

Programmation sous Visual Basic

Programmation sous Visual Basic 6.0 composé d'une présentation comprenant plusieurs tutoriels. Dans ces tutoriels, des étudiants sans expérience préalable en programmation apprendront à concevoir et à créer des applications.

I. Introduction

La première tâche que nous allons étudier car une des plus facile sera de créer un écran de droits d'auteurs (Copyright).

Cet écran vous servira d'écran de présentation à chaque application que vous allez créer. L'écran de droits d'auteur va identifier l'auteur de l'application et l'année de copyright, et affichera le logo de votre entreprise ou organisation. Nous pourrions éventuellement le doter d'un numéro de série "intelligent" permettant de créer une protection logicielle assez sophistiquée pour rebuter la majorité des casseurs de code.

Quoique cette première tâche soit légère, elle va vous donner l'opportunité d'apprendre les bases de Visual Basic sans vous inquiéter des questions de conceptions et des concepts de programmation appliqués dans les réalisations plus complexes.

II. Démarrage

II.1. Démarrage de Visual Basic

Avant de pouvoir commencer à écrire votre écran de présentation de droits d'auteur, vous devez démarrer Visual Basic. (Nous supposons qu'il est déjà installé sur votre machine)

II.2. Pour démarrer Visual Basic:

Cliquez sur le bouton Démarrer de la barre des tâches.

Dans le menu Démarrer, pointez sur Programmes. Dans la liste des Programmes, pointez sur Microsoft Visual Basic 6.0(ou éventuellement Microsoft Visual Studio 6.0)

L'écran des droits d'auteur de Visual Basic apparaît momentanément, puis la boîte de dialogue Nouveau projet s'ouvre (figure 1.1)

La première fenêtre qui s'affiche lorsque vous lancez Visual Basic vous propose de choisir le type d'application que vous voulez créer.

Sélectionnez la première icône marquée Exe standard située dans l'onglet "Nouveau", puis cliquez sur le bouton "Ouvrir" pour afficher un nouveau projet.

L'écran de création de Visual Basic apparaît.



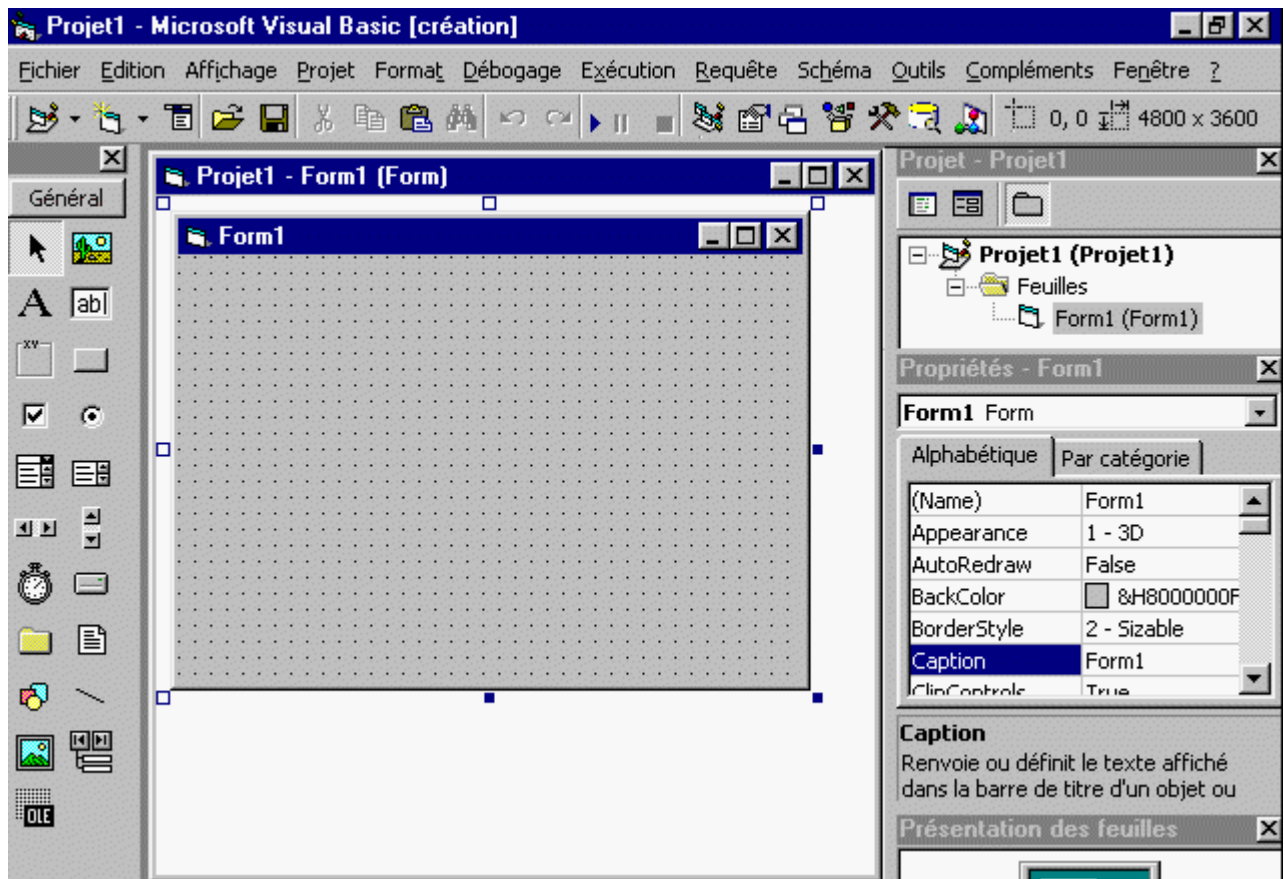
Figure 1-1



Exe.standard" représente le modèle le plus courant pour réaliser la plupart des applications sous Visual basic

III. L'interface de travail de Visual Basic 6.0 édition professionnelle

Comme toutes les feuilles sous plate forme Windows l'interface de travail se présente avec une barre de titre, suivie d'une barre de menu et d'une barre d'outils. Diverses fenêtre s'ouvrent suivant la configuration souhaitée, à savoir les plus utilisées, la boîte à outils, la feuille de travail(Form1), la fenêtre de projet, la fenêtre de propriété.



Dans la prochaine leçon nous verrons comment paramétrer l'organisation de notre plan de travail.

Tutoriel 2 : L'environnement de développement

Date de publication : Lundi 18 mars 2003, Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilmir.developpez.com\)](http://gilmir.developpez.com)

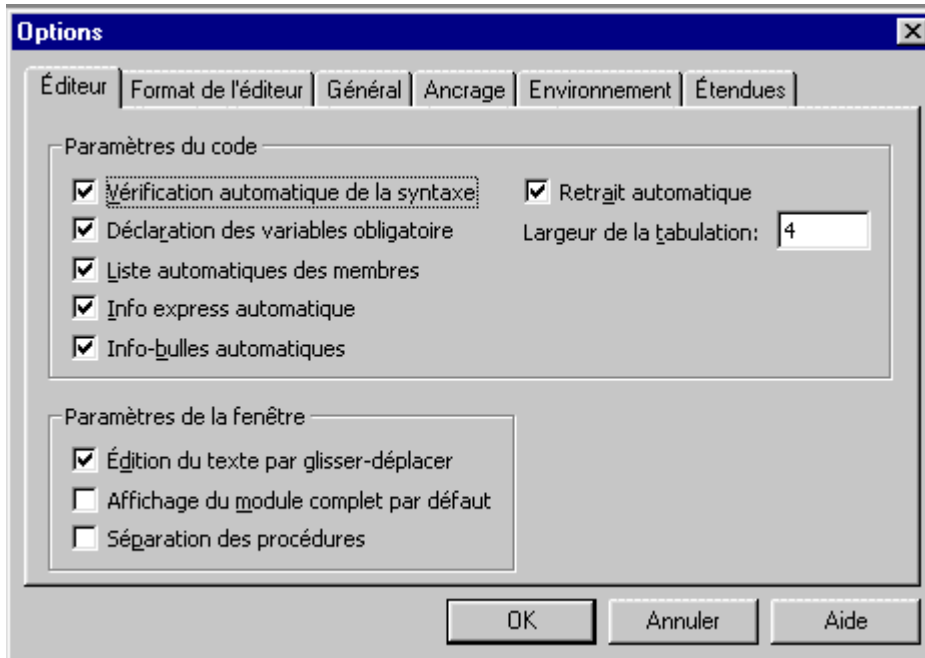
[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. L'environnement de développement

L'environnement de développement intégré de Visual Basic (**IDE**: **I**ntegred **D**evelopment **E**nvironment) peut être personnalisé, pour ce qui concerne sa présentation et son mode de fonctionnement.

Il est toutefois préférable avant de commencer votre premier projet, d'effectuer quelques réglages :

1)- Dans la barre de menu, cliquez sur l'étiquette "**Outils**", puis sur "**Options**", dans l'onglet "**Editions**" vous cochez toutes les cases à l'exception de "Affichage du module complet" et dans la largeur de la tabulation vous saisissez la valeur "2".



2)- Sélectionnez l'onglet "**Général**", vous modifiez les valeurs des unités de la grille à "30"

3)- Sélectionnez l'onglet "**Ancrage**" et désactivez toutes les coches des cases à cocher.

Lorsque vous serez plus à l'aise avec votre environnement de travail, vous pourrez ajuster les options pour les personnaliser à votre goût.

Nous verrons plus tard d'autres réglages intéressants.

II. Votre premier projet VISUAL BASIC

II.1. Création de l'interface

La plupart des applications réalisées avec Visual Basic comportent au moins une feuille. Les feuilles (Form en Anglais) constituent les éléments de base de l'interface utilisateur. Elles forment les conteneurs (containers) recevant des contrôles, comme des boutons, des zones de textes, des cases à cocher, etc...Qui peuvent être adaptés en fonction de vos aptitudes créatrices.

Lorsque vous avez démarré Visual Basic, la fenêtre de l'éditeur de feuille présente une feuille vide nommée **Form1** (Nom que vous retrouvez dans la barre de titre).

II.2. Définir la taille de la feuille

Il y a plusieurs façons de créer notre feuille, la plus facile est de réaliser une image et de l'incorporer à la feuille, l'autre façon est de composer pièce par pièce notre présentation et c'est cette option qui retiendra notre faveur. Le résultat est le même à la différence près que la solution avec l'image nous consommera un peu plus de mémoire.

La taille de la feuille sera fonction des éléments que nous allons y insérer et nous allons lui donner une taille approximative tout en sachant que nous pourrions éventuellement la modifier.

En regardant la feuille (Form1) vous constatez des petits carrés noirs situés au centre de chaque côté de la feuille. C'est ce que l'on appelle des poignées qui vous permettent en les étirant de modifier les dimensions respectives de chaque côté de la feuille.

Vous avez les possibilités de contrôler à tout moment les dimensions que vous accordez à votre feuille par l'intermédiaire du compteur situé dans la barre d'outils.

Nous lui donnerons comme valeur les dimensions de :6195 x 3840.

Pour modifier la taille de cette fenêtre, utilisez le pointeur de la souris.

Placez-le sur le coin inférieur droit de la fenêtre (ce qui a pour effet de le transformer en une double flèche oblique), appuyez sur le bouton gauche de la souris, déplacez le pointeur jusqu'à ce que la fenêtre atteigne les dimensions souhaitées, puis relâchez la souris et contrôlez les dimensions affichées.

Nous venons de modifier les valeurs initiales de la feuille manuellement, nous aurions pu le faire également par l'intermédiaire de la fenêtre de propriétés.

III. Les propriétés

III.1. Ça se complique, qu'est ce que les propriétés ?

Chaque objet en Visual Basic dispose d'un jeu de caractéristiques, appelées propriétés, qui lui sont associées. Ces propriétés, listées dans la fenêtre Propriétés, contrôlent l'apparence et le comportement de l'objet.

Visual Basic attribue une valeur par défaut aux propriétés de chaque objet. Ces valeurs peuvent être modifiées pendant la conception du programme ou en mode actif de l'application par programmation.

Nous ne nous intéresserons pour l'instant qu'aux propriétés que nous allons utiliser. Je vous conseille néanmoins de faire des essais pour vous familiariser avec toutes les propriétés de chaque objet.

III.2. La feuille de propriétés



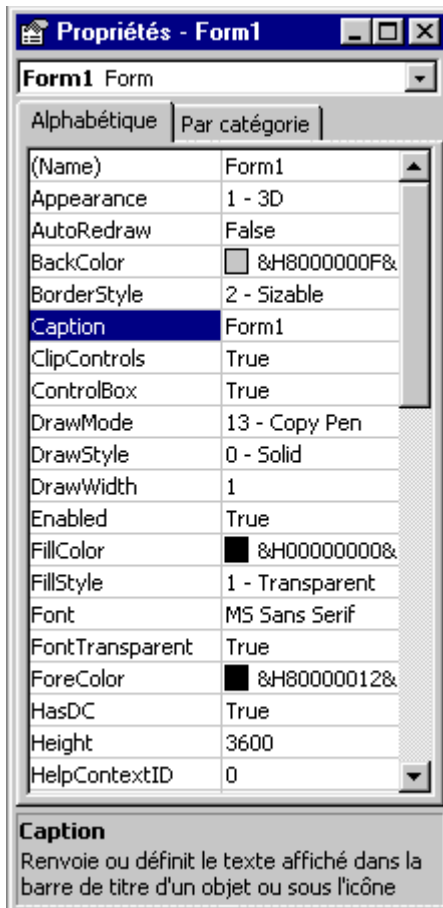
Pour afficher la fenêtre de propriétés, cliquez depuis la barre d'outils sur l'icône qui représente une main qui tient un fichier (le 2ème à partir de la gauche) ou appuyez sur la touche de fonction **F4** de votre clavier.

Dans le champ de saisie en haut de la feuille, vous lisez le nom de la feuille auxquelles se rapportent les propriétés. Il s'agit dans cet exemple de la feuille Form1.



Si j'insiste sur ce point c'est que tout simplement lorsque vous aurez un programme qui contiendra une dizaine de feuilles, il faudra afficher précisément la feuille dont les propriétés sont à modifier. Ne pas se tromper de feuille!

Nous garderons l'option représentée par l'onglet "Alphabétique"



La première propriété à modifier est le nom :

Name : Form1 sera remplacée par : frmAbout(Nom de la feuille)

Puis les autres propriétés caractéristiques :

BackColor : en cliquant sur le carré grisé nous ouvrons la palette et nous choisissons la couleur de fond souhaitée. (Noire)

BorderStyle : 2 - Sizable c'est à dire modifiable par l'utilisateur sera remplacée par : 1 - fixe l'utilisateur ne pourra pas modifier les dimensions de la feuille.

Caption : Form1 sera remplacé par : "A propos de... ". Ce champs de saisie correspond à l'écriture qui se trouve dans la barre de titre de votre feuille.

Icon : (Icon) permet d'insérer une icône, n'importe quelle icône fera l'affaire pourvu qu'elle ait une extension "*.ico"
Icône: *Fichier Image qui comporte une extension ico.*

Height : 3840 - Dimensions en hauteur de la feuille

Width : 6195 - Dimensions en largeur de la feuille

Une nouveauté par rapport aux premières versions de VB le champs de saisie en bas de la feuille de propriétés qui vous indique les caractéristiques de la propriété sélectionnée.

Sur cet exemple nous avons l'explication de la propriété **caption**

Chaque propriété à une valeur attribuée par défaut et qui est modifiable au choix du programmeur.

Cette modification peut se faire en cours de programmation comme nous le verrons plus loin.

La propriété **Caption** vous permet d'affecter une touche d'accès rapide à un contrôle.

Dans la légende, tapez le signe & juste avant le caractère que vous souhaitez désigner comme touche d'accès rapide. Ce caractère apparaîtra souligné. Il vous suffira d'appuyer sur la touche ALT et sur le caractère souligné pour placer le focus sur ce contrôle.

(nous reparlerons du focus en temps utile)

IV. Insertion d'objets

IV.1. La boîte à outils



Pour sélectionner un objet de la boîte à outils, déplacez le curseur de la souris sur l'élément que vous voulez choisir une info bulle apparaît et vous indique le nom de l'objet sur lequel vous pointez la souris.

