Créer des boites de dialogue avec VBA

Introduction

Le langage VBA permet d'afficher cinq sortes de boîtes de dialogue que vous pourrez parfois utiliser à la place des boîtes de dialogue personnalisées. Elles sont plus ou moins modifiables, mais sans offrir toutes les possibilités d'une véritable boîte de dialogue personnalisée.

- La fonction MsgBox.
- La fonction InputBox.
- La méthode GetOpenFilename.
- La méthode GetSaveAsFilename.
- La méthode FileDialog.

La fonction MsgBox

Afficher une boîte de message simple

Une fonction MsgBox est utilisable de deux manières :

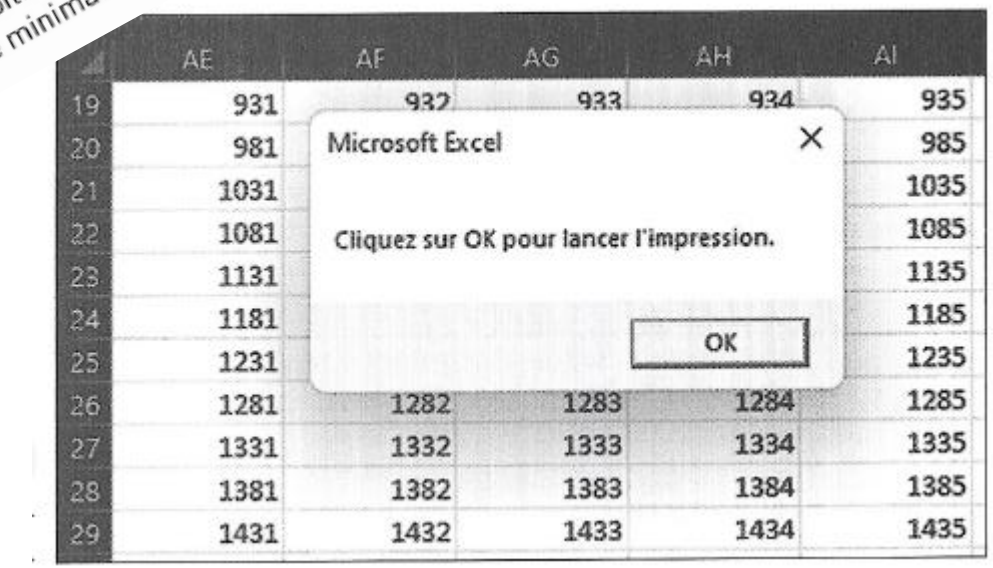
» **Pour afficher un message à destination de l'utilisateur : Dans ce cas, vous ne vous souciez pas du résultat renvoyé par la fonction.**

» **Obtenir une réponse de la part de l'utilisateur : Dans ce cas, vous tenez compte du résultat retourné par la fonction. Il dépend du bouton sur lequel l'utilisateur a cliqué.**

Si vous désirez utiliser cette fonction pour elle-même, ne mettez pas ses arguments entre parenthèses. L'exemple suivant se contente d'afficher un message sans retourner de résultat. Dès que le message est affiché, le code s'arrête jusqu'à ce que l'utilisateur ait cliqué sur OK :

```
Sub DémoMsgBox()
    MsgBox "Cliquez sur OK pour lancer l'impression."
    Sheets("Résultats").PrintOut
End Sub
```

Une boîte de mes-
sage minimaliste.



l'impression commence une fois que l'utilisateur a cliqué sur OK. Mais rien n'est prévu pour annu ler cette opération.

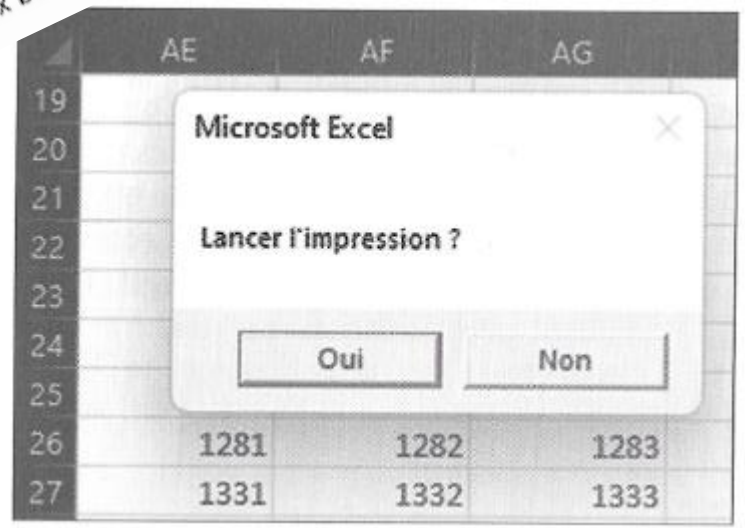
Obtenir une réponse d'une boîte de message

Lorsque le message qui devra être affiché contient plus de boutons qu'un simple bouton OK, le programme VBA doit savoir sur lequel l'utilisateur a cliqué. La fonction MsgBox est heureusement capable de retrouver une valeur associée au bouton qui a été cliqué. Le résultat de la fonction MsgBox peut être affecté à une variable.

Dans le code suivant, quelques constantes prédéfinies facilitent le travail à partir des valeurs retournées par MsgBox :

```
Sub ObtenirUneRéponse()
    Dim Réponse As Integer
    Réponse = MsgBox("Lancer l'impression ?", vbYesNo)
    Select Case Réponse
        Case vbYes
'       ...[code si la réponse est Oui]...
        Case vbNo
'       ...[ code si la réponse est Oui]...
    End Select
End Sub
```

Une boîte de message simple, avec deux boutons.



La Figure montre le résultat. Quand vous exécutez cette procédure, la variable Réponse reçoit la valeur produite, soit vbYes ou vbNo selon le bouton qui a été cliqué. L'instruction Select Case utilise la valeur Réponse pour déterminer l'action que la routine doit accomplir.

Les constantes de la fonction MsgBox.

Constante	Valeur	Description
vbOKOnly	0	N'affiche que le bouton OK.
vbOKCancel	1	Affiche les boutons de commande OK et Annuler.
vbAbortRetryIgnore	2	Affiche les boutons de commande Abandonner, Recommencer et Ignorer.
vbYesNoCancel	3	Affiche les boutons de commande Oui, Non et Annuler.
vbYesNo	4	Affiche les boutons de commande Oui et Non.
vbRetryCancel	5	Affiche les boutons de commande Recommencer et Annuler.
vbCritical	16	Affiche l'icône de message critique (cercle rond avec le X blanc) et émet le son associé.
vbQuestion	32	Affiche l'icône de question (phylactère bleu avec un point d'interrogation) et émet le son associé.
vbExclamation	48	Affiche l'icône d'alerte (triangle jaune avec le point d'exclamation) et émet le son associé.
vbInformation	64	Affiche l'icône d'information (phylactère bleu avec un i) et émet le son associé.
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 quatrième bouton est le bouton par défaut.

Vous pouvez aussi utiliser le résultat de la fonction MsgBox sans passer par une variable, comme le démontre la variante suivante :

```
Sub ObtenirUneRéponse2()
    If MsgBox("Lancer l'impression ?", vbYesNo) = vbYes Then
'       ...[code si clic sur Oui]...
    Else
'       ...[ code si pas de clic sur Oui]...
    End If
End Sub
```

Personnaliser les boîtes de message

La souplesse de l'argument Boutons facilite la personnalisation des boîtes de message. Vous pouvez en effet spécifier quels boutons doivent être affichés, décider si une icône doit apparaître et également quel sera le bouton par défaut (celui qui sera « cliqué » si l'utilisateur appuie sur la touche Entrée).

Pour utiliser plusieurs de ces constantes comme argument, connectez-les simplement avec un opérateur + (plus). Par exemple, pour afficher une boîte de message avec seulement les boutons Oui et Non et l'icône d'exclamation, utilisez l'expression suivante comme deuxième argument MsgBox :

```
vbYesNo + vbExclamation
```

L'exemple suivant utilise une combinaison de constantes pour afficher une boîte de message avec des boutons Oui et Non (vbYesNo) ainsi qu'une icône de questionnement (vbQuestion). La constante vbDefaultButton2 désigne le deuxième bouton (Non) comme bouton par défaut, c'est-à-dire celui qui est pris en compte en appuyant sur la touche Entrée. Par souci de simplicité, j'affecte ces constantes à la variable Config, qui est ensuite utilisée comme second argument de la fonction MsgBox :

```
Sub ObtenirUneRéponse3()
    Dim Config As Long
    Dim Réponse As Integer
                                        ltButton2
    Réponse = MsgBox("Traiter le rapport mensuel ?", Config)
    If Réponse = vbYes Then LancerRapport
End Sub
```

```
Sub ObtenirUneRéponse3()
    Dim Config As Long
    Dim Réponse As Integer
    Config = vbYesNo + vbQuestion + vbDefaultButton2
    Réponse = MsgBox("Traiter le rapport mensuel ?", Config)
    If Réponse = vbYes Then LancerRapport
End Sub
```

L'argument Bouton de la fonction MsgBox définit le bouton par défaut.

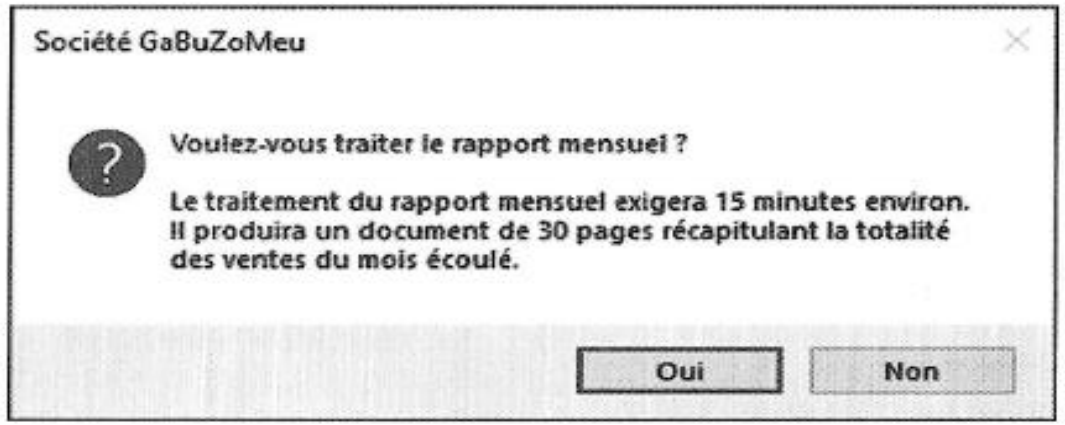
La Figure montre la boîte de message affichée par Excel lorsque la procédure ObtenirUneRéponse3 est exécutée. Si l'utilisateur clique sur le bouton Oui, la routine exécute la procédure nommée LancerRapport (elle n'est pas mentionnée dans le listing). S'il clique sur le bouton Non, ou s'il appuie sur Entrée, la routine s'achève sans qu'aucune action soit entreprise. Comme l'argument Titre a été omis dans la fonction MsgBox, Excel affiche le titre par défaut : Microsoft Excel.

un autre exemple de fonction MsgBox :

```
Sub ObtenirUneRéponse4()
    Dim Msg As String, Title As String
    Dim Config As Integer, Réponse As Integer
    Msg = "Voulez-vous traiter le rapport mensuel ?"
    Msg = Msg & vbNewLine & vbNewLine
    Msg = Msg & "Le traitement du rapport mensuel "
    Msg = Msg & "exigera 15 minutes environ. Il "
    Msg = Msg & "produira un document de 30 pages "
    Msg = Msg & "récapitulant la totalité des "
    Msg = Msg & "ventes du mois écoulé."
    Title = "Société GaBuZoMeu"
    Config = vbYesNo + vbQuestion
    Réponse = MsgBox(Msg, Config, Title)
    If Réponse = vbYes Then LancerRapport
End Sub
```

Cet exemple montre comment placer efficacement un texte long dans une boîte de message. Une variable (Msg) et un opérateur de concaténation (&) ont été utilisés pour construire le message par une succession d'instructions. La constante vbNewLine insère un retour à la ligne (il faut donc l'utiliser deux fois pour produire une ligne vierge). Le nom de la société a été placé dans la barre de titre grâce à l'argument de titre de MsgBox.

