l'ouverture simultanée de plusieurs documents. Cette possibilité est intéressante à plus d'un titre, en particulier :

- Elle facilite les copier-coller entre documents.
- Elle permet de voir plusieurs parties d'un même document.
- Elle permet de construire un document en ayant un ou plusieurs autres documents sous les yeux.

En utilisant Visual Basic 2010, il est très simple de définir des applications comprenant plusieurs documents. Ces applications sont dites MDI (*Multiple Document Interface*). Elles partagent en général un système de menus, une barre d'outils et/ou une barre d'état. Les fenêtres d'un document sont également appelées fenêtres enfants (*child*). La fenêtre de base, qui contient le système de menus, la barre d'outils et la barre d'état, est la fenêtre parent.

Pour concevoir une application MDI, vous devez au minimum définir une feuille parent et une feuille enfant. Chaque document ouvert ou créé dans l'application utilisera la même feuille parent en arrière-plan et une feuille enfant personnelle à l'intérieur de la feuille parent.

La fenêtre parent est une feuille Visual Basic traditionnelle pour laquelle la propriété `IsMDIContainer` a été initialisée à `True`. La fenêtre enfant est matérialisée par une feuille Windows Form traditionnelle.

Création d'un formulaire parent

Le formulaire MDI parent est le fondement d'une application MDI. Il donne accès et contient toutes les fenêtres MDI enfants.

Pour définir un formulaire MDI parent, commencez par créer une nouvelle application avec la commande Nouveau/Projet dans le menu Fichier. Choisissez .NET Framework 4 dans la liste déroulante Framework. L'entrée Visual Basic/Windows étant sélectionnée sous Modèles installés, choisissez le modèle Application Windows Forms, nommez le projet Bloc Notes et validez en cliquant sur OK.

Dans la fenêtre des propriétés, attribuez la valeur `True` à la propriété `IsMDIContainer`. Cette simple action désigne le formulaire en cours d'édition comme un conteneur MDI de fenêtres enfants.

Ajoutez un contrôle `MenuStrip` au formulaire et définissez le menu suivant :

Élément de menu principal	*Élément de menu secondaire*
Fichier	Nouveau
Fichier	Quitter
Fenêtre	

La commande de menu Fichier/Nouveau sera utilisée pour créer des fenêtres enfants, et le menu Fenêtre, pour basculer entre les diverses fenêtres enfants ouvertes.

Cliquez sur le contrôle `MenuStrip` dans la feuille de l'application et affectez la valeur `FenêtreToolStripMenuItem` à sa propriété `MdiWindowListItem`. Cette simple action affecte au menu Fenêtre la liste des fenêtres MDI enfants ouvertes et indique par une coche la fenêtre enfant active.

Création d'un modèle de fenêtre enfant

Lancez la commande Ajouter un formulaire Windows dans le menu Projet pour ajouter un nouveau formulaire au projet. Choisissez Windows Form dans la boîte de dialogue Ajouter un nouvel élément, donnez le nom Modèle Enfant au nouveau formulaire et cliquez sur Ajouter.

Faites glisser un contrôle `RichTextBoxControl` de la Boîte à outils jusqu'au nouveau formulaire. Dans la fenêtre des propriétés, déroulez la liste `Dock` et cliquez sur le rectangle central (voir Figure 5.1). Cette action affecte la valeur `Fill` à la propriété `Dock` et la valeur `Top, Left` à la propriété `Anchor`. Le contrôle `RichTextBox` remplira totalement la zone du formulaire MDI enfant, même si ce dernier est redimensionné.

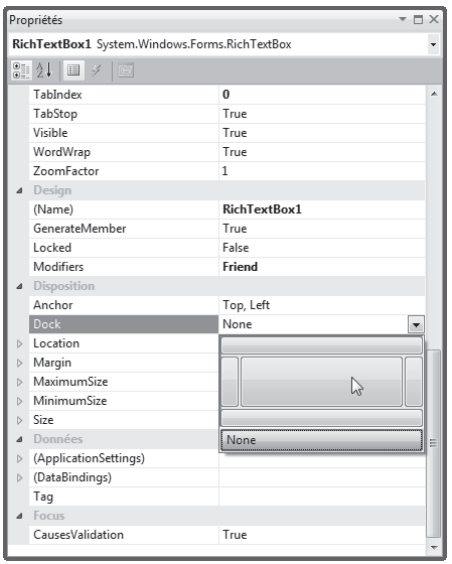

Figure 5.1
Cliquez ici pour maximiser le contrôle `RichTextBox`.

