PROGRAMMATION EVENEMENTIELLE AVEC VISUAL BASIC

1. LA PROGRAMMATION EVENEMENTIELLE

L'introduction des interfaces graphiques a introduit de profondes modification du mode de programmation des application interactives.

PROGRAMMATION CLASSIQUE	PROGRAMMATION EVENEMENTIELLE
L'utilisateur est guidé par le programme et le développeur contrôle ses actions et leur succession par le biais de menus ou d'instructions d'affichage ou de lecture de données	Les interfaces graphiques ont introduit la notion d'objets interactifs proposés à l'utilisateur et réagissant aux actions de celui-ci, qui sont appelées des évènements. La tâche du programmeur consiste désormais à choisir ces objets d'interaction, et à définir le comportement de l'application en réaction aux évènements susceptibles de se produire
Existence d'un programme principal appelant des procédures et fonctions.	Le programmeur n'écrit plus de programme principal : il se contente d'écrire le code correspondant aux réactions à ces évènements. Les initialisations sont en général liées à l'évènement "lancement de l'application"

LA GESTION DES EVENEMENTS

Chaque composant de l'application reconnait un certain nombre d'évènements. L'environnement prend en charge la détection de ces évènements et le déclenchement des actions associées.

LES DIFFERENTS TYPES D'EVENEMENTS

Les évènements externes
- liés au clavier et à la souris
- liés au temps (terminaison d'un délai, échéance d'une date)

Les évènements internes
- modification d'un objet ou d'une donnée (modification d'un enregistrement dans une base de donnée, modification du contenu d'une zone de saisie)
- évènements inter-application (échanges de données)

LE CONTROLE DU DIALOGUE HOMME MACHINE

Comme le programmeur ne peut prévoir dans quel ordre l'utilisateur va interagir avec les composants de l'application, il lui appartient de n'offrir que les possibilités d'interaction

adéquates, en utilisant notamment la possibilité de jouer sur la visibilité des composant ou sur leur "activabilité".

2. INTRODUCTION A VISUAL BASIC

Visual Basic est un environnement de développement d'applications pour WINDOWS : il met en oeuvre les principes de la **programmation événementielle** et permet la réalisation d'applications selon les techniques et les méthodes dites "**RAD**" (Rapid Application Development).

LES OBJETS GRAPHIQUES

Visual Basic manipule essentiellement des objets visuels qui sont regroupés en 2 classes :

- la classe des feuilles ou fenêtres ;
- la classe des contrôles (ou widgets : window gadgets) : objets placés dans les feuilles.

A chaque objet feuille ou contrôle sont associés :

- un **nom** : il permet d' identifier l' objet dans les programmes.
- des **propriétés** : ce sont les caractéristiques propres à l' objet (exemples : sa taille, sa position, sa couleur etc...) qui décrivent l'aspect et l'état interne des objets.
- des **méthodes** : ce sont des programmes inclus dans l' objet lui-même ; en programmation événementielle ces méthodes ne sont pas modifiables (exemple : la méthode Hide associée à un objet fenêtre permet de cacher une fenêtre) et régissent le comportement de l'objet lorsque certains évènements se produisent.
- des **événements externes** liés au clavier et à la souris **et internes** liés au changement d'état de l' objet.
- des **procédures événementielles** : ces programmes s'exécutent dès que l' événement auquel ils sont attachés survient.

LES COMPOSANTS D'UNE APPLICATION VISUAL BASIC

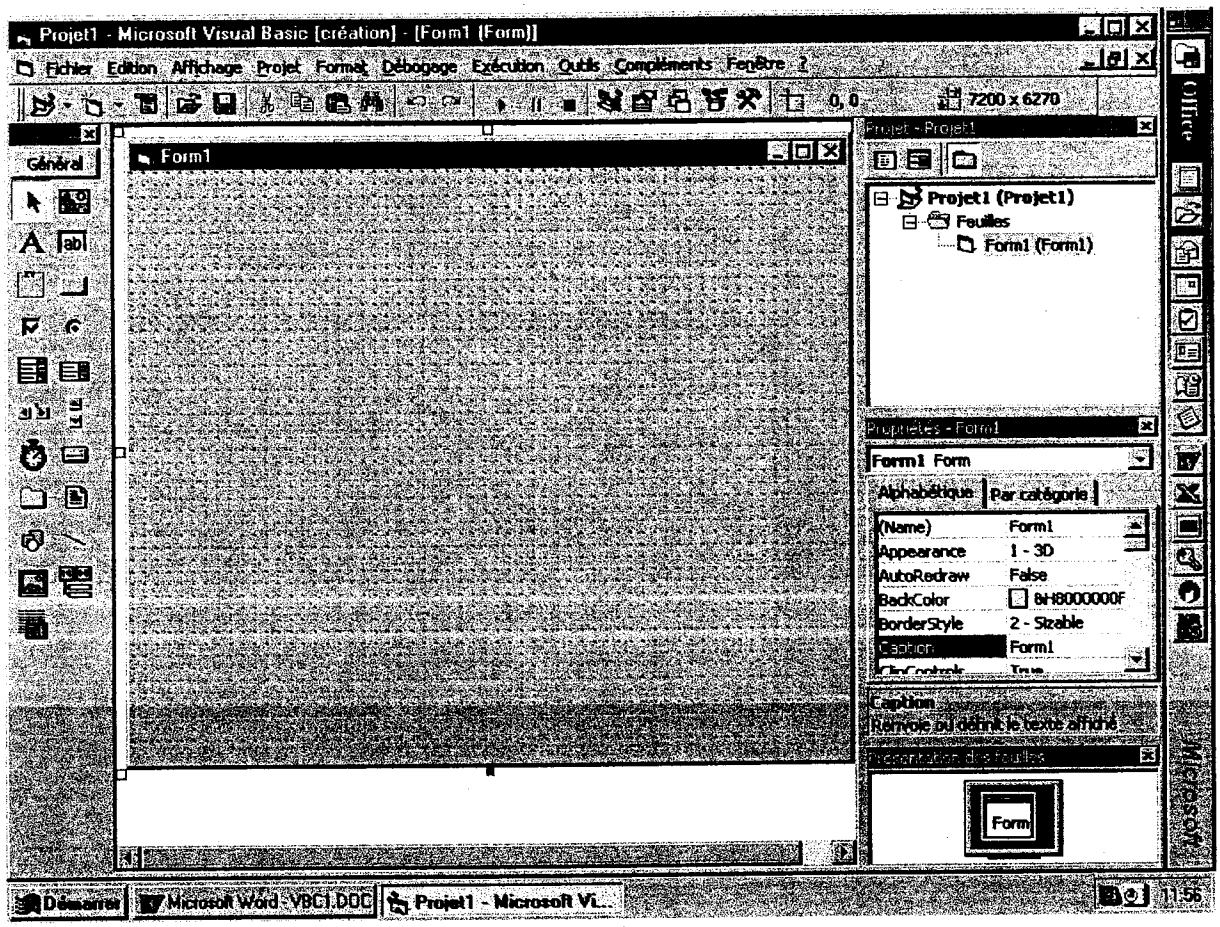
Un projet développé en Visual Basic contient :

