Chapitre 5

Dialoguer avec un ordinateur

Les boutons ou les *TextBox*, que vous avez utilisés dans les programmes précédents, sont des contrôles qui permettent une interaction avec l'utilisateur. Dans ce chapitre, vous allez apprendre à maximiser cette interaction, à rendre vos programmes plus vivants, à ajouter du contenu multimédia et à naviguer sur le Web. Les possibilités sont sans fin. Le principe est simple. Lorsque vous lancez Visual Basic, vous arrivez dans le designer de fenêtres. Ces fenêtres sont appelées "formulaires", car elles sont utilisées pour capturer des informations de l'utilisateur et lui donner des réponses en conséquence. Dans les chapitres précédents, vous avez par exemple double-cliqué sur un bouton pour vous placer dans l'édition de code de l'événement correspondant au clic du bouton. La programmation d'applications pour Windows est fondée sur ce simple concept : un événement, une réponse. L'utilisateur du programme clique sur un bouton ? Les instructions que vous avez prévues pour cet événement sont exécutées.

Comme vous allez le voir, chaque contrôle possède ses propres événements.

Les différents contrôles

Définition

Vous avez déjà utilisé la boîte à outils de Visual Basic Express au cours des différents chapitres, pour ajouter un bouton ou une *TextBox* à votre application. Ces éléments sont des contrôles Windows. Ils possèdent chacun des propriétés, qui sont visibles dans la fenêtre des propriétés, et des événements, qui sont visibles d'un clic sur l'icône en forme d'éclair de la fenêtre des propriétés.

Les propriétés permettent de modifier l'état de chaque contrôle. Par exemple, ouvrez un nouveau projet avec Visual Basic Express puis ajoutez un bouton en le faisant glisser sur le formulaire qui se trouve dans le designer depuis la boîte à outils. Cliquez sur le bouton et appuyez sur la touche F4. La fenêtre des propriétés s'affiche et vous pouvez modifier l'état du bouton, de son nom jusqu'à sa couleur de fond. Commençons par exemple par changer le texte de ce bouton.

1. Sélectionnez le bouton d'un simple clic.
2. Appuyez sur la touche F4.

3. Dans la liste qui s'affiche, descendez jusqu'à l'élément *Text*.
4. À droite de *Text*, entrez le texte que vous voulez, par exemple `MonBouton`.
5. Validez avec ↵. Le changement est immédiat.

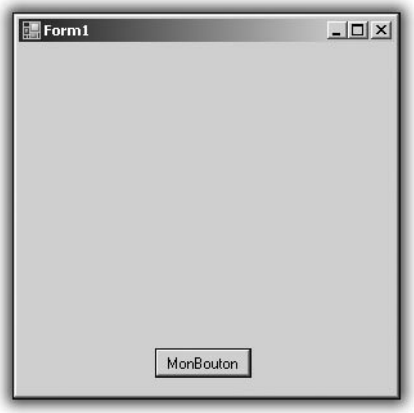

Figure 5.1 : *Changement de texte du bouton*

Vous savez maintenant manipuler les propriétés d'un contrôle. Le principe est analogue quel que soit le type du contrôle puisque tous disposent de la même fenêtre de propriétés. Attention toutefois, certaines propriétés sont disponibles pour plusieurs contrôles différents, mais ne modifient pas la même chose. Par exemple, la propriété *Text* que vous venez de manipuler change le texte d'un bouton, mais si vous modifiez cette même propriété pour un formulaire, c'est son titre qui est changé.

Les événements

Chaque contrôle possède son lot d'événements modifiables de manière à donner au programme le comportement souhaité. Pour modifier un événement d'un contrôle, vous avez deux solutions. Vous avez déjà vu la première, qui consiste à double-cliquer sur le contrôle, ce qui place l'éditeur de texte dans un événement par défaut. C'est le cas du clic sur un bouton ou du changement de texte d'une *TextBox*. En revanche, pour modifier le comportement d'un programme lorsque le pointeur de la souris quitte la zone d'un bouton, il suffit d'aller dans la liste des événements en cliquant sur l'icône en forme d'éclair de la fenêtre des propriétés et de parcourir la liste. Affichons par exemple un message à chaque fois que l'utilisateur entre dans la zone d'un bouton.

1. Lancez Visual Basic Express et créez un nouveau projet.
2. Dans le designer de formulaires, ajoutez un bouton par glisser-déposer.
3. Sélectionnez le bouton en cliquant dessus puis appuyez sur F4.
4. Dans la fenêtre des propriétés, modifiez la propriété *Text* par `Ne cliquez pas ici !`.
5. Cliquez sur l'icône *Événements* de la fenêtre des propriétés (le petit éclair), puis cherchez l'événement *MouseEnter*.
6. Double-cliquez sur la case vide dans la colonne de droite qui correspond à l'événement.
7. Une fois dans l'éditeur de code, ajoutez l'instruction `MessageBox.Show("NE PAS CLIQUER !").`
8. Lancez le programme avec F5, puis passez la souris au-dessus du bouton, sans cliquer. Le résultat est immédiat.