Une boîte de dialogue affichée par la fonction MsgBox, avec un titre, une icône et deux boutons.




```
Sub ObtenirUneRéponse4()
    Dim Msg As String, Title As String
    Dim Config As Integer, Réponse As Integer
    Msg = "Voulez-vous traiter le rapport mensuel ?"
    Msg = Msg & vbNewLine & vbNewLine
    Msg = Msg & "Le traitement du rapport mensuel "
    Msg = Msg & "exigera 15 minutes environ. Il "
    Msg = Msg & "produira un document de 30 pages "
    Msg = Msg & "récapitulant la totalité des "
    Msg = Msg & "ventes du mois écoulé."
    Title = "Société GaBuZoMeu"
    Config = vbYesNo + vbQuestion
    Réponse = MsgBox(Msg, Config, Title)
    If Réponse = vbYes Then LancerRapport
End Sub
```

Les constantes utilisées comme valeurs de retour par la fonction MsgBox.

Constante	Valeur	Signification
vbOK	1	L'utilisateur a cliqué sur OK.
vbCancel	2	L'utilisateur a cliqué sur Annuler.
vbAbort	3	L'utilisateur a cliqué sur Abandonner.
vbRetry	4	L'utilisateur a cliqué sur Recommencer.
vbIgnore	5	L'utilisateur a cliqué sur Ignorer.
vbYes	6	L'utilisateur a cliqué sur Oui.
vbNo	7	L'utilisateur a cliqué sur Non.

La fonction InputBox

La fonction InputBox du langage VBA sert à obtenir une valeur unique saisie par l'utilisateur. Il peut s'agir d'un nombre, d'une chaîne de caractères, et même d'une plage d'adresses. C'est une excellente alternative au développement de boîtes de dialogue personnalisées (les objets UserForms) quand le but est de récupérer une seule valeur.

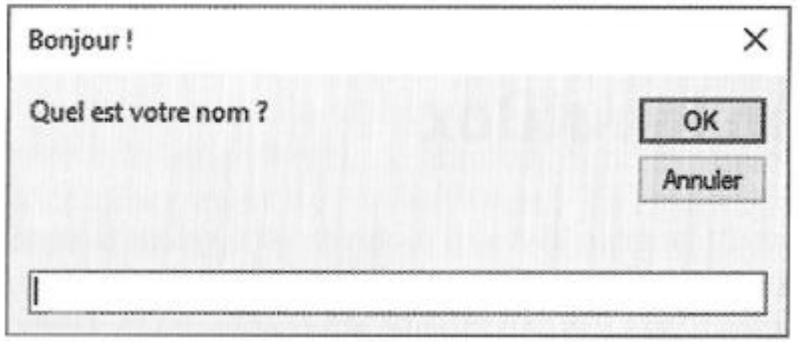
La syntaxe de InputBox

Voici une version simplifiée de la syntaxe de la fonction InputBox :

```
InputBox(invite[, titre][, parDéfaut])
```

Argument	Description
invite	Le texte affiché dans la boîte de saisie.
titre	Spécifie le texte affiché dans la barre de titre de la boîte de saisie (facultatif).
parDéfaut	Définit la valeur par défaut (facultatif).

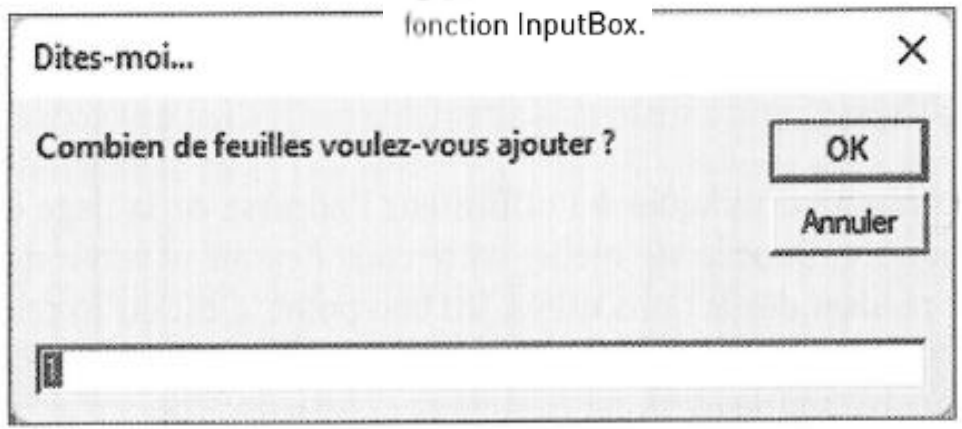
```
LeNom = InputBox("Quel est votre nom ?", "Bonjour !")
```



La fonction VBA InputBox retourne toujours une chaîne. Vous devrez au besoin la convertir en valeur numérique en effectuant les contrôles nécessaires. L'exemple qui suit fait appel à la fonction InputBox pour demander un nombre. Si la chaîne renvoyée contient effectivement une valeur numérique, tout continue normalement. Sinon, le code affiche un message d'erreur.

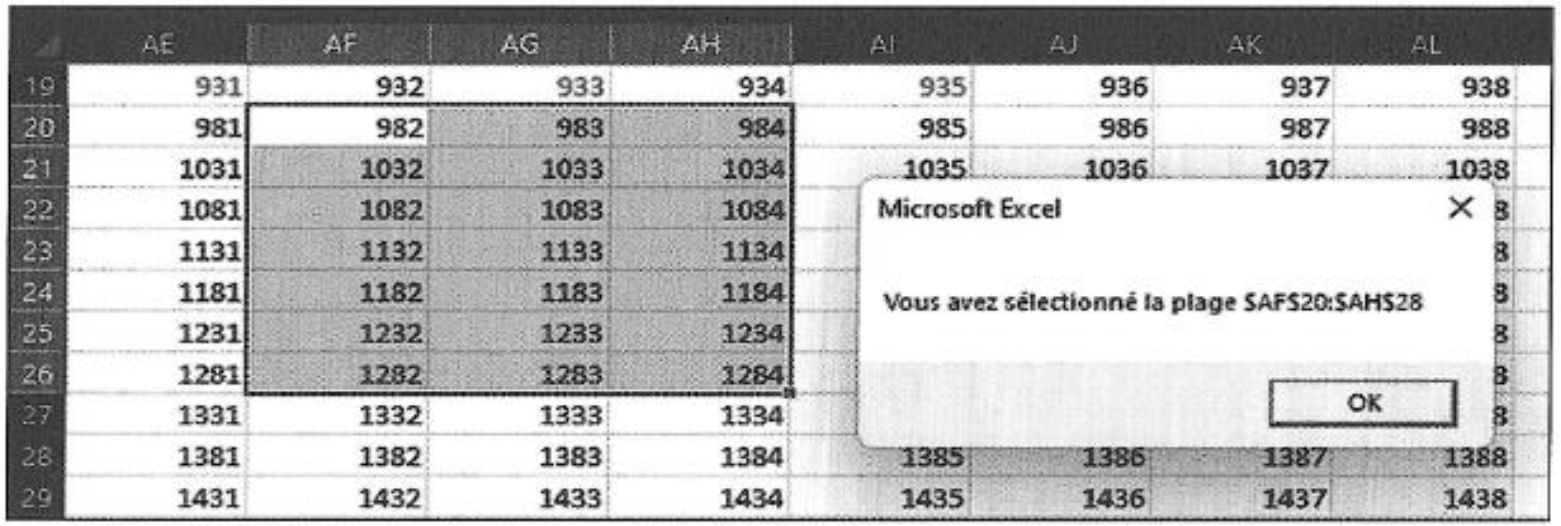
```
Sub AjouteFeuilles()
    Dim Invite As String
    Dim Titre As String
    Dim DefValeur As Integer
    Dim NombFeuilles As String

    Invite = "Combien de feuilles voulez-vous ajouter ?"
    Titre = "Dites-moi..."
    DefValeur = 1
    NombFeuilles = InputBox(Invite, Titre, DefValeur)

    If NombFeuilles = "" Then Exit Sub 'Annulation
    If IsNumeric(NombFeuilles) Then
        If NombFeuilles > 0 Then Sheets.Add Count:=NombFeuilles
    Else
        MsgBox "Entrez un nombre valide !"
    End If
End Sub
```

Un autre exemple d'utilisation de la fonction InputBox.



L'un des grands avantages de la méthode Application InputBox est la possibilité de demander à l'utilisateur de sélectionner une plage au clavier ou avec la souris. Voici un bref exemple :

```
Sub LitPlage()
    Dim Rng As Range
    On Error Resume Next
    Set Rng = Application.InputBox _
      (prompt:="Spécifiez une plage:", Type:=8)
    If Rng Is Nothing Then Exit Sub
    MsgBox "Vous avez sélectionné la plage " & Rng.Address
End Sub
```



Utiliser la méthode InputBox de l'objet Application pour sélectionner une plage de cellules.

La fonction GetOpenFileName

Si une procédure VBA doit demander à l'utilisateur de spécifier un nom de fichier, vous *pourriez* le faire avec la fonction InputBox. Mais ce n'est généralement pas l'outil le plus approprié pour ce genre de tâche, car la plupart des utilisateurs ont des difficultés à se souvenir des chemins et des noms de répertoires, sans compter les risques d'erreurs de saisie.

Pour éviter ces problèmes, utilisez de préférence la méthode GetOpenFilename de l'objet Application. Elle garantit en effet que l'application obtiendra un nom de fichier valide, y compris son chemin complet. Cette méthode affiche la classique boîte de dialogue Ouvrir, c'est-à-dire celle qui apparaît en choisissant la commande Fichier > Ouvrir > Parcourir.

La méthode GetOpenFilename n'ouvre pas véritablement le fichier. Elle ne fait que retourner le nom du fichier sélectionné par l'utilisateur. Vous pourrez ensuite écrire du code qui fera ce que vous désirez à partir de ce nom de fichier.

Exemple d'utilisation de GetOpenFilename

L'argument filtrageFichiers sert à indiquer ce qui doit apparaître dans la liste Types de fichiers, en bas de la boîte de dialogue Ouvrir. Il est constitué d'une paire de chaînes pour le filtrage de fichiers, suivie par la spécification de celui-ci grâce à des caractères de substitution. Les deux éléments sont séparés par une virgule. Si l'argument est omis, l'argument suivant est utilisé par défaut :

```
Tous les fichiers (*.*), *.*
```

Remarquez que la chaîne est constituée de deux parties :

```
Tous les fichiers (*.*)
```