Déroulez le menu Fichier dans la feuille de l'application et double-cliquez sur la commande Nouveau. Cette action affiche le squelette de la procédure `NouveauToolStripMenuItem_Click()`. Complétez cette procédure comme suit :

```
Private Sub NouveauToolStripMenuItem_Click(ByVal sender As System.Object,
```

```
➥ByVal e As System.EventArgs) Handles NouveauToolStripMenuItem.Click
    Dim NewMDIChild As New Modèle_Enfant
    NewMDIChild.MdiParent = Me
    NewMDIChild.Show()
End Sub
```

La première ligne définit l'objet fenêtre fille MDI `NewMDIChild` :

```
Dim NewMDIChild As New Form2()
```

La deuxième instruction associe la fenêtre fille avec le conteneur mère :

```
NewMDIChild.MDIParent = Me
```

Enfin, la troisième instruction affiche la fenêtre fille à l'intérieur de la fenêtre mère :

```
NewMDIChild.Show()
```

Pour compléter le programme, déroulez le menu Fichier dans la feuille de l'application et double-cliquez sur la commande Quitter. Complétez la procédure `QuitterToolStripMenuItem_Click()` comme suit :

```
Private Sub QuitterToolStripMenuItem_Click(ByVal sender As System.Object,
➥ByVal e As System.EventArgs) Handles QuitterToolStripMenuItem.Click
    End
End Sub
```

Vous pouvez maintenant compiler et lancer l'application en appuyant sur la touche F5. La commande Fichier/Nouveau affiche une fenêtre fille dans la fenêtre principale de l'application. Si vous ouvrez plusieurs fenêtres avec cette commande, vous pouvez utiliser le menu Fenêtres pour basculer de l'une à l'autre (voir Figure 5.2).

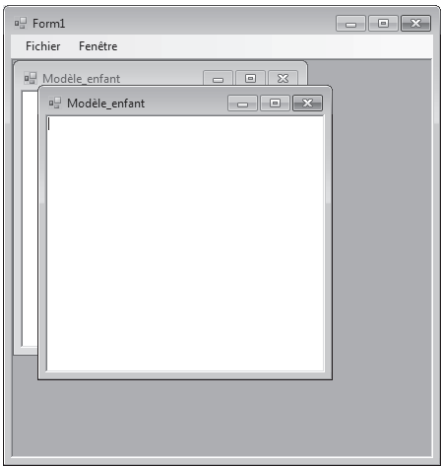

Figure 5.2
Deux fenêtres filles ont été ouvertes dans cette application MDI.

Le code de l'application se réduit aux deux procédures événementielles `NouveauToolStripMenuItem_Click ()` et `QuitterToolStripMenuItem_Click ()`. L'application se trouve dans le dossier Bloc Notes après installation des sources de l'ouvrage :

```
Public Class Form1

    Private Sub NouveauToolStripMenuItem_Click(ByVal sender As System.Object,
    ➥ByVal e As System.EventArgs) Handles NouveauToolStripMenuItem.Click
        Dim NewMDIChild As New Modèle_Enfant
        NewMDIChild.MdiParent = Me
        NewMDIChild.Show()
    End Sub

    Private Sub QuitterToolStripMenuItem_Click(ByVal sender As System.Object,
    ➥ByVal e As System.EventArgs) Handles QuitterToolStripMenuItem.Click
        End
    End Sub

End Class
```

Polices de caractères

Les polices installées relèvent de la classe `FontFamily` et l'affichage de texte utilise le système d'affichage GDI+.

Pour mieux comprendre comment utiliser ces deux éléments, nous allons définir un petit projet dans lequel les polices installées seront accessibles à travers une liste déroulante. Lorsque l'utilisateur sélectionnera une entrée dans cette liste, un texte de corps 20 s'affichera dans la police choisie (voir Figure 5.3).