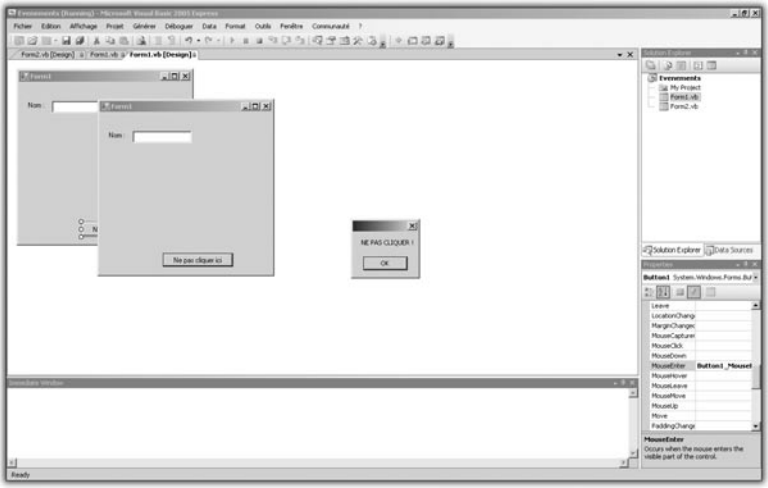

Figure 5.2 : *Résultat de l'événement MouseEnter*

Le contrôle Label

Dans la boîte à outils, faites glisser sur le formulaire le contrôle nommé *Label* ("étiquette" en anglais). Il est couramment utilisé pour spécifier à quoi correspond une zone de texte.

Figure 5.3 :
Label avec une TextBox

Dans ce cas, nous utilisons un *Label* pour guider l'utilisateur : en regardant la fenêtre, il sait tout de suite qu'il faut entrer un nom dans la *TextBox*.

Pour modifier le texte d'un *Label*, il suffit de le sélectionner en cliquant dessus puis, après avoir appuyé sur F4 pour afficher les propriétés, de modifier la propriété *Text*.

Les propriétés principales d'un *Label* sont *Text*, qui représente le contenu du *Label*, et *(Name)*, qui représente le nom du contrôle pour le programme. Vous pouvez accéder aux propriétés d'un contrôle via la fenêtre des propriétés, mais également dans le code, ce qui permet de modifier un contrôle pendant que le programme s'exécute. Voici un exemple :

1. Ouvrez un projet Visual Basic Express et ajoutez un bouton et un *Label* à votre formulaire dans le designer.
2. Dans les propriétés du *Label*, modifiez le texte par `Vous n'avez pas cliqué sur le bouton`.
3. Modifiez la propriété *(Name)* de ce *Label* par `Label_Click`.
4. Double-cliquez sur le bouton pour éditer l'événement "clic".
5. Une fois dans l'éditeur de code, ajoutez l'instruction `Label_Click.Text = "Vous avez cliqué sur le bouton"`.
6. Lancez le programme avec F5 et cliquez sur le bouton du formulaire.

Figure 5.4 :
Avant le clic

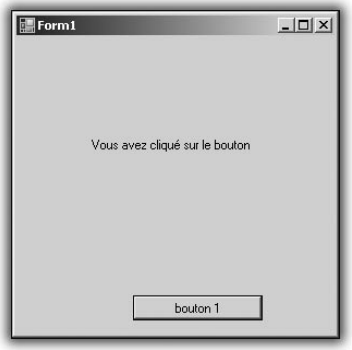

Figure 5.5 :
Après un clic sur le bouton

Le contrôle Button

Ce contrôle est l'un des plus utilisés. Dès lors que l'on veut un retour de l'utilisateur, cela passe souvent par un clic sur un bouton.

L'événement le plus courant et le plus modifié pour ce contrôle est évidemment le clic, mais vous pouvez changer le comportement associé au passage du curseur ou, à l'inverse, lorsque l'utilisateur quitte la zone du bouton sans cliquer dessus.

Les propriétés les plus utiles pour un bouton sont *Text*, qui représente le texte inscrit sur le bouton, et *(Name)*, qui représente le nom du bouton pour le programme. Vous pouvez accéder à ces propriétés (comme à celles d'un *Label*) directement depuis le code. Dans l'exemple suivant, nous allons réaliser un compteur de clics à l'aide d'un *Button* et d'un *Label*.

1. Ouvrez un nouveau projet Visual Basic Express, puis ajoutez un bouton et un *Label* au formulaire.
2. Double-cliquez sur le bouton pour éditer le code de l'événement "clic".
3. Ajoutez les instructions suivantes :

```
Dim i As Integer
i = Int32.Parse(Label1.Text)
i = i + 1
Label1.Text = i.ToString
```

4. Lancez le programme avec F5 puis cliquez sur le bouton. Le texte du *Label* est modifié en fonction du nombre de clics effectués.