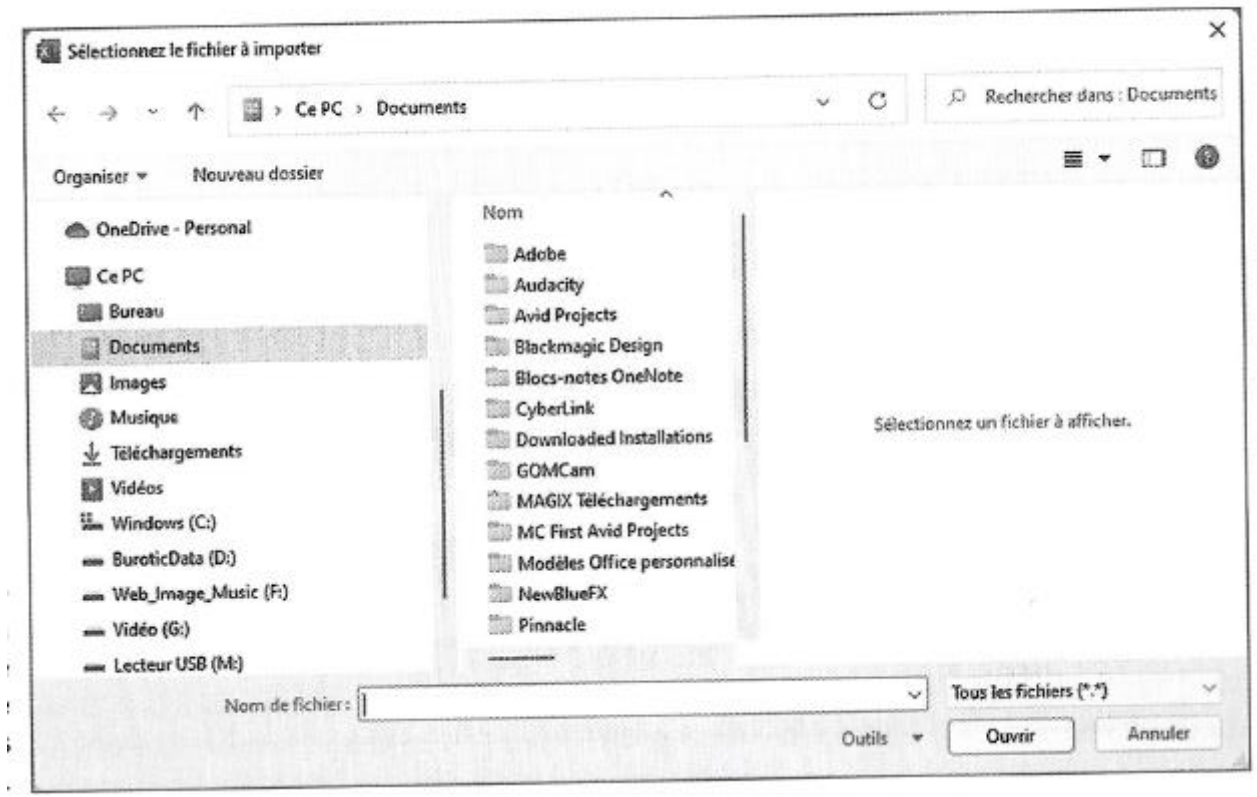
et

```
*.*
```

La première partie de cette chaîne est le texte affiché dans la liste déroulante Types de fichiers. La seconde partie définit quels sont les fichiers affichés dans la boîte de dialogue. Par exemple, *.* signifie *tous les fichiers.*

```
Sub RécupNomFichier()
    Dim typeFichier As String
    Dim filtreIndex As Integer
    Dim Titre As String
    Dim nomFichier As Variant

'   Etablissement de la liste des filtres de fichiers
    typeFichier = "Fichiers texte (*.txt),*.txt," & _
                  "Fichiers Lotus (*.prn),*.prn," & _
                  "Texte (séparateur : point-virgule) (*.csv),*.csv," & _
                  "Fichiers ASCII (*.asc),*.asc," & _
                  "Tous les fichiers (*.*),*.*"

'   Affichage par défaut = *.*
    filtreIndex = 5

'   Définition du message de la barre de titre
    Titre = "Sélectionnez le fichier à importer"

'   Récupération du nom de fichier
    nomFichier = Application.GetOpenFilename(typeFichier, _
       filtreIndex, Titre)

'   Gestion des données provenant de la boîte de dialogue

    If nomFichier = False Then
        MsgBox "Aucun fichier n'a été sélectionné."
    Else
        MsgBox "Vous avez sélectionné le fichier " & nomFichier
    End If
End Sub
```

La méthode GetOpenFilename affiche une boîte de dialogue personnalisée. Elle retourne aussi le nom du fichier sélectionné et son chemin. Elle n'ouvre toutefois pas le fichier.



Dans une application opérationnelle, vous feriez quelque chose de plus utile, comme ouvrir le fichier avec une instruction comme celle-ci :

```
Workbooks.Open nomFichier
```

La fonction GetSaveAsFileName

La méthode GetSaveAsFilename fonctionne comme la méthode GetOpenFile-name, sauf qu'elle affiche la boîte de dialogue Enregistrer sous d'Excel à la place d'Ouvrir. Elle récupère le chemin et le nom de fichier indiqués par l'utilisateur, mais n'en fait rien. Autrement dit, c'est à vous d'écrire le code qui sauvegarde effectivement le fichier.

Voici la syntaxe de cette méthode :

```
object.GetSaveAsFilename([nomFichierInitial], [filtrageFichiers],
        [indexFiltrage], [titre], [texteBouton])
```

Argument	Description
nomFichierInitial	Spécifie le nom par défaut qui apparaît dans le champ Nom de fichier.
filtrageFichiers	Définit les types de fichiers affichés par Excel, comme par exemple *.txt. Plusieurs filtres peuvent être utilisés afin de proposer un choix plus large à l'utilisateur.
indexFiltrage	Définit le filtre de fichier qu'Excel doit proposer par défaut.
titre	Contient le texte affiché dans la barre de titre de la boîte de dialogue.

La procédure qui suit affiche une boîte de dialogue qui permet à l'utilisateur de sélectionner un dossier. Le nom de ce dossier (ou le message « Annulé ») est ensuite affiché à l'aide de la fonction MsgBox :

```
Sub RecupDossier()
    With Application.FileDialog(msoFileDialogFolderPicker)
      .InitialFileName = Application.DefaultFilePath & "\"
      .Title = "Sélectionnez un emplacement pour la sauvegarde"
      .Show
      If .SelectedItems.Count = 0 Then
          MsgBox "Annulé"
      Else
          MsgBox .SelectedItems(1)
      End If
    End With
End Sub
```

L'objet FileDialog permet de spécifier le chemin d'accès initial en fournissant ce-lui-ci dans la propriété InitialFileName. En l'occurrence, ce code se sert de point de départ du chemin par défaut utilisé par Excel.

Afficher les boites pré définies d’Excel

Vous pouvez écrire du code VBA qui exécute des actions comme si elles avaient été sélectionnées dans le ruban ou dans une boîte de dialogue, sans même qu'Excel affiche quoi que ce soit de particulier.

Par exemple, l'instruction ci-dessus est équivalente au fait de choisir, dans le groupe Noms définis de l'onglet Formules, la commande Définir un nom pour afficher la boîte de dialogue Nouveau nom, puis de spécifier dans le champ Nom la valeur NomsMois, et enfin de cliquer sur OK :

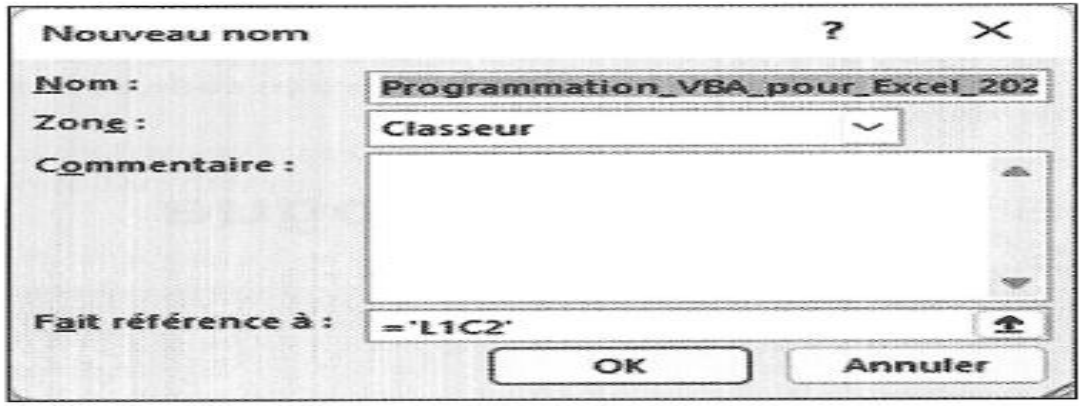
```
Range("A1:A12").Name = "NomsMois"
```

Quand vous exécutez cette instruction, la boîte de dialogue Nouveau nom n'apparaît pas. Et c'est pratiquement toujours ce que vous désirez. Il ne faudrait en effet pas qu'une boîte de dialogue surgisse tout d'un coup sur l'écran lors de l'exécution de la macro.

À la différence du précédent, l'exemple suivant affiche la boîte de dialogue Nouveau nom. La plage qui apparaît dans le champ fait référence à est celle qui était sélectionnée au moment où la commande est exécutée

Le code pour afficher cette boîte de dialogue est le suivant :

```
Sub NouvNom()
    Application.CommandBars.ExecuteMso "NameDefine"
End Sub
```



Afficher une boîte de dialogue d'Excel en utilisant VBA.

ExecuteMso est une méthode de l'objet CommandBars. Elle accepte un seul argument, qui est un paramètre idMso représentant un contrôle du ruban. Malheureusement, ces paramètres ne sont pas listés dans l'aide de VBA. Et comme le ruban est un concept relativement récent, le code qui fait appel à la méthode ExecuteMso n'est pas compatible avec les versions antérieures à Excel 2007.

Boite de dialogue personnalisée: UserForm

Créer une boîte de dialogue personnalisée : vue d'ensemble

Pour créer une boîte de dialogue personnalisée, vous suivrez généralement ces étapes :

1. **Détermination de la manière dont la boîte de dialogue sera utilisée et de l'endroit où elle apparaîtra dans la macro VBA.**

2. **Appui sur Alt+F11 afin d'activer l'éditeur VBE et d'insérer un nouvel objet UserForm.**

   Un objet UserForm ne contient qu'une seule boîte de dialogue personnalisée.

3. **Ajout de contrôles à l'objet UserForm.**

   Les contrôles sont notamment des boîtes de texte, des boutons, des cases à cocher et des zones de liste.

4. **Utilisation de la fenêtre Propriétés pour modifier les propriétés des contrôles ou l'objet UserForm lui-même.**

5. **Écriture de procédures de gestion des événements pour les contrôles (par exemple, une macro est exécutée lorsque l'utilisateur clique sur un bouton dans la boîte de dialogue).**

   Ces procédures sont stockées dans la fenêtre Code de l'objet UserForm.

6. **Écriture d'une procédure – stockée dans un module VBA – qui affiche la boîte de dialogue pour l'utilisateur.**

Travailler avec les objets UserForm

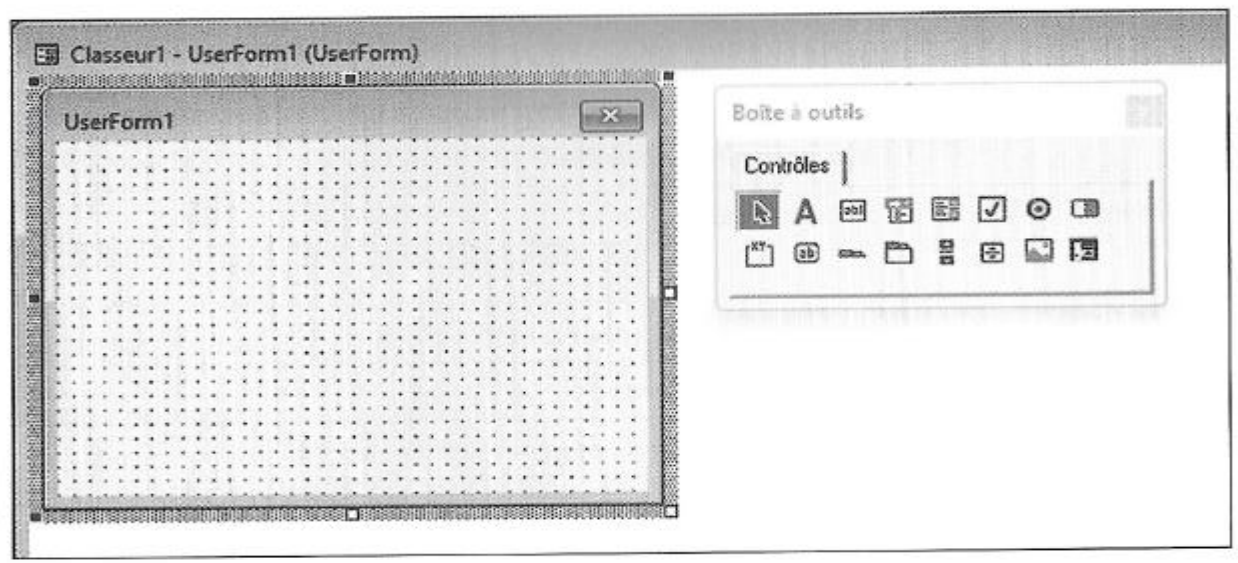
Chacune des boîtes de dialogue personnalisées que vous créez est stockée dans son propre objet UserForm (à raison d'une boîte par objet). Vous créez ces objets UserForm, et vous y accédez dans Visual Basic Editor.

Insérer un nouvel objet UserForm

Procédez comme suit pour insérer un objet UserForm :

1. **Activez l'éditeur VBE en appuyant sur Alt+F11.**
2. **Sélectionnez le classeur dans la fenêtre Projet.**
3. **Choisissez Insertion > UserForm.**

   L'éditeur VBE insère un nouvel objet UserForm contenant une boîte de dialogue vide.



Un nouvel objet UserForm.

outils que vous allez ajouter des contrôles à vos boîtes de dialogue personnalisées. Si vous ne voyez pas cette fenêtre, choisissez la commande Affichage > Boîte à outils.

Pour ajouter un contrôle, cliquez dessus puis faites-le glisser sur la boîte de dialogue pour l'insérer à l'intérieur de celle-ci. Vous pouvez ensuite le déplacer et le redimensionner à l'aide des techniques habituelles.

Les contrôles de la boîte à outils.