- un fichier général de description du projet. Il mémorise la liste de tous les fichiers nécessaires au projet. C'est un fichier texte ASCII d' extension ".VBP".
- un ou plusieurs fichiers feuilles d' extension ".FRM". Dans un fichier sont mémorisés en format binaire, tous les éléments attachés à une feuille : la description de la fenêtre, la description des contrôles inclus dans la fenêtre, les procédures et fonctions liées à la feuille.
- zéro ou plusieurs fichiers modules : les procédures et fonctions communes à tout le projet sont stockées dans ces fichiers au format texte ASCII simple. Ils ont l' extension ".BAS".
- zéro ou plusieurs fichiers de contrôles personnalisés : 22 contrôles standard sont intégrés dans le programme Visual Basic. Des contrôles personnalisés peuvent être ajoutés. Ils sont stockés sur disque dans des fichiers d' extension ".VBX" ou ".OCX". pr crées des nx objets.

Le projet terminé et mis au point peut être transformé en un fichier disque exécutable ".EXE". Ce fichier s' exécute seul mais il est nécessaire de disposer de plusieurs fichiers ".DLL" de Visual Basic et des fichiers ".VBX" et ".OCX" le cas échéant.

Il appartient au développeur de construire la ou les feuilles nécessaires à l'application, d'y placer les contrôles voulus, d'en fixer les propriétés et d'écrire le code des procédures déclenchées par les événements jugés significatifs.

L'ENVIRONNEMENT DE DEVELOPPEMENT



Barre de menus et barre d'outils : les icônes de la barre d'outils servent de raccourcis à quelques options des menus. La barre d'outils est affichée ou masquée via le menu Affichage.

Boîte à outils : contient les icônes servant à placer les contrôles sur une feuille de projet.

La fenêtre de projet : contient la liste de toutes les feuilles et de tous les fichiers utilisés par l'application.

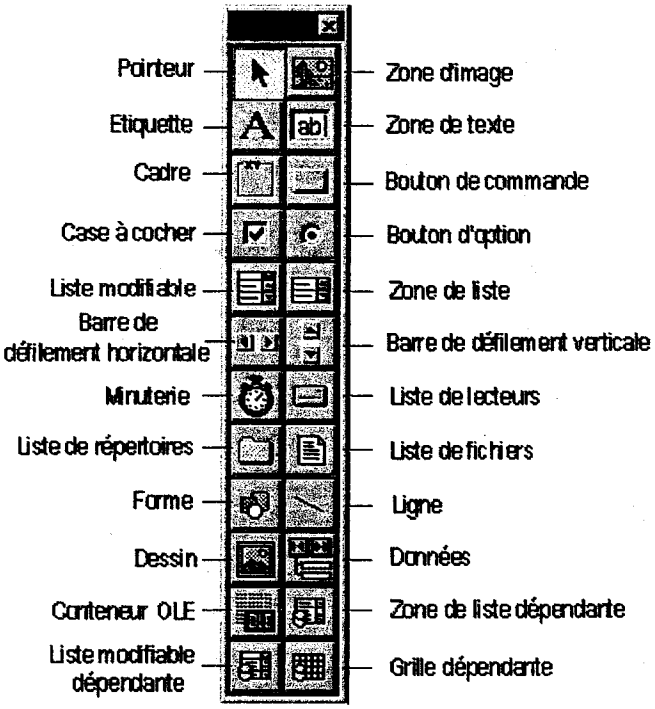
La feuille courante : fenêtre appartenant à l'application. Elle peut être remplacée par la fenêtre de code qui affiche le code source associé aux évènements des objets de cette fenêtre.

La fenêtre de propriétés : donne accès à la liste de tous les objets de la feuille courante, avec pour chacun, la liste des propriétés et leurs valeurs.

La fenêtre de débogage : (non visible) permet de tester les valeurs de propriétés et des variables lorsqu'une application est interrompue.

LES CONTROLES

La boîte à outils de Visual Basic contient les outils que vous pouvez utiliser pour dessiner des contrôles sur vos feuilles. Chacun de ces outils (consultez la figure suivante) représente un contrôle. Elle peut être enrichie par l'ajout d'autres contrôles fournis avec l'environnement ou par des développeurs tiers.



Le tableau suivant décrit brièvement les contrôles de Visual Basic figurant dans la boîte à outils standard.

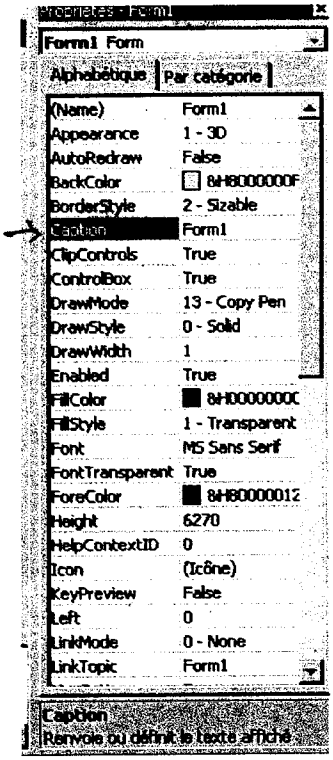
Icône	Classe	Description
	Pointeur	Permet de déplacer et de redimensionner des feuilles et des contrôles. (Il ne s'agit pas d'un contrôle.)
	PictureBox	Affiche des images en mode point (bitmaps), des icônes ou des métafichiers Microsoft Windows. Affiche du texte ou agit comme un conteneur visuel pour d'autres contrôles.
	Label	Affiche du texte que l'utilisateur ne peut pas modifier et avec lequel il ne peut pas dialoguer.
	TextBox	Fournit une zone pour taper ou afficher du texte.
	Frame	Constitue un conteneur visuel et fonctionnel pour des contrôles.
	CommandButton	Exécute une commande ou une action lorsque l'utilisateur le choisit.
	CheckBox	Affiche une option Vrai/Faux ou Oui/Non. Le nombre de cases à cocher que vous pouvez activer simultanément est illimité.