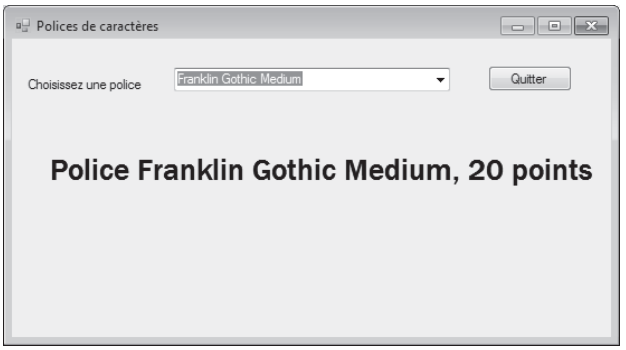

Figure 5.3
Le programme Polices de caractères en mode Exécution.

Définissez un nouveau projet avec la commande Nouveau/Projet dans le menu Fichier. Choisissez .NET Framework 4 dans la liste déroulante Framework. Sélectionnez Visual Basic/Windows. Choisissez le modèle Application Windows Forms et validez. Ajoutez un `Label`, un `ComboBox` et un contrôle `Button` à la feuille du projet. Modifiez les propriétés de ces contrôles comme suit :

Contrôle	*Propriété*	*Valeur*
`Form1`	`Text`	Polices de caractères
`Label1`	`Text`	Choisissez une police
`Button1`	`Text`	Quitter

Double-cliquez sur un emplacement inoccupé de la feuille et complétez la procédure `Form1_Load()` comme suit :

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles MyBase.Load
    Dim strFonts As New Collections.ArrayList(10)
    Dim Font As FontFamily

    For Each Font In FontFamily.Families
        strFonts.Add(Font.Name)
    Next
    ComboBox1.DataSource = strFonts
End Sub
```

La première instruction définit l'objet collection `strFonts`, dans lequel seront stockés les noms des polices installées :

```
Dim strFonts As New Collections.ArrayList(10)
```

La deuxième instruction définit l'objet `FontFamily Font`, qui sera utilisé pour accéder à chacune des polices installées :

```
Dim Font As FontFamily
```

Pour parcourir toutes les polices installées, le plus simple consiste à utiliser une instruction `For Each` :

```
For Each Font In FontFamily.Families
```

Le nom de chacune des polices est alors ajouté à la collection `strFonts` :

```
strFonts.Add(Font.Name)
```

Lorsque toutes les polices ont été parcourues, la liste déroulante est remplie en initialisant sa propriété `DataSource` avec la collection `strFonts` :

```
Next
ComboBox1.DataSource = strFonts
```

Pour être en mesure de faire référence aux fonctions de manipulation de texte de GDI+, vous devez inclure une instruction `Imports` en tête de code :

```
Imports System.Drawing
```

Définissez également l'objet `Graphics` `g`, juste après la déclaration de classe `Public Class Form1`. Cet objet sera utilisé pour réaliser l'affichage du texte :

```
Dim g As Graphics
```

Pour être en mesure d'afficher un texte dès le lancement du programme, vous allez définir la procédure `Form1_Paint()`. Sélectionnez l'entrée `(Form1 Evénements)` dans la première liste déroulante de la fenêtre de code et l'entrée `Paint` dans la seconde. Complétez la procédure `Form1_Paint` comme suit :

```
Private Sub Form1_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.
➥PaintEventArgs) Handles MyBase.Paint
    g = Me.CreateGraphics
    g.DrawString("Police " + ComboBox1.Text + ", 20 points", New Font(ComboBox1.
    ➥Text, 20), Brushes.Black, New PointF(30, 100))
End Sub
```

La première instruction donne accès au système graphique `g` :

```
g = Me.CreateGraphics
```

La seconde instruction utilise la procédure `DrawString()` pour afficher du texte dans la fenêtre :

```
g.DrawString("Police " + ComboBox1.Text + ", 20 points", New Font(ComboBox1.Text, 20),
➥Brushes.Black, New PointF(30, 100))
```

La procédure `DrawString()` existe en plusieurs déclinaisons :