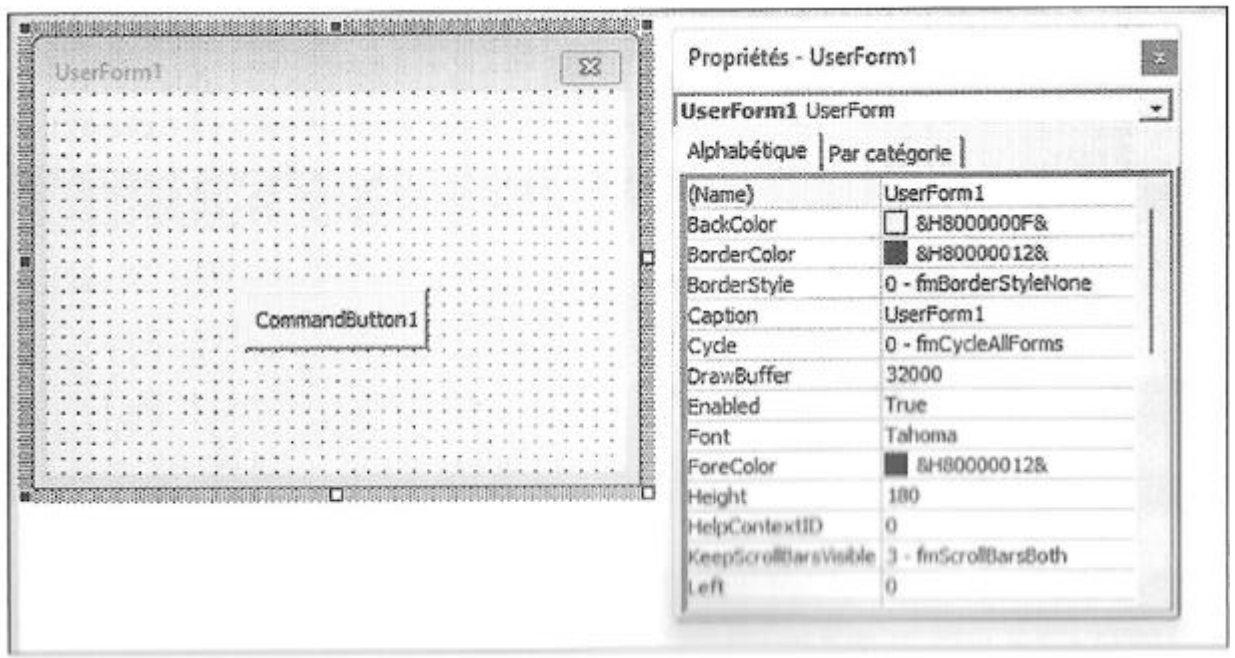
Contrôle	Description
Intitulé	Stocke du texte.
Zone de texte	Permet à l'utilisateur d'entrer du texte.
Zone de liste modifiable	Affiche une liste déroulante.
Zone de liste	Affiche une liste d'éléments.
Case à cocher	Commode pour les options actif/inactif et oui/non.
Bouton d'option	N'autorise le choix que d'une option parmi d'autres du groupe de boutons.

Contrôle	Description
Bouton bascule	Commutateur à deux positions.
Cadre	Contient d'autres contrôles.
Bouton de commande	Bouton cliquable.
Contrôle onglet	Onglets.
Multipage	Conteneur à onglets pour d'autres objets.
Barre de défilement	Barre avec un ascenseur.
Toupie	Bouton cliquable servant généralement à modifier une valeur.
Image	Contient une image.
RefEdit	Permet à l'utilisateur de sélectionner une plage.

Modifier les propriétés d'un contrôle UserForm

Chacun des contrôles d'un objet UserForm contient des propriétés qui déterminent son apparence ou son comportement. Ces propriétés sont modifiables.

Utilisez la fenêtre Propriétés pour modifier les propriétés d'un contrôle UserForm.



La fenêtre Propriétés apparaît en appuyant sur F4. Les propriétés affichées dépendent évidemment de ce qui a été sélectionné et changent par exemple d'un contrôle à un autre. Pour la masquer, cliquez sur sa case de fermeture. La touche F4 la ramènera à la vie lorsque vous en aurez à nouveau besoin.

Voici quelques propriétés de contrôles, suivies de leur traduction :

- **Name (nom)**
- **Width (largeur)**
- **Height (hauteur)**
- **Value (valeur)**
- **Caption (légende)**

Pour modifier une propriété d'un contrôle, procédez comme suit :

1. **Commencez par sélectionner le contrôle voulu dans le cadre de l'objet UserForm.**
2. **Assurez-vous que la fenêtre Propriétés est visible (si besoin est, appuyez sur F4).**
3. **Dans la fenêtre Propriétés, cliquez sur la ligne qui définit la propriété que vous voulez éditer.**
4. **Apportez les changements voulus dans la partie droite de la fenêtre Propriétés.**

Si vous sélectionnez l'objet UserForm lui-même (et non un de ses contrôles), la fenêtre Propriétés affiche les propriétés de la boîte de dialogue proprement dite.

Utiliser les informations fournies par une boîte de dialogue personnalisée

L'éditeur VBE attribue un nom à chacun des contrôles que vous ajoutez à un objet UserForm. Ce nom est celui qui est présent dans la propriété Name. Utilisez-le dans votre code pour faire référence à tel ou tel contrôle. Par exemple, si vous avez placé un contrôle Case à cocher dans un objet UserForm nommé UserForm1, par défaut le contrôle Case à cocher sera nommé CheckBox1. L'instruction ci-dessous fait apparaître le contrôle déjà coché :

```
UserForm1.CheckBox1.Value = True
```

La plupart du temps, votre code sera inséré dans le module de code l'objet UserForm lui-même. Vous pouvez alors omettre son nom et simplifier vos instructions, comme ici :

```
CheckBox1.Value = True
```

La fenêtre Code de l'objet UserForm

Chaque objet UserForm est doté d'une fenêtre Code qui contient le code VBA (les procédures d'événement) exécuté lorsque l'utilisateur interagit avec la boîte de dialogue. Appuyez sur F7 pour afficher la fenêtre Code. Elle reste vide tant que vous n'avez pas ajouté de procédures. Appuyez sur Maj+F7 pour revenir à la boîte de dialogue.

Un autre moyen de passer de la fenêtre Code à la fenêtre UserForm consiste à cliquer, dans la barre de titre de la fenêtre Projet, sur les boutons Afficher le code et Afficher l'objet. Il est également possible de cliquer du bouton droit sur l'objet UserForm et de choisir Code dans le menu qui apparaît. Faites alors un double-clic sur le nom de l'objet UserForm dans la fenêtre Projet pour le réafficher.

Afficher une boîte de dialogue personnalisée

Vous pouvez afficher une boîte de dialogue personnalisée en utilisant la méthode Show de l'objet UserForm, dans une procédure VBA.

La macro qui affiche la boîte de dialogue doit se trouver dans le module VBA, et non dans la fenêtre Code de l'objet UserForm.

La procédure suivante affiche la boîte de dialogue nommée UserForm1 :

```
Sub AfficherBoîteDialogue()
    UserForm1.Show
'   [Placez ici les autres instructions]
End Sub
```

Les contrôles des boites de dialogue avec VBA

Un utilisateur répond à une boite de dialogue personnalisée (UserForm dans VBE) grâce à différents **contrôles** (boutons de commande et d'option, champs de saisie, …) qu'elle contient. Le code VBA exploite ensuite ses réponses pour déterminer l'action qu'il doit entreprendre.

Ajouter des contrôles

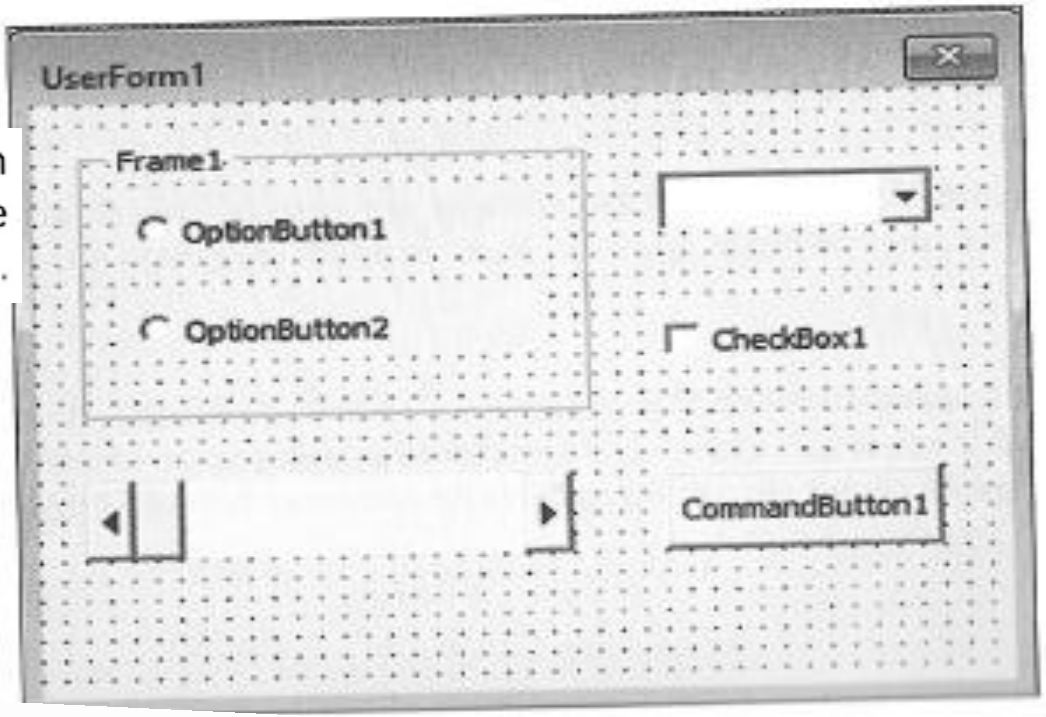
Procédez comme suit pour placer un contrôle dans une boîte de dialogue personnalisée :

1. **Dans la boîte à outils, cliquez sur le contrôle à ajouter.**
2. **Cliquez dans l'objet UserForm.**

   Le contrôle y apparaît.
3. **Agissez sur ses poignées de redimensionnement pour modifier la taille du contrôle.**

Ou alors, glissez-déposez le contrôle afin de conserver ses dimensions

Un objet UserForm agrémenté de quelques contrôles.



Un objet UserForm peut afficher un réseau de points qui facilitent l'alignement des objets. Ils s'accrochent spontanément à ces repères, comme s'ils étaient magnétisés. Si cette fonctionnalité vous gêne – elle empêche les positionnements intermédiaires –, vous pouvez la désactiver :

1. **Dans l'éditeur VBE, choisissez Outils > Options.**
2. **Dans la boîte de dialogue Options, cliquez sur l'onglet Général.**
3. **Définissez vos préférences dans la rubrique Paramètres de la grille de la feuille.**