	OptionButton	Fait partie, avec d'autres boutons d'option, d'un groupe qui permet d'afficher plusieurs choix. L'utilisateur ne peut sélectionner qu'un de ces boutons à la fois.
	ComboBox	Allie les fonctions d'une zone de texte et d'une zone de liste. Permet à l'utilisateur de taper dans un élément sélectionné ou d'en sélectionner un dans une liste déroulante.
	ListBox	Affiche une liste d'éléments parmi lesquels l'utilisateur peut faire un choix.
	HScrollBar	
	VScrollBar	Permettent à un utilisateur de sélectionner une valeur dans une plage de valeurs. (Il s'agit de deux contrôles distincts, qui sont en outre différents des barres de défilement intégrées dans de nombreux contrôles.)
	Timer	Exécute des événements Timer à des intervalles définis
	DriveListBox	Affiche une liste de lecteurs de disque valides parmi lesquels l'utilisateur peut faire un choix.
	DirListBox	Affiche une liste de répertoires et de chemins d'accès parmi lesquels l'utilisateur peut faire un choix.
	FileListBox	Affiche une liste de fichiers parmi lesquels l'utilisateur peut faire un choix.
	Shape	Ajoute un rectangle, un carré, une ellipse ou un cercle sur une feuille.
	Line	Ajoute un segment de droite sur une feuille.
	Image	Affiche des images en mode point, des icônes ou des métafichiers Microsoft Windows. Agit comme un bouton de commande lorsque l'utilisateur clique dessus.
	Data	Permet de se connecter à une base de données existante et d'afficher les informations qu'elle contient sur une feuille
	OLE Objet lié	Incorpore des données dans une application Visual Basic.
	DBCombo	Le contrôle DBCombo offre la plupart des fonctionnalités propres au contrôle ComboBox standard ainsi que des fonctions d'accès aux données améliorées.
	DBList	Le contrôle DBList offre la plupart des fonctionnalités propres au contrôle ListBox standard ainsi que des fonctionnalités d'accès aux données améliorées.
	Menu	Crée des menus dans vos applications Visual Basic. C'est dans le Créateur de menus que vous travaillez avec les contrôles Menu. Vous accédez à cet outil en choisissant la commande Créateur de menus dans le menu Outils ou en cliquant sur l'icône « Créateur de menus » de la barre d'outils.

LES PROPRIETES

Chacun des objets manipulés par Visual Basic possède des propriétés dont les valeurs déterminent son apparence et son état.
A titre d'exemple voici quelques propriétés de l'objet feuille (form) :

titre de l'objet →



La valeur d'une propriété peut être donnée lors de la conception de l'application et modifiée de manière dynamique par programme.

Certaines propriétés d'objets n'apparaissent pas dans la fenêtre des propriétés parce que leur valeur n'est pas modifiable. Elles peuvent cependant être consultées dans un programme (exemple, nombre d'éléments courant d'une liste).

LES EVENEMENTS

A chaque feuille et contrôle est associé un ensemble prédéfini d'événements.

Toujours à titre d'exemple, voici la liste des événements auxquels réagit un objet de type Bouton de commande :

Nom d'événement	Description
Click	Se produit lorsque l'utilisateur a cliqué sur l'objet
DragDrop	Se produit lorsqu'une opération glisser/déplacer a eu lieu

DragOver	Se produit lorsqu'une opération glisser/déplacer est en cours (l'utilisateur n'a pas encore relâché le bouton de la souris)
MouseMove	Le pointeur de la souris est déplacé sur le contrôle
Mouse Down	Un bouton de souris est enfoncé
MouseUp	Un bouton de souris est relâché
GotFocus	Se produit lorsque le bouton a obtenu le "focus", soit parce que l'utilisateur a cliqué dessus ou l'a activé avec la touche TAB, soit parce qu'un programme a activé la méthode SetFocus du bouton. Lorsqu'un objet a le focus, tout ce qui est tapé au clavier le concerne directement.
LostFocus	Le bouton a perdu le focus
KeyPress	Se produit lorsque l'utilisateur a appuyé sur une touche alors que le bouton avait le Focus
KeyDown	idem, mais se produit avant que la touche ait été relâchée (utile pour tester si une touche MAJ, CTRL, etc. est enfoncée)
KeyUp	idem, mais se produit lorsque la touche est relâchée Ces trois événements mettent à la disposition du programmeur le code ASCII de la touche concernée.

Procédure événementielle

A chaque événement, peut être associé une suite d'instructions à exécuter lorsque l'événement se produit : il s'agit de **la procédure événementielle**.

On peut donc voir ce type de procédure comme étant un sous-programme qui est automatiquement appelé lorsque l'événement correspondant se produit.