Nous allons positionner sur la feuille les autres objets nécessaires à la présentation de notre feuille, à savoir :

- Un objet Label (Label1) qui nous servira de titre
- un objet Image (Image1) pour insérer votre logo sous forme d'icône
- un second objet Label (Label2) qui indiquera la version du programme
- un autre Label (Label3) qui indiquera l'usage du programme
- une autre image (Image2) pour insérer le logo sous forme d'image de notre structure
- un trait récupéré dans la boîte à outils
- un Label (Label4) devant nous donner le nom du programme ainsi que son copyright
- deux boutons (CommandButton1 et CommandButton2)
- un autre Label (Label5) qui servira d'étiquette
- 2 derniers Labels (Label6 et Label7) qui contiendront notre numéro de série

IV.2. Création de l'interface

Nous allons positionner sur l'interface de travail les objets nécessaires à la présentation de notre feuille, à savoir :

Label1 positionné en haut à gauche de la feuille, propriétés :

Height :390

Width : 4630

Caption : Help System Restorer ou le nom de votre choix

BackColor : Choisir la couleur Noire

ForeColor : Choisir la couleur Rouge

Ces deux dernières propriétés seront identiques à tous les objet positionnés dans cette feuille.

Label2 positionné juste en dessous de la première étiquette comprendra dans sa propriété :

Caption : Version 1.0

Je vous laisse le soin de lui donner les dimensions adéquates

Label3 sera placé en dessous de **Label2** et aura comme propriété :

Caption : Description de l'application

Suivi de :

Label4 avec comme propriété :

Caption : Programme de sauvegarde de la base de registre, ou descriptif selon votre choix.

Image1 sera placé juste à côté de **Label1** et dans son prolongement.

Les propriétés dimensionnelles s'ajusteront aux dimensions de l'image

Nous insérerons le fichier Image en cliquant sur la propriété Picture qui a pour effet d'ouvrir une boîte de dialogue de Windows qui vous permet de charger l'image de votre choix au format Bmp, Gif ou wmf.

Si vous n'avez pas de "Logos" sous la main téléchargez les images et l'exemple de l'application : [Télécharger ici](#)

Nous insérerons un objet **Line** (trait) en dessous de l'objet Image1, sans commentaires.

Label5, positionné en dessous du trait aura comme propriété :

Caption : Help System Restorer[Savereg] Copyright © 2001

Tous droits réservés Gilbert Miralles 30980 Langlade France

Ndl : Il est bien entendu que vous pouvez modifier cette saisie en la remplaçant par vos propres coordonnées.

Le sigle Copyright "©" s'obtient en appuyant sur les touches Alt du clavier alphanumérique + 0169 du clavier numérique.

Il ne nous reste plus qu'à positionner les deux derniers "Label" qui restent, à savoir :

Label6 qui est une étiquette qui aura comme propriété :

Caption :Sérial N° :

Label7 qui sera juxtaposé à l'étiquette **Label6** devant recevoir la valeur du numéro de série de votre réalisation.

Pour terminer, nous positionnons les deux boutons qui auront comme propriété :

CommandButton1

Caption : OK

CommandButton2

Caption : Infos Systeme...

Vous leur donnez comme dimensions des valeurs en rapport avec la présentation de votre feuille.

Vous pouvez agrémenter la présentation en modifiant la propriété "color" de certains textes.



La prochaine leçon sera réservée à l'écriture du code.

Tutoriel 3 : L'environnement de programmation de Visual Basic

Date de publication : Lundi 18 mars 2003 , Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilmir.developpez.com\)](http://gilmir.developpez.com)

[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. Faisons connaissance avec le langage de programmation objet

I-A. Généralités

Visual Basic, est en parfaite harmonie avec Windows. Le développement d'une application passe par les étapes suivantes :

- Dessin de l'interface d'utilisation, c'est à dire les fenêtres et leur constituants, à l'aide d'un outil interactif de dessin, "**l'environnement Visual Basic**".
- Valorisation initiale des propriétés qui sont des attributs ou caractéristiques de chaque élément de l'interface.

I-B. Ecriture du code BASIC

La programmation des applications Visual Basic est dite événementielle, par opposition à la programmation linéaire traditionnelle.

Ainsi, une application Visual Basic est constituée d'un ensemble de procédures indépendantes les unes des autres.

I-C. Les procédures

Une procédure comprend des instructions écrites à l'aide du langage *BASIC*. Elle est associée à un objet, c'est à dire à un des éléments d'une feuille, la feuille elle même, ou bien un bouton, une liste, un champ de saisie etc...

La procédure est appelée par Visual Basic lorsqu'il se produit un événement pour l'objet correspondant.

Si vous n'avez pas écrit de code dans une procédure chargée de traiter un type d'événement pour un objet donné, il ne se passe rien de particulier lorsque l'événement est généré.

Pour écrire le code d'une application, il convient donc de déterminer les événements auxquels on souhaite réagir, et pour quels objets.

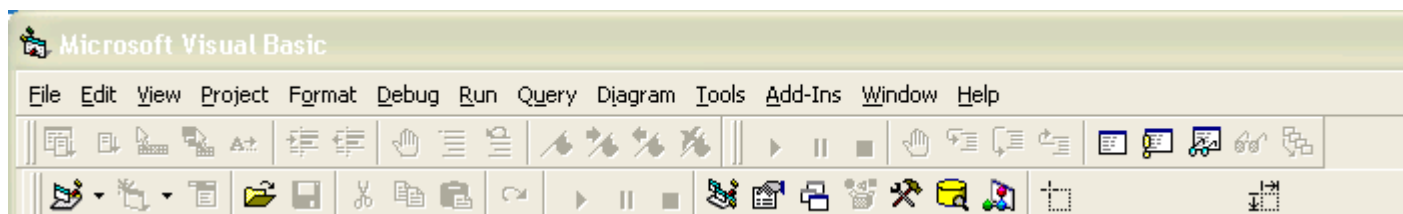
Cela détermine les procédures dans lesquelles le code est écrit.

II. La fenêtre principale (Interface graphique de conception)

II-A. 3 barres horizontales

La fenêtre principale comprend trois barres horizontales :

- **la barre de titre**, comme toute application Windows.
- **la barre de menu**, permettant la saisie des commandes, et
- **la barre d'outils** donnant un accès rapide aux principales commandes



II-B. La boîte à outils

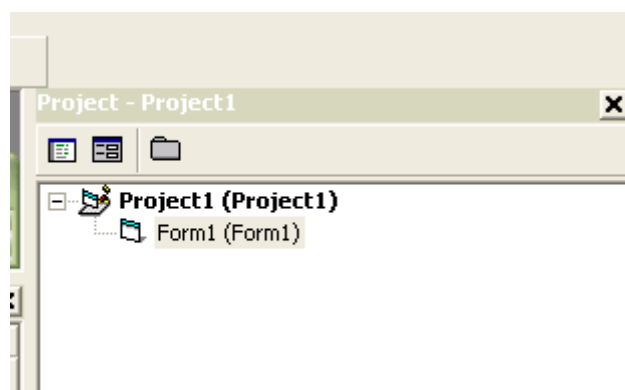
La boîte à outils, est une fenêtre qui comprend initialement tous les icônes visualisant des contrôles personnalisés (application complémentaire dont certains contrôles sont en option selon la version que vous possédez) pour la version standard 20 icônes.



Elle n'est accessible qu'en phase de conception (Mode Arrêt)

II-C. La fenêtre de projet

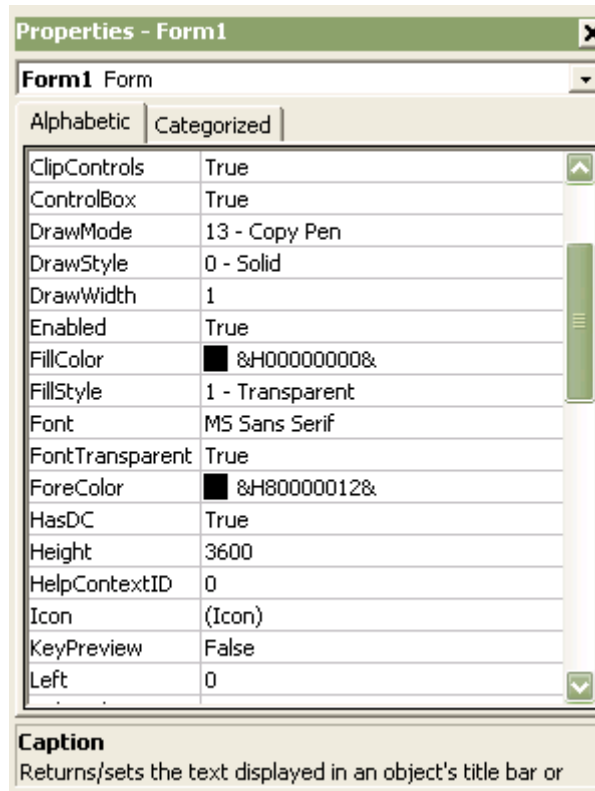
La fenêtre de projet comprend la liste des divers fichiers constituant une application.



(Pour l'instant nous n'avons qu'un seule fichier "Form1")

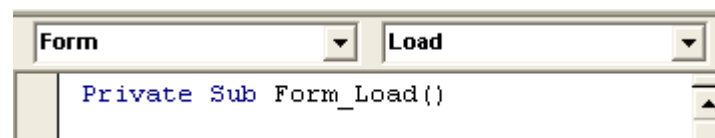
II-D. La fenêtre de propriétés

La fenêtre de propriétés comprend la liste des propriétés de l'objet sélectionné dans la feuille courante (feuille) ou contrôle) ainsi que leurs valeurs.



II-E. Procédures

En haut la liste des objets, présent définis dans le fichier sélectionné dans la fenêtre projet et à droite la procédure appelée par Visual Basic lorsqu'il se produit un événement pour l'objet correspondant



III. Avançons dans l'application

A ce stade de l'application, nous n'avons créé qu'une seule fenêtre, c'est donc la fenêtre principale. La fenêtre principale que j'appelle l'interface d'accueil est la feuille que l'utilisateur voit en premier lorsque le programme est lancé.

Nous pouvons construire autant de feuilles que l'on veut et déterminer par la suite qu'elle est la feuille qui deviendra l'interface d'accueil.

Cette possibilité s'obtient par le paramétrage de l'environnement de travail de Visual Basic.

Cliquez dans la barre de menus sur l'étiquette "Projet", puis sur "Propriétés de projet1", sélectionnez "Objet de démarrage".

Si le champ de saisie comprend la valeur "Sub main", modifiez et écrivez "Form1" qui est le nom de la feuille que nous avons créée et sur laquelle nous sommes en train de travailler.

Par la suite nous modifierons la valeur de l'étiquette pour saisir le nom de la feuille que nous souhaitons lancer au démarrage du programme.

Avant de nous lancer dans la super production qui va nous tenir en haleine pendant quelques heures, je me dois de vous instruire de quelques notions qui me paraissent indispensable pour la bonne compréhension de la suite du programme.

III-A. Un tout petit peu de Théorie

III-A-1. Méthodes

Sous Visual Basic, certaines fonctions ne sont accessibles que combinées avec des objets. Ces instructions sont dénommées méthodes. Par exemple la méthode [Show](#) charge et affiche une fenêtre définie par l'utilisateur (vous en l'occurrence) (se référer aux sources pour les autres méthodes)

III-A-2. Programmation événementielle

Les programmes Visual Basic sont commandés par les événements générés par les objets de l'interface utilisateur.

Par exemple, le chargement d'une feuille est un événement auquel certaines instructions peuvent être rattachées. Comme par exemple l'effacement d'une feuille dans la mémoire.

L'événement sera causé par un clic de souris, dans le cas d'un bouton, ou par une modification du contenu d'une zone de texte.

III-A-3. Domaine de validité des variables et des procédures

Comme tout langage moderne, Visual Basic connaît les variables globales, locales et statiques. Les variables globales sont accessibles depuis le programme entier mais les variables locales ne sont connues que dans leur procédure. Les variables statiques conservent leur valeur à la sortie de la procédure, ce qui les distingue des variables locales.

III-A-4. Compilation

Quand un projet a été compilé, son extension devient EXE, et il n'est plus possible de le "démonter" : tous les fichiers qui le composent sont désormais intimement associés.

Les images installées dans un contrôle sont elles aussi intégrées à l'exécutable. Seuls les fichiers OCX ou TXT restent autonomes, et devront être livrés à l'utilisateur.

Les fichiers DLL devront également être fournis avec l'application.

III-A-5. Les types de données

Suffixe	Type de données	Taille	Limites d'utilisation
%	Integer	2 oct	de -32.768 à 32767
&	Long	4 oct	de -2.147.483.648 à 2.147.483.647
!	Single (à virgule flottante en simple précision)	4 oct	entre -3,402823E38 et -1,401298E-45 pour les nombres négatifs et entre 1,401298E-45 et 3,402823E38 pour les positifs
#	Double (à virgule flottante en double précision)	8 oct	entre -1,79769313486232E308 et -4,94065645841247E-324 pour les nombres négatifs et entre 4,94065645841247E-324 et 1,79769313486232E308 pour les positifs
@	Currency (Virgule fixe 15 chiffres à gauche du séparateur décimal et	8 oct	entre -922 337 203 685 477,5808 et 922 337 203 685 477,5807

	4 chiffres à droite)		
\$	String	1 oct	par caractère
rien	Variant	nombre requis	

III-A-6. Quelques précisions

De ce tableau on retient que si l'on doit déclarer une variable alphanumérique (caractères textes) on choisira une variable de type **String**

Pour une opération avec des chiffres ou des opérations de comptage simples, on choisira une variable numérique de type **Integer**, ou **long** pour des opérations importantes (voir limites).

Pour les calculs monétaires, on choisira la variable de type **Currency**.

Le type par défaut d'une variable est **VARIANT**. Ce type indique à Visual Basic que la donnée est susceptible de contenir divers types de données.

Je déconseille d'utiliser la variable de type **Variant**, sauf cas de force majeure.

III-A-7. Qu'est ce qu'une variable ?

Les instructions sont constituées de verbes qui agissent sur des variables.

Une variable peut être considérée comme une case mémoire. Elle possède un nom et une valeur.

Le nom est constitué de lettres, de chiffres et de caractères <_> (le souligné), le premier caractère est obligatoirement une lettre et il ne peut y avoir plus de 40 caractères dans un nom de variable.

III-A-8. Déclarations de variables

La déclaration d'une variable se fait de plusieurs façons , à savoir :

Dim <nom de la variable> As <type de la variable>

Ex : pour déclarer une variable de type String (alphanumérique), que vous avez nommée <Bazile>, vous écrivez

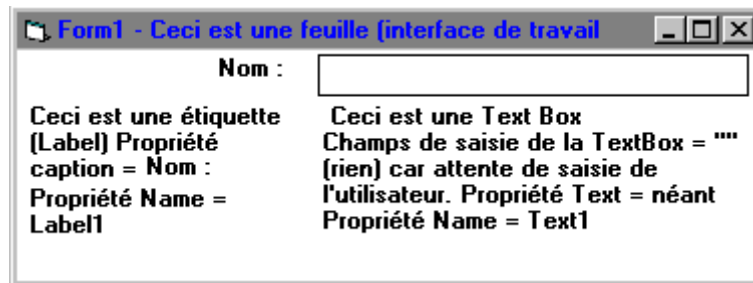
Dim Bazile As String représentée dans le programme par <Bazile\$>

Si la même variable est de type Integer (numérique), vous écrivez :

Dim Bazile As Integer représentée dans le programme par <Bazile%>

III-A-9. Propriétés d'un objet

Exemple d'un champ de saisie comportant une étiquette ayant comme **propriété Caption** : Nom, et comme **propriété Name** : Label1.



Ne pas confondre la propriété Name qui est le nom par défaut de l'objet, et la propriété Caption qui peut avoir n'importe quel nom et qui dans la cas présent est le Nom de l'utilisateur à saisir. Vous mettez un certain temps à vous y faire, mais, vous comprendrez avec un peu d'expérience.