Le contrôle ListBox

Ce contrôle permet de proposer une liste à l'utilisateur. Pour l'utiliser, faites-le glisser sur le formulaire. Avec une *ListBox*, vous allez réaliser un répertoire téléphonique.

Une fois le contrôle *ListBox* ajouté au formulaire, sélectionnez-le en cliquant dessus, puis appuyez sur F4 pour visualiser les propriétés. En face de la propriété *Items*, l'inscription *Collection* apparaît. Une collection est un ensemble de valeurs. Vous allez éditer cette collection avec l'outil approprié. Pour cela, cliquez sur l'icône décorée de trois points de suspension, à droite de la mention *Collection*.

Figure 5.6 : *L'éditeur de collection*

Dans la fenêtre qui apparaît, entrez un nom et un numéro de téléphone, séparés par un espace, en effectuant un retour à la ligne entre chaque saisie.

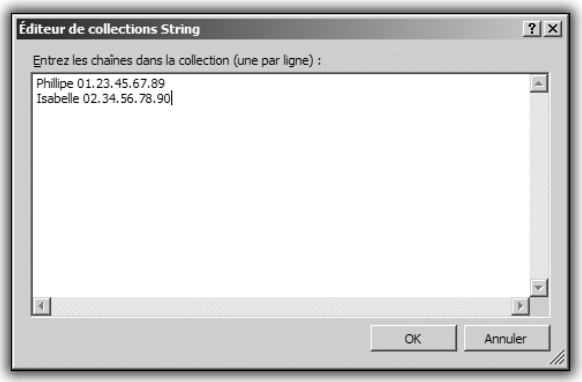

Figure 5.7 : *L'éditeur une fois rempli*

Cliquez ensuite sur le bouton OK de l'éditeur, puis lancez le programme avec F5. La fenêtre s'affiche et la *ListBox* présente les noms et numéros saisis.

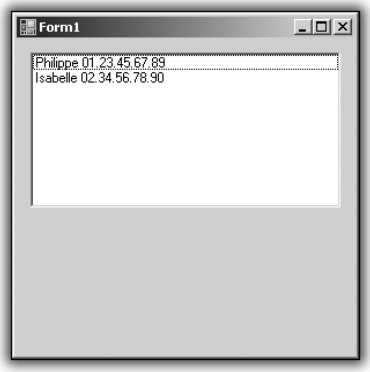

Figure 5.8 :
Le programme une fois lancé

Maintenant, il serait interessant d'ajouter des éléments à la *ListBox*. Pour cela, introduisez un bouton et une *TextBox* dans le formulaire principal.

Double-cliquez sur le bouton, puis ajoutez le code suivant dans l'éditeur :

```
ListBox1.Items.Add(TextBox1.Text)
```

Cette instruction ajoute à la collection d'objets de la *ListBox* le texte qui est saisi dans la *TextBox*.

Imaginons maintenant que vous vouliez effacer l'une des entrées de la *ListBox*. Cette opération est des plus simples.

Fermez l'application puis revenez au designer. Ajoutez à votre formulaire un deuxième bouton, qui servira à effacer une entrée. Dans les propriétés du bouton, modifiez le texte pour qu'il affiche "Suppression".

Double-cliquez ensuite sur ce deuxième bouton pour entrer dans l'éditeur de code à l'endroit du code exécuté lors d'un clic sur ce bouton.

Entrez alors l'instruction suivante :

```
ListBox1.Items.Remove(ListBox1.SelectedItem)
```

Cette ligne spécifie au programme qu'il faut retirer de la collection d'objets de la *ListBox* l'objet qui est sélectionné au moment du clic (`ListBox1.SelectedItem`). Lancez ensuite le programme avec F5 et testez la suppression via le nouveau bouton.

Vous savez maintenant comment ajouter et supprimer des objets dans une *ListBox*.

Renommer les contrôles

Vous pouvez simplifier l'édition d'un programme en renommant les contrôles à l'aide de leur propriété *name*. En renommant par exemple *Label1* en *MaLabel*, vous pourrez modifier sa propriété `Text` dans le code en saisissant `MaLabel.text`.

Les contrôles PictureBox et OpenFileDialog

Ce contrôle est simplement une boîte à image. Vous pouvez par ce biais afficher les images que vous voulez. Il dispose d'une propriété qui permet de sélectionner l'image par défaut. Elle est logiquement nommée *Image*. Vous pouvez la modifier dans la fenêtre des propriétés, qui permet de sélectionner une image sur votre ordinateur et de l'incorporer au fichier ressource. Cliquez sur les trois points de suspension à côté de la propriété pour ouvrir l'éditeur.