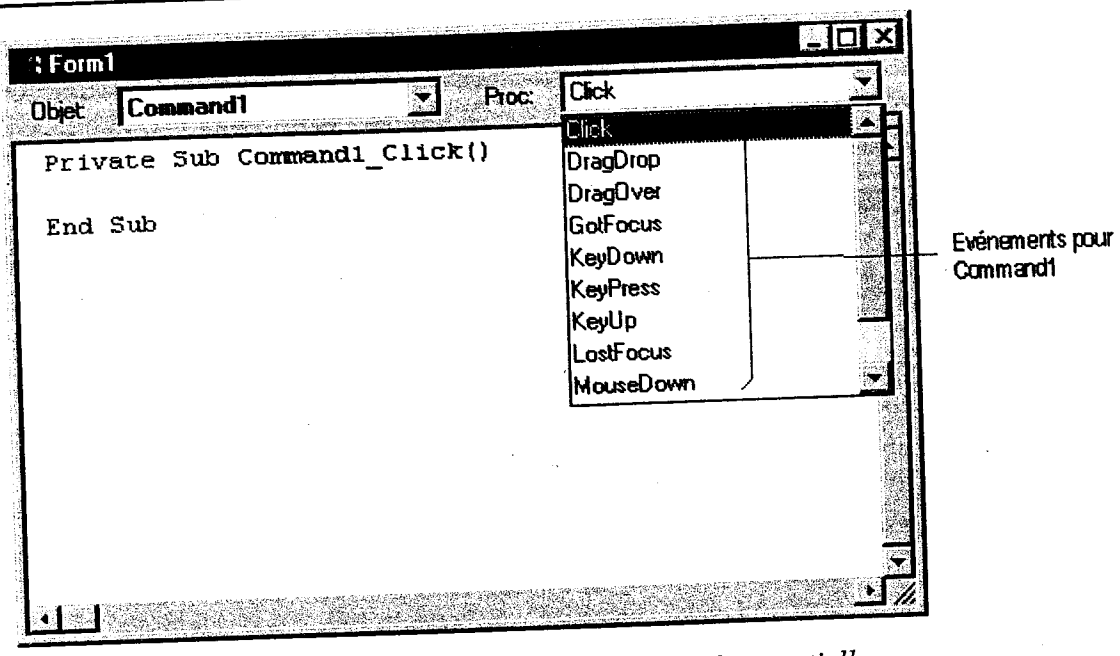
Comment écrire les procédures événementielles ?

Le code Visual Basic de votre application s'écrit dans la fenêtre Code. Un code est constitué d'instructions, de constantes et de déclarations. La fenêtre Code vous permet de visualiser et de modifier rapidement n'importe quelle partie du code de votre application.

On accède à la fenêtre de code de plusieurs manières :

- Cliquez deux fois sur la feuille ou le contrôle pour lequel vous souhaitez écrire du code.
- OU BIEN : Dans la fenêtre Projet, sélectionnez le nom d'une feuille ou d'un module, puis choisissez le bouton « Code ».
- OU BIEN ; Sélectionnez « Code » dans le menu contextuel de la feuille ou le contrôle pour lequel vous souhaitez écrire du code.

La figure suivante illustre la fenêtre Code qui apparaît lorsque vous cliquez deux fois sur le contrôle CommandButton, ainsi que les événements associés à cette commande.



Pour écrire une procédure événementielle

1. Dans la zone de liste « Objet », sélectionnez le nom d'un objet dans la feuille active. (La feuille *active* est celle qui vient d'être sélectionnée.)

   Dans cet exemple, choisissez le bouton de commande « Command1 ».

2. Dans la zone de liste « Proc », sélectionnez un nom d'événement pour l'objet sélectionné.

   Ici, la procédure **Click** est déjà sélectionnée, puisqu'il s'agit de la procédure associée par défaut à un bouton de commande. Vous pouvez remarquer qu'un *modèle* de procédure est maintenant affiché dans la fenêtre Code.

3. Tapez le code suivant entre les instructions **Sub** et **End Sub** :

```
Text1.Text = "Bonjour tout le monde !"
```

La procédure d'événement doit ressembler à celle-ci :

```
Private Sub Command1_Click ()
 Text1.Text = "Bonjour tout le monde !"
End Sub
```

propriété

Vous pouvez remarquer que dans ce code seule la valeur de la propriété Text du contrôle Text1 est changée en « Bonjour tout le monde ! ». La syntaxe de cet exemple prend la forme objet.propriété, dans laquelle Text1 est l'objet et Text la propriété. Vous pouvez utiliser cette syntaxe pour modifier les valeurs des propriétés de toute feuille ou de tout contrôle, en réponse à des événements qui se produisent pendant l'exécution de votre application.

La procédure d'événement d'un contrôle associe le nom réel du contrôle (spécifié par la propriété Name), un trait de soulignement (_) et le nom de l'événement. Par exemple, si vous souhaitez qu'un bouton de commande appelé « Command1 » invoque une procédure d'événement lorsque l'utilisateur clique dessus, utilisez la procédure Command1_Click.

LES METHODES

Chaque objet, feuille et contrôle, présente un ensemble prédéfini de méthodes appelables depuis le code de l'application. Une méthode n'est ni plus ni moins qu'un sous-programme avec éventuellement des paramètres formels éventuels et/ou des une valeur de retour dans le cas où le sous-programme est une fonction.

Si l'on veut rendre visible une feuille, on dispose de la méthode show :

```
form1.show
```

Si l'on veut rendre actif le contrôle zone de texte, text1, de la feuille form1, on dispose de la méthode setfocus :

```
text1.setfocus
```

Il faut distinguer les méthodes prédéfinies attachées à un objet et celles que le programmeur écrit en réaction aux événements utiles.
Ainsi dans le cas d'un bouton de commande, il est possible d'associer un sous-programme à chacun des événements que peut recevoir un bouton de commande : ce sont les procédures événementielles.
Par ailleurs, un bouton de commande dispose de méthodes prédéfinies, appelables dans toute procédure.

Par exemple, au chargement de la feuille form1, on souhaite rendre active la zone de texte Text1 :

Il faut donc écrire la procédure événementielle Form1_Load correspondant au code à exécuter lors de la réception de l'événement Load sur la feuille Form1 :