Introduction aux propriétés des contrôles

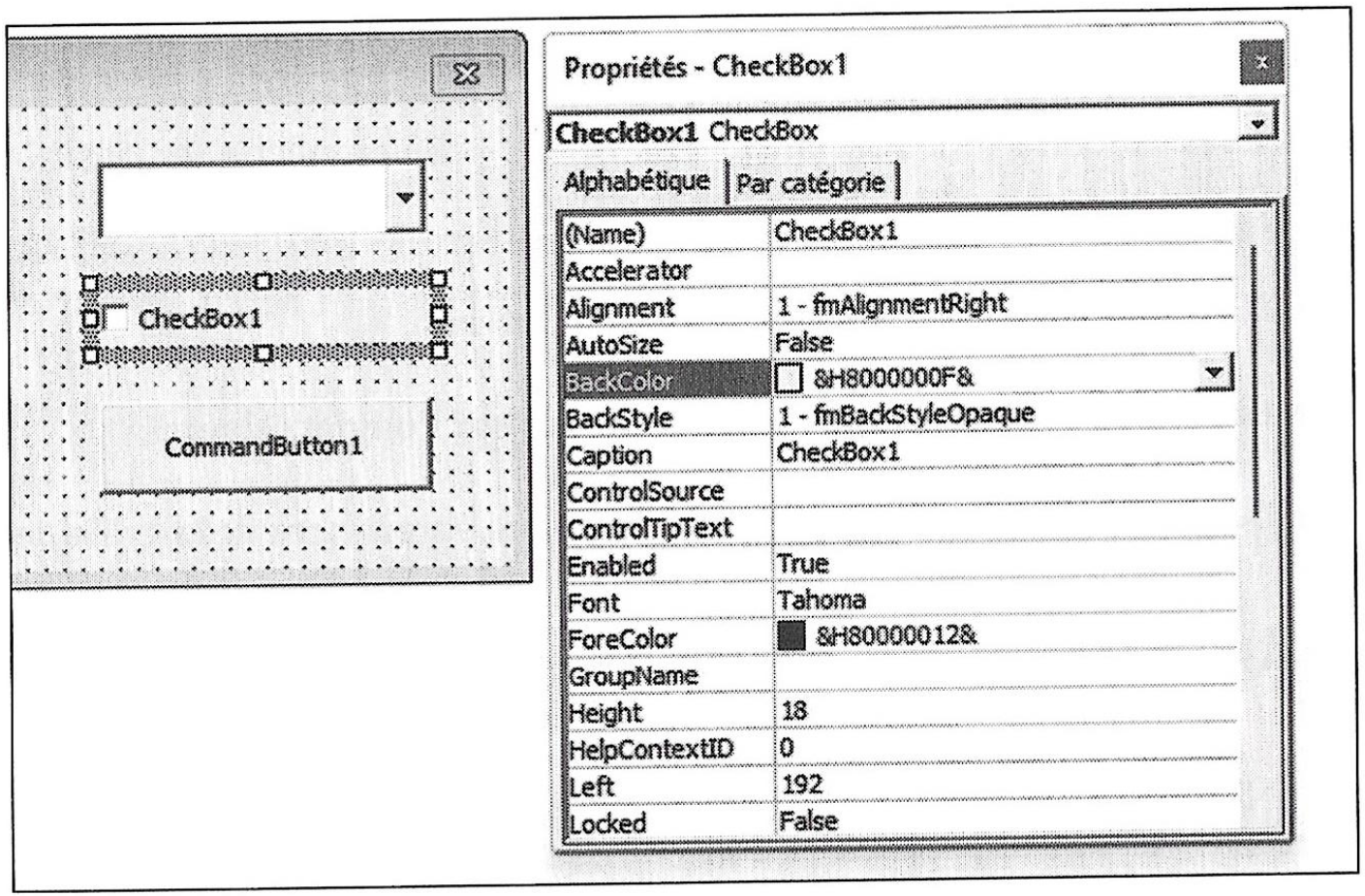
Chaque contrôle que vous placez dans un objet UserForm a des propriétés qui définissent son apparence et son comportement. Les propriétés d'un contrôle peuvent être modifiées à deux moments :

» **Lors de la conception de l'objet UserForm. Les modifications sont effectuées manuellement, dans la fenêtre Propriétés.**

» **Pendant l'exécution. Vous écrivez à cette fin un programme VBA spécifique. Ces changements sont toujours temporaires, car ils sont effectués dans la *copie* de la boîte de dialogue personnalisée qui est montrée par le programme, et non dans l'objet UserForm tel qu'il est enregistré dans le classeur.**

Quand vous ajoutez un contrôle à un objet UserForm, vous devez presque toujours procéder à quelques paramétrages de ses propriétés. Ils s'effectuent dans la fenêtre Propriétés, affichée en appuyant sur F4.

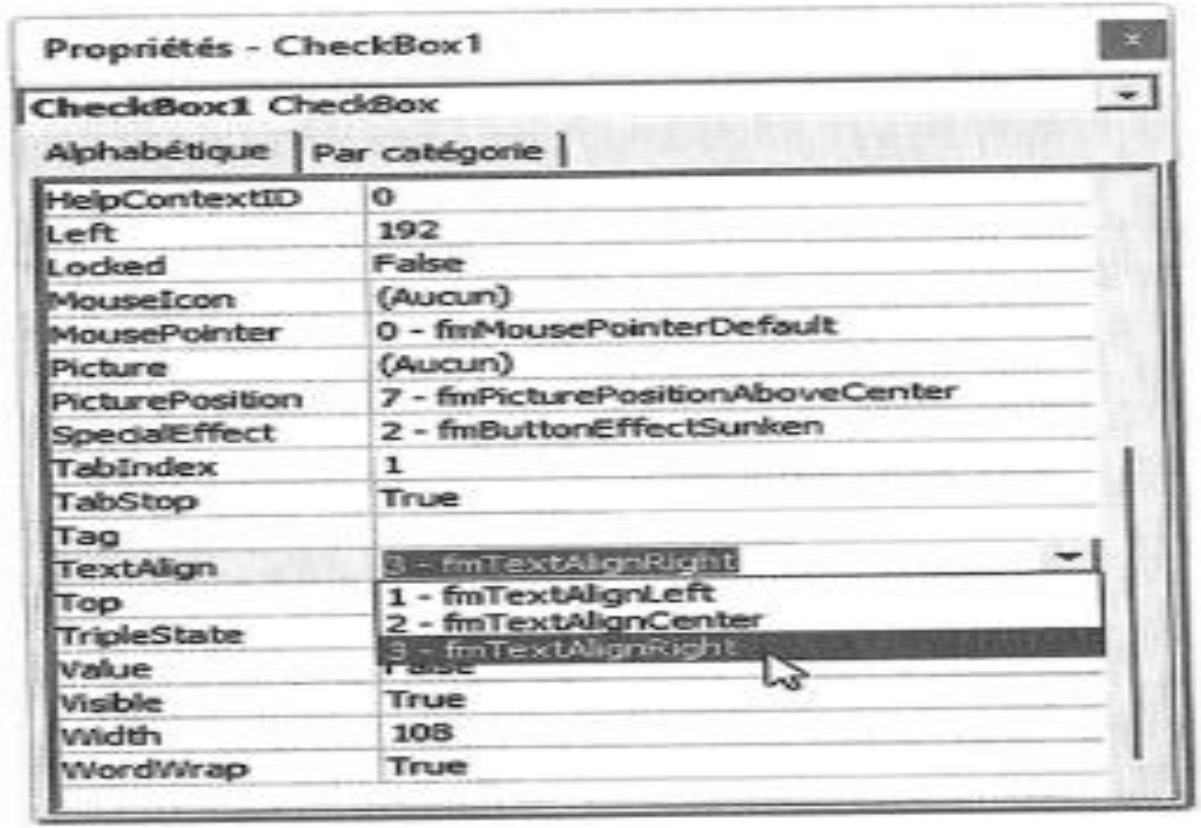
Pendant la conception, les propriétés d'un contrôle sont paramétrées dans la fenêtre Propriétés.



Les propriétés communes des contrôles.

Propriété	Action
Accelerator	Souligne la première occurrence de la lettre dans le libellé, pour en faire un raccourci clavier. L'utilisateur appuie sur Alt+*lettre* pour sélectionner la commande.
AutoSize	Si elle est sur True, le contrôle se redimensionne automatiquement selon la longueur du texte présent dans son intitulé (Caption).
BackColor	Définit la couleur d'arrière-plan du contrôle.
BackStyle	Définit le style (transparent ou opaque) de l'arrière-plan.
Caption	Texte du contrôle.
Left et Top	Valeurs définissant la position du contrôle (bord gauche, bord supérieur).
Name	Le nom du contrôle. Par défaut, le nom d'un contrôle est basé sur son type. Vous pouvez choisir n'importe quel nom valide (la syntaxe est la même que celle des noms de macro). Chaque contrôle doit toutefois avoir un nom unique dans la boîte de dialogue.
Picture	Image à afficher. Elle peut se trouver dans un fichier graphique. Vous pouvez également sélectionner la propriété Picture et coller une image préalablement copiée dans le Presse-papiers.
Value	Valeur du contrôle.
Visible	Si False, le contrôle est masqué.
Width et Height	Valeurs définissant la largeur et la hauteur du contrôle.

Quand vous sélectionnez un contrôle, ses propriétés apparaissent dans la fenêtre Propriétés. Pour en modifier une, sélectionnez-la puis modifiez son contenu. Certaines vous aident un peu. Par exemple, si vous désirez modifier la valeur de TextAlign, la fenêtre Propriétés déroulera un menu contenant toutes les valeurs valides

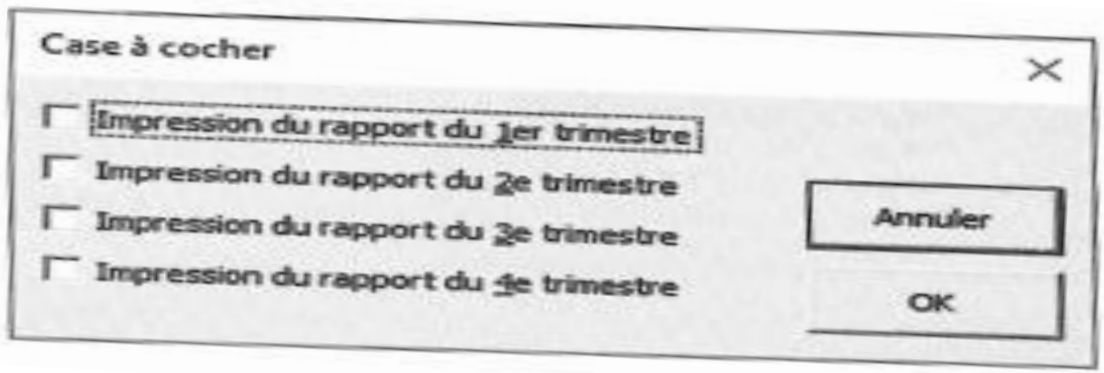


Les contrôles en détail

Le contrôle Case à cocher

Un contrôle Case à cocher est commode pour exprimer un choix binaire : oui ou non, vrai ou faux, actif ou inactif, *etc.*

Des contrôles Case à cocher.



Voici une description des propriétés de contrôle Case à cocher les plus utiles :

» **Accelerator** : souligne la première occurrence de la lettre dans le libellé, pour en faire un raccourci clavier. Si la lettre est a, appuyer sur Alt+A modifie la valeur du contrôle Case à cocher : la case est alors affichée cochée ou décochée.

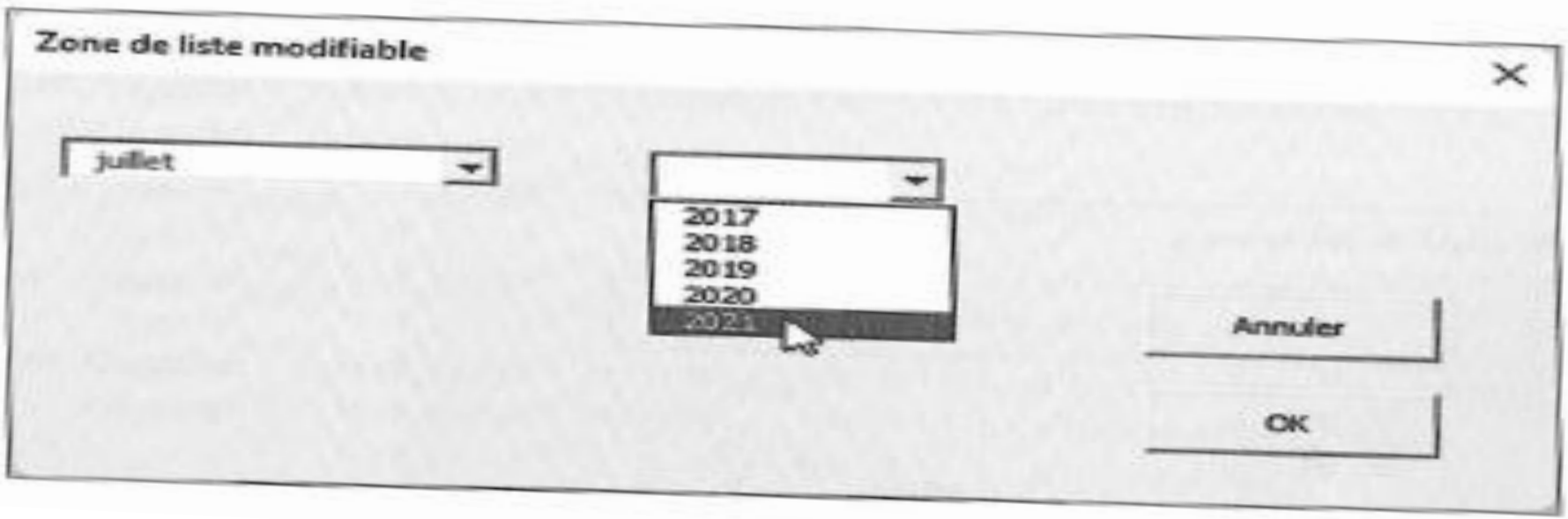
» **ControlSource** : adresse de la cellule d'une feuille de calcul liée au contrôle Case à cocher. Dans Excel, la cellule affiche VRAI si le contrôle est coché ou FAUX s'il ne l'est pas. Cette propriété est facultative, et, la plupart du temps, une case à cocher n'est pas liée à une cellule.

» **Value** : si elle est sur True, la case contient une coche. Si elle est sur False, la case est vide.