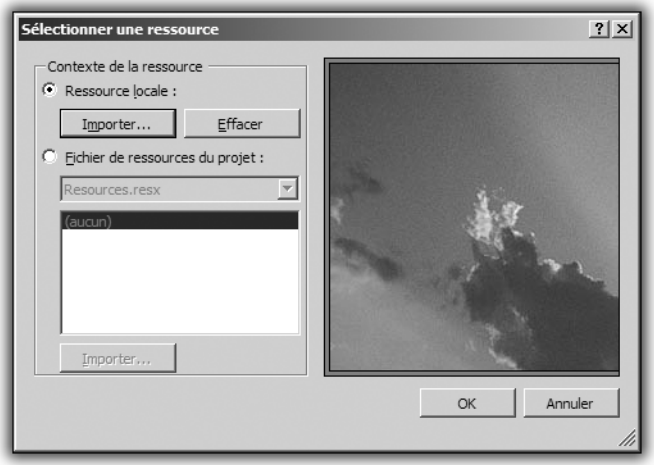

Figure 5.9 : *L'éditeur de la propriété Image*

Dans cet éditeur, sélectionnez *Ressource locale* puis cliquez sur le bouton **Importer**. Parcourez ensuite votre disque dur a la recherche d'une image qui vous plaît, puis double-cliquez dessus. L'image s'affiche dans la *PictureBox*. Mais si cette image est plus grande que la boîte insérée dans le formulaire, elle ne sera pas entièrement affichée. Pour remédier à cela, dans la fenêtre des propriétés, descendez dans la liste jusqu'à trouver la propriété *SizeMode*, puis changez-la pour la valeur *StretchImage*, qui permet d'ajuster automatiquement la taille de l'image à la taille de la *PictureBox*.

Lancez maintenant le programme avec la touche F5. La fenêtre principale se lance et contient l'image que vous avez sélectionnée. Cependant, si vous devez éditer le code du programme à chaque fois que vous voulez changer l'image affichée, c'est contraignant. Ne serait-il pas pratique d'avoir un bouton un peu spécial, qui permette de sélectionner un fichier et de le traiter en conséquence ?

Le contrôle *OpenFileDialog* est justement fait pour cela. Il est un peu particulier en ce sens qu'il permet à un utilisateur de choisir un fichier en parcourant le disque dur et de traiter le fichier en question.

Faites glisser un contrôle *OpenFileDialog* de la boîte à outils vers le formulaire. Vous pouvez constater qu'il ne s'ajoute pas au formulaire, comme les autres contrôles, mais qu'il va se nicher en dessous.

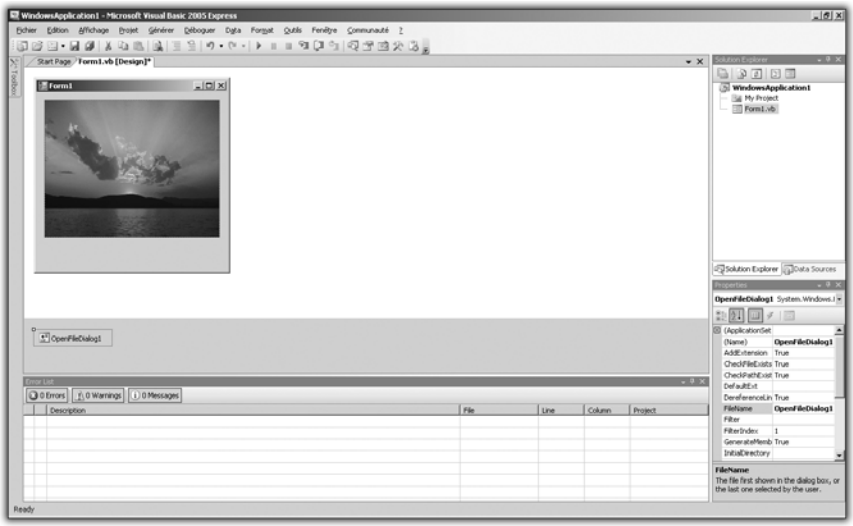

Figure 5.10 : *Le contrôle OpenFileDialog*

En fait, ce contrôle ne se manipule pas directement, mais au travers d'autres contrôles, comme un bouton par exemple. Pour illustrer le fonctionnement, ajoutez un bouton à votre application puis double-cliquez dessus pour afficher le code de l'événement "clic". Ajoutez la ligne d'instruction `OpenFileDialog1.ShowDialog()`. Ensuite, lancez votre programme en appuyant sur la touche F5. Une fois le programme lancé, cliquez sur le bouton. Une boîte de dialogue qui vous propose de choisir un fichier s'affiche.

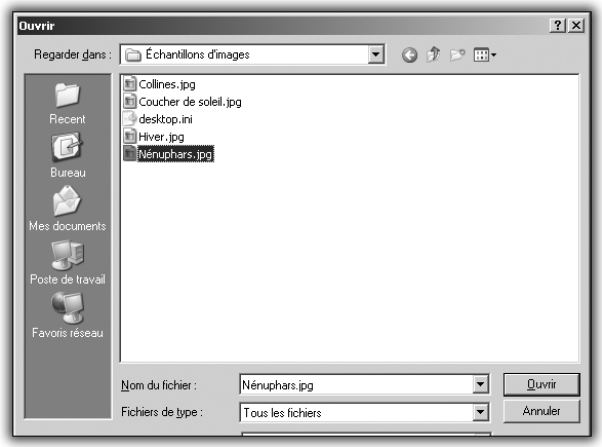

Figure 5.11 : *La boîte de dialogue*

Toutefois, lorsque vous sélectionnez un fichier à l'aide de cette boîte de dialogue, il ne se passe rien. Pour remédier à cela, ajoutez le code suivant à l'événement "clic" :

```
PictureBox1.Image = Image.FromFile(OpenFileDialog1
✂ .FileName)
```

Cette ligne demande simplement à la *PictureBox* de changer l'image en en construisant une à partir d'un fichier sélectionné via la boîte de dialogue générée par le contrôle *OpenFileDialog*.

Lancez le programme pour vérifier que tout fonctionne et admirez le fruit de votre programmation.

Vous disposez maintenant d'une visionneuse d'image, par exemple pour regarder les clichés issus d'un appareil photo numérique branché à votre ordinateur. Mais ne serait-il pas intéressant de choisir les fichiers dans le répertoire *Mes Images* au lieu de chercher dans le répertoire à partir duquel vous lancez l'application ? Ce serait un plus indéniable.

Pour cela, il existe la propriété *InitialDirectory* du contrôle *OpenFileDialog*. Elle ne peut pas être directement modifiée dans la fenêtre des propriétés. Vous devez intervenir sur le code. Pour cela, double-cliquez sur le formulaire pour éditer le code lancé lors du chargement de la fenêtre. Une fois dans l'éditeur de code, saisissez cette instruction :

```
OpenFileDialog1.InitialDirectory = Environment
✂ .SpecialFolder.MyPictures
```

Elle permet d'éditer le répertoire par défaut dans lequel chercher les images et de spécifier que ce répertoire sera *Mes Images*, créé par Windows pour chaque utilisateur de l'ordinateur. Il existe également le même genre de raccourci pour le répertoire *Mes Documents*, *Ma Musique*, le Bureau, etc.

Figure 5.12 :
Le programme avec l'image une fois changée

Vous êtes maintenant capable de sélectionner un fichier à l'aide d'une boîte de dialogue.

Le contrôle WebBrowser

Le contrôle *WebBrowser* permet de développer en un temps record des applications s'appuyant sur la puissance du Web. Grâce à lui, vous pouvez disposer d'un navigateur Internet à l'intérieur de votre application.

Vous allez vous en servir pour développer rapidement un navigateur Internet personnalisé.

Ouvrez un nouveau projet avec Visual Basic Express. Une fois dans le designer, ajoutez une *TextBox* dont vous changerez la propriété *(name)* en `address`, puis un contrôle *WebBrowser* que vous placerez sur le formulaire.

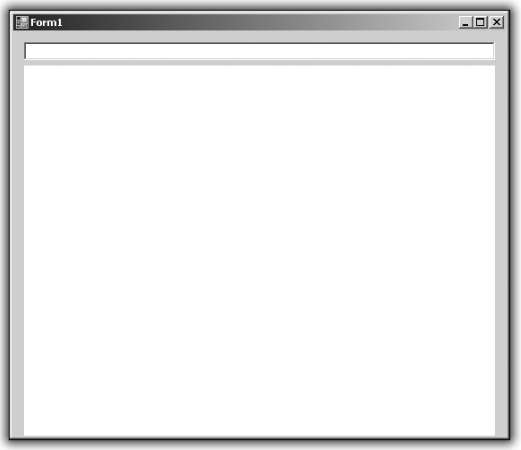

Figure 5.13 : *L'ébauche du navigateur*

Redimensionner un formulaire

N'hésitez pas à redimensionner le formulaire à l'aide de la souris en plaçant le curseur sur l'un des coins et en bougeant la souris tout en maintenant le bouton gauche enfoncé.

Ajoutez un bouton à droite de la *TextBox* puis changez la propriété *Text* du bouton par `Go`. Modifiez le code de l'événement "clic" en double-cliquant sur ce dernier et ajoutez l'instruction suivante :

```
WebBrowser1.Navigate(address.Text)
```

Lancez votre programme en appuyant sur F5. Saisissez une adresse dans la *TextBox* puis cliquez sur le bouton **GO**. Vous avez maintenant votre propre navigateur.

Figure 5.14 : *Votre navigateur terminé*

Vous avez, en 5 minutes, réalisé une application fonctionnelle, d'une qualité quasiment égale à celle d'une application professionnelle. Est-il encore possible de dire que la programmation est réservée à une élite d'informaticiens ?

Les contrôles FontDialog et ColorDialog

Vous allez apprendre à réaliser un éditeur de texte simple permettant de saisir du texte, de changer sa police et sa couleur.

Commencez par créer un nouveau projet avec Visual Basic Express. Puis, dans l'éditeur de formulaire, ajoutez deux boutons, un contrôle *TextBox* et les contrôles *FontDialog* et *ColorDialog*. Comme le contrôle *OpenFileDialog* que vous avez manipulé précédemment, ces deux derniers contrôles ne viennent pas s'ajouter au formulaire, mais juste en dessous. Il faudra donc utiliser les deux boutons pour faire le lien entre les contrôles et leur utilisation. Renommez le texte d'un des boutons en `Police` et le texte du second bouton en `Couleur`. Double-cliquez sur le bouton **Police** et ajoutez les instructions suivantes au code de l'événement "clic" :

```
FontDialog1.ShowDialog()
TextBox1.Font = FontDialog1.Font
```

Revenez à l'éditeur de formulaire. Double-cliquez sur le bouton **Couleur** pour éditer l'événement "clic". Ajoutez les instructions suivantes :

```
ColorDialog1.ShowDialog()
TextBox1.ForeColor = ColorDialog1.Color
```

Lancez le programme avec la touche F5, saisissez du texte dans la *TextBox*, puis utilisez les deux boutons pour modifier la couleur et la police du texte.

Les instructions se limitent ici à appliquer une police au texte sélectionnée via la boîte de dialogue **Font**. Quand vous cliquez sur OK, la police choisie remplace celle de la *TextBox*. Quand vous cliquez sur l'autre bouton, il se passe exactement la même chose, mais la couleur sélectionnée dans la boîte de dialogue **Couleurs** s'applique au premier plan de la *TextBox*, à savoir au texte.

Soyez professionnel !

Pour donner un look plus professionnel à cet éditeur de texte, sélectionnez la *TextBox* d'un simple clic, puis cliquez sur son petit triangle en haut à droite. Cochez la case *Multiline*. Il est maintenant possible de modifier la hauteur en nombre de lignes de cette *TextBox*.

Vous pouvez appliquer un changement de couleur à tous les éléments de votre formulaire. Par exemple, vous pouvez accorder la couleur de fond de l'application avec la couleur de texte que vous venez de sélectionner en ajoutant la ligne suivante à l'événement approprié :

```
Me.BackColor = ColorDialog1.Color
```

`Me` fait référence à la fenêtre qui est en cours d'exécution. Vous appliquez ici la couleur à l'arrière-plan de la fenêtre. Ainsi, le texte que vous allez saisir sera de la même couleur. La méthode de changement de couleur fonctionne avec n'importe quel contrôle disposant d'une propriété *Color*. Pour vérifier que le contrôle en possède une, il suffit de regarder dans la liste de la fenêtre des propriétés. La deuxième méthode consiste à saisir le nom du contrôle dans l'éditeur de code, suivi d'un point, puis d'appuyer sur les touches Ctrl+Barre d'espace pour lister toutes les propriétés et méthodes des contrôles.

Le contrôle TreeView

Si vous vous êtes déjà servi de l'Explorateur de fichiers Windows, vous avez dû remarquer que les répertoires se présentent sous la forme d'une arborescence. Il est possible d'ajouter ce style d'arborescence dans vos programmes à l'aide du contrôle *TreeView*. Il permet d'organiser des données sous forme de nœuds hiérarchiques. Le nœud de premier niveau s'appelle "nœud racine" et les nœuds de niveaux inférieurs sont les fils du nœud racine, chaque nœud fils pouvant lui-même servir de nœud racine. Un exemple sera certainement plus parlant.

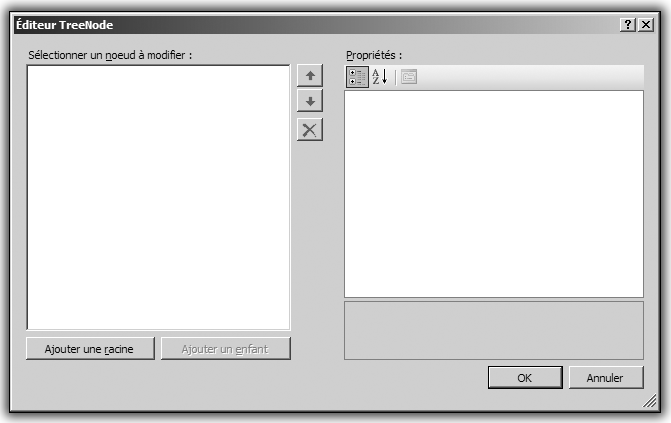

Figure 5.15 : *L'éditeur de nœud*

Créez un nouveau projet, puis ajoutez au formulaire un contrôle *TreeView*. Une fois le *TreeView* ajouté, sélectionnez-le en cliquant dessus, puis affichez la fenêtre des propriétés en appuyant sur F4. L'une des propriétés s'appelle *Nodes*. Comme la collection *Items* du contrôle *ListBox*, il s'agit d'une collection de valeurs, qui vont représenter cette

fois des nœuds de l'arborescence. Comme pour la *ListBox*, vous pouvez éditer la collection de nœuds graphiquement, en cliquant sur les trois points de suspension à droite de *Nodes*, dans la fenêtre des propriétés.

Cliquez sur le bouton **Ajouter une racine** pour ajouter un premier nœud au *TreeView*. Une fois ce nœud ajouté, vous pouvez lui adjoindre soit des enfants, soit une autre racine. Mais encore une fois, vous pouvez modifier un *TreeView* avec le code de l'application.

L'objectif ici est de créer une bibliothèque de films.

En ce sens, ajoutez à votre application trois *TextBox* et deux boutons.

Figure 5.16 : *Squelette de l'application*

Nommez le premier bouton `Ajout Catégorie` et le second `Ajout Film`. Il faut maintenant modifier les événements "clic" pour que les boutons ajoutent chacun une catégorie à un niveau de hiérarchie différent.

Double-cliquez sur le bouton **Ajout Catégorie** de manière à vous placer dans l'éditeur de code, puis recopiez l'instruction suivante :

```
TreeView1.Nodes.Add(TextBox1.Text)
```

Cette instruction ajoute un nœud au niveau hiérarchique le plus haut du *TreeView*. Double-cliquez sur le bouton **Ajout Film** et, une fois dans l'éditeur de code, saisissez l'instruction suivante :

```
TreeView1.SelectedNode.Nodes.Add(textbox2.Text)
```

Cette ligne ajoute un nœud en considérant le nœud sélectionné comme racine d'un arbre.

Lancez maintenant l'application en appuyant sur F5. Entrez une catégorie dans la première *TextBox* puis cliquez sur le bouton **Ajout Catégorie**. Sélectionnez ensuite le nœud que vous venez d'ajouter, puis saisissez un titre de film dans la deuxième *TextBox*. Cliquez ensuite sur le deuxième bouton et le nom du film viendra s'ajouter à la hiérarchie.

Figure 5.17 : *Votre bibliothèque de films*

Le contrôle ComboBox

Ce contrôle permet d'éviter bon nombre de saisies répétées en proposant une liste déroulante de choix possibles. Vous allez tester son utilité en remplaçant le bouton **Ajout Catégorie** de la bibliothèque de films par une *ComboBox*. Créez un nouveau projet et ajoutez au formulaire les contrôles suivants : un *TreeView*, deux *Label*, une *TextBox*, une *ComboBox* et un bouton.

Nommez le premier *Label* `Catégories` et placez-le au-dessus de la *ComboBox*. Nommez le deuxième *Label* `Film` et placez-le au-dessus de la *TextBox*. Placez le bouton juste en dessous.

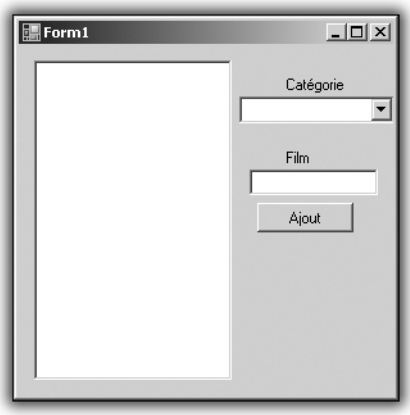

Figure 5.18 : *Bibliothèque améliorée*

Sélectionnez maintenant la *ComboBox* et affichez ses propriétés en appuyant sur F4. Dans la liste, vous retrouvez une propriété *Items* éditable, comme pour la *ListBox*. Le fonctionnement de ces deux contrôles est par bien des points similaire. Pour éditer le contenu de la *ComboBox*, cliquez sur les trois points de suspension à droite de la propriété *Items*.

Une fenêtre s'affiche alors.

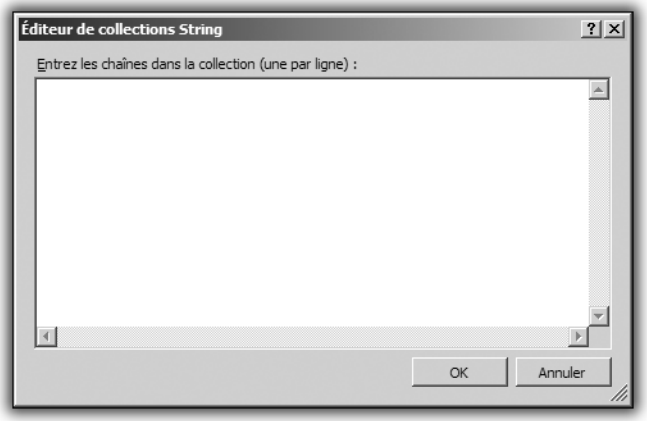

Figure 5.19 : *Ajoutez des éléments*

Ajoutez des catégories de films, une par ligne, et appuyez sur OK. Votre liste de genres est gardée dans la collection *Items* de la *ComboBox*. Vous allez maintenant modifier le code de l'événement "clic" du bouton pour ajouter un film à la liste.

En ce sens, placez-vous dans l'éditeur de code et saisissez le code suivant :

```
Dim n As TreeNode
n = New TreeNode
n.Text = ComboBox1.SelectedItem
n.Nodes.Add(New TreeNode(TextBox1.Text))
TreeView1.Nodes.Add(n)
```

Cette portion de code déclare une variable de type `TreeNode` (nœud d'un arbre) et fait en sorte que le texte affiché sur le *TreeView* soit celui de la catégorie choisie dans la *ComboBox*. Ensuite est ajouté, en tant que nœud enfant du nœud `n`, un nœud qui porte le nom du film que l'on va saisir dans la *TextBox*.

Figure 5.20 :
Le résultat de votre travail

5.2. Les formulaires

Les formulaires permettent à l'utilisateur de saisir des informations et également de lui en renvoyer en utilisant les différents contrôles mis à disposition par Visual Basic Express. Jusqu'à présent, vous avez développé des applications qui utilisaient un seul formulaire. Vous allez maintenant utiliser les formulaires pour organiser au mieux une application.

Lorsque vous créez un projet sous Visual Basic Express, un formulaire nommé *Form1.vb* est ajouté au projet. Si vous développez des applications complexes, vous aurez souvent recours à d'autres fenêtres à l'intérieur de votre programme. Pour cela, vous devrez ajouter des

formulaires au projet. Pour ajouter un formulaire, cliquez du bouton droit dans l'Explorateur de solutions situé au-dessus de la fenêtre des propriétés, puis cliquez sur **Ajouter/Nouvel élément**. Dans la liste de choix qui est alors proposée, choisissez *Formulaire Windows* (Windows Form). Un nouvel onglet, nommé *Form2.vb*, est alors ajouté au projet. Les manipulations possibles sur ce nouveau formulaire sont exactement les mêmes que sur le formulaire qui est ajouté par défaut à la création d'un nouveau projet.

Pour illustrer ce principe, reprenez la bibliothèque de films développée précédemment et améliorez la saisie de films en utilisant un deuxième formulaire.

Créez un nouveau projet puis, comme pour le projet précédent, ajoutez un contrôle de type *TreeView*, une *TextBox* et deux boutons. Ajoutez une *TextBox* avec un bouton **Ajout Catégorie**. Ajoutez un deuxième bouton que vous nommerez `Ajout Film`.

Ajoutez maintenant un nouveau formulaire au projet grâce à la méthode décrite précédemment. Dans le designer, sélectionnez l'onglet du nouveau formulaire et ajoutez-y deux boutons et une *TextBox*. À gauche de la *TextBox*, ajoutez un *Label* `Film`.

Sélectionnez le premier bouton, puis affichez les propriétés en appuyant sur F4. Dans la propriété *Text*, inscrivez `OK`. Changez le texte du deuxième bouton par `Annuler`. Il faut maintenant faire en sorte que, lorsque l'utilisateur clique sur le bouton OK, les informations du film s'ajoutent à l'arborescence.

Modifiez le code de l'événement "clic" du bouton **Ajout Catégorie** sur le formulaire 1. Recopiez le code suivant :

```
TreeView1.Nodes.Add(TextBox1.Text)
```

Cela permet de créer un nœud de catégorie. Modifiez ensuite le code de l'événement "clic" du bouton **Ajout Film**. Insérez les instructions suivantes :

```
Dim Form_ajout As Form2
Form_ajout = New Form2
Form_ajout.ShowDialog()
```

Il s'agit ici de déclarer une variable qui soit de type `Form2`, `Form2` étant le formulaire que vous avez ajouté à votre projet. De cette manière, un

clic sur le bouton **Ajout Film** ouvre une nouvelle fenêtre avec le contenu du formulaire 2.

Il faut maintenant modifier les événements "clic" du deuxième formulaire pour ajouter un film dans la catégorie sélectionnée.

Pour le bouton OK, saisissez le code suivant :

```
Form1.TreeView1.SelectedNode.Nodes.Add(TextBox1.Text)
Me.Dispose()
```

Pour le bouton **Annuler**, utilisez l'instruction suivante :

```
Me.Dispose()
```

De cette manière, un clic sur le bouton OK ajoute le film à la catégorie sélectionnée, et un clic sur le bouton **Annuler** ferme le formulaire. La méthode `Dispose` ferme le formulaire et efface toute trace de ce dernier dans la mémoire de l'ordinateur. Après l'exécution de la méthode `Dispose`, il est impossible d'accéder au contenu du formulaire.