```
DrawString(str, police, pinceau, PF)
DrawString(str, police, pinceau, RF)
DrawString(Str, police, pinceau, PF, Format)
DrawString(Str, police, pinceau, RF, Format)
DrawString(Str, police, pinceau, Sx, Sy)
DrawString(Str, police, pinceau, Sx, Sy, Format)
```

Voici la signification des arguments de cette procédure :

- `str` est la chaîne à afficher.

- `police` est le nom de la police à utiliser.
- `pinceau` est le pinceau avec lequel les caractères doivent être peints.
- `PF` est une référence à un objet `PointF` qui identifie le coin supérieur gauche de la zone de tracé.
- `RF` est une référence à un objet `RectangleF` qui identifie la zone de tracé (coordonnées du point supérieur gauche, longueur et hauteur).
- `Sx` et `Sy` sont les coordonnées Single du coin supérieur gauche de la zone de tracé.
- `Format` est une référence à un objet `StringFormat` qui spécifie le format d'affichage (espacement, alignement, etc.).

Dans cet exemple, la procédure `DrawString()` est utilisée dans sa plus simple expression. Remarquez la définition de la police et de l'objet `PointF`, directement dans l'appel à la procédure :

```
g.DrawString("Police " + ComboBox1.Text + ", 20 points", New Font(ComboBox1.Text, 20),
➥Brushes.Black, New PointF(30, 100))
```

Pour modifier l'affichage lorsque l'utilisateur sélectionne une police dans la liste déroulante, nous allons utiliser des instructions du même type. Double-cliquez sur le contrôle `ComboBox1` et complétez la procédure `ComboBox1_SelectedIndexChanged()` comme suit :

```
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
    g = Me.CreateGraphics
    g.Clear(Me.BackColor)
    g.DrawString("Police " + ComboBox1.Text + ", 20 points", New Font(ComboBox1.
    ➥Text, 20), Brushes.Black, New PointF(30, 100))
End Sub
```

La première instruction donne accès à l'objet graphique `g` :

```
g = Me.CreateGraphics
```

Pour ne pas interférer avec l'affichage précédent, la zone graphique de la fenêtre est effacée à l'aide de la procédure `Clear()`. La couleur utilisée est la couleur de fond actuelle (`Me.BackColor`) :

```
g.Clear(Me.BackColor)
```

Enfin, un appel à la procédure `DrawString()` affiche le nouveau texte en utilisant la police sélectionnée :

```
g.DrawString("Police " + ComboBox1.Text + ", 20 points", New Font(ComboBox1.Text, 20),
➥Brushes.Black, New PointF(30, 100))
```

Pour terminer cette application, double-cliquez sur le bouton de commande et ajoutez une instruction `End` dans la procédure `Button1_Click()` :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Button1.Click
    End
End Sub
```

Voici le listing complet de l'application. Vous trouverez les fichiers correspondants dans le dossier Polices après installation des sources de l'ouvrage.

```
Imports System.Drawing

Public Class Form1

    Dim g As Graphics

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles MyBase.Load
        Dim strFonts As New Collections.ArrayList(10)
        Dim Font As FontFamily

        For Each Font In FontFamily.Families
            strFonts.Add(Font.Name)
        Next
        ComboBox1.DataSource = strFonts
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object,
    ➥ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        g = Me.CreateGraphics
        g.Clear(Me.BackColor)
        g.DrawString("Police " + ComboBox1.Text + ", 20 points",
        ➥New Font(ComboBox1.Text, 20), Brushes.Black, New PointF(30, 100))
    End Sub

    Private Sub Form1_Paint(ByVal sender As Object, ByVal e As System.Windows.
    ➥Forms.PaintEventArgs) Handles MyBase.Paint
        g = Me.CreateGraphics
        g.DrawString("Police " + ComboBox1.Text + ", 20 points",
        ➥New Font(ComboBox1.Text, 20), Brushes.Black, New PointF(30, 100))
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.
EventArgs) Handles Button1.Click
        End
    End Sub
End Class
```

Les possibilités offertes par GDI+ au niveau des polices de caractères sont très étendues. Il est par exemple possible d'effectuer des rotations ou des inclinaisons sur le texte ou encore d'utiliser un pinceau non uniforme pour dessiner les caractères.