Ne confondez pas cases à cocher et boutons d'option. Ces deux contrôles se ressemblent, mais ils répondent à des besoins différents.

Le contrôle Zone de liste modifiable

Un contrôle Zone de liste modifiable est similaire à la commande Zone de liste décrite plus loin. Il s'en distingue par sa liste déroulante qui n'affiche qu'un seul élément à la fois. Autre différence : l'utilisateur peut entrer une valeur qui ne figure pas dans la liste. Sur la Figure , vous voyez deux contrôles de ce type : un pour le mois et l'autre pour l'année. Celui de droite est sélectionné et il affiche sa liste d'options.



Voici une description des principales propriétés de zone de liste modifiable :

» **ControlSource** : cellule contenant la valeur affichée dans le champ de saisie.

» **ListRows** : nombre d'éléments à afficher lorsque la liste est déployée.

» **ListStyle** : apparence de la liste d'éléments.

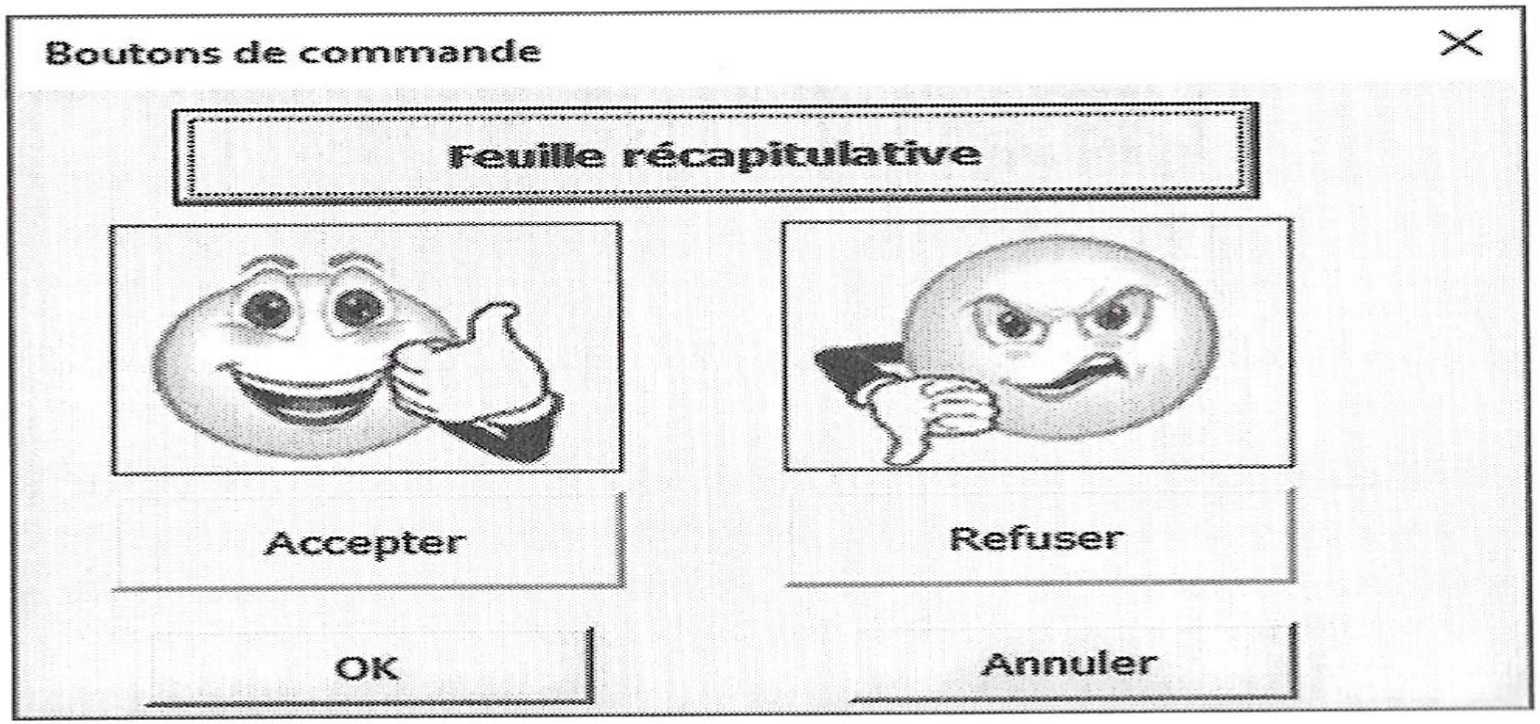
» **RowSource** : plage d'adresses (dans la feuille de calcul) contenant la liste des éléments affichés dans la Zone de liste modifiable.

» **Style** : indique si le contrôle doit agir comme liste déroulante ou comme zone de liste. Une simple liste déroulante n'autorise pas l'utilisateur à entrer manuellement une nouvelle valeur.

» **Value** : texte de l'élément sélectionné par l'utilisateur et affiché dans la zone de liste modifiable.

Le contrôle Bouton de commande

Le contrôle Bouton de commande est un simple bouton cliquable. Il ne sert à rien tant que vous ne lui avez pas attribué une procédure d'événement à exécuter lors du clic. La Figure montre une boîte de dialogue contenant quelques boutons. Vous remarquerez que deux de ces boutons contiennent une image qui a été insérée par copier/coller dans leur champ Picture.



Quand un bouton de commande est cliqué, il exécute une macro dont le nom est celui du bouton de commande, suivi d'un signe de soulignement et du mot *Click*. Par exemple, si le nom du contrôle Bouton de commande est CeBouton, cliquer dessus exécutera la macro nommée CeBouton_Click. Cette macro est stockée dans la fenêtre Code de l'objet UserForm.

Voici la description des principales propriétés du contrôle Bouton de commande :

- **Annuler : si True, appuyer sur Échap exécute la macro associée au bouton. Un seul bouton devrait être associé à cette valeur.**

- **Default : si True, appuyer sur Entrée exécute la macro associée au bouton. À nouveau, un seul bouton devrait être associé à cette valeur.**

Le contrôle Cadre

Le contrôle Cadre entoure d'autres contrôles. Vous vous en servirez, soit pour des raisons esthétiques, soit pour regrouper un ensemble de contrôles. Le contrôle Cadre est particulièrement utile lorsque la boîte de dialogue contient plusieurs jeux de boutons d'option ou de cases à cocher qu'il faut différencier

Les principales propriétés du contrôle Cadre sont :

» **BorderStyle : définit l'apparence du cadre.**

» **Caption : texte affiché en haut à gauche du cadre. La légende peut être une chaîne vide si vous désirez que le contrôle n'affiche aucun texte.**

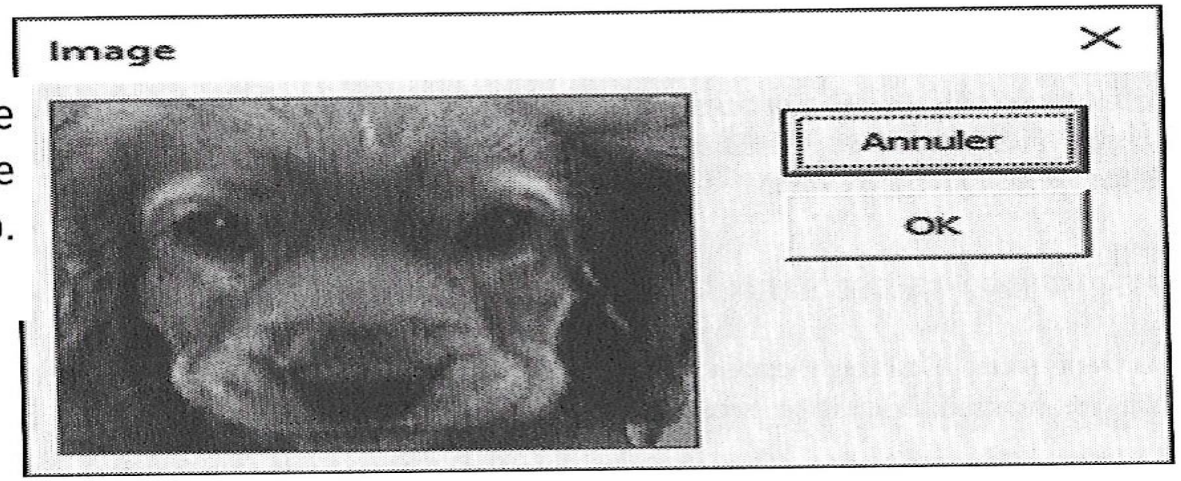
Le contrôle Image

Un contrôle Image affiche une image (le contraire eut été étonnant). Vous l'utiliserez par exemple pour afficher le logo de l'entreprise ou personnaliser une boîte de dialogue avec une photo d'identité.

Les principales propriétés du contrôle Image sont :

» **Image : le fichier d'image à afficher.**

» **PictureSizeMode : le mode d'affichage de l'image si ses dimensions ne sont pas homothétiques avec celles de la fenêtre définie dans la boîte de dialogue.**



Un contrôle Image peut afficher une photo.

Quand vous cliquez dans la propriété Image, l'éditeur VBE demande un nom de fichier. L'image choisie est stockée dans le classeur de sorte que, si vous remettez ce classeur à une tierce personne, il n'est pas nécessaire de joindre le fichier graphique.

Voici un moyen rapide de configurer la propriété Image : copiez l'image dans le Presse-papiers et sélectionnez Image dans la fenêtre Propriétés. Appuyez ensuite sur Ctrl+V pour coller l'image. Vous pouvez aussi vous servir des boutons du groupe Illustrations, sous l'onglet Insertion, pour choisir une image sur votre disque dur ou sur le Web.

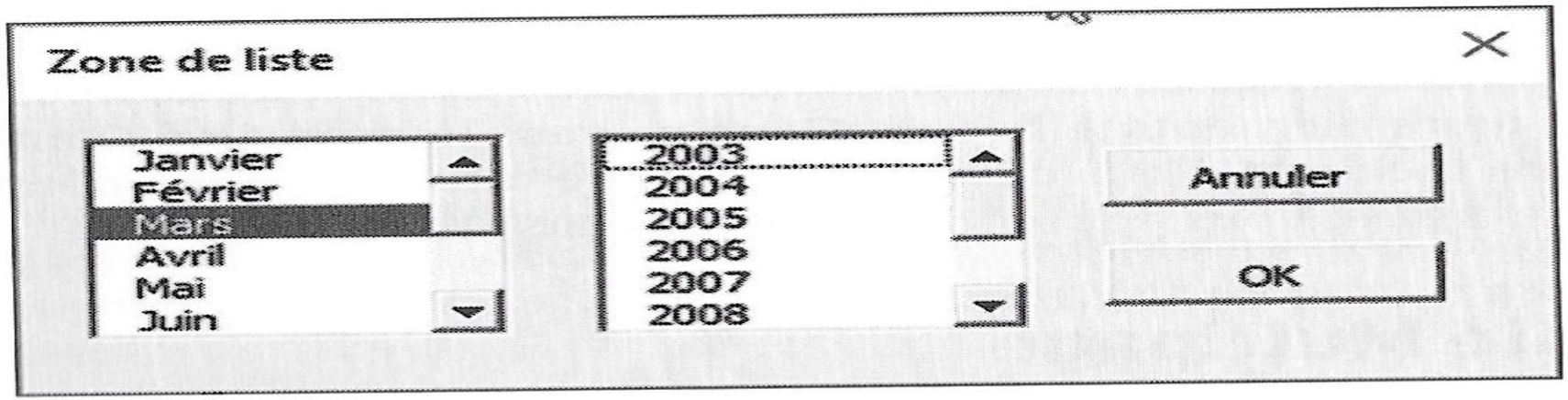
Certaines images peuvent augmenter considérablement la taille du classeur en termes d'octets. C'est pourquoi vous devriez choisir de préférence un fichier d'image dans un format compressé (le format JPEG par exemple).

Le contrôle Intitulé

Le contrôle Intitulé se contente d'afficher du texte dans la boîte de dialogue.

Le contrôle Zone de liste

Un contrôle Zone de liste présente une liste d'éléments dans laquelle l'utilisateur fait son choix. La Figure montre une boîte de dialogue contenant deux contrôles Zone de liste.



Les contrôles Zone de liste sont très souples. Vous pouvez par exemple spécifier la plage qui, dans une feuille de calcul, contient les éléments à prendre en compte. Cette plage peut comporter plusieurs colonnes. La liste peut également être produite par du code VBA.

Si tous les éléments d'une liste ne peuvent pas être affichés simultanément, une barre de défilement apparaît de manière à ce que l'utilisateur puisse faire défiler la liste.