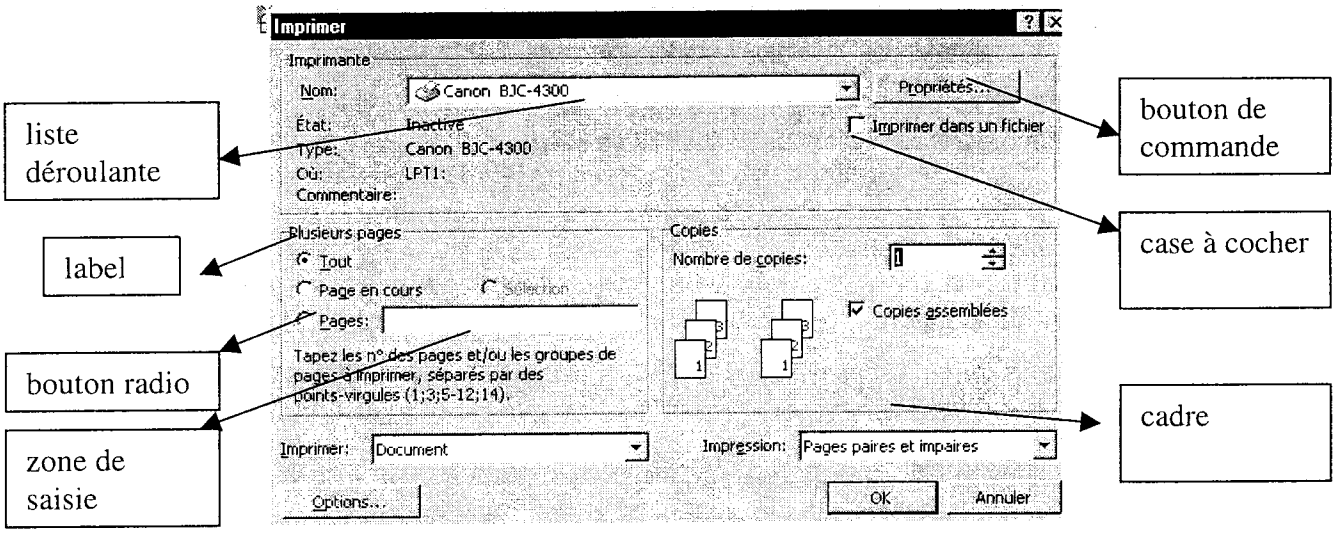
```
Sub Form1_Load
     Text1.SetFocus
End Sub
```

Pour exemple, on donne ici la liste des méthodes prédéfinies attachées à un objet de type bouton de commande :

Nom de méthode	Description
Drag	Permet de faire glisser le bouton
Move	Permet de déplacer le bouton
Refresh	Permet de réafficher le bouton, par exemple après en avoir modifié des propriétés
SetFocus	Attribue le Focus au bouton
Z-order	Permet d'organiser les objets superposés en termes d'arrière ou d'avant-plan

VISUAL BASIC –ETUDE DES CONTROLES (1)

La boîte de dialogue ci-dessous comporte un certain nombre de contrôles :



REMARQUE GENERALE CONCERNANT L'ENSEMBLE DES CONTROLES : LA PROPRIETE NAME

Tous les contrôles VB ont une propriété Name qui a pour valeur le nom interne du composant. Lorsqu'on pose un contrôle sur une feuille, VB lui attribue automatiquement un nom formé du nom du contrôle et d'un numéro. Il appartient au programmeur de modifier la valeur de cette propriété de manière à lui donner un nom significatif. Ainsi, dans l'exemple ci-dessus, le bouton portant l'étiquette "OK" peut avoir été "baptisé" par VB "Command4". Il serait plus judicieux de lui attribuer un nom plus significatif.

Usuellement, les conventions suivantes sont à appliquer :

Objet	Préfixe	Exemple
Feuille	frm	frmFileOpen
Case à cocher	chk	chkReadOnly
Liste modifiable	cbo	cboEnglish
Liste modifiable dépendante	dbc	dbcEnglish
Bouton de commande	cmd	cmdCancel
Données	dat	datBiblio
Liste de répertoires	dir	dirSource
Liste de lecteurs	drv	drvTarget
Liste de fichiers	fil	filSource
Cadre	fra	fraLanguage
Grille	grd	grdPrices
Grille dépendante	dbg	dbgPrices
Barre de défilement horizontale	hsb	hsbVolume
Dessin	img	imgIcon
Etiquette	lbl	lblHelpMessage
Ligne	lin	linVertical
Zone de liste	lst	lstPolicyCodes
Zone de liste dépendante	dbl	dblPolicyCode
Menu	mnu	mnuFileOpen

Conteneur OLE	ole	oleObject1
Bouton d'option	opt	optFrench
Zone d'image	pic	picDiskSpace
Forme (cercle, carré, ovale, rectangle, rectangle arrondi et carré arrondi)	shp	shpCircle
Zone de texte	txt	txtGetText
Minuterie	tmr	tmrAlarm
Barre de défilement verticale	vsb	vsbRate

LE CONTROLE LABEL

Un contrôle Label est un contrôle graphique qui permet d'afficher du texte que l'utilisateur ne peut pas modifier directement.

Propriétés :

caption : le texte affiché dans le contrôle

LE CONTROLE TEXTBOX

Un contrôle TextBox, parfois appelé champ d'édition ou contrôle d'édition, affiche des informations entrées au moment de la création, tapées par l'utilisateur ou affectées au contrôle au moment de l'exécution.

PROPRIETES

text	le contenu de la zone de saisie ou d'affichage
Multiline	autorise ou non (true ou false) du texte sur plusieurs lignes. Une barre de défilement vertical est automatiquement ajoutée si le texte dépasse la taille de la zone.
Alignment	détermine le mode de justification. Est sans effet si la propriété MultiLine est à false

METHODES

SetFocus	permet de donner le focus au contrôle : tout événement clavier sera dirigé vers ce contrôle (un curseur y clignote). Un seul composant peut avoir le focus à un instant donné.

EVENEMENTS

DblClick	l'utilisateur double-clique dans la zone
keyDown	se produit dès qu'un utilisateur appuie sur une touche du clavier alors que la zone a le focus)
KeyUp	se produit dès qu'un utilisateur relache une touche du clavier alors que la zone a le focus)
keypress	se produit dès qu'un utilisateur tape sur une touche du clavier alors que la zone a le focus). Lors d'une frappe, les évènements se déclenchent dans l'ordre suivant : KeyDown, KeyUp, Keypress
MouseDown MouseUP Click	L'utilisateur appuie sur un bouton de la souris (MouseDown), le relache(MouseUp), et a finalement cliqué (Click).

LE CONTROLE COMMANDBUTTON

Les boutons de commande permettent d'offrir à l'utilisateur de choisir une action. Ils sont pour comportement prédéfini de donner l'illusion de s'enfoncer lorsqu'on clique dessus.

PROPRIETES

Caption	Le libellé affiché sur le bouton
Picture	L'image affichée sur le bouton (incompatible avec caption)
Default	La touche ENTREE a le même effet qu'un click. Il ne peut y avoir qu'un bouton dont la propriété default est à true sur une feuille
Cancel	La touche ECHAP a le même effet qu'un click. Il ne peut y avoir qu'un bouton dont la propriété cancel est à true dans une feuille
Enabled	Si cette propriété est à False, le bouton est inactif et son libellé est grisé.

METHODES

Comme le contrôle TextBox

EVENEMENTS

Comme le contrôle TextBox

LE CONTROLE CHECKBOX

PROPRIETES

caption	détermine le texte à afficher en regard de la case à cocher
Value	Détermine l'état de la case (cochée, non cochée ou grisée : checked, unchecked ou grayed)

Méthodes et événements comme ci-dessus.

LE CONTROLE OPTIONBUTTON

Les boutons d'options ou boutons radio ont des caractéristiques semblables à celles des cases à cocher. La différence réside dans le fait que, si on a un groupe de plusieurs cases à cocher, on peut en cocher plusieurs, alors dans un groupe de boutons d'options, on ne peut en cocher qu'un.

LE CONTROLE LISTBOX

Un contrôle ListBox affiche une liste dans laquelle l'utilisateur peut sélectionner un ou plusieurs éléments. Si le nombre d'éléments est trop grand pour tenir dans la zone affichée, une barre de défilement lui est automatiquement ajoutée.

PROPRIETES

List	désigne l'ensemble des éléments présents dans la liste
ListIndex	désigne le numéro de l'élément sélectionné dans la liste. vaut –1 si aucun élément n'a été sélectionné
ListCount	Le nombre d'éléments figurant dans la liste. les éléments sont numérotés de 0 à ListCount-1

METHODES

Additem	permet d'ajouter un élément dans la liste Syntaxe object.AddItem item index
RemoveItem	permet de supprimer un élément de la liste Syntaxe object.RemoveItem index

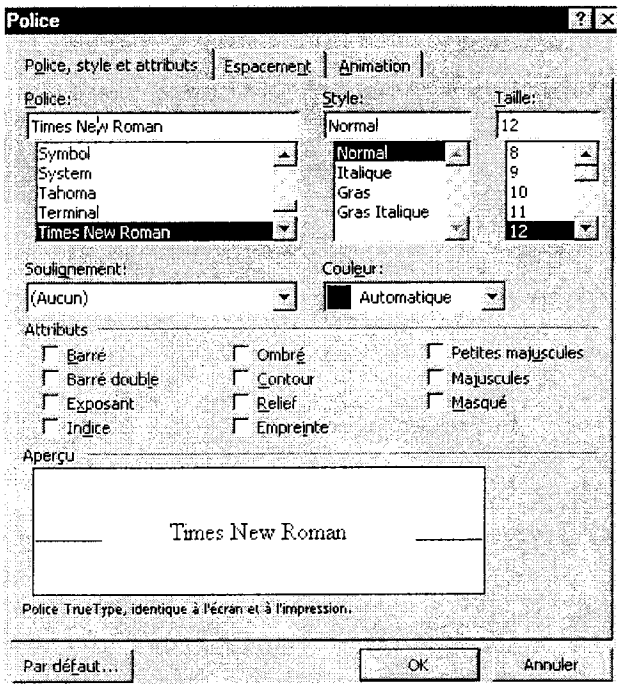
EVENEMENTS
Comme ci-dessus

LE CONTROLE COMBOBOX

Un contrôle ComboBox réunit les fonctionnalités des contrôles TextBox et ListBox. Il permet aux utilisateurs d'entrer des informations dans la partie zone de texte (TextBox) ou de sélectionner un élément dans la partie zone de liste (ListBox).

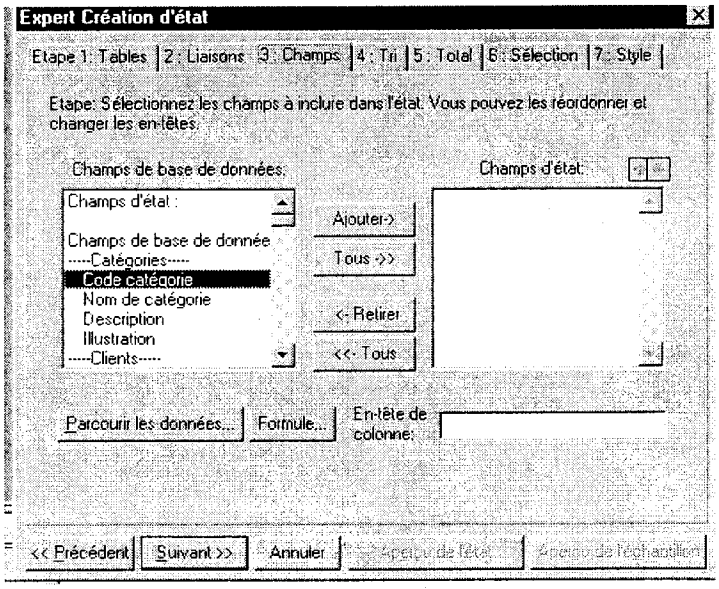
On y trouve l'ensemble des propriétés, méthodes et évènements de ces deux types de contrôle.

EXERCICE 1



Donner le type de chacun des composants de la boîte de dialogue ci-dessus.

EXERCICE 2



Ecrire l'algorithme correspondant aux évènements click sur les boutons "Ajouter->", "Tous->>", "<-Retirer" et "<<-Tous" de la boite de dialogue ci-dessus. On suppose que les deux zones de listes s'appellent respectivement LstChamps et LstEtat.

Ecrire l'algorithme associé à l'événement click sur les deux boutons "cmdFlechehaut" et "cmdFlechebas" a côté du label "Champs d'états".

EXERCICE 3

Durant les TP en C, a été développée une application simplifiée de gestion d'un stock de produits. Elle offrait l'interface utilisateur suivante :

```
Voulez-vous
     1 - Ajouter un produit au stock
     2 - Modifier la quantité en stock d'un produit
     3 - Supprimer un produit du stock
     4 - Visualiser les produits en stock
Votre choix : 1
Référence du produit ? EMM456
Quantité : 10
etc...
```

Concevoir un écran VB offrant les mêmes fonctionnalités. Décrire chacun des composants et le comportement attendu de l'application.

VISUAL BASIC – LE LANGAGE DE PROGRAMMATION

LES VARIABLES ET LES TYPES DE DONNEES

Une variable simple se déclare de la manière suivante :

```
Dim Nom_de_var As Type_de_Var
```

Les types de données prédéfinis en VB sont :

Type de données	Description
Byte	Entier de 0 à 255
Boolean	True ou false
Integer	entier de -32768 à +32767 (2 octets)
Long	entier sur 4 octets
Single	réel en simple précision
Double	réel en double précision
Currency	Monétaire
Date	du 1er janvier 100 au 31 décembre 9999
Object	Référence d'une donnée de type Objet
String	Chaine de de 0 à 65400 caractères
Variant	Tout type de valeur
Type	L'instruction type permet de définir des types structurés. La plage de valeurs de chaque champ dépend de son type de données.

Vous pouvez placer plusieurs déclarations de variable sur une même ligne, à condition de les séparer les unes des autres par une virgule et de préciser pour chacune son type. Si celui-ci est omis, le type de la variable est le type variant.

```
Dim i, j As Integer, msg As String
```

i est déclaré en tant que Variant, j en tant qu'entier, et msg en tant que chaîne de caractères

LES CONSTANTES

Const nom_const **=** valeur_const

Exemples :

```
Const Pi = 3.14159265358979
Const MaxPlanets = 9
Const ReleaseDate = #1/1/95#        ' date
Const Version = "07.10.A"           ' chaine de caractères
Const CodeName = "Enigma"
```

Vous pouvez placer plusieurs déclarations de constante sur une même ligne, à condition de les séparer les unes des autres par une virgule :

```
Const Pi = 3.14, MaxPlanets = 9, WorldPop = 6E+09
```

L'expression située à droite du signe égal (=) est souvent un nombre ou un littéral chaîne, mais il peut également s'agir d'une expression qui a pour résultat un nombre ou une chaîne (bien que l'expression ne puisse pas contenir d'appel de fonction). Vous pouvez même définir des constantes en réutilisant des constantes déjà définies :

```
Const Pi2 = Pi * 2
```

LES TABLEAUX

Un tableau unidimensionnel peut se déclarer de plusieurs façons :

`Dim tab(20) as integer` : déclare un tableau de 21 éléments numérotés de 0 à 20. Les indices des tableaux commencent par défaut à 0.

On peut cependant utiliser l'instruction Option Base pour faire commencer les tableaux à l'indice 1

```
Option base 1        Attribue la valeur 1 au premier indice des tableaux.
Dim tab(20) as integer
```

: le tableau contient maintenant 20 éléments numérotés de 1 à 20.

Pour lever toute ambiguïté on peut préciser à la déclaration les deux bornes du tableau (recommandé).

`Dim tab(0 to 20) as integer` déclare un tableau de 21 éléments numérotés de 0 à 20.

On accède à chaque élément en notant :
Nomtableau(indice)
Exemple :

```
tab(15) = 8
```

Les tableaux multidimensionnels obéissent à la même syntaxe, on donne les numéros des derniers éléments séparés par des virgules :
Exemple

```
Dim Eleves(1 To 15,1 to 2) as String
      15 lignes pour les 15 élèves
      colonne 1 : noms, colonne 2 : prenoms
Eleve(15, 1) = "Durand"
Eleves(15,2) = "Georges"
```

LES STRUCTURES

La déclaration d'un type structuré se note comme suit :

```
Type NomType
    NomChamp As NomType
    ....
End Type
```

Exemple

```
Type t_eleve
     nom as string
     prenom as string
     age as insteger
EndType
```

On pourra accéder aux champs d'une variable structurée au moyen de la notation pointée ou de l'instruction with :

Exemple :

```
uneleve as t_eleve
With uneleve
     .nom = "Durand"
     .prenom = "Georges"
     .age = 18
End With
```

```
uneleve as t_eleve
uneleve.nom = "Durand"
uneeleve.prenom="Georges"
uneleve.age = 18
```

LES INSTRUCTIONS DE CONTROLE

Dans la description ci-dessous, on adopte les conventions suivantes :

- en caractères gras figurent les mots-clés du langage
- entre crochets les éléments facultatifs

On retrouve les instructions classiques de tout langage de programmation :

Si...Alors...Sinon...Fsi:

```
If condition then
     instructions
     ....
[ elseif condition then
     instructions
     .... ]
[ else
     instructions
     .... ]
end if
```

Selon. (ou suivant) :

```
Select Case expression
     Case liste d'expressions
          instructions
          ....
     Case else
          instructions.....
End Select
```

Pour....FinPour :

```
For var_compteur = debut To fin [ Step incrément ]
     instructions.....
     .......
Next var_compteur
```

Repeter....jusqu'a...

```
Do Until condition
     instructions
     ......
Loop
```

Tant que.....Ftq

```
Do while condition
     instructions
     ......
Loop
```

L'affectation se note :

```
var = valeur
```

Le signe d'égalité sert à la fois à l'affectation et à la comparaison. VB se débrouille en fonction du contexte!

Les expressions booléennes :

Les comparateurs sont les comparateurs usuels. La différence s'écrit <>. Les connecteurs sont AND et OR. Le nom logique est NOT

L'UTILISATION DES PROPRIETES ET DES METHODES DES OBJETS

Les propriétés et méthodes d'un objet s'utilisent via **une notation dite "pointée"**.
Exemple :
Soit une feuille contenant un bouton de commande "Changer" et une zone de saisie texte1 de type textBox. On veut que lorsque l'utilisateur clique sur le bouton, la zone de saisie affiche "Salut toto".
On suppose que la zone de saisie a pour valeur "texte1" dans sa propriété Name .
Pour obtenir cela, on écrira

```
Texte1.Text = "Salut toto"
```

Cette instruction figurera dans la méthode attachée à événement Click du bouton. Elle a pour effet d'affecter la valeur "Salut toto" à la propriété Text du contrôle Texte1.