Les rotations et inclinaisons se font à l'aide de la procédure `RotateTransform()`. Par exemple, l'instruction suivante provoque une inclinaison de 45 degrés du texte :

```
G.RotateTransform(45)
```

Pour utiliser un pinceau non uniforme, il suffit de le définir comme tel. À titre d'exemple, l'instruction suivante définit un objet `Bitmap` à partir du fichier c:\test\colorwir.bmp :

```
Dim Bmap = New Bitmap("C:\\test\\colorwir.bmp")
```

Cet objet `Bitmap` est utilisé pour définir un pinceau `TextureBrush B` :

```
Dim B = New TextureBrush(Bmap)
```

Il suffit maintenant d'utiliser ce pinceau dans la procédure `DrawString()` pour obtenir un texte texturé :

```
g.DrawString(ComboBox1.Text, New Font(ComboBox1.Text, 120, FontStyle.Bold), B,
➥New PointF(30, 60))
```

La Figure 5.4 représente l'effet obtenu dans le programme précédent lorsque la police `Agency FB` est sélectionnée.

Figure 5.4
Un exemple de texturage à l'aide d'un fichier bitmap.

Voici le code de l'application. Les fichiers correspondants se trouvent dans le dossier Polices2 après installation des sources de l'ouvrage :

```
Imports System.Drawing.Text


Public Class Form1

    Dim g As Graphics
```

```
    Dim Bmap = New Bitmap("C:\\test\\colorwir.bmp")
    Dim B = New TextureBrush(Bmap)

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles MyBase.Load
        Dim strFonts As New Collections.ArrayList(10)
        Dim Font As FontFamily

        For Each Font In FontFamily.Families
            strFonts.Add(Font.Name)
        Next
        ComboBox1.DataSource = strFonts
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object,
    ➥ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        g = Me.CreateGraphics
        g.Clear(Me.BackColor)
        g.DrawString(ComboBox1.Text, New Font(ComboBox1.Text, 120, FontStyle.
        ➥Bold), B, New PointF(30, 60))
    End Sub

    Private Sub Form1_Paint(ByVal sender As Object, ByVal e As System.Windows.
    ➥Forms.PaintEventArgs) Handles MyBase.Paint
        g = Me.CreateGraphics
        g.DrawString(ComboBox1.Text, New Font(ComboBox1.Text, 120, FontStyle.
        ➥Bold), B, New PointF(30, 60))
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Button1.Click
        End
    End Sub
End Class
```

Le Presse-papiers

Pour accéder au presse-papiers de Windows, vous pouvez utiliser :

- les procédures et fonctions traditionnelles de .NET, telles que `SetDataObject()` et `GetDataObject()` ;
- les procédures et fonctions apparues dans la version 2.0 de .NET, relatives à l'objet `My.Computer.Clipboard`.

Nous allons examiner ces deux approches dans les pages suivantes.

Accès traditionnel au presse-papiers

Visual Basic 2010 possède un certain nombre de fonctions qui lui permettent de manipuler le presse-papiers de Windows :

- `SetDataObject()` place un objet dans le presse-papiers.
- `GetDataObject().GetDataPresent("typeObjet")` permet de tester le type de l'objet présent dans le presse-papiers.
- `GetDataObject().GetData("typeObjet")` permet de récupérer le contenu du presse-papiers sous la forme d'un objet dont le type est spécifié dans la fonction `GetData()`.

L'application développée dans ce chapitre montre comment copier un objet bitmap dans le presse-papiers, le récupérer et l'affecter à un contrôle `PictureBox`.

Définissez un nouveau projet de type Application Windows Forms avec la commande Nouveau/Projet dans le menu Fichier. Choisissez .NET Framework 4 dans la liste déroulante Framework. L'entrée Visual Basic/Windows étant sélectionnée dans le volet gauche, choisissez le modèle Application Windows Forms, nommez le projet OutilsEtat et validez en cliquant sur OK. Ajoutez un contrôle `PictureBox` sur la feuille du projet. Maximisez la taille de ce contrôle en donnant la valeur `Fill` à sa propriété `Dock` (pour cela, développez la liste déroulante de la propriété `Dock` et cliquez sur le rectangle central). Affectez une image quelconque au contrôle `PictureBox` par l'intermédiaire de sa propriété `Image` (dans cet exemple, le programme utilise les images Colorwir.bmp et Vide.bmp, qui se trouