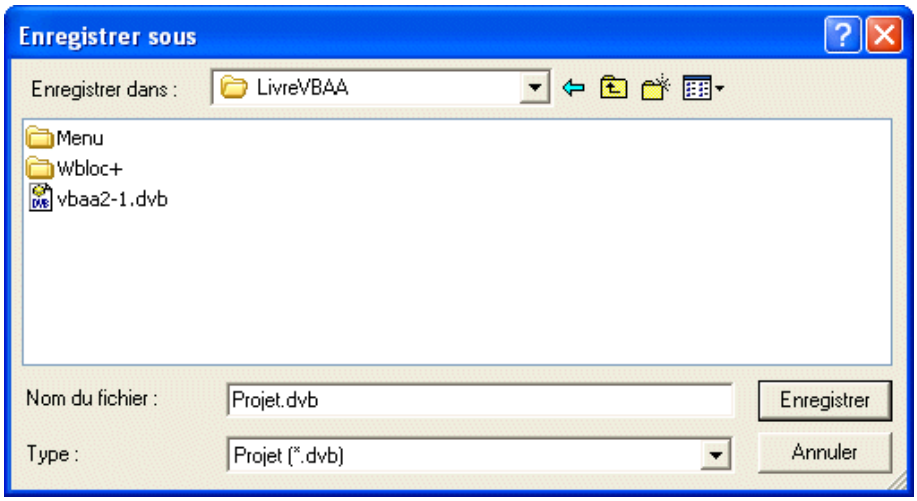

Figure 2-14 La fenêtre *Enregistrer sous* apparaît lors du premier enregistrement.

L'ensemble du projet, feuilles, modules et contrôles est enregistré dans un fichier unique avec l'extension *.dvb*.

Dans ce menu déroulant *Fichier*, vous pourrez enregistrer à nouveau au fur et à mesure de vos modifications mais vous ne pourrez pas là changer le nom de l'enregistrement. Si vous désirez changer de nom à votre projet, il vous faudra passer par le *Gestionnaire VBA* pour avoir à nouveau la fenêtre *Enregistrer sous*.

2.5 – Incorporer un projet dans un dessin AutoCAD

Vous pouvez incorporer un projet sélectionné dans le dessin en cours qui est sélectionné. Ainsi lorsque vous chargé votre dessin, la macro incorporée est immédiatement disponible.

C'est par le *Gestionnaire VBA* que vous aurez cette possibilité, en cliquant sur le bouton *Incorporer*. Choisissez le dessin où incorporer la macro dans la liste déroulante *Dessin*, si vous avez plusieurs dessins d'ouverts en même temps. Il vous faudra bien sûr enregistrer votre dessin également.

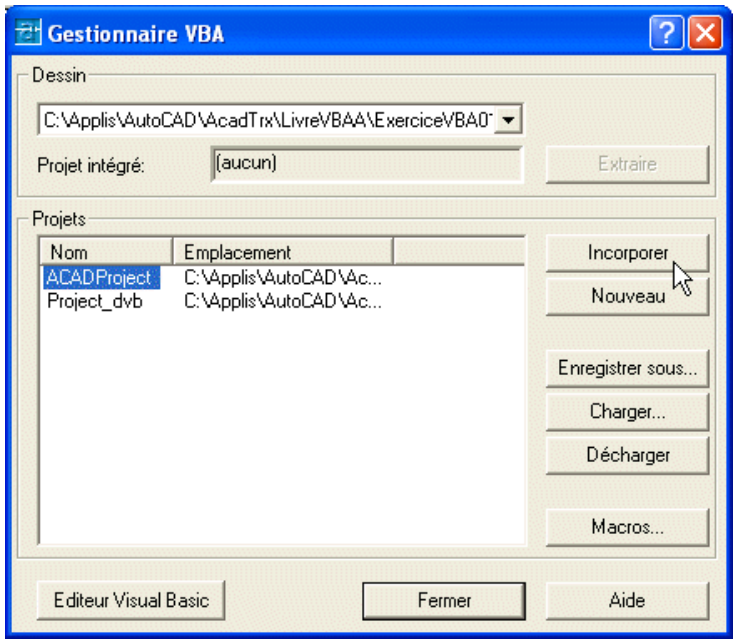

Figure 2-15 Le *Gestionnaire VBA* permet d'incorporer un projet dans un dessin.

Vous ne pouvez incorporer qu'un seul projet dans votre dessin.

Inversement si le dessin sélectionné dans la fenêtre possède une macro incorporée et que vous vouliez la supprimer ou la changer, le bouton ***Extraire*** du *Gestionnaire VBA* vous permettra de le faire.