La propriété Name est le nom que vous donnez à l'objet dans la phase de développement. Cette propriété n'est pas visible par l'utilisateur. Par contre la propriété Caption est le nom que vous donnez à la valeur de l'étiquette, ici dans l'exemple c'est Nom : mais cela aurait pu être <Adresse> ou <Ville> etc...et cette propriété est visible par l'utilisateur, mais au contraire de la TextBox celle-ci n'est pas modifiable.

III-C. Première lignes de code

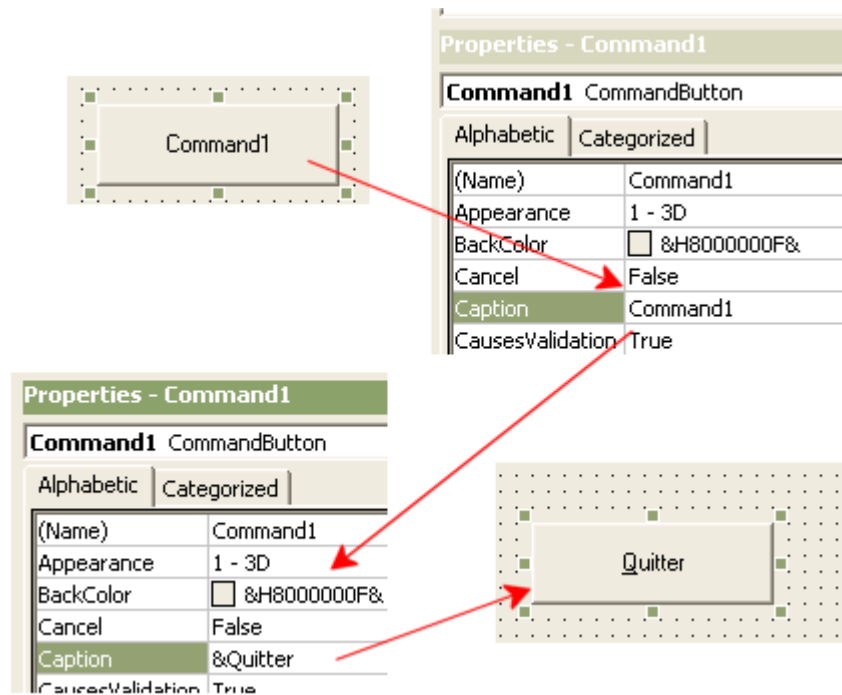
Nous allons essayer de mettre en pratique les formules précédemment acquises.

Lorsque vous travaillez sur l'interface de travail vous vous trouvez en mode développement.

Vous pouvez contrôler le bon déroulement des instructions ou fonctions écrites dans les pages de code en passant en mode exécution.

Nous allons écrire notre première ligne de code et visualiser le fonctionnement en mode exécution.

Ecrire dans la propriété Caption de bouton de commande "Command1" &Quitte.



Nous allons nous servir à titre de démonstration de ce bouton pour quitter notre application.

En effectuant un "DoubleClick" sur le bouton, nous ouvrons une feuille qui est la feuille de code de l'objet qui vient d'être sélectionné.



Vu de la feuille de code correspondante au bouton de commande "Command1"

Cette feuille se présente avec une "ComboBox"(Command1) qui vous permet en cliquant sur le petit bouton de droite de dérouler une liste qui comprend tous les noms d'objets utilisés dans cette feuille(Form1).

La "ComboBox de droite(Click) liste toutes les propriétés disponibles pour l'objet sélectionné.

Nous devons générer un événement lorsque l'utilisateur appuiera sur le bouton, et lorsqu'il générera un clic avec le curseur de la souris.

Nous sélectionnerons donc l'événement "Click".

Ecriture du code dans la feuille de code du bouton de commande "Command1" avec l'événement "Click"

Nous écrivons notre première procédure en saisissant:

```
Private Sub Command1_Click ( )  
    'Sortie du programme  
        End  
End Sub
```

Vous avez vu que j'ai mis une apostrophe devant la ligne de texte "'Sortie du programme"
Ces lignes commençant par une apostrophe sont nommées "lignes de commentaires. Il est conseillé en effet d'indiquer sous forme de texte l'explication de l'événement que l'on a voulu effectuer. Cela sera très utile surtout lorsque notre programme sera composé de plusieurs dizaines de procédures.

Ne pas s'en priver d'autant que le compilateur ignore complètement les lignes de commentaires.

Si vous êtes un peu curieux (il faut l'être en programmation) et que vous avez cliqué sur "Général" dans la feuille de code vous constatez que l'interface comprend une instruction "Option Explicite". Cette instruction force le développeur à déclarer toutes les variables à utiliser dans son programme. Si vous oubliez de déclarer une variable en phase de conception, le programme vous rappellera à l'ordre et vous indiquera une erreur.

Puisque nous venons de créer notre première procédure, essayons de la lancer pour visualiser ce qu'il se passe.

Dans la barre d'outils, cliquez sur le bouton exécuter



formé d'un petit triangle.

La feuille doit s'afficher, et lorsque vous cliquez sur le bouton "Quitter", l'application doit se fermer.



Visual Basic écrit sous la couleur bleue tous les mots dits "Réservés", vous ne pouvez pas les utiliser en dehors de la fonction établie par l'environnement de programmation.

La prochaine leçon sera réservée à la création de tableaux.

Tutoriel 4 : Les tableaux

Date de publication : Lundi 18 mars 2003 , Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilbert.miralles.com\)](http://gilbert.miralles.com)

[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. Les tableaux

I.1. Définition

Un tableau, c'est dans le cas d'une réalisation graphique, la déclaration de l'espace écran sous forme d'emplacement graphique.


je m'explique, si nous prenons un écran de configuration graphique de 800x600, cela veut dire que l'écran comprend 800 pixels sur une ligne, et 600 pixels sur une colonne. Un pixel représentant un point sur l'écran.

Ex : Un pixel = ceci [.]

Chaque partie de l'écran étant représentée, si nous voulons animer un graphique, nous devons déclarer dans le programme, c'est à dire à Visual Basic, la configuration d'écran que nous allons utiliser.

Dans l'une des applications que j'ai déjà développées ("Patman"), j'ai décidé que le tableau comprendrait 456 cases, comment j'ai fait pour trouver ce chiffre ?

J'ai dessiné plusieurs éléments graphiques devant représenter l'interface de travail, pour

l'exemple, je vous reproduis un de ces éléments  (le voilà), il mesure 23x23 pixels. Pour construire l'interface graphique dans l'espace écran dont je disposais, j'ai calculé que je pouvais mettre dans la longueur de l'écran 19 éléments, et dans la largeur de l'écran 24 éléments. Le calcul est simple 19 que multiplie 24 égale 456.

Ces chiffres peuvent être modifiés en fonction de la configuration d'écran et des dimensions des éléments que vous avez dessinés.

I.2. Comment allons nous déclarer ce tableau ?

Principe :

Dim <nom du tableau><liste de paramètres> **As** <type du tableau>

Généralement nous déclarons par exemple :

```
Dim Mon_tableau (1 To 12) As Integer
```

Dans le cas de l'exemple précité, nous déclarerons notre tableau avec les instructions suivantes :

- Global Const NB_CASES = 456
- Global Const NB_LIGNES = 19
- Global Const NB_COLONNES = 24

Si nous nous référons aux explications précédentes, nous en concluons que nous venons de déclarer un tableau qui comprend 19 lignes, 24 colonnes, 456 cases, qui sont des constantes donc qui ont des valeurs fixes et qui sont déclarées globalement dans tous le programme, donc qui sera pris en compte par toutes les feuilles de travail s'il y en a plusieurs.

Nous déclarerons ces informations dans un module que nous nommerons par exemple : "Global.bas"

II. Les affectations ou traitements conditionnels

Nous terminerons ces exercices par les traitements conditionnels :

Etudions le " IF Then Else " :

<nom_de_la_variable>=<fonction><nom_de_la_variable><valeur><opération>

Explication :

Si (**If**) inclure une condition Alors (**Then**) donner instructions pour appliquer la condition si la condition peut être exécutée, **Else** (nouvelle condition) donner de nouvelles instructions pour exécuter une nouvelle condition si la première n' est pas exécutée. **End If** Fin du traitement conditionnel.

If <condition> Then <instruction> Else <instruction>

```
If tableau(ligne, colonne) = Patman Then
    'Traduction = si le tableau représenté par les lignes et les
    colonnes est égal à patman(objet) alors, Lire les Instructions
    suivantes</li>
    ligne Patman = Patman 'la ligne du patman devient le patman
    (objet)
    colonne Patman = Patman 'la colonne du patman devient le patman
    (objet)<a name="contents"></li>
Else
    'Instructions (si la première condition n'est pas remplie)
    'Donner de nouvelles instructions</li>
End if
```



Prenez note que lorsque la condition n' est pas remplie, le pointeur de Visual Basic ignore les instructions et saute à l' instruction suivante.

Nous pouvons utiliser également le terme **Elseif** (ou bien) qui vous permet d'utiliser une autre instruction si celle utilisée précédemment n'est pas exécutée!

Vous constatez que devant le mot *Instruction*, j'ai mis une apostrophe ' ; en effet Visual Basic considère que toute écriture précédée d'une apostrophe n'est pas pris en compte par le compilateur.

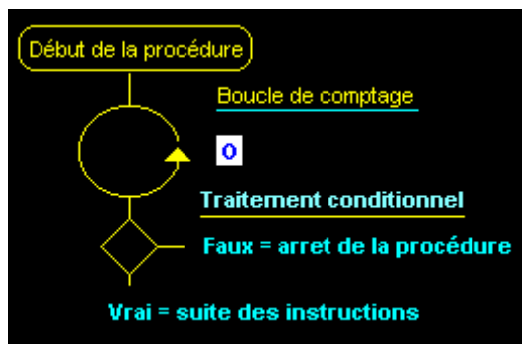
Donc toutes les lignes de commentaires sont précédées d'une apostrophe et servent au développeur à expliquer la forme et l'idée de son texte.

Je vous conseille de mettre le plus de commentaires possibles.

III. Les compteurs de boucles

III-1. Qu'est ce qu'une boucle ?

On peut comparer une boucle à un point qui circule dans un cercle et qui tourne sans fin.
On peut lui adjoindre un compteur qui incrémentera (ajouter 1) le nombre de tours chaque fois que celui-ci passera devant la case départ.
On peut également le faire sortir de la boucle au bout d'un certain nombre de tours.



- 1 - Démarrage du point
- 2 - exécuter un tour
- 3 - Contrôle du nombre de tours
- 4 - Si la valeur compteur est égale ou supérieure à 3, alors le point sort de la boucle
- 5 - Si la valeur compteur est inférieure à 3
- 6 - Ajouter 1 à la valeur du compteur et retourner à la case départ

III-2. Etudions la boucle "For ... Next"

```
For <compteur> = <debut> To <fin> Step <pas>  
<bloc d'instructions>  
Exit For  
<bloc d'instructions>  
Next <compteur>
```

Exemple d'utilisation : On veut comptabiliser les mois de l'année dans une boucle, on écrira :

```
For Mois = 12 To 1 Step -1  
'Instructions  
Total = Total + Montant(Mois)  
Next Mois
```

Dans cet exemple on décrémente (Comptage négatif) de 12 à 1 la variable **Step** indique le pas de décrémentation, et dans ce cas nous souhaitons décrémentation de 1 (mois) si nous avons souhaité effectuer des calculs basés sur un trimestre nous aurions mis comme "valeur de pas" le chiffre 3 (Step - 3)

La ligne instruction exécute le programme demandé et le **Next** est la variable de fin de procédure.

Si vous souhaitez comprendre au mieux l'opération de comptage avec les boucles, cliquez sur le lien qui vous ouvre le fichier [\[EXERCICE1\]](#)

IV. Les principaux opérateurs arithmétiques et de comparaison

IV.1. Les opérateurs arithmétiques

Opérateur	Signification	Exemple
+	Ajoute deux nombres	$a + b$
-	Soustrait deux nombres ou rend négatif un nombre	$a - b$ ou -92
*	Multiplie deux nombres	$a * b$
/	Divise deux nombres	a / b

IV.2. Les opérateurs de comparaison

Opérateur	Signification	Exemple
<	Inférieur	$a < b$ (a plus petit que b)
<=	Inférieur ou égal	$a <= b$
>	Supérieur	$a > b$
>=	Supérieur ou égal	$a >= b$
=	Egal	$a = b$
<>	Différent	$a <> b$
&	Retourne une chaîne de caractère qui est la concaténation des deux autres	Bonjour & messieurs, retourne "Bonjour messieurs"

D'autres traitements conditionnels seront étudiés en cours de programmation

Tutoriel 4bis : Notion d'algorithmie

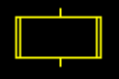
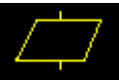

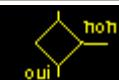


I. Introduction

Un algorithme est une règle, il s'exprime par une suite ordonnée de directives composée d'**actions** et de **décisions** qu'il faut exécuter en séquence suivant un enchaînement strict pour accomplir une Tache donnée, conforme à un cahier des charges.

Dans un automatisme, la succession des tâches logiques constituent l'algorithme de sa fonction globale.

L'algorigramme reproduit dans un langage graphique normalisé tous les cheminements du raisonnement logique qui détermine la composition de l'algorithme.

II. Structures algorithmiques

Symbole	Désignation
	Sous programme - Portion de programme considéré comme une simple opération.
	Entrée - Sortie Mise à disposition d'une information à traiter ou enregistrement d'une information traitée.
	Préparation - Opération qui détermine partiellement ou complètement la voie à suivre dans un embranchement ou un sous programme.
	Embranchement - Exploitation de conditions variables impliquant le choix d'une voie parmi plusieurs.
	Début, fin ou interruption d'un organigramme, point de contrôle, etc...
	Opération ou groupe d'opérations sur des données, instructions, etc..., ou opération pour laquelle il n'existe aucun symbole normalisé.

II-A. Sens conventionnel des liaisons

Le sens général des lignes de liaison doit être :

- de haut en bas
- de gauche à droite

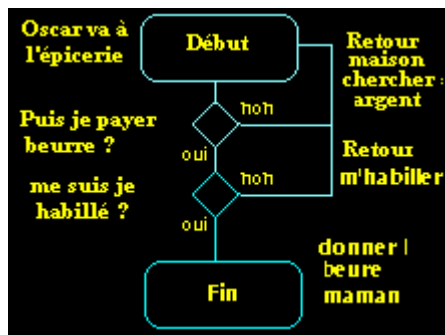
II-B. Deux exemples de structures algorithmiques

Premier exemple

C'est la maman d'Oscar qui l'appelle et qui lui dit :

Oscar cours vite à l'épicerie, achète moi un pain de beurre, prend de l'argent dans mon porte monnaie, habille toi bien parce qu'il fait froid.

Si on crée une application en fonction du cahier des charges à notre disposition, et ceci sans créer l'algorithme, il va s'ensuivre ce qui suit :



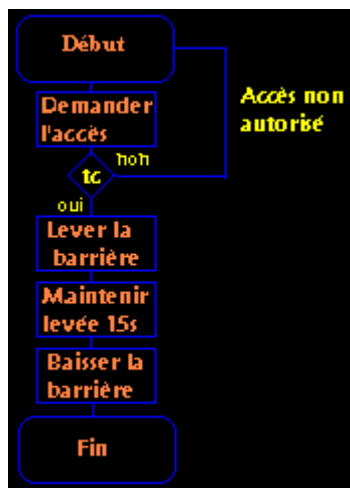
Oscar va à l'épicerie, demande du beurre,
traitement conditionnel1(Tc1)à t'il de l'argent ?
, non, pas d'argent,
alors il retourne chercher l'argent,
traitement conditionnel1(Tc1), à t'il de l'argent ?
oui, alors il prend le beurre, (T2c2) est il habillé ?
, non, retour à la maison On recommence, s'habiller,
aller à l'épicerie, premier Tc, réponse oui, deuxième Tc, réponse oui, retour à la maison et donner beurre à maman.
FIN

Cela peut paraître cousu de fil blanc pour certains, pourtant combien de fois à t'il fallut reprendre un programme tout simplement parce que l'on avait oublié une étiquette, une feuille, ou une réponse à un message d'erreur.

Ce petit exemple veut vous montrer qu'un programme est une suite logique d'événement qui s'enchaîne les uns aux autres comme un maillon d'une chaîne de vélo, et que, s'il manque un maillon, ou si le maillon n' est pas à sa place, et bien la chaîne déraile.

Deuxième exemple :

Cela se passe de commentaires

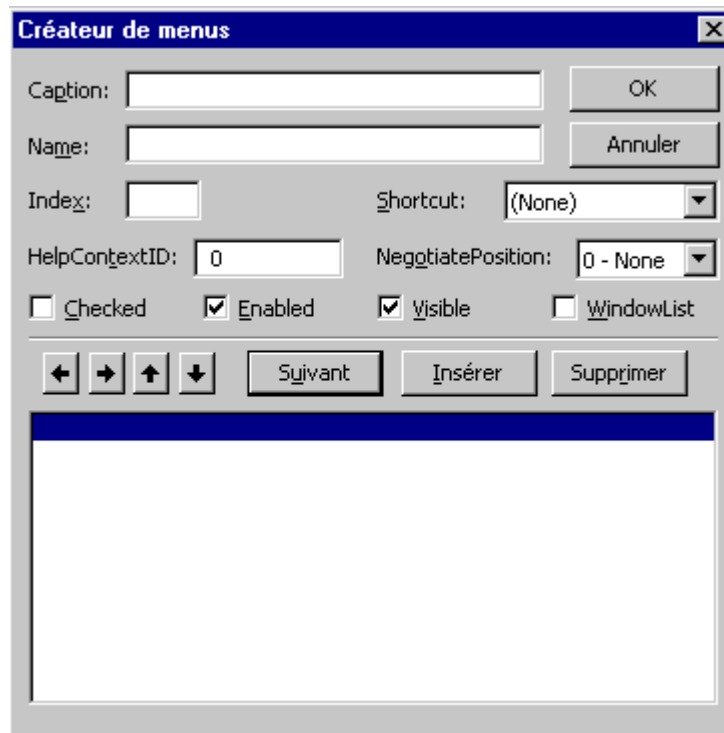


I. Introduction

Un contrôle **Menu** affiche un menu personnalisé pour votre application. Un menu peut inclure des commandes, des titres de sous-menus et des barres de séparation. Vous pouvez créer des menus dans lesquels figurent jusqu'à quatre niveaux de sous-menus.

II. Créateur de menus, boîte de dialogue

Pour créer un contrôle **Menu**, utilisez le Créateur de menus. Entrez le nom du **Menu** dans la zone Caption. Pour créer une barre de séparation, tapez simplement un trait d'union (-) dans la zone Caption. Pour afficher une coche à gauche du nom d'un élément de menu, activez la case à cocher Checked.



Bien que certaines propriétés des contrôles **Menu** puissent être définies dans le Créateur de menus, toutes les propriétés de ces contrôles sont disponibles dans la fenêtre Propriétés. Pour afficher les propriétés d'un contrôle de **Menu**, sélectionnez son nom dans la zone Objet de la partie supérieure de la fenêtre Propriétés.

Lorsque vous créez une application MDI, la barre de menus de la feuille MDI fille remplace celle de l'objet **MDIForm** quand la feuille fille est activée.

II-A. Options de la boîte de dialogue

Caption

Vous permet d'entrer les noms de menu ou de commande qui apparaîtront sur votre barre de menus ou dans un menu.

Si vous voulez créer une **barre séparatrice** dans votre menu, tapez un simple trait d'union (-) dans la zone Caption.

Pour permettre à l'utilisateur d'accéder au menu par l'intermédiaire du clavier, insérez un caractère (&) avant une lettre. Au moment de l'exécution, cette lettre apparaît soulignée (le caractère & reste invisible), et l'utilisateur peut accéder au menu ou à la commande en appuyant sur ALT et la lettre. Si vous souhaitez qu'un caractère & apparaisse dans le menu, tapez deux caractères & consécutifs dans la légende.

Name

Vous permet d'entrer un nom de contrôle pour l'élément de menu. Ce nom est un identificateur utilisé uniquement pour accéder à l'élément de menu dans le code. Il n'apparaît pas dans le menu.

Index

Vous permet d'affecter une valeur numérique qui détermine la position du contrôle à l'intérieur d'un groupe de contrôles. Cette position n'a aucun rapport avec la position à l'écran.

ShortCut

Vous permet de sélectionner un raccourci clavier pour chaque commande.

HelpContextID

Vous permet d'affecter une valeur numérique unique pour l'identificateur de contexte. Cette valeur est utilisée pour trouver la rubrique appropriée dans le fichier d'aide identifié par la propriété **HelpFile**.

NegotiatePosition

Vous permet de sélectionner la propriété **NegotiatePosition** du menu. Cette propriété détermine si le menu apparaît dans une feuille conteneur, et, si oui, à quel emplacement.

Checked

Vous permet d'ajouter une coche à la gauche d'un élément de menu. Celle-ci est généralement utilisée pour signaler si une option à bascule est validée ou non.

Enabled





Vous permet de décider si l'élément de menu doit répondre à des événements, ou doit être grisé si vous voulez qu'il soit indisponible.

Visible

Vous permet de rendre l'élément visible dans le menu.

WindowList

Détermine si le contrôle Menu contient une liste des feuilles MDI fille ouvertes dans une application MDI.

-  **Flèche vers la droite**
Déplace le menu sélectionné d'un niveau vers le bas lorsque vous cliquez dessus. Vous pouvez créer jusqu'à quatre niveaux de sous-menus.
-  **Flèche vers la gauche**
Déplace le menu sélectionné d'un niveau vers le haut lorsque vous cliquez dessus. Vous pouvez créer jusqu'à quatre niveaux de sous-menus.
-  **Flèche vers la haut**
Déplace l'élément de menu sélectionné d'une position vers le haut à l'intérieur d'un même niveau de menu à chaque fois que vous cliquez dessus.
-  **Flèche vers la bas**
Déplace l'élément de menu sélectionné d'une position vers le bas à l'intérieur d'un même niveau de menu à chaque fois que vous cliquez dessus.

Zone de liste du menu

Zone de liste qui affiche une liste hiérarchique des éléments de menu. Les éléments de sous-menus sont indentés pour indiquer leur position ou leur niveau hiérarchique.

Suivante

Déplace la sélection vers la ligne suivante.

Insérer

Insère une ligne dans la liste, au-dessus de la ligne actuellement sélectionnée.

Supprimer

Supprime la ligne actuellement sélectionnée.

OK

Ferme le Créateur de menus et applique toutes les modifications à la dernière feuille sélectionnée. Le menu est disponible au moment de la création ; toutefois le fait de sélectionner un menu à la création ouvre la fenêtre Code pour l'événement Click de ce menu, sans exécuter aucun code d'événement.

Annuler

Ferme le Créateur de menus et annule toutes les modifications.

III. Apprenons à réaliser des barres de menu dans notre interface.

Donnons le "Focus" à notre feuille de travail en la sélectionnant tout simplement, la barre de titre s'affiche avec la couleur bleu que vous connaissez bien.

Cliquez dans la barre de menu de V.B sur l'étiquette "Fenêtre" puis sur l'étiquette "Création de menus".

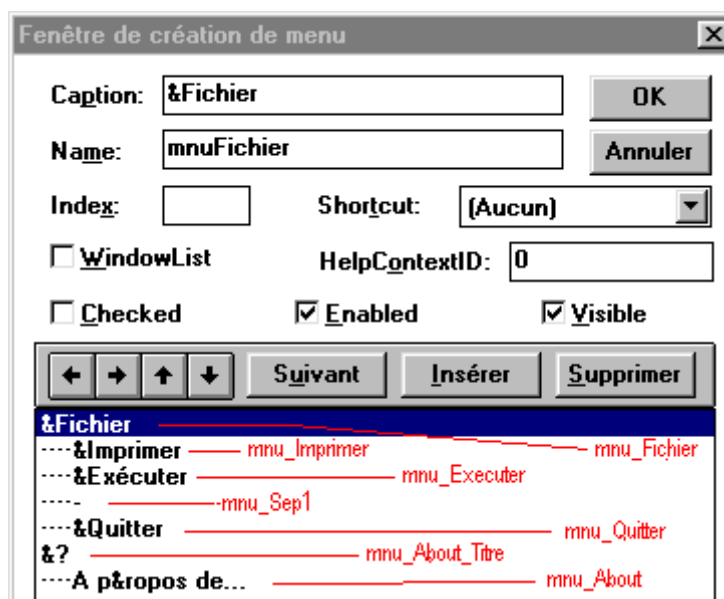
Ou bien,

Dans la barre d' outils de l'interface de VB, cliquez sur l'icône qui représente un fichier



le troisième en partant de la gauche. (Il est nécessaire qu'une feuille soit affichée)

Vous visualisez le créateur de menu.



La fenêtre de création de menus s'ouvre.

- 1ère étiquette - l'étiquette intitulée 'Fichier' (Etiquette Parent)
 - Dans la propriété "Caption"
 - Ecrire : [&Fichier] (sans les []) puis appuyez sur la touche tabulation (Tab) de votre clavier
 - votre curseur se positionne dans le champ de saisie intitulé "Name"
 - Ecrire : [mnu_Fichier], dans la propriété "Name", appuyez sur Tab(bouton clavier)

1ère sous étiquette - l'étiquette 'Imprimer' (Etiquette Fille)

- Appuyez sur le bouton " Suivant" pour ouvrir une nouvelle fenêtre de saisie.
- Ecrire : [&Imprimer] dans la propriété "Caption" appuyez sur Tab, puis,
- Ecrire : [mnu_Print], dans la propriété "Name", appuyez sur Tab,
- ensuite cliquez sur le bouton flèche droite pour créer une étiquette fille qui sera logée dans l'étiquette (Parent) intitulée "Fichier"

2ème sous étiquette - l'étiquette intitulée 'Exécuter' (Etiquette Fille)

- Appuyez sur le bouton " Suivant" pour ouvrir une nouvelle fenêtre de saisie.
- Ecrire : [&Exécuter] dans la propriété "Caption" appuyez sur la Tab puis,
- Ecrire : [mnu_Execut, dans la propriété "Name", appuyez sur Tab
- Si les quatre petits points avant la commande &Exécuter ne sont pas représentés alors, cliquez sur le bouton flèche droite pour créer une étiquette fille qui sera logée dans l'étiquette (Parent) intitulée "Fichier" sinon,

Insérer une ligne de séparation (Etiquette Fille)

- Appuyez sur le bouton " Suivant" pour ouvrir une nouvelle fenêtre de saisie. (les 4 petits points s'inscrivent automatiquement)
- Ecrire : [-] (tiret de séparation situé sur la touche "6" de votre clavier) dans la propriété "Caption ", appuyez sur "Tab"
- Ecrire : [mnu_Sep1, dans la propriété "Name", appuyez sur Tab

3ème sous étiquette - l'étiquette intitulée 'Quitter' (Etiquette Fille)

- Appuyez sur le bouton " Suivant" pour ouvrir une nouvelle fenêtre de saisie.
- Ecrire : [&Quitter] dans la propriété "Caption" appuyez sur la Tab puis,
- Ecrire : [mnu_Quitter, dans la propriété"Name", appuyez sur Tab

2ème étiquette - l'étiquette intitulée "?" (Etiquette Parent)

- Appuyez sur le bouton "Suivant" pour ouvrir une nouvelle fenêtre de saisie.
- Ecrire : [?] dans la propriété"Caption" appuyez sur la Tab puis,
- Ecrire : [mnu_About_Titre, dans la propriété "Name", appuyez sur Tab (observez que devant le caractère " ? " nous n'avons pas les 4 petits points (étiquette Parent)

1ère sous étiquette - l'étiquette intitulée "A propos de..." (Etiquette Fille)

- Appuyez sur le bouton " Suivant" pour ouvrir une nouvelle fenêtre de saisie.
- Ecrire : [&A propos de...] dans la propriété "Caption" appuyez sur la Tab puis,,
- Ecrire : [mnu_About, dans la propriété "Name", appuyez sur Tab ensuite cliquez sur le bouton flèche droite pour créer une étiquette fille qui sera logée dans l'étiquette (Parent)
- Appuyer sur le bouton "OK" ,contrôlez votre travail et sauvegardez le..Vous avez écrit votre première " **barre de menus**"

• Tutoriel 6 : Un environnement basé sur l'objet

• Environnement Objet...

- Vous avez dit objet ? Mais... de quel objet parlez-vous?
En déplaçant le curseur de votre souris, sur les différents objets insérés dans la boîte à outils, un petit rectangle de couleur apparaît et vous indique le nom de l'objet.



Les différents objets de la boîte à outils.

Dans cet exemple nous avons survolé avec la souris la dernière icône de la dernière rangée de droite, ce qui a provoqué l'apparition de la bulle qui nous indique que cet icône correspond à un contrôle "Data".

Etudions les différents contrôles de la boîte à outils.

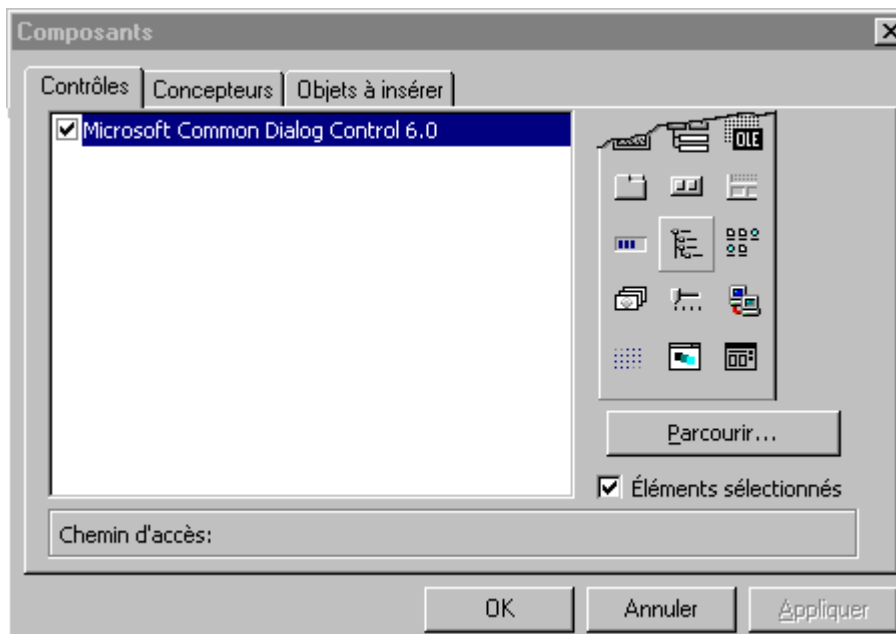
Nom du contrôle	Description
PictureBox	C'est un conteneur d'autres objets. Vous pouvez insérer une image ou par exemple un groupe d'objets en une seule opération
Label	On utilise ce contrôle pour placer du texte comme affichage simple ou comme étiquette.(L'utilisateur ne peut pas modifier le texte saisi dans ce contrôle)
TextBox	C'est le seul objet qui vous permet de saisir du texte, des nombres ou des dates.
Frame	Ce contrôle sert à enjoliver votre présentation. Un autre contrôle remplit le même rôle mais n'est pas livré en

	standard, c'est le contrôle Sheridan 3D Controls qui est généré par le fichier Threed32.ocx.(A posséder impérativement)
CommandButton	C'est un bouton poussoir qui enclenchera une action par l'intermédiaire d'une procédure événementielle.
CheckBox	C'est une case à cocher
OptionButton	C'est un bouton radio(Option)
ComboBox	Il s'agit d'une liste modifiable qui permet de choisir un seul élément dans une liste mais aussi de taper une valeur qui n'est pas affichée
ListBox	C'est une liste dans laquelle vous pouvez choisir un ou plusieurs éléments, sans pouvoir saisir de nouvelle valeur.
HScrollBar	C'est un ascenseur horizontal
VScrollBar	C'est un ascenseur vertical
Timer	C'est une minuterie, genre chronomètre qui vous permet d'enclencher une action toutes les n milli-secondes
DriveListBox	Ce contrôle permet d'afficher la liste de tous les lecteurs disponibles sur l'ordinateur
DirListBox	Ce contrôle permet d'afficher la liste de tous les répertoires d'un lecteur sélectionné
FileListBox	Ce contrôle permet d'afficher la liste de tous les fichiers d'un répertoire sélectionné
Shape	Permet de dessiner des formes, rectangles, cercles
Line	Permet de dessiner des lignes
Image	Sert à insérer des images, nous le préférons à l'objet PictureBox
Data	Ce contrôle permet de programmer l'accès aux bases de données
OLE	Il permet de placer des applications OLE(Object Link and Embedding)

- Avant de programmer avec Visual Basic, vous devez vous familiariser avec son environnement graphique, ses menus, ses barres d'outils ainsi que les fenêtres qui permettent de programmer et de paramétrer les différents objets. Si vous possédez la version "Pro", vous pouvez aller chercher des contrôles supplémentaires comme celui que j'ai cité dans le paragraphe précédent.

Dans la barre de menus, cliquez sur l'étiquette "**Projets**", "**Composants**"

-



En cliquant sur le bouton "Parcourir", vous ouvrez une boîte de dialogue de Windows. Il ne vous reste plus qu'à aller chercher le fichier qui vous intéresse, en l'occurrence "Threed32.ocx" qui se trouve dans le répertoire de c:\Windows\system. Vous pouvez également visualiser tous les fichiers avec extension *.ocx pour connaître les contrôles que vous avez à disposition.

- Nous essayerons à titre pédagogique de réaliser quelques petites applications qui englobent les contrôles "Treed32.ocx" et "Spin32.ocx" cela nous permettra de constater l'évolution esthétique de nos présentations.

Tutoriel 7 : Etude des principales propriétés d'une feuille Form1

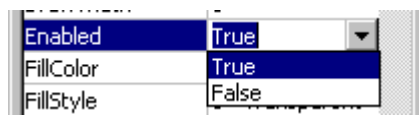
Date de publication : Lundi 18 mars 2003 , Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilmir.developpez.com\)](http://gilmir.developpez.com)

[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. Etude des principales propriétés d'une feuille Form1

Pour modifier une propriété, sélectionnez l' objet désiré(Form1), cliquez sur la ligne de la propriété que vous voulez modifier et utilisez la flèche dirigée vers le bas pour afficher les différentes valeurs



Comme indiqué sur la figure ci-contre. Modification des valeurs des propriétés en double cliquant sur le nom de la propriété ou en utilisant la flèche dirigée vers le bas.

II. Appel de la fenêtre des propriétés

Si la fenêtre des propriétés n' apparaît pas, appuyez sur la touche de fonction F4 ou sélectionnez dans le menu **Affichage/Fenêtre des propriétés**.

III. Propriétés d'une feuille FORM1 (les principales)

Propriétés	Signification
BackColor	Définit la couleur d' arrière plan
BorderStyle	Définit le style de la bordure d' un objet
Caption	Définit le texte dans la barre de titre de la feuille
ControlBox	Détermine si les boutons du menu système en haut à droite dans la barre de titre de la feuille sont affichés.
Enabled	Détermine si l' objet peut répondre aux événements, c'est à dire s'il est utilisable.
Font	Définit une police, un style et une taille de caractère
ForeColor	Définit la couleur de premier plan
Height	Définit la hauteur de l' objet à partir du haut
Icon	Indique l' icône qui est affichée quand la feuille est réduite
Left	Indique la position horizontale gauche à l' exécution de la feuille
MaxButton	Définit si une feuille contient un bouton "Agrandissement"
MidiChild	Définit si une feuille est considérée comme feuille enfant MIDI
MinButton	Définit si une feuille contient un bouton "Réduction"
MouseIcon	Définit une icône de souris personnalisée
MousePointer	Définit le type de pointeur de souris
tag	Permet de stocker des informations supplémentaires

Top	Détermine la position verticale de l' objet à exécution
Width	Indique la largeur de l' objet
WindowState	définit l' état visuel d' une feuille au moment de l' exécution

Vous pouvez tester toutes ces propriétés en vous positionnant en mode Exécution. Par contre pour modifier un paramètre, vous devrez vous placer en mode conception.

IV. Le mode conception

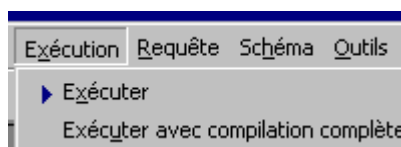
Vous réalisez votre interface, mise en place des objets, des propriétés, et écriture du code devant générer les événements souhaités.

Vous modifiez tous les paramètres qui vous semblent utiles au fonctionnement de votre réalisation.

Lorsque vous désirez visualiser le résultat de votre travail, vous devrez passer en mode exécution.

V. Le mode exécution

Vous exécutez la feuille par la commande du menu Exécution/Exécuter ou bien en appuyant sur la touche "F5" de votre clavier



Dans la barre de menus, cliquez sur l' étiquette Exécution, puis sur Exécuter.

Vous pouvez également exécuter votre application en mode pas à pas, c' est à dire en suivant ligne par ligne, procédure par procédure le cheminement de la lecture de votre code. Pour ce faire, appuyez sur la touche "F8" de votre barre de fonctions de votre clavier.

En exécutant cette manœuvre, vous avez exécuté votre programme qui se trouve positionné sur votre bureau derrière la feuille de procédure que vous avez visualisé.

Il faut faire un peu d'acrobatie pour déplacer la feuille de procédure pour rendre visible l'application que l' on veut exécuter afin de contrôler le cheminement des instructions et des fonctions du code.

C'est la procédure habituelle que vous emploieriez pour déboguer votre programme en phase finale de réalisation.

La compilation complète permet à votre application de compiler tout le code nécessaire dans l'environnement de développement. Quand vous choisissez Exécution/Exécuter ou bien quand vous appuyez sur la touche "F5" de votre clavier, seul le code nécessaire au démarrage de l'application est compilé.

VI. Comment placer des objets sur la feuille ?

Placer un contrôle

Double-cliquez sur l'objet. Celui-ci se place au centre de la feuille.
Cliquez une fois sur l'objet et dessinez le sur la feuille avec la souris. Quand vous relâchez le bouton de la souris, le contrôle est placé.

Sélectionner un ou plusieurs contrôles

Cliquez sur le contrôle. Des poignées s'affichent tout autour du contrôle.
Cliquez sur un contrôle, maintenez la touche "**Maj**" du clavier, puis cliquez sur un autre contrôle.
Dessinez un cadre fictif avec la souris tout autour des contrôles à sélectionner.

Déplacer un ou plusieurs contrôles

Sélectionnez les contrôles comme expliqué ci-dessus, puis déplacez en un.
Vous constaterez qu'ils se déplacent tous.

Cadrer un groupe de contrôles

Sélectionnez les contrôles à cadrer. Utilisez la commande du menu "**Format/Aligner**" et choisissez l'option que vous désirez..

Supprimer un contrôle

Sélectionnez le contrôle que vous voulez supprimer puis appuyez sur la touche "**Suppr**" du clavier ou cliquez sur le bouton "**Couper**" dans la barre d'outils.

NDL'A-

Je vous conseille d'utiliser les procédures que je viens de décrire pour dessiner des tableaux importants

devant comprendre 2, 3 voire 400 objets sur la feuille.

La marche à suivre est de réaliser manuellement la première ligne de votre tableau, de la sélectionner, puis de faire un copier coller de la ligne sélectionnée qui viendra se placer au dessous, et ainsi de suite jusqu' à la ligne finale.

Ex pour réaliser un tableau devant contenir 456 objets images de dimensions 32x32, on réalise la première ligne, on la sélectionne, puis on la colle en dessous de la première.(Procédure Copier/Coller) Les objets devront être indexés et "Vb" incrémentera automatiquement les objets.

Contrôler néanmoins le résultat. Le premier objet devant avoir l'index "0", et pour une grille de 256 objets, le dernier devra avoir l'index "255".

Visual Basic 6 est assez précis dans ce domaine contrairement à certaines versions précédentes ou le contrôle était vraiment nécessaire.



Tutoriel 8 : Utilisation primaire des contrôles

I. Introduction

Dans les pages qui vont suivre nous allons apprendre à utiliser les contrôles avec pour chacun leur code correspondant afin de les rendre opérationnels.

Certains contrôles sont faciles à programmer, d'autres le sont moins surtout pour les débutants aussi nous accompagnerons les explications de ces modules avec des exemples à télécharger.

Libre à vous d'utiliser cette option.

II. Les contrôles Visual basic

Les contrôles Visual basic sont des outils graphiques qui servent à construire l'interface utilisateur d'un programme. Ils sont regroupés dans la boîte à outils de l'environnement de programmation.

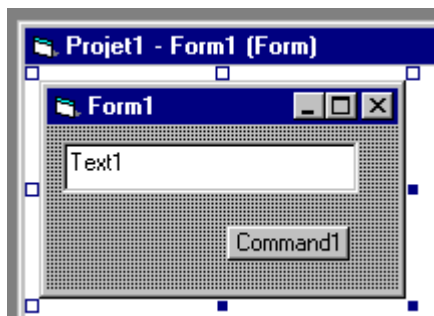
Il suffit de quelques clics et de déplacement de souris pour créer des objets sur une feuille. Vous apprendrez dans les prochaines leçons à afficher des informations dans une zone de texte, à parcourir les lecteurs et dossiers présents sur votre système, à traiter les données utilisateurs, à démarrer des applications Windows et à visualiser des enregistrements de bases de données.

Les exercices vous aideront à concevoir vos propres applications et vous feront étudier en détail les objets, propriétés et codes de programme.

II-A. Utilisation de l' objet "TextBox"

Dans cet exercice nous utiliserons deux contrôles, le contrôle "**TextBox**" (zone de texte) et le contrôle "**CommandButton**" (bouton de commande), ils sont tous deux présents dans la boîte à outils.

1. Démarrez Visual Basic et créez une nouvelle application "EXE standard".
2. Positionnez sur la feuille(Form1) nouvellement créée un contrôle TextBox
Placez le pointeur de la souris au milieu de la feuille (il se transforme en croix), et tracez une zone de texte similaire au cliché ci-après.
3. Positionnez sur la feuille un contrôle "CommandButton".



Une zone de texte sert à afficher du texte sur une feuille ou de traiter les saisies de l'utilisateur pendant l'exécution d'un programme.
L'utilisateur peut modifier le contenu de la saisie.

A la conception du programme ou en cours d'utilisation l'on peut modifier les propriétés de l'objet soit par la fenêtre de propriétés, soit en programmant les lignes de codes nécessaires à l'application de nouvelles instructions.

Dans notre exemple, l'objet zone de textes sera utilisé pour afficher un message lorsque l'utilisateur appuiera sur le bouton de commande.

II-B. Utilisation de l'objet "CommandButton"

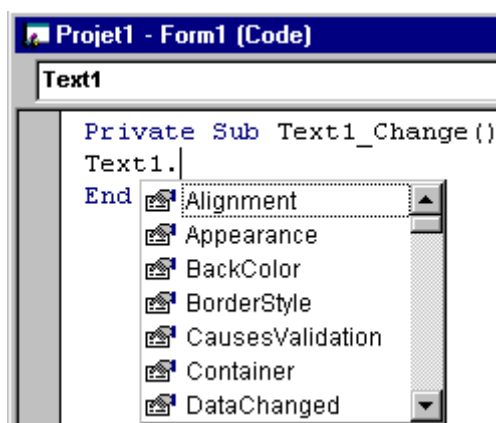
Un bouton de commande permet de traiter l'événement utilisateur le plus courant. Comme dans la vie courante lorsque vous appuyez sur un interrupteur dans votre salon, le lampadaire s'allume. Vous avez provoqué un événement, vous attendiez une réponse à celui-ci, en l'occurrence que le lampadaire s'allume. En programmation objet sous Visual Basic, il en va de même, l'utilisateur provoque un événement (lorsqu'il appuie sur le bouton) et attend en réponse l'exécution immédiate d'une action spécifique.

Le comportement du bouton doit répondre aux attentes de l'utilisateur.

Vous pouvez comme pour la feuille Form1 étudiée précédemment, modifier les caractéristiques des boutons de commande (comme celles de tous les objets) à l'aide des propriétés du contrôle utilisé. Vérifiez les possibilités offertes dans la feuille de propriétés. Lorsque l'on a positionné les objets sur la feuille comme indiqué sur la figure ci-dessus, on va apporter les modifications souhaitées, à savoir :

Contrôle	Propriété	Paramètre
Text1	Text	Vide(rien)
Command1	Caption	&Ok

Une particularité de la version 6.0 de Visual basic qui n'est pas à négliger pour les débutants, c'est la zone de liste qui affiche toutes les propriétés valides pour l'objet que vous utilisez. (Le caractère (perluette) écrit devant le "O" de "OK" sert simplement à souligner le premier caractère de chaque mot afin de créer un lien devant lancer la procédure par l'appui de ce caractère depuis le clavier.)



Cette zone de liste s'affiche lorsque vous saisissez dans la feuille de code le nom de l'objet, suivi du point.

Dans le cas présent il s'agit du contrôle "TextBox" qui, lorsque l'on écrit Text1 suivi du point(Text1.) déroulera la zone de liste.

Celle-ci vous indique toutes les propriétés disponibles valides pour ce contrôle.

Ecriture du code devant générer l'événement souhaité :

Effectuez un double-clic sur le bouton de commande "OK".

Une fenêtre s'ouvre, c'est la feuille des procédures (écriture du code) qui correspond à l'objet à traiter. Dans cette feuille nous trouvons déjà les instructions par défaut de l'objet.

```
Private Sub Command1_Click( )
```

```
End Sub
```

Si vous n'écoutez rien, il ne se passe rien.

Comme nous souhaitons générer un événement, nous allons écrire : Votre prénom suivi de votre nom! Ce qui donne :

```
Private Sub Command1_Click( )  
'Commentaires (écrire ici l' idée de l' événement à générer) je  
mettrais par Ex : 'Ecrire dans la TextBox  
Text1.Text = "Gaston Lagaffe"  
End Sub
```

Explications : **Private sub** est le début de la procédure "Command1_Click" Suivi par les commentaires (je précise que ceux-ci ne sont pas pris en compte par le compilateur, donc ne vous en privez pas) qui doivent obligatoirement commencer par une apostrophe.

Viennent ensuite les instructions qui doivent être prises en compte et exécutées par le programme.

Je traduirais cette instruction plus simplement pour les néophytes par :

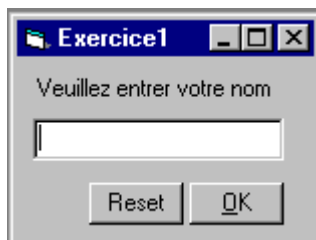
Dans l'objet intitulé Text1 et particulièrement dans la procédure Text, écrit la chaîne de caractères suivante "Gaston Lagaffe" (qui s'écrira obligatoirement entre guillemets).

Et **End Sub** qui indique que la procédure est terminée.

Vous pouvez également pour donner du cachet à votre réalisation créer un second bouton qui effacera le texte déjà écrit dans la TextBox comme par exemple :

Contrôle	Propriété	Paramètre
Command2	Caption	Reset

```
Private Sub Command2_Click( )  
'Effacer la zone de texte  
Text1.Text = ""  
End Sub
```



Nous pouvons également à titre pédagogique utiliser le contrôle "Label" Nous l'utiliserons particulièrement pour créer des étiquettes car contrairement au contrôle TextBox, le "Label" ne peut pas être modifié par l'utilisateur.

Exemple : Vous placez un "Label"(étiquette) au dessus de la "TextBox" et vous écrivez dans la propriété "Caption" de celui-ci : "Veuillez entrer votre nom " Ce qui modifiera la présentation de notre réalisation par la figure suivante.

Nous terminons notre réalisation en lui apportant les modifications décrites en aparté de la figure jointe et nous procédons aux essais en passant du mode "**Conception**" au mode "**Exécution**" comme expliqué dans les leçons précédentes.

IV. QCM




Vous pouvez si vous le souhaitez répondre à votre premier QCM (questions à choix multiples) réalisé à

l'intention de ceux qui suivent ces tutoriels. [QCM1 - Visual Basic 6.0](#)

Tutoriel 9 : Utiliser des objets systèmes de fichiers

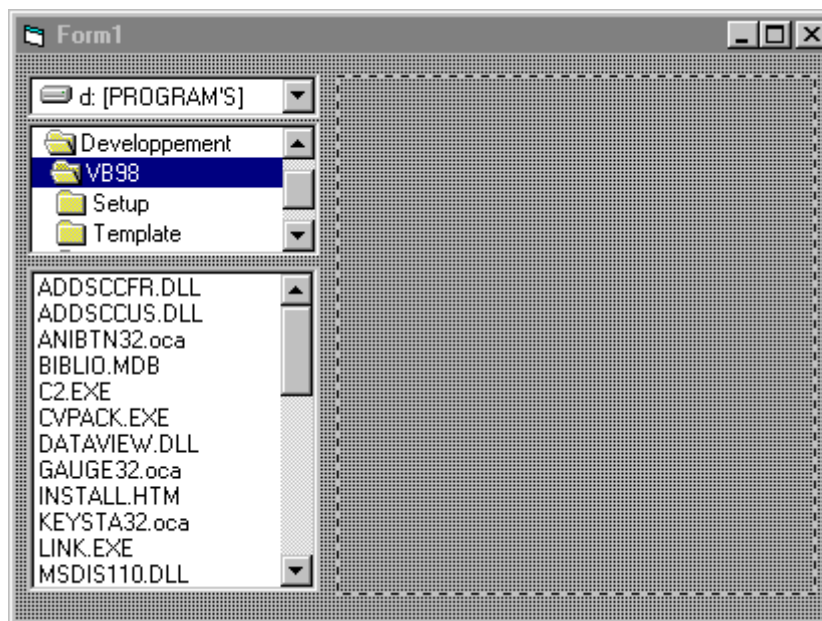
I. Introduction

Visual Basic propose trois objets très utiles permettant d'accéder au système de fichiers. Il s'agit des contrôles "DriveListBox"(liste de lecteurs), "DirListBox"(liste de répertoires), et "FileListBox"(liste de fichiers).

-  **La liste de lecteur** permet de parcourir les lecteurs valides de votre système.
-  **La liste de répertoires** permet de naviguer parmi les dossiers d'un lecteur particulier.
-  **La liste de fichiers** permet de sélectionner un fichier spécifique dans un dossier.

II. Application Explorateur

Vous pourrez télécharger en option le fichier compressé de l'application "Explorateur" qui reprend les contrôles que nous venons d'évoquer.



Voici le prototype de l'application "**Explorateur**"

Les dimensions n'ont pas grande importance, à vous d'ajuster les paramètres en fonction de vos aptitudes créatrices.

Vous pouvez passer en mode Exécution pour voir ce qui se passe ?

Un objet "FileListBox"(liste de fichiers) permet à l'utilisateur de sélectionner un fichier spécifique dans le système de fichier.

Visual basic insère alors le nom du fichier sélectionné dans la propriété "FileName" de l'objet.

Les sélections appropriées effectuées par l'utilisateur sont récupérées également à l'identique par les objets :

DriveListBox qui sera inséré dans la propriété "Drive" de l'objet et, DirListBox sera inséré dans la propriété "Path" de l'objet.

Nous utiliserons ces trois propriétés dans le programme Explorateur.

L'utilisateur change le paramètre d'un objet durant l'exécution d'un programme, cette modification engendre un événement qui est traité par Visual Basic dans le code du programme.

Nota - Les propriétés "Drive", "Path" et "FileName" ne sont accessibles qu'en phase d'exécution. Il est impossible de les définir à l'aide de la fenêtre de propriétés.

Nous allons définir les propriétés suivantes en utilisant la fenêtre "Propriétés"

Objets	Propriétés	Paramètres
File1	Pattern	*.bmp ; *.* ; *.wmf ; *.ico
Image1	Stretch	True
Image1	BorderStyle	1 - Fixed Single

Si nous n'avons pas utilisé la propriété "Pattern" la "FileListBox" aurait affichée tous les types de fichiers contenus dans un dossier avec le risque que l'utilisateur sélectionne un fichier qui n'est pas supporté par **Visual Basic**. Ce qui aurait pour effet de provoquer une erreur. Il est préférable dans la mesure du possible d'essayer d'éviter tout risque d'erreurs, tout au moins celles qui sont prévisibles.

Nous allons ajouter quelques lignes de code aux procédures qui sont associées aux objets. Ces procédures sont appelées "**Procédures d'événement**" car elles sont engendrées par un événement, par exemple *un clic de souris*.

On affiche par un double clic sur l'objet "DriveListBox" la fenêtre de code de l'objet (Drive1) et particulièrement dans la procédure "Change" on écrit :

```
Private Sub Drive1_Change ( )
'L' utilisateur à cliqué sur l'objet
Dir1.Path = Drive1.Drive
End Sub
```

Cette instruction actualise la propriété "**Path**" dans la liste de lecteurs lorsque l'utilisateur sélectionne un lecteur de disque dans la zone.

Avec cette simple ligne de code nous constatons que nous pouvons changer de lecteur et lire dans le contrôle DirListBox les répertoires se trouvant dans le lecteur sélectionné. Essayez de passer en mode Exécution pour visualiser le fonctionnement.

Nous allons effectuer la même opération avec la "**DirListBox**"

```
Private Sub Dir1_Change ( )
'L' utilisateur à cliqué sur l'objet
File1.Path = Dir1.Path
End Sub
```

Cette instruction lie la liste des fichiers à celle des répertoires de manière à ce que les fichiers présentés correspondent à ceux du dossier sélectionné.

Nous allons effectuer la même opération avec la "**FileListBox**" et ajouter le code suivant :

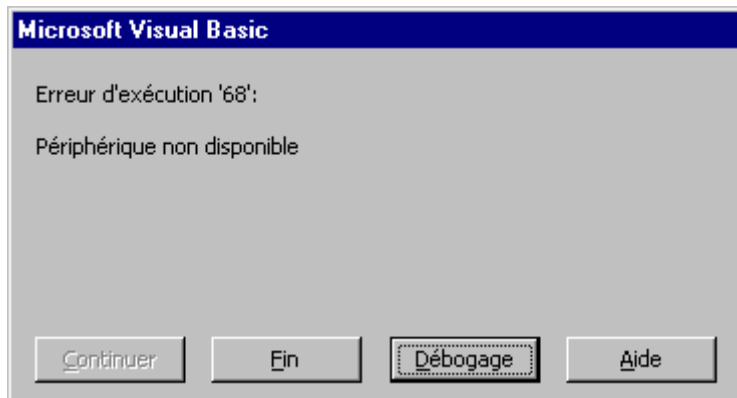
```
Private Sub File1_Click()
Dim SelectedFile As String
'L'utilisateur a sélectionné l'objet
SelectedFile = File1.Path & "\" & File1.FileName ' (&) Voir
opérateurs
Image1.Picture = LoadPicture(SelectedFile)
End Sub
```

Nous avons déclaré la variable SelectedFile, concaténé les instructions afférentes au fichier sélectionné puis rajouté l'anti-slash pour le cas ou ? Et avons donné au programme le chemin du fichier qui nous intéresse en l'occurrence celui que l'utilisateur a sélectionné dans la liste des fichiers valides.

Le paramètre 1(Fixed Single) de la propriété BorderStyle de l'objet "Image" nous permet de cadrer l'image sélectionnée en fonction de ces propres dimensions (pour l'esthétique).

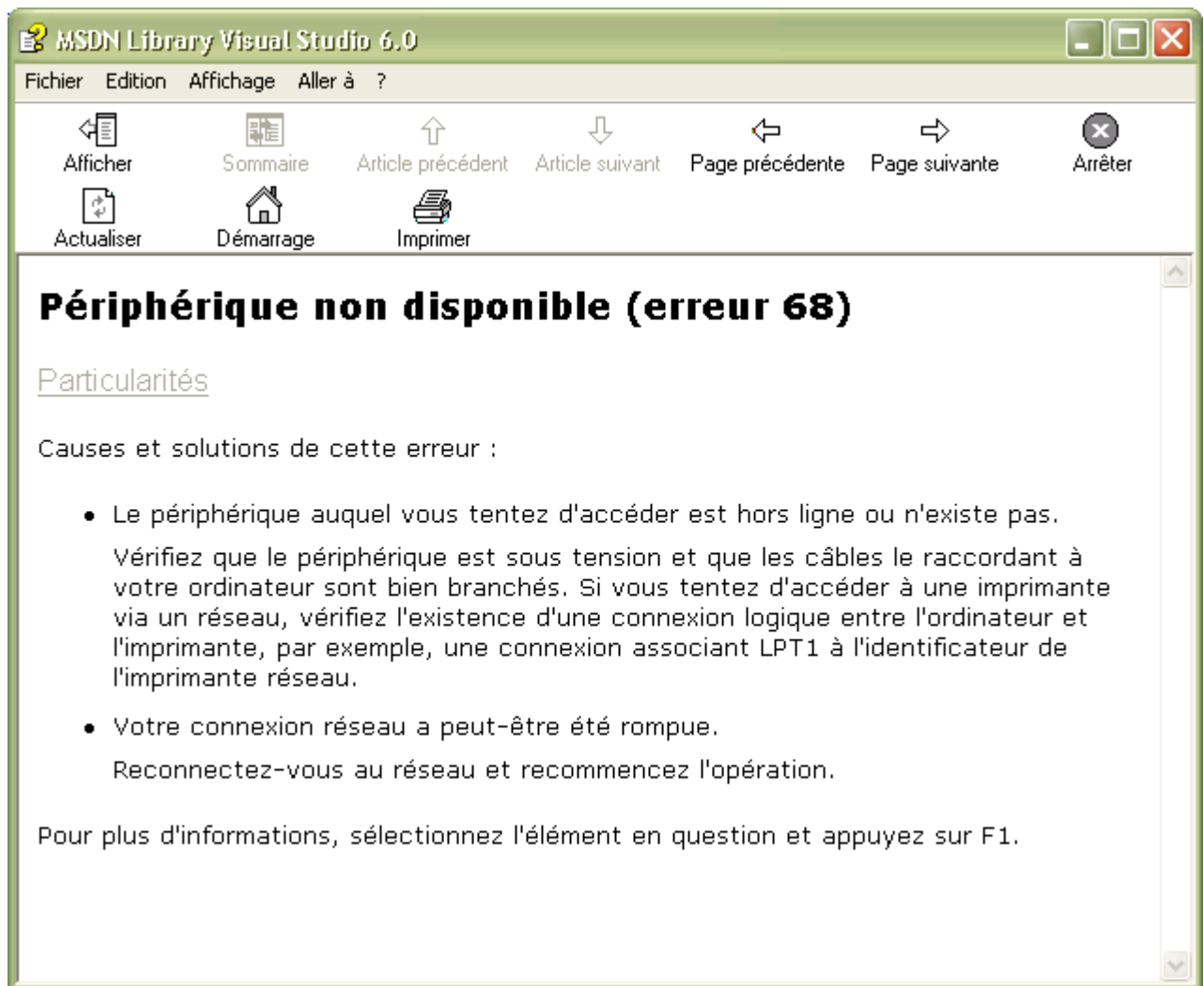
Exécutons le programme en procédant comme d'habitude.

Lorsque le programme se trouve en mode Exécution, sélectionnez le contrôle "Drive1" et choisissez le lecteur "A:", vous constatez que le programme(VB) génère une erreur d'exécution en ouvrant et affichant une fenêtre du style :



Voici la fenêtre affichée par Visual Basic lorsque vous générez l'erreur d'exécution n° 68 qui stipule que le périphérique est non disponible.

Pour ceux qui possèdent le programme Visual basic d'origine Microsoft, vous pouvez cliquer sur le bouton d'aide, ce qui vous permet de savoir que vous disposez d'une aide en ligne bien ficelée qui vous aidera dans bien des cas à trouver la procédure ou l'instruction recherchée. Cliquez sur le bouton d'aide et vous devez voir apparaître la fenêtre suivante:



Périphérique non disponible (erreur 68)

PARTICULARITES, Causes et solutions de cette erreur :

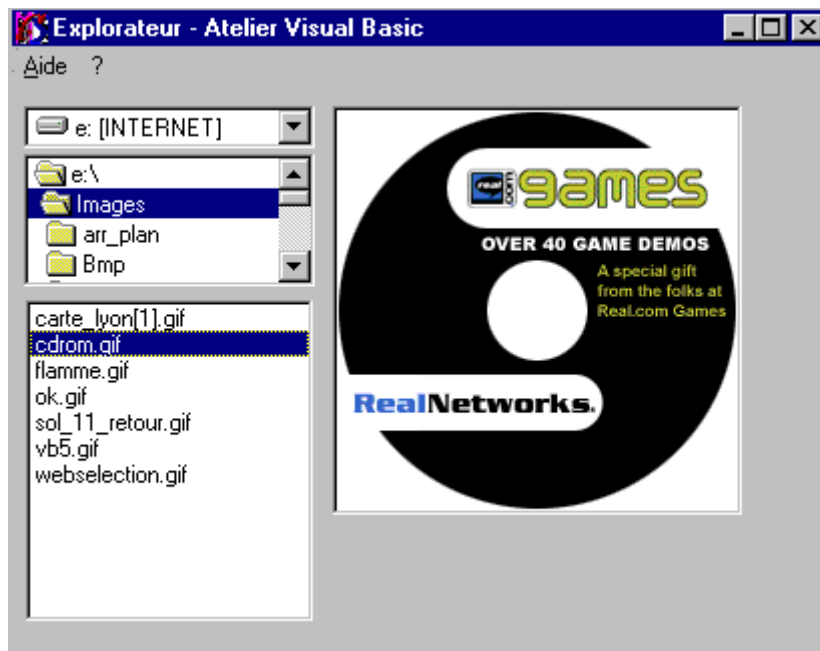
- Le périphérique auquel vous tentez d'accéder est hors ligne ou n'existe pas.
Vérifiez que le périphérique est sous tension et que les câbles le raccordant à votre ordinateur sont bien branchés. Si vous tentez d'accéder à une imprimante via un réseau, vérifiez l'existence d'une connexion logique entre l'ordinateur et l'imprimante, par exemple, une connexion associant LPT1 à l'identificateur de l'imprimante réseau.
- Votre connexion réseau a peut-être été rompue.
Reconnectez-vous au réseau et recommencez l'opération.

Pour plus d'informations, sélectionnez l'élément en question et appuyez sur F1.

Il est possible de générer d'autres erreurs avec ce programme, aussi sachez que pour résoudre ces problèmes, il faut utiliser des instructions qui contournent la cause de l'erreur, ou créer des routines spéciales nommées "Gestionnaires d'erreurs" qui vous permettront de contourner les problèmes rencontrés par cette application qui demande à être déboguée.

Nous apprendrons plus tard à écrire un gestionnaire d'erreurs et à rechercher les erreurs éventuelles.

Le programme "Explorer" est terminé, nous affichons le résultat de notre réalisation.



Le programme en lui même est très simple, mais si vous le souhaitez, vous pouvez télécharger les sources [ici](#)

Ndl'A- L'image affichée par le programme Visual Basic est un Logo de la firme Real.Com

Sauvegardez votre réalisation.

Dans le menu, cliquez sur l'étiquette "Enregistrez le projet sous" : vous pouvez modifier le nom de la feuille "Form1" par "Explorer" qui prendra aussitôt l'extension "Explorer.frm" et le projet que nous venons de réaliser "Projet1.vbp" sera sauvegardé sous le nom de "Explorer.vbp"

Vous pouvez également créer votre première application exécutable

Dans la barre de menu cliquez sur l'étiquette "Fichier" vous devez avoir dans la fenêtre qui vient de s'ouvrir une étiquette comportant la saisie : "Créer Projet1.exe" En cliquant sur cette étiquette vous ouvrez une fenêtre de dialogue de Windows qui comporte dans le champs de saisie le nom de votre projet.

Vous le renommez en lui donnant le nom de votre choix.

Dans l'exemple que je viens d'évoquer, je l'ai appelé "Explorat.exe" Vous avez pu constater que dans le libellé du nom, je n'ai saisi que 8 caractères.

Cela évite qu'il puisse être renommé par "Dos" (Si je l'avais affublé d'un nom Long")pour le cas ou un utilisateur le lancerait depuis cette plate forme.

Tutoriel 10 : La case à cocher, le bouton d'options et la zone de liste


Date de publication : Lundi 18 mars 2003 , Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilmir.developpez.com\)](http://gilmir.developpez.com)

[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. Le contrôle "CheckBox" (case à cocher)

La plupart des applications laissent une certaine marge de choix à l'utilisateur.

Le contrôle "CheckBox" (case à cocher)  permet d'affecter à une option ou un paramètre une valeur qui peut être Vrai ou Faux (True ou False). Les cases à cocher peuvent être rassemblées en groupe, chaque case étant activable individuellement et sans incidence pour les autres cases du même groupe.

Vous pouvez utiliser la propriété "ToolTip" qui permet d'ajouter au contrôle un court texte explicatif sous forme d'une Info-bulle lorsque le pointeur de la souris survole le contrôle un court instant. Une nouveauté dans la version 6.0 de Visual Basic qui propose un contrôle "CheckBox" graphique.

Utilisez pour cela la propriété "Picture" pour affecter à la case à cocher une icône pour l'état "non enfoncé", puis de la propriété "DownPicture" pour l'état "enfoncé" vous pouvez utiliser également la propriété "DisabledPicture" pour l'état désactivé.

Contrairement aux contrôles d'options boutons, la case à cocher est dotée de 3 états, l'état désactivé qui correspond à la valeur "0", l'état activé = 1 et l'état grisé = 2.

C'est la raison pour laquelle, nous ne pourrions pas utiliser des valeurs booléennes pour coder ce contrôle, nous utiliserons les valeurs citées précédemment.

Déterminer l'état de la case à cocher

Le code à utiliser est :

```
If CheckBox1.Value = 0 Then
    'Instructions
    Beep 'Emet un son sur le haut parleur
du PC lorsque le "Beep" est activé
Else
    'Instructions
    Beep
End If
```

Les états possibles des contrôles Case à cocher		
Valeur	Description	Constante
0	Désactivée(par défaut)	vbUnchecked
1	Activée	vbChecked
2	Etat intermédiaire	vbGrayed

En règle générale, et si vous ne savez pas quelle instruction écrire ou dans le cas d'un traitement conditionnel, ou si vous ne savez pas quelle direction va prendre votre pointeur(surtout si vous utilisez "Else" dans votre procédure), saisissez l'instruction "Beep" qui n'aura aucune influence sur votre programme mais qui vous indiquera exactement où se dirige votre pointeur, puis, si vous lancez l'application et que vous entendez le **Beep**, vous aurez la confirmation que l'instruction aura été prise en compte.

Vous n'aurez plus qu'à remplacer "Beep" par les instructions définitives.

Si vous voulez fixer en interne l'état d'une case à cocher comme étant "non cochée", c'est à

dire en écrivant du code, vous écrivez l'instruction suivante :

CheckBox1.Value = 0

Pour quelle soit cochée, vous écrivez :

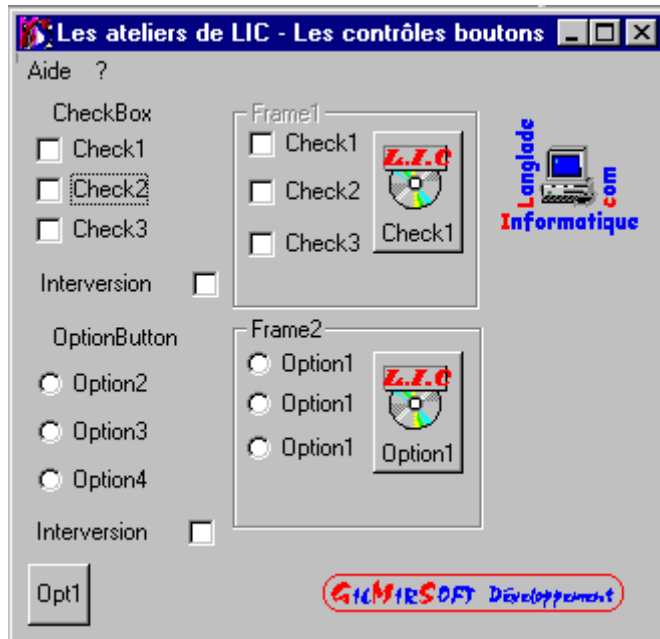
CheckBox1.Value = 1

Pour quelle apparaisse en grisé, vous écrivez :

CheckBox1.Value = 2

Le texte affiché à coté de la case à cocher est déterminé par la propriété "**Caption**".


Vous pouvez modifier l'alignement du texte à droite ou à gauche avec la propriété "**Alignment**".



Tous nos Tutoriels sont accompagnés d'exemples que vous pouvez télécharger sur notre site. L'interactivité du module vous permet de comprendre aisément le fonctionnement des exemples fournis [ici](#)

II. Le contrôle "OptionButton"

Le contrôle "**OptionButton**" accepte deux états pour l'option Vrai ou Faux (True ou False).

Ces "**OptionsButtons**" ou boutons d'options  peuvent être rassemblés en groupes de contrôles dans une "**Frame**" ce type de contrôle bénéficie d'une sélection exclusive. Vous pouvez également comme le contrôle décrit précédemment bénéficier de l'option graphique avec la propriété "**Picture**" en insérant une image en position bouton sélectionné et d'une autre image en position bouton désactivée.

Déterminer l'état du bouton d'option

Les boutons d'options connaissent deux états, l'état cochée qui correspond à la valeur booléenne "True" et l'état non cochée qui correspond à la valeur booléenne "False". Dans un groupe de contrôle seul un bouton peut être coché.

Exploitation d'un événement utilisateur avec 2 boutons d'option :

```
Sub Option1_Click ( )  
    If Option1.Value = True Then  
        'Instructions si l'option1 est activée  
    Else
```

```

        'Instructions si l'option2 est activée
    End If
End Sub

```

Exploitation d'un événement utilisateur avec 3 boutons d'option :

```

Sub Option1_Click ( )
    If Option1.Value = True Then
        'Instructions si l' option1 est activée
    ElseIf
        'Instructions si l' option2 est activée
    ElseIf
        'Instructions si l' option3 est activée
    End If
End Sub

```

Vous pouvez également comme le bouton précédent utiliser l' option "InfoBulle" avec la propriété "**ToolTip**".

On peut également affecter une valeur par l' intermédiaire du code, comme par exemple : Option1.Value = 1 ou dans l' hypothèse d' affecter une valeur dans une zone de liste de fichiers pour sélectionner une extension de fichier bien déterminée nous utiliserons le masque de recherche:


```

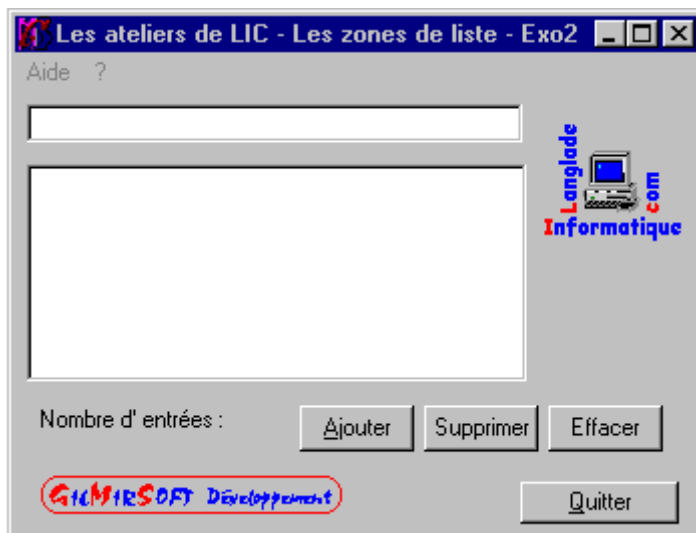
Sub Option1_Click ( )
    Fichier1.Pattern = "*.exe"
End Sub

```

III. La zone de liste et zone modifiable

Le contrôle zone de liste affiche une liste contenant

des entrées dont l' utilisateur peut sélectionner un ou plusieurs éléments.  Si le nombre d' éléments contenus dans la liste dépasse celui qui peut être affiché, une barre de déplacement(ascenseur) s' affiche automatiquement.



Vous pouvez [télécharger le module FileList](#) qui explique le fonctionnement des zones de liste. Téléchargez le module et saisissez un certain nombre d'entrées pour voir apparaître la barre de déplacement.

Une zone de liste peut être créée en mode "Conception" comme décrit dans l'exemple exo2, dans ce cas le remplissage de la zone de liste est réalisé par la procédure événementielle "Load" qui est appelée lors du chargement de la feuille.

En mode "Exécution" nous utiliserons la méthode "**AddItem**".


```
Objet.AddItem "Element"
```

L'argument "Objet" désigne le nom du contrôle, c'est à dire celui de la zone de liste.

L'argument "Element" est l'entrée qui doit être ajoutée.

Ex : lstZonedeliste.AddItem "**Help System Restorer**"

L'application décrite montre l'utilisation d'une zone de liste.

Elle comprend une "**TextBox**", une zone de liste, deux "**Label**", 4 boutons poussoirs.

Nous avons utilisé dans ce module les "**Méthodes**" "**RemoveItem**" et "**Clear**". La méthode "**RemoveItem**" nous permet après avoir sélectionné la ligne qui nous intéresse de pouvoir sélectivement la supprimer.

La méthode "**Clear**" quant à elle efface complètement le contenu de la liste.

Nous avons également utilisé la propriété "**ListCount**"

La propriété "List" est liée à la propriété "ListCount", qui renvoie le nombre d'entrées de la liste. Dans l'exercice Exo2 la propriété "ListCount" renvoie le nombre d'entrée que vous saisissez et rectifie le nombre indiqué lorsque vous effacez une ou plusieurs entrées. Vous pourrez étudier tous les autres contrôles dans les cours que nous diffusons sur notre site.

Tutoriel 11 : MsgBox - Instruction ou fonction ?

Date de publication : Lundi 18 mars 2003 , Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilmir.developpez.com\)](http://gilmir.developpez.com)

[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. Introduction

Cette instruction très utilisée dans Visual Basic permet de poser une question à l'utilisateur au travers d'une boîte de dialogue.

Celle-ci peut être paramétrée en définissant le nombre de boutons poussoir, l'icône de la boîte et le bouton par défaut, c'est à dire celui qui sera enfoncé quand vous appuyez sur la touche "Entrée".

La syntaxe de l'instruction MsgBox est la suivante :

```
Variable = MsgBox("Message", Type, "Titre de la fenêtre")
```

- **Variable** : Variable de type Entier(Integer) recevant la valeur du bouton.
- **Message** : Chaîne de caractères affichée comme message dans la boîte de dialogue.
- **Type** : Expression numérique qui contrôle les boutons et les icônes à afficher.
- **Titre** : Expression chaîne affichée dans la barre de titre de la boîte de dialogue.

Le paramètre Type représente l'addition de plusieurs constantes intégrées de Visual Basic.

Pour répondre à la question posée, MsgBox est utilisée en Instruction lorsque la syntaxe utilisée n'attend pas une réponse du système par exemple :

Instruction :

MsgBox Message\$, 16 Titre\$ Dans ce cas nous avons utilisé une instruction de VB.

Fonction :

Dans le cas d' une fonction, (qui attend une réponse du système) nous écrivons :

```
T$ = "Attention"           'Titre du message
M$ = "Désirez-vous vraiment quitter le programme ?"      'Message à
afficher
```

si votre message à afficher est plus long, vous pouvez utiliser la concaténation suivante,

```
M$ = M$ + " suite du message" (vous pouvez aussi utiliser le
caractère "&" à la place du caractère "+")
Reponse% = MsgBox(M$, 4 + 32, T$)
```

et voici la réponse que vous pouvez éventuellement utiliser :

```
If Reponse% = 6 Then End           'Sortie du programme
```

J'utilise depuis Vb 3.0 la fonction précédente qui fonctionne sous toutes les versions connues à ce jour.

Nous allons les étudier en détail avec les tableaux suivants.

Les constantes pour les boutons :

Constante	Chiffre	Description
vbOKOnly	0	Affiche le bouton OK uniquement
vbOKCancel	1	Affiche les boutons Ok et Annuler
vbCancelRetryIgnore	2	Affiche les boutons Abandonner, Répéter et Ignorer
vbYesNoCancel	3	Affiche les boutons Oui, Non et Annuler
vbYesNo	4	Affiche les boutons Oui et Non
vbRetryCancel	5	Affiche les boutons Répéter et Annuler

Les constantes pour les icônes :

Constante	Chiffre	Description
vbCritical	16	Affiche l'icône message critique
vbQuestion	32	Affiche l' icône requête d' avertissement
vbExclamation	48	Affiche l'icône message d' avertissement
vbInformation	64	Affiche l'icône message d' information

Les constantes pour le bouton par défaut :

Constante	Chiffre	Description
vbDefaultButton1	0	Le premier bouton est le bouton par défaut
vbDefaultButton2	256	Le deuxième bouton est le bouton par défaut
vbDefaultButton3	512	Le troisième bouton est le bouton par défaut
vbDefaultButton4	768	Le troisième bouton est le bouton par défaut

Chaque bouton lorsqu'il est activé, renvoie une valeur que l'on pourra interpréter au travers de la variable.

Si vous ne mettez pas le paramètre "Type" l'instruction MsgBox n'affichera que le bouton "OK"

Les constantes renvoyées par MsgBox :

Constante	Chiffre	Description
vbOk	1	OK
vbCancel	2	Annuler
vbAbort	3	Abandonner
vbRetry	4	Réessayer
vbIgnore	5	Ignorer
vbYes	6	Oui
vbNo	7	Non

Le code suivant montre un exemple de paramétrage de boîte de dialogue MsgBox avec retour de valeur d'une variable

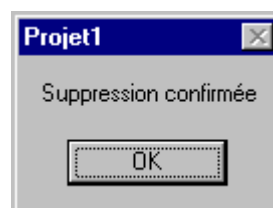
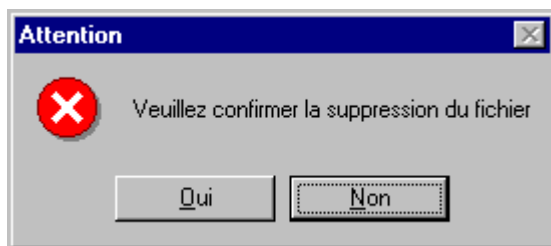
```
Private Sub Button_Supprime_Click ( )
Dim Sup
Sup = MsgBox("Veuillez confirmer la suppression du fichier",
vbCritical + vbYesNo + 256, "Attention")
If Sup = vbYes Then
MsgBox "Suppression confirmée"
End If
End Sub
```

Nous aurions pu écrire :

```
Sup = MsgBox("Veuillez confirmer la suppression du fichier", 16 + 4
+ 256, "Attention")
```

Faites des essais avec toutes les combinaisons possibles ou bien téléchargez le module d'exemples qui vous permettra de constater de visu toutes les combinaisons possibles de MsgBox [\[ici\]](#)

Voici les fenêtres que vous devez obtenir :



Tutoriel 12 : Les boîtes de dialogues de Windows

Date de publication : Lundi 18 mars 2003 , Date de mise à jour : Lundi 28 janvier 2008

Par [Gilbert Miralles \(gilmir.developpez.com\)](http://gilmir.developpez.com)

[Version PDF \(Miroir\)](#) [Version hors-ligne \(Miroir\)](#)

I. La boîte de dialogue "InputBox"

Comme pour les messages, Visual Basic propose une fonction simple permettant de créer une boîte de dialogue InputBox. Elle est activée par l'appel suivant: Variable\$ = InputBox\$(Prompt\$, Titre\$, Defaut\$, X%, Y%)

Le paramètre **Prompt\$** est le texte saisi par l'utilisateur.

Vous pouvez utiliser les caractères Chr\$(13) + Chr\$(10) pour forcer une ligne.(comme dans MsgBox)

Le paramètre **Titre\$** contient éventuellement le titre de la boîte de dialogue.

Le texte **Defaut\$** est proposé au moment de l'ouverture de la boîte de dialogue.

X% et **Y%** permettent de positionner la boîte, en l'absence de ces valeurs, la boîte de dialogue est centrée.

Un exemple de boîte de dialogue InputBox

Le message est passé dans le paramètre "Message\$", le titre dans "Titre\$".

Tous les paramètres sont définis librement à l'exception de "Message\$"

Si vous définissez une chaîne de caractère "Texteimplicité\$", ce texte sera affiché dans la zone de texte dès l'ouverture de la fenêtre.

Les variables numériques "x%" et "y%" permettent de positionner l'angle supérieur gauche de la fenêtre dans des coordonnées exprimés en "Twips" à partir de l'angle supérieur gauche de l'écran.

Une application envisageable de "InputBox" est la saisie d'un mot de passe.

```
Titre$ = "Saisie du mot de passe"  
Message$="Tapez quelque chose pour accéder au programme"  
MotDpass$=InputBox$(Message$, Titre$)'La boite de dialogue s'ouvre,  
' inscrit la valeur de, Message$ et de Titre$ à l'emplacement prévu  
par VB
```

```
If MotDpass$ = MotDpass$ Then 'Explication du traitement  
conditionnel : '  
'si la donnée saisie par l'utilisateur est égale à la donnée
```

```

enregistrée par le concepteur, alors....
    'Instructions (autoriser l'accès)
Else
    'Instructions (refuser l'accès et quitter le programme)
'Fin de la procédure
End If

```

Source [Téléchargez l'exemple de "InputBox"](#)

II. Les boîtes de dialogues de windows

Vos applications doivent présenter les mêmes menus et les mêmes boîtes de dialogue que les applications Windows standards.

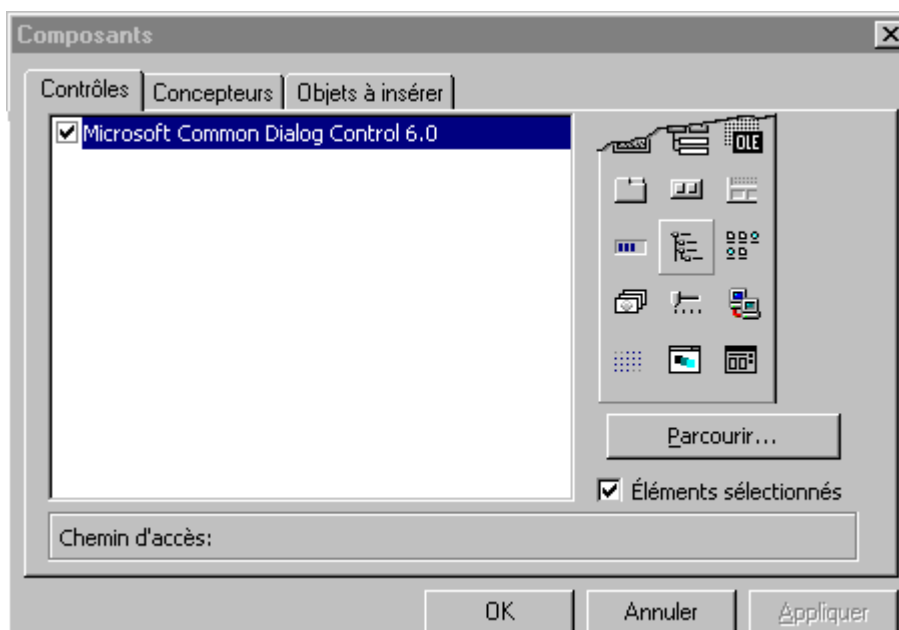
Exemple de boîte de dialogue (Ouvrir, Enregistrer, Couleur, Polices, Imprimer)

Ce sont des applications de Windows, et oui j'ai bien dit Windows, qui peuvent être utilisées par Visual Basic pour le choix d'un nom de fichier, d'une couleur ou d'une police de caractères.

Cela permet aux diverses applications d'avoir une apparence homogène, et simplifie la tâche du programmeur.

La boîte à outils ne contient pas tous les contrôles disponibles, pour pouvoir l'utiliser, vous devez aller le chercher et l'ajouter dans la boîte à outils.

Pour utiliser le contrôle "Common Dialog" cliquez dans la barre de menus sur la commande du menu intitulée Projet, puis Composants



cliquez sur le bouton **parcourir** et recherchez le fichier intitulé : [Comdlg32.ocx](#) vous devez voir apparaître la saisie que vous apercevez en surbrillance dans le cliché ci-contre. Cochez la case et cliquez sur OK Le contrôle se trouvera dans la boîte à outils.

Le contrôle se trouvant positionné maintenant dans votre boîte à outils, vous le copiez dans votre interface de travail avec le principe du glisser, poser, n'importe où puisque' il n'est pas visible en mode "Exécution".

Il ne vous reste plus qu'à codifier les instructions nécessaires à l'exécution des différents paramètres que vous souhaitez utiliser.

Exemples de la commande d'affichage des Boîtes de dialogues de WINDOWS :

- ShowOpen pour la boîte "Ouvrir"
- ShowSave pour la boîte "Enregistrer"
- ShowColor pour la boîte "Couleurs"
- ShowFont pour la boîte "Polices"
- ShowPrinter pour la boîte "Imprimer"
- ShowHelp pour la boîte "Aide Windows"

Comme vous le voyez, un seul contrôle Common Dialog suffit pour afficher divers types de boîtes de dialogues.

Toutefois, avant de déclencher le contrôle par une méthode, il est indispensable de définir quelques propriétés.

L'utilisation de ses propriétés est largement expliquée dans nos cours en ligne.

Vous pouvez télécharger les routines d'exemples de boîtes de dialogues en cliquant ci après :
[Source](#) [Téléchargement de "cmdialog.zip"](#)

Tutoriel 13 : Les structures de contrôles

I. Introduction

Il est nécessaire de pouvoir contrôler le bon déroulement des instructions dans le code et de tester les variables. Visual Basic propose plusieurs instructions de contrôle qui permettent de déterminer ce que le programme doit réaliser par la suite.

- La construction IF - THEN
- La construction SELECT CASE
- La construction IFF
- La boucle FOR - NEXT
- La boucle WHILE
- La boucle DO WHILE UNTIL

L'étude de toutes ces structures ne rentrent pas dans le cadre de ce programme, je vous recommande pour de plus amples conseils les cours que nous diffusons au sein de notre centre de formation. [Visiter...](#)

La construction IF THEN

L'instruction la plus utilisée en programmation est la construction If - Then
Cette instruction courante est à sens unique et permet des prises de décision.
La syntaxe de base de la construction If Then est la suivante :

```
If Condition logique Then
    'Instructions si la condition est vraie
Else
    'Instruction si la condition est fausse
End If      'Fin de l' instruction
```

La construction If Then est employée pour exécuter une ou plusieurs instructions conditionnelles. Le mot Else est facultatif, si celui-ci est inclus dans la construction c'est qu'il permet d'exécuter une ou plusieurs instructions quand la condition que vous testez est fausse. Une autre construction If Then Else permet d'effectuer de multiples conditions à l'intérieur de la structure If Then.

Voici la syntaxe :

```
If Condition1 Then
    'Instructions si la condition est vraie
ElseIf Condition2 Then
    'Instructions si la condition est vraie
ElseIf Condition3 Then
    'Instructions si la condition est vraie
Else
    'Instructions si la condition est fausse
End If
```

Exemples :

```
If AgeFrederic > 30 Then
    'Instructions si la condition est vraie
    msg = "Frédéric a plus de 30 ans"
    MsgBox msg
ElseIf AgeFrederic >25 Then
    'Instructions si la condition est vraie
    msg = "Frédéric a plus de 25 ans"
    MsgBox msg
ElseIf AgeFrederic >20 Then
    'Instructions si la condition est vraie
    msg = "Frédéric a plus de 20 ans"
    MsgBox msg
ElseIf AgeFrederic > 15 Then
    'Instructions si la condition est vraie
    msg = "Frédéric a plus de 15 ans"
    MsgBox msg
Else
    'Instructions si la condition est fausse
    msg = "Frédéric a 15 ans ou moins de 15 ans"
    MsgBox msg
End If
```

Source [Téléchargeons le programme d'exemple : agefreddy](#)

Nous n'avons pas voulu dans cet exemple indiquer exactement l'âge de Frédéric, mais connaissant le principe vous pouvez maintenant à votre tour, déterminer exactement l'âge qui

est indiqué dans le champs de saisie. A vous de jouer!

Il est préférable lorsque vous avez de nombreuses structures imbriquées d'utiliser la construction SELECT CASE que vous pourrez étudier dans nos cours.

Une des structures les plus employées en programmation est la boucle FOR NEXT

Une boucle est un processus de répétition d'un bloc d'instructions.

Vous utiliserez For...Next lorsque vous voulez répéter des instructions dans la boucle un nombre précis de fois.

Vous pouvez forcer un arrêt des répétitions avec l'instruction Exit For.

La syntaxe est la suivante :

```
For pointeur = ValeurDepart To ValeurFin Step Pas
    'Instructions
Next
```

Vous pouvez utiliser le mot clé STEP pour changer la valeur du compteur qui s'incrémente(1) ou pour indiquer que le compteur va dans une direction négative. Par exemple le code suivant compte de 5 à 100 par pas de 5 en montrant les valeurs du compteur.

Vous incrémentez de 5	Vous décrémentez de 5
<pre>Dim A As Integer For A = 5 To 100 Step 5 MsgBox (A) Next A</pre>	<pre>Dim A As Integer For A = 100 To 5 Step -5 MsgBox (A) Next A</pre>

Vous pouvez modifier le pas à votre convenance.

Sans le sigle STEP l'incréméntation aura comme valeur 1 point.

Un très bon exercice à réaliser est la fameuse calcullette de conversions Francs/Euros vous trouverez sur notre site de nombreux exemples à développer.

Tutoriel 14 : De l'intelligence dans les objets

De l'intelligence dans les objets

Je souhaitais continuer ces Tutoriels avec une approche du processus de l'intelligence artificielle qui est contrairement aux idées reçues relativement simple.

Sans rentrer dans les algorithmes compliqués, on peut développer des jeux simples comme Le bandit Manchot" ou bien un jeu de "Morpions".

Historiquement l'Intelligence artificielle est aussi vieille que le concept de jeu vidéo

Quand les amateurs commencent à écrire leurs propres programmes, beaucoup rêvent du jour où ils seront capables de programmer un jeu d'échecs ?

Mais l'écriture d'un jeu d'échecs tourne parfois à l'obsession, même parmi les programmeurs les plus érudits et même familiarisés avec les échecs.

Il convient de ne pas perdre de vue que ces jeux n'ont que peu de choses à voir avec les jeux d'arcades, d'aventures ou de simulations, qui tous demandent des techniques différentes de programmation et de l'imagination dans les procédures d'adresses.

Nous commencerons cette analyse des jeux "Intelligents" avec un exemple que d'aucuns

trouverons un peu simpliste, mais qui a l'avantage de recouvrir la plupart des principes d'écriture de ces jeux.

Si vous avez étudié les fonctions aléatoires, vous avez appris que générer une séquence réellement aléatoire est une tâche impossible pour les êtres humains et pour les ordinateurs, bien que ces derniers arrivent à faire une meilleure approximation.

Dans une longue série de coups, le joueur humain choisit invariablement un objet plus souvent que les autres. On peut donc inscrire dans le programme un sous programme qui garde en mémoire les choix du joueur en utilisant un tableau à trois éléments que l'on pourrait appeler :

Choix1, Choix2, Choix3.

L'ordinateur peut alors calculer quel objet est le plus souvent joué, et ensuite jouer l'objet qui l'emporte sur celui-ci.

Le plus grave inconvénient de cet algorithme apparaît au moment où le joueur réussit à percer la stratégie de l'ordinateur.

Les êtres humains sont incapables de prendre une décision totalement irrationnelle ou aléatoire. Il s'ensuit que chaque choix dépend des choix précédents.

Si l'ordinateur réussit à élaborer une approximation sur ces choix, alors il devrait pouvoir gagner assez régulièrement. Le programme doit être écrit de telle façon qu'il puisse interpréter la formule pendant le déroulement de la partie.

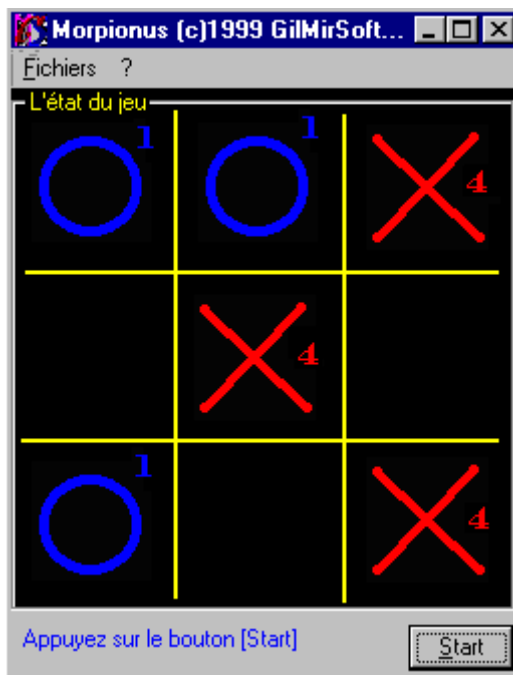
Les programmes qui sont capables d'apprendre de cette façon s'appellent :

Des programmes "Heuristiques"

Un programme heuristique permet à l'ordinateur de détecter des changements dans la stratégie de son adversaire et de modifier son algorithme en conséquence.

Il applique une technique statistique appelée "Corrélation"

Nous allons faire abstraction de toutes ces techniques rébarbatives pour nous consacrer à l'étude simpliste d'un jeu de morpions, tout en sachant que nous aurons par la suite la possibilité d'améliorer ses performances en suivant les techniques décrites précédemment.



Une évaluation correcte des positions est fondamentale à tout programme de jeu même si ce jeu est aussi simple que le morpion. Sur ce tableau 3x3 les zéros du joueur sont représentés par le chiffre 1, les croix de l'ordinateur par le chiffre 4. A l'aide de ces chiffres, on peut évaluer n'importe quelle situation de jeu.

Il suffit de faire les totaux pour chaque rangée, colonne, diagonale. Un total de 12 indique que l'ordinateur peut donc gagner, et un total de 3 que c'est le joueur qui est sur le point de conclure. Les valeurs 1 et 4 sont utilisées parce qu'elles fournissent des totaux différents pour chaque combinaison de coup. Dans le cas de cette figure, c'est celui qui va jouer le premier qui va gagner puisqu'il va aligner (12) ou (3).

Il est souhaitable de faire simple dans un premier temps, mais il m'apparaît indispensable d'utiliser un générateur aléatoire pour générer le carré de départ lorsque c'est à l'ordinateur de jouer. Cela désoriente le joueur (un certain temps) qui ne sait pas où le computer va positionner son pion.

On pourrait choisir un algorithme compliqué de façon que l'on ne tombe pas sur des chiffres déjà utilisés tout au moins dans la limite des 9 disponibles.

Mais faisons simple, et l'on pourrait coder le générateur sous la forme de :

```
Private Sub Form_Load( )
    'Centrage des feuilles
    CenterForm Me 'Utiliser dans un module la
procédure de centrage des feuilles
    Form1!ctlQuiJoue.Caption = "Appuyez sur le bouton [Start]"
    Randomize 'Commande du générateur
aléatoire
End Sub
```

```
Sub Ordi_A_Jouer ( )
    'Déterminer aléatoirement la sélection d'une case
    Valeur% = Int(Rnd * 9) 'Formule simple pour générer un
chiffre aléatoirement dans la limite de 9 nombres.
    'Ecrit la valeur dans l'étiquette
    Form1!Label10.Caption = Valeur%
```

le programme a sélectionné une case, contrôler si la case Index est vide puis, affichons la case choisie. [\(1\)](#)

```
If Form1!ctlControlEtat(Index).Caption = "0" Then
    'Instructions, la case est vide, alors affichons notre pion
    Form1!Image1(Valeur%).Picture = Form2!Image_X.Picture
    'Mise à jour du compteur de sélection
    Form1!ctlControlSelect.Caption = Valeur%
    'Mise à jours des compteurs individuels
    Iteration_Compteur_Ordi
Exit Sub
Else
    'La case est occupée, alors allons voir ailleurs
    Control_Resultat
End If
End Sub
```