Si au contraire, lorsque l'utilisateur clique sur le bouton, on souhaite vider la zone de saisie et lui donner la main pour saisir un nouveau texte, on écrira dans la même méthode :

```
Texte1.Text = ""          ' Chaine vide
Texte1.SetFocus
```

L'appel de la méthode prédéfinie SetFocus a pour effet de placer le curseur dans la zone de saisie et d'y afficher ce que l'utilisateur tapera au clavier.

Enfin, pour obtenir que, si la zone est vide, on y affiche "salut toto" et que sinon on la vide et on attende une saisie :

```
If texte1.text = "" then
     texte1.text = "Salut toto"
else
     texte1.text = ""
     texte1.SetFocus
end if
```

D'une manière générale, une propriété ou une méthode prédéfinie d'un objet s'utilise en la faisant précéder du **nom interne (propriété name)** de cet objet.

PROCEDURES ET FONCTIONS

On distingue les procédures événementielles, dont le nom est prédéfini par VB et qui se déclenchent lorsque événement se produit (ex : Command1_Click()) et les procédures personnalisées, définies par le développeur et appelées par lui.

Procédure

```
Sub Nom_Proc ( liste d'arguments )
     [ déclarations de variables locales ]
     instructions
End sub
```

Les procédures peuvent être appelées de deux manières :

```
     call proc(param1, param2)
```

ou

```
     proc param1,param2
```

L'utilisation du mot-clé CALL impose de noter les paramètres entre parenthèses.

Fonction

```
Function Nom_Fonc ( liste d'arguments ) As Type_result
     [ déclarations de variables locales ]
     instructions
     Nom_Fonc = resultat
End Function
```

Les arguments sont données sous la forme suivante :

```
     Nom_arg As Type_Arg
```

Un argument peut être précédé du mot-clé **ByRef** (passage par référence, option par défaut) ou **ByVal** (passage par valeur). Un bon développeur s'astreindra à toujours préciser le mode de transmission (ByRef pour les données modifiées et les résultats, ByVal pour les données).

VISIBILITE DES VARIABLES

Une variable peut être déclarée :

- à l'intérieur d'une procédure ou d'une fonction : elle est visible seulement à l'intérieur de la procédure ou de la fonction
- dans la section des déclarations d'un module de feuille (accessible dans la fenêtre de code : objets General et procs Déclarations) : elle sera visible dans toutes les procédures et fonctions de la feuille.
- Dans un module, elle sera visible par toutes les procédures et fonctions du module, et par toutes les procédures et fonctions de l'application si elle est déclarée "PUBLIC" (Le mot-clé PUBLIC remplace alors le mot DIM.)

BOITES DE SAISIE ET DE DIALOGUE

Pour dialoguer avec l'utilisateur autrement qu'avec les contrôles placés sur une feuille le programmeur dispose de boîtes de dialogues.
On ne s'intéresse ici qu'à celles que nous utiliserons : les boîtes d'information et les boîtes de saisie.

L'instruction **MsgBox** permet de faire afficher une boîte de dialogue affichant un message et munie de boutons de commande.

Syntaxe

```
MsgBox ( message [, Type-Boutons ] [, Titre ] )
```

La syntaxe complète et la signification des arguments peut être consultée dans l'aide en ligne de Visual Basic.

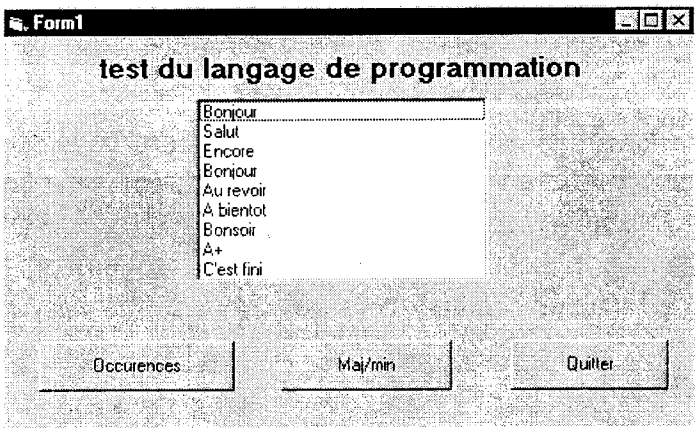
La fonction **InputBox** affiche une boîte de dialogue comportant une zone de saisie et retourne la valeur entrée par l'utilisateur (de type *Variant*). La fonction **InputBox$** se comporte de manière identique mais rend un résultat de type *String*.

Syntaxe

```
Var = InputBox[$]( message, [ titre ], [réponse par défaut ],
                   [xpos, ypos] )
```

EXERCICES

Soit l'interface suivante :



Le bouton Occurrences a pour comportement le suivant :

- une chaine de caractères est demandée à l'utilisateur
- on compte le nombre d'occurrences de cette chaine dans la liste déroulante et on affiche le résultat.

Le bouton Maj/min a pour effet de demander à l'utilisateur s'il veut une conversion en majuscules ou en minuscules (saisie contrôlée) et convertit les éléments de la liste en fonction de la réponse.

Coder en VB les procédures correspondantes.

On dispose de deux fonctions Lcase et Ucase qui reçoivent chacune une chaine de caractères et rendent cette chaine convertie en minuscules (Lcase) ou en majucules (Ucase).

```
Private Sub Command1_Click()
Dim i, n As Integer
Dim str As String

str = InputBox("Chaine à rechercher")
n = 0
For i = 0 To List1.ListCount - 1
   If List1.List(i) = str Then
      n = n + 1
   End If
Next i
MsgBox ("Il y a " & CStr(n) & " Occurrrences de " & str)

End Sub

Private Sub Command2_Click()
Dim rep As String
Dim i As Integer
rep = InputBox("Majuscules (maj) ou Minuscules (min)")

Do Until rep = "maj" Or rep = "min"
rep = InputBox("Majuscules (maj) ou Minuscules (min)")
Loop
If rep = "maj" Then
   For i = 0 To List1.ListCount - 1
      List1.List(i) = UCase(List1.List(i))
   Next i
Else
   For i = 0 To List1.ListCount - 1
      List1.List(i) = LCase(List1.List(i))
   Next i
End If
List1.Refresh
End Sub

Private Sub Command3_Click()
Unload Me
End Sub
```