Voici les principales propriétés d'un contrôle Zone de liste :

- **ControlSource** : cellule contenant la valeur affichée dans la zone de liste.
- **IntegralHeight** : True si la hauteur de la liste doit s'ajuster automatiquement pour afficher la totalité d'une longue ligne de texte lors d'un défilement vertical. Si la valeur est False, le texte est tronqué si nécessaire. Notez que la hauteur réelle de la liste peut légèrement différer de ce que vous avez spécifié lors de sa conception. En fait, Excel ajuste cette hauteur de manière à ce que la dernière entrée soit entièrement visible.
- **ListStyle** : apparence des éléments de la liste.
- **MultiSelect** : autorise ou non l'utilisateur à sélectionner plusieurs éléments dans la liste.
- **RowSource** : plage d'adresses de la feuille de calcul contenant les éléments affichés dans la zone de liste.
- **Value** : texte de l'élément sélectionné par l'utilisateur, tel qu'il est affiché dans la zone de liste.

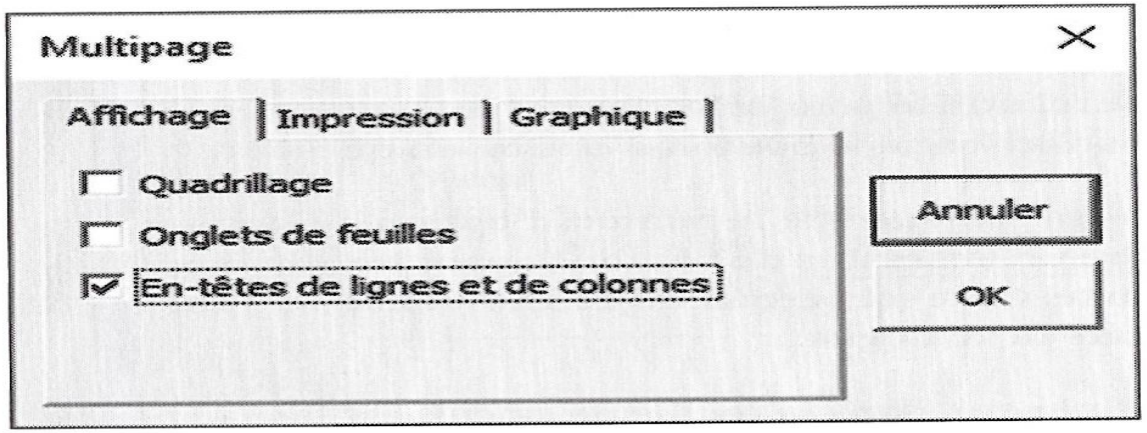
Le contrôle Multipage

Un contrôle Multipage permet de créer des boîtes de dialogue à onglets. La Figure montre une boîte de dialogue personnalisée comportant trois pages, ou onglets.

Les principales propriétés d'un contrôle Multipage sont :

» **Style : définit l'apparence du contrôle. Les onglets peuvent apparaître comme normalement (en haut), ou à gauche, ou sous la forme de boutons ou encore masqués. Dans ce cas, les onglets ne sont pas affichés. La page affichée par défaut est définie par la propriété Value**

» **Value : définit la page ou onglet à afficher par défaut. La valeur 0 affiche la première page, la valeur 1 la deuxième, et ainsi de suite.**

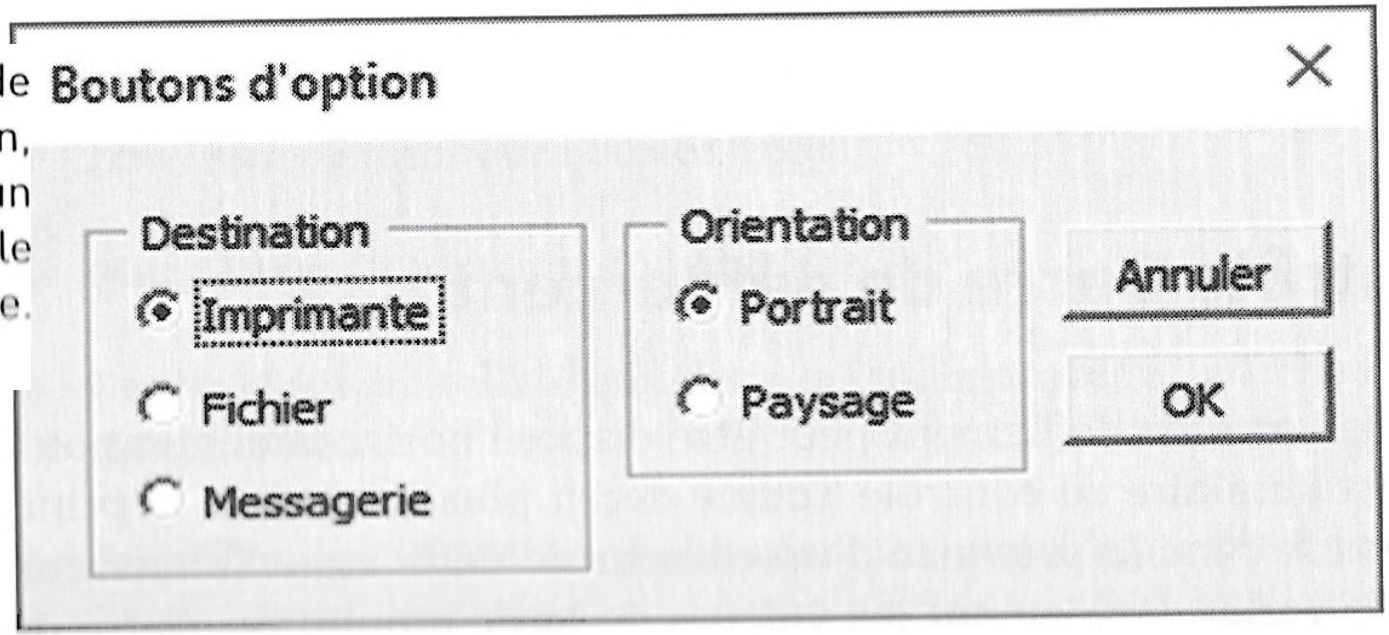


Par défaut, le contrôle Multipage affiche deux pages. Pour en ajouter, cliquez du bouton droit sur un onglet et, dans le menu contextuel, choisissez Nouvelle page. Pour modifier l'ordre des pages, procédez de la même façon, mais choisissez la commande Déplacer.

Le contrôle Bouton d'option

Les boutons d'option sont commodes lorsqu'il s'agit de sélectionner un petit nombre d'éléments. Ils sont toujours utilisés par groupe d'au moins deux. La Figure . montre deux groupes nommés Destination Rapport et Orientation. L'un permet de choisir la sortie du document (une seule), l'autre l'orientation du papier.

Deux groupes de boutons d'option, entourés chacun par un contrôle Cadre.



Les principales propriétés d'un contrôle Bouton d'option sont :

- **Accelerator** : raccourci utilisé pour sélectionner un bouton. Si la lettre est c, appuyer sur Alt+C sélectionne le bouton d'option en question.

- **GroupName** : nom qui indique que le bouton d'option appartient à un même groupe, c'est-à-dire un ensemble de boutons ayant tous le même nom dans la propriété GroupName. Celle-ci permet de définir plusieurs groupes de boutons dans une même boîte de dialogue.

- **ControlSource** : cellule qui, dans la feuille de calcul, est liée au bouton d'option. La cellule affiche VRAI si le bouton est sélectionné et FAUX s'il ne l'est pas.

- **Value** : si True, le bouton d'option est sélectionné. Si False, il ne l'est pas.

Si votre boîte de dialogue doit contenir plusieurs ensembles de boutons d'options, vous devrez obligatoirement définir la propriété GroupName de chacun de ces ensembles. Sinon, tous les boutons appartiendraient au même groupe. Si un ensemble de boutons est placé dans un contrôle Cadre, tous les boutons qui en font partie seront automatiquement regroupés.

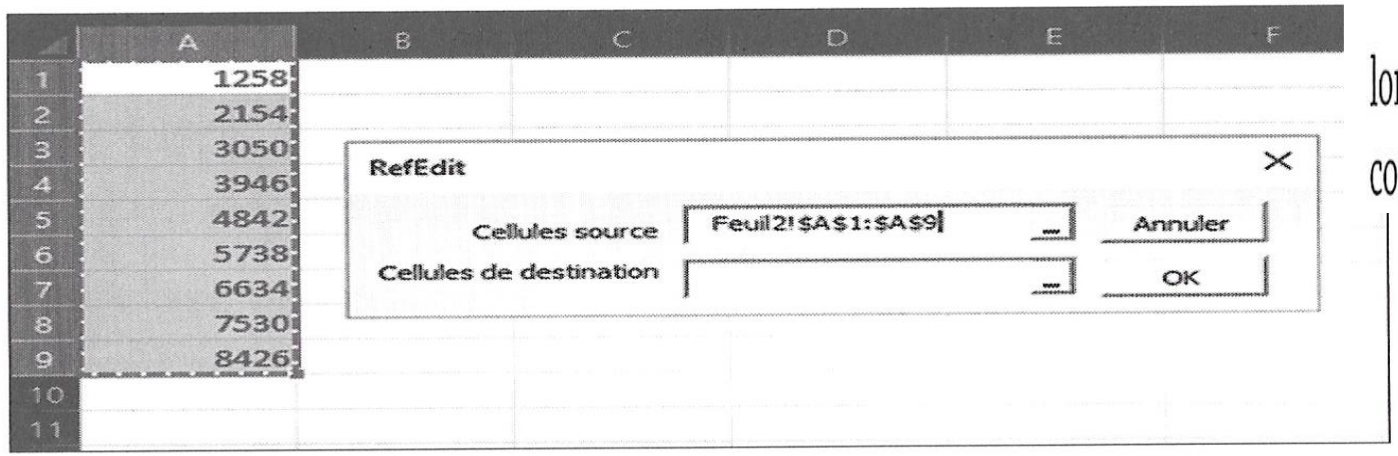
Le contrôle RefEdit

Le contrôle RefEdit est utilisé lorsque vous voulez permettre à l'utilisateur de sélectionner une plage dans une feuille de calcul. La Figure montre une boîte de dialogue dotée de deux contrôles RefEdit. Sa propriété Value contient l'adresse de la plage sélectionnée.

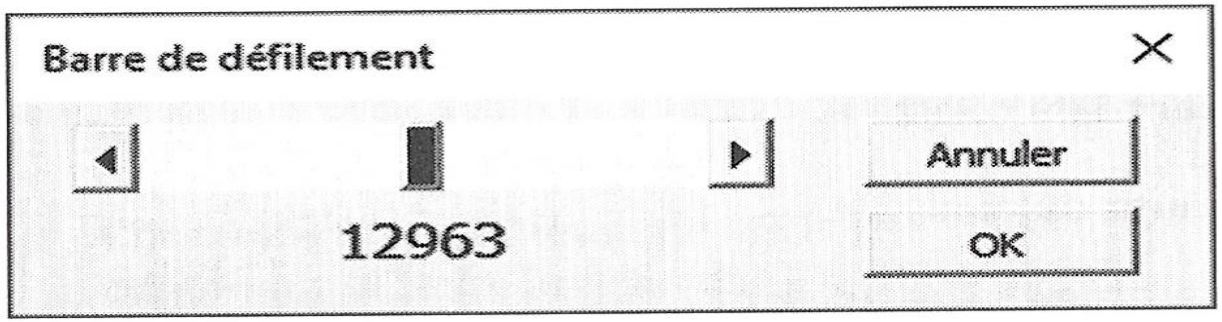
Le contrôle RefEdit peut poser des problèmes dans des UserForms plus complexes. Pour de meilleurs résultats, ne placez pas le contrôle RefEdit dans un contrôle Frame ou MultiPage.

Le contrôle Barre de défilement

Le contrôle Barre de défilement peut être disposé horizontalement ou verticalement. Il est similaire au contrôle Toupie décrit plus loin, mais la principale différence réside dans la présence d'un curseur de défilement (ou ascenseur) permettant de varier rapidement les valeurs de large amplitude. Autre différence :



lorsque vous cliquez sur le bouton du haut dans une barre de défilement, la valeur correspondante décroît.



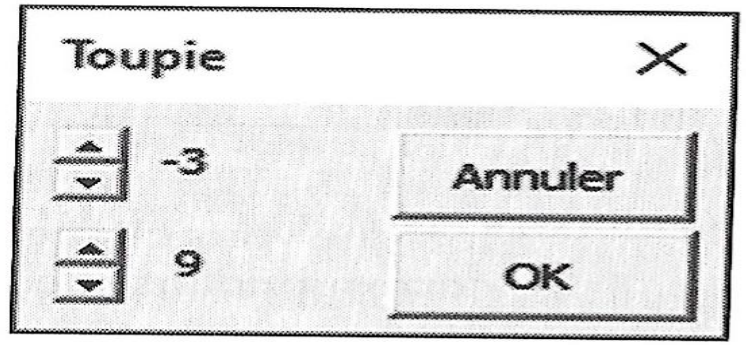
Les principales propriétés d'un contrôle Barre de défilement sont :

- **Value : valeur courante du contrôle.**
- **Min : valeur minimum du contrôle.**
- **Max : valeur maximum du contrôle.**
- **ControlSource : cellule qui, dans la feuille de calcul, affiche la valeur courante du contrôle.**
- **SmallChange : variation de la valeur du contrôle produite par un clic.**
- **LargeChange : variation de la valeur du contrôle produite par un clic sur l'un des boutons d'extrémité.**

Le contrôle Barre de défilement sert surtout à spécifier une valeur dans une plage de valeurs de grande amplitude.

Le contrôle Toupie

Le contrôle Toupie permet à l'utilisateur de sélectionner une valeur en cliquant sur l'un de ses deux boutons fléchés (l'un qui augmente la valeur, l'autre qui la réduit). La Figure montre une boîte de dialogue comportant deux contrôles Toupie. Les chiffres à droite sont affichés par un contrôle Intitulé. Dans un contexte opérationnel, ils sont liés aux toupies par une procédure VBA.



Deux contrôles Toupie.

Les principales propriétés d'un contrôle Toupie sont :

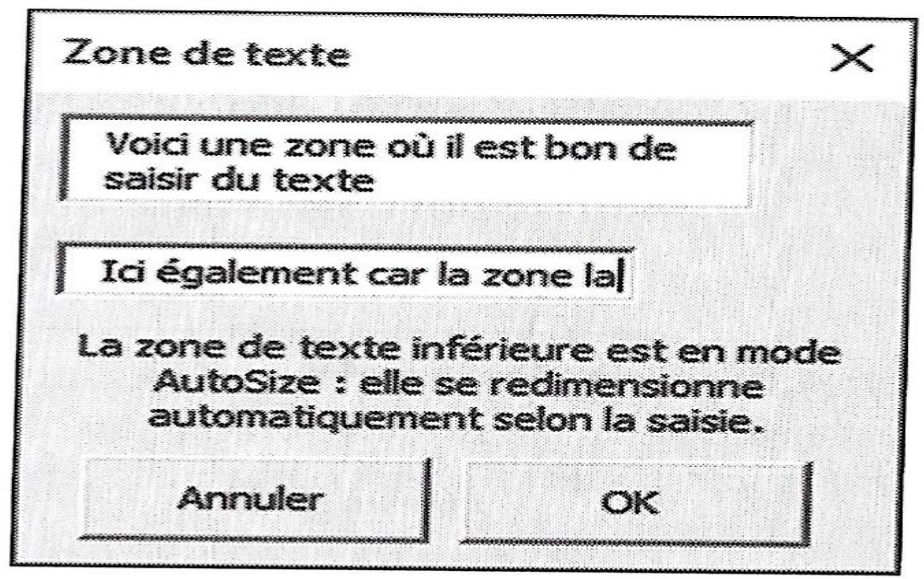
- **Value :** valeur courante du contrôle.
- **Min :** valeur minimum du contrôle.
- **Max :** valeur maximum du contrôle.
- **ControlSource :** cellule qui, dans la feuille de calcul, affiche la valeur courante du contrôle.
- **SmallChange :** variation de la valeur du contrôle produite par un clic. Par défaut, cette valeur est 1, mais vous pouvez en spécifier une autre.

Le contrôle Onglet

Un contrôle Onglet est similaire au contrôle Multipage, sauf qu'il n'est pas si facile à utiliser. Je me demande même pourquoi ce contrôle est proposé. Vous pouvez l'ignorer et utiliser à la place le contrôle Multipage.

Le contrôle Zone de texte

Un contrôle Zone de texte permet à l'utilisateur d'y placer du texte. La Figure montre une boîte de dialogue contenant deux de ces contrôles.



Deux contrôles Zone de texte.

Les principales propriétés d'un contrôle Zone de texte sont :

» **AutoSize : si True, la zone de texte se redimensionne automatiquement selon la quantité de texte saisie (dans ce mode, le renvoi à la ligne n'est plus géré).**

» **ControlSource : adresse de la cellule contenant le texte affiché dans la zone de texte.**

» **IntegralHeight** : si True, la zone de texte s'ajuste automatiquement pour afficher des lignes de texte complètes lors d'un défilement vertical. Si False, des lignes peuvent être tronquées.

» **MaxLength** : nombre maximum de caractères autorisés dans le champ. Si la valeur est zéro, le nombre de caractères est illimité.

» **MultiLine** : si True, la zone de texte peut afficher plusieurs lignes (ne s'applique pas avec la propriété AutoSize).

» **TextAlign** : définit l'alignement du texte dans la zone de texte (à gauche, centré ou à droite).

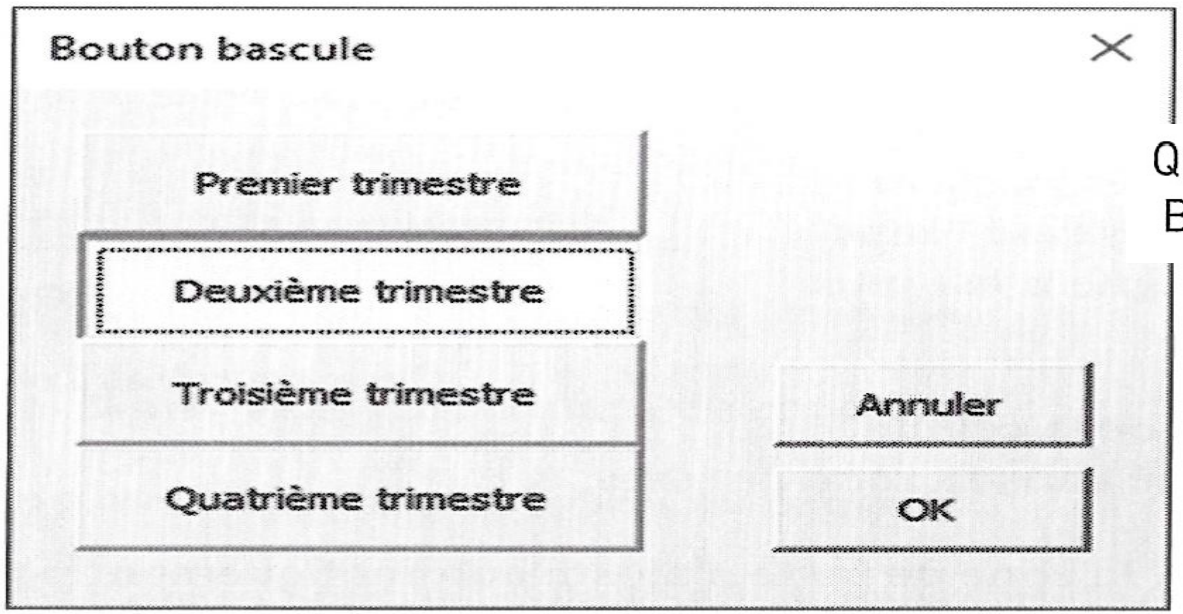
» **WordWrap** : autorise le renvoi à la ligne automatique (sauf en mode AutoSize).

» **Scrollbars** : permet d'afficher une barre de défilement verticale et/ou horizontale.

Quand vous insérez un contrôle Zone de texte, sa propriété Wordwrap est True, et celle de MultiLine est False. Pour que le texte revienne automatiquement à la ligne, vous devez donc mettre vous-même la propriété MultiLine sur True.

Le contrôle Bouton bascule

Un bouton bascule a deux états : actif et inactif. Cliquer dessus fait passer d'un état à l'autre, selon que la valeur est True (enfoncé) ou False (relâché). Ce type de contrôle peut être utilisé à la place des cases à cocher. La Figure montre une boîte de dialogue avec quatre contrôles Bouton bascule. Le second est enfoncé.



Quatre contrôles Bouton bascule.

Travailler avec les boîtes de dialogue

Déplacer et redimensionner des contrôles

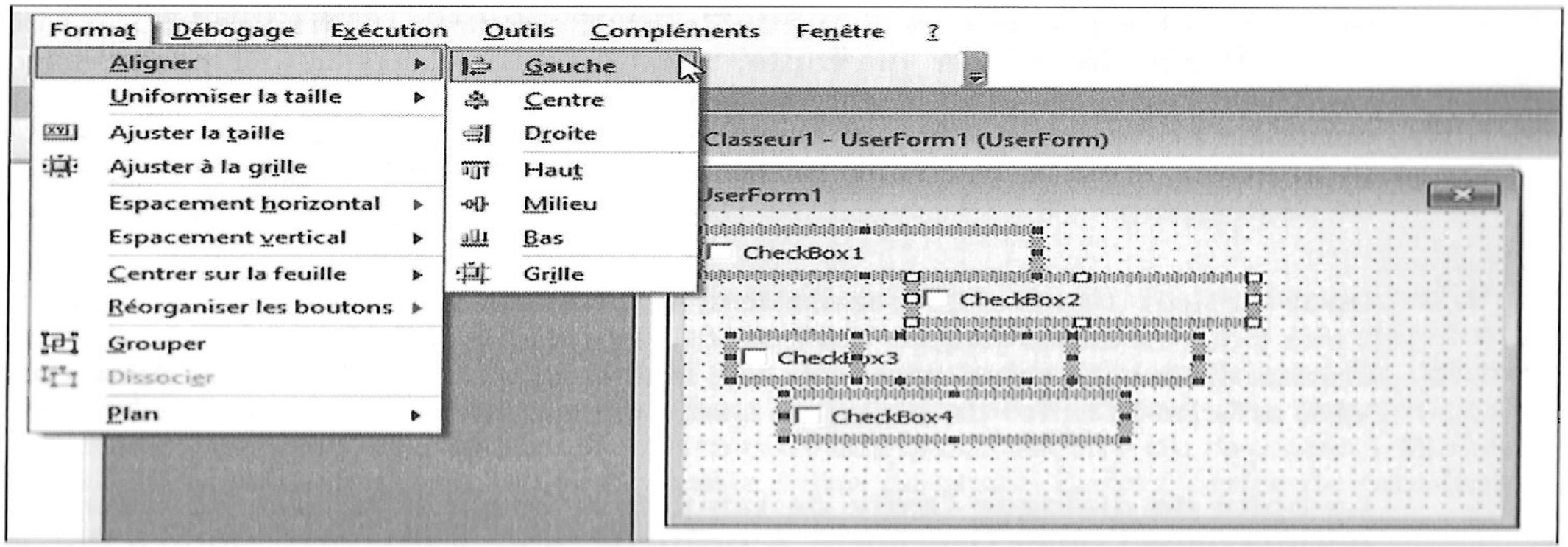
Une fois un contrôle placé dans une boîte de dialogue, il peut être déplacé et redimensionné à la souris à l'aide des techniques standard. Ou alors, pour un contrôle plus précis, vous entrerez des valeurs exactes dans ses propriétés Height, Width, Left ou Top.

Vous pouvez aussi sélectionner plusieurs contrôles à la fois en cliquant dessus avec la touche Ctrl enfoncée, ou encore sélectionner un groupe de contrôles au lasso. Quand plusieurs contrôles sont sélectionnés, la fenêtre Propriétés n'affiche que leurs propriétés communes. Si vous y changez quelque chose, la nouvelle valeur de propriété s'appliquera à tous les contrôles sélectionnés, ce qui est bien plus rapide que de le faire individuellement.

Un contrôle peut en cacher un autre, car ils sont empilables. À moins d'avoir de bonnes raisons de le faire, éviter de les faire se chevaucher.

Aligner et espacer des contrôles

Dans l'éditeur VBE, le menu Format contient plusieurs commandes qui facilitent l'alignement et l'espacement précis des contrôles dans une boîte de dialogue. Vous devez bien sûr sélectionner préalablement les contrôles auxquels vous les appliquerez. Ces commandes n'appelant pas de commentaires particuliers, je ne m'y attarderai pas. La Figure montre des contrôles Case à cocher sur le point d'être alignés.



Bibliographie

- Pour la mise en œuvre https://arkham46.developpez.com/articles/office/userformdialog/
- Programmation VBA pour Excel, J. Walkenbach