2.6 – Ouvrir (ou charger) un projet

Avant de lancer une macro VBA, elle doit être chargée. Si vous travaillez sur une routine et que vous vouliez l'exécuter pour vérifier comment elle fonctionne — ce que vous ferez souvent— vous n'avez pas besoin de la charger, elle est déjà en mémoire.

Pour ouvrir ou charger un projet – dans ce cas, les deux mots ont le même sens – plusieurs moyens sont à votre disposition :

- Cliquez sur l'option *Outils* ⇒ *Macro VBA* ⇒ *Charger projet*…
- Sur la ligne de commande d'AutoCAD, tapez **chargvba** ou **_vbaload** _↑
- Cliquez sur le bouton *Charger* du *Gestionnaire VBA*.

- Cliquez sur la quatrième icône de la barre d'outils VBAA, si vous l'avez chargée (voir chapitre 1)

Dans tous ces cas vous obtenez la fenêtre *Ouvrir un projet VBA*.

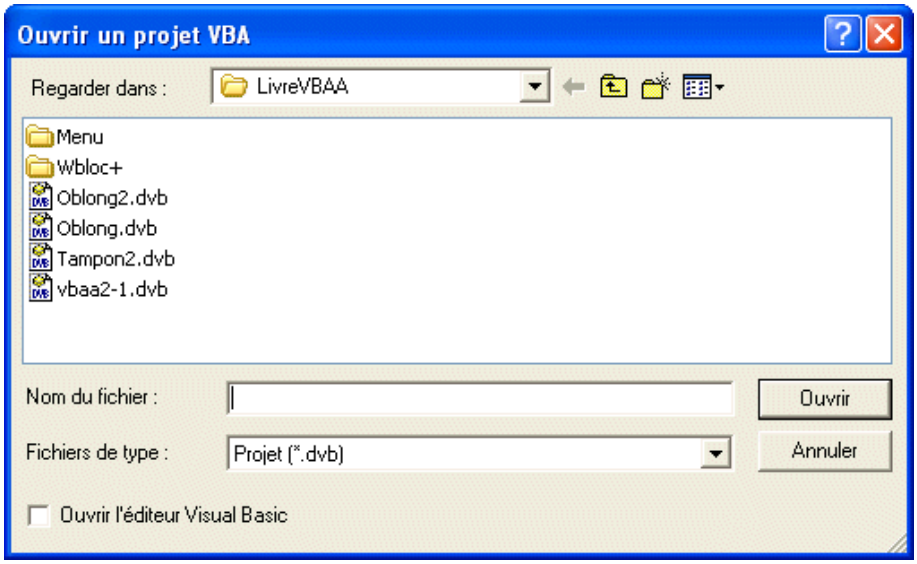

Figure 2-16 La fenêtre *Ouvrir un projet VBA*.

Cochez la case *Ouvrir l'éditeur Visual Basic*, en bas à gauche de la boîte de dialogue, pour ouvrir automatiquement l'environnement VBA IDE au chargement d'un projet VBA. Cette option reste activée jusqu'à ce que vous supprimiez la coche de la case.

Si la boîte de dialogue n'apparaît pas, la variable système FILEDIA est sans doute désactivée dans AutoCAD. Cette variable active et désactive l'affichage des boîtes de dialogue. Pour réactiver la variable FILEDIA, donnez-lui la valeur 1.

Vous pouvez utilisez un autre moyen pour ouvrir un projet : Choisissez *Outils* ⇒ *Charger une application*. AutoCAD ouvre alors la boîte de dialogue *Charger/Décharger les applications*. Naviguez jusqu'à votre projet, choisissez-le, et cliquez *Charger*. Ensuite cliquez *Fermer* pour quitter la boîte de dialogue. Le projet est maintenant chargé.

Note

Depuis la version 2000 d'AutoCAD, le nombre de projets VBA pouvant être chargés en même temps n'est plus limité.

Si vous avez eu des problèmes avec ce premier exemple, vous savez maintenant comment faire pour charger la version qui se trouve sur le CD d'accompagnement du livre. C'est le fichier *vbaa2-01.dvb* qui se trouve dans le répertoire *Exercices\Chap02*

Note

N'oubliez pas que les fichiers du CD sont en lecture seule et ne peuvent être modifiés tant qu'ils sont lancés à partir du CD. Si vous désirez les modifier, copiez-les sur votre disque dur et changez-en l'attribut par l'option *Propriétés* du menu contextuel (le menu obtenu en cliquant avec le bouton de droite) de l'Explorateur Microsoft.

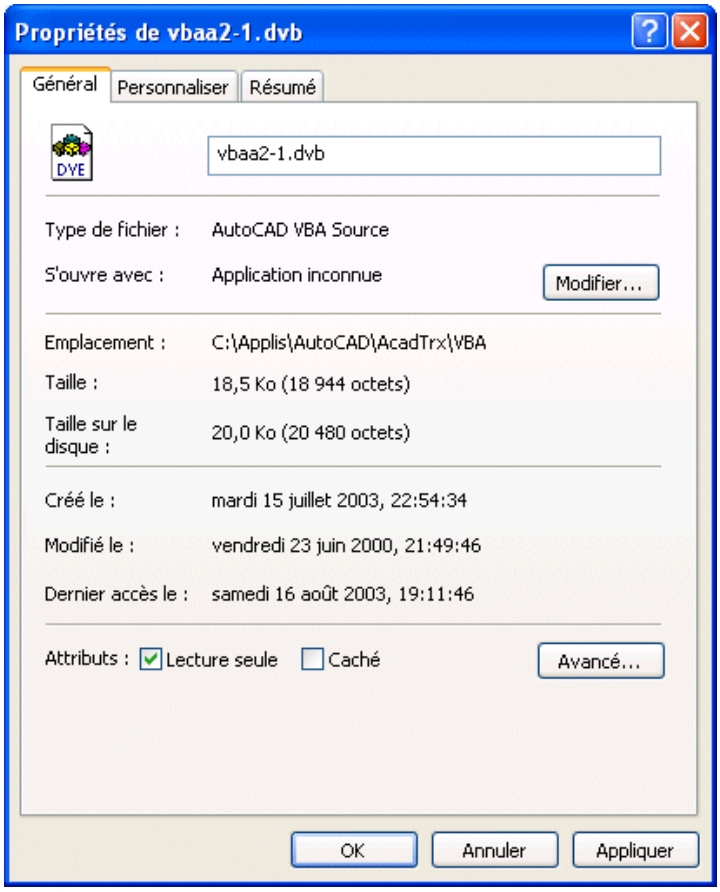

Figure 2-17 Décochez l'attribut *Lecture seule* dans la fenêtre *Propriétés* de *l'Explorateur Microsoft* avant de modifier un fichier copié sur le CD.

2.7 – Un coup de tampon : le second exemple

Je sens que d'aucuns vont faire la fine bouche. Un exemple, c'est sans doute bien mais celui-là ne sert à rien !

OK. Voici sans plus attendre et avant de faire un peu de théorie, un petit programme vraiment utile qui est souvent réclamé sur les forums AutoCAD. C'est une amélioration du programme précédent. Il s'agit d'insérer un texte qui donnera automatiquement :

- Le nom du dessin y compris le chemin.
- Le nom d'utilisateur de la machine
- La date
- L'heure

Ensuite, la commande Traceur sera lancée automatiquement.

Plusieurs formes de cet utilitaire, écrites en AutoLISP, existent sous le nom de *Stamp*. C'est pourquoi nous l'avons appelé *Tampon. (Vous pourrez notamment trouvez une version écrite par mes soins, en AutoLISP, sur le site Le Coin des AutoCADiens).*

2.7.1 – Rien qu'un module

Cette fois-ci, il n'y aura ni feuille, ni contrôles. Ils seraient inutiles. Nous utiliserons uniquement un module pour écrire notre programme.

Pour commencer, pour éviter les confusions, déchargez tous les autres programmes en cours à l'aide du gestionnaire VBA.

- Puis retournez dans la fenêtre 'EDI' de Microsoft Visual Basic.
- Dans le menu *Insertion*, cliquez sur ***Module***.
- Choisissez *Insertion* ⇒ *Procédure*. Dans la zone de texte *Nom*, tapez **Tampon.** Le type doit être *Sub* et la portée doit être *Public*. Cliquez OK.

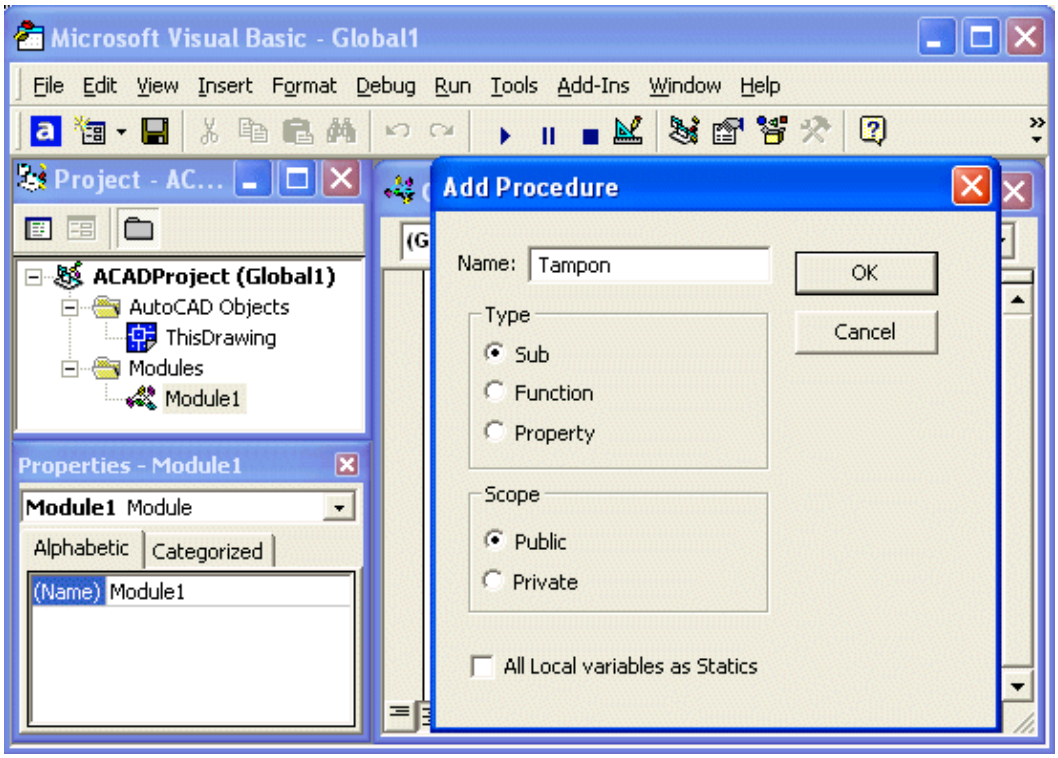

Figure 2-18 Ajouter une procédure Tampon

- Il vous reste maintenant à tapez le texte (ou à faire un copier-coller à partir du fichier texte *vbaa0202.txt* qui se trouve dans le répertoire *Exercices\Chap02* du CD

```
Public Sub Tampon()
    ' Déclaration des variables
    Dim Pt1(0 To 2) As Double
    Dim txtTout As String
    Dim txtTexte1 As String
    Dim txtTexte2 As String
    Dim txtTexte3 As String
     ' Définition du point d'insertion du texte          Ligne 8
    Pt1(0) = 5           'coordonnée en X
    Pt1(1) = 20          'coordonnée en Y
    Pt1(2) = 0           'coordonnée en Z
    ' Définition du texte à insérer dans le dessin      Ligne 12
    txtTexte1 = ThisDrawing.FullName
    txtTexte2 = " par Laurellée Hardy "
    txtTexte3 = Now
    txtTout = txtTexte1 & txtTexte2 & txtTexte3
    ' Ajout du texte                                    Ligne 17
    ThisDrawing.ModelSpace.AddText txtTout, Pt1, 3
```

```
    ' Sortie traceur                                    Ligne 19
    ThisDrawing.Plot.PlotToDevice
End Sub
```

Avant de tester la macro, mettez une apostrophe devant la ligne 20, sinon vous aurez une sortie traceur du dessin en cours à chaque fois :

```
    ' ThisDrawing.Plot.PlotToDevice
```

Vous savez maintenant que ce qui suit une apostrophe devient un commentaire et n'est pas pris en compte par VBA. On peut également remplacer l'apostrophe par ***Rem*** (pour Remarque).

Pour lancer la macro, revenez à AutoCAD, sinon vous ne verrez rien. Dans le menu déroulant *Outils* ⇒ *Macro VBA*, cliquez sur *Macros*...puis choisissez votre macro.

Je sais, ce n'est pas des plus simples pour lancer une macro. Plus tard nous apprendrons à insérer nos programmes dans des barres d'outils et il suffira de cliquer sur l'icône correspondante pour les lancer.

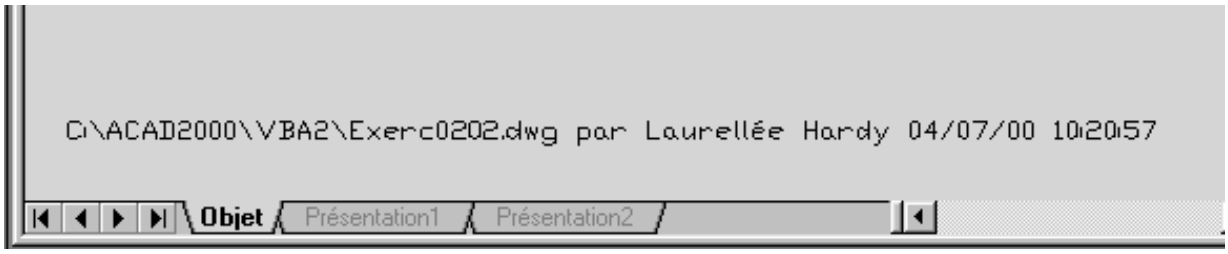

Figure 2-19 Et voilà le résultat de votre macro.

2.7.2 – Quelques commentaires

Voici quelques précisions pour comprendre le programme et pour l'adapter à votre goût.

Lignes 9 à 11 : Coordonnées du point d'insertion du texte. Remplacez ici les valeurs indiquées par celles que vous voulez. Si vous désirez laisser l'utilisateur choisir le point d'insertion, vous devez remplacer ces lignes par la méthode utilisée dans le premier exercice. Dans ce cas, n'oubliez pas de modifier également la déclaration de la variable Pt1.

Ligne 13 : *FullName* est une propriété de *ThisDrawing* (le dessin en cours) qui donne le nom de ce dessin en cours ainsi que le chemin complet. Si vous ne désirez pas mettre le chemin, remplacez *FullName* par la propriété *Name*

```
txtTexte1 = ThisDrawing.Name
```

Note

Si vous n'avez pas encore donné de nom à votre dessin, la propriété ***FullName*** est vide.

Ligne 14 : Bien sûr, remplacez ici le nom de Laurellée par celui que vous désirez. Plus tard nous apprendrons comment récupérer directement le nom d'utilisateur de la machine et l'affecter à la variable txtTexte2.

Ligne 15 : Vous connaissez déjà la fonction *Now*. Nous verrons plus tard comment utilisez la fonction *Format* pour modifier le résultat à notre convenance.

Ligne 18 : Vous avez déjà vu la méthode *AddText*. Ici la hauteur du texte est fixée à 3. Mettez ce que vous voulez.

Ligne 20 : `ThisDrawing.Plot.PlotToDevice.` La méthode PlotDevice lance une sortie traceur du dessin en cours. Le traçage doit déjà avoir été configuré. Vous pouvez bien sûr supprimer cette ligne si vous ne voulez pas d'impression du dessin.

2.8 – Un petit rappel

Comme avec tout langage de programmation, vous devez apprendre la syntaxe, connaître les différents termes et savoir quand les utiliser.

Le tableau 2-1 vous rappelle les termes principaux qui ont déjà été vus et qu'il est important de se rappeler pour comprendre plus facilement.

Tableau 2-1
Quelques termes utilisés en VBA (par ordre alphabétique)

Terme	Définition
Contrôle *(Control)*	Elément que l'on sélectionne dans la boîte à outils et que l'on dispose sur une feuille.
Evénement	Actions, telles que clics de souris, qui permettent de contrôler le déroulement du programme
Feuille *(UserForm)*	Un récipient pour les composants visuels, tels que boutons et zones de texte d'une boîte de dialogue que vous ajoutez dans votre projet VBA.

Fonction *(Function)*	Une procédure, écrite en code VBA, qui retourne une valeur.
Macro	Un programme public qu'un utilisateur peut exécuter directement.
Méthode	Procédure qui agit sur un objet.
Module	Un ensemble de sous-programmes et fonctions qui habituellement forment un composant distinct dans un projet.
Procédure *(Procedure)*	Code qui fait quelque chose et qui porte un nom. Une procédure peut être un sous-programme, une fonction ou une propriété.
Projet *(Project)*	Un ensemble de feuilles et modules.
Propriété *(Property)*	Une procédure, écrit en code VBA qui spécifie une valeur (la propriété d'un objet).
Sous-programme *(Subroutine)*	Une procédure, écrite en code VBA qui ne retourne pas de valeur.

2.9 – Résumé

Dans ce chapitre vous avez écrit votre premier programme en apprenant à utiliser l'éditeur de texte, créer une feuille et placer un contrôle. Vous avez ensuite appris à exécuter le programme, enregistrer, incorporer ou ouvrir un projet.

Vous avez écrit une seconde macro qui peut vraiment servir et fait une petite révision des termes utiles à savoir.

Dans le prochain chapitre nous saurons tout sur les variables et leur portée, les tableaux et les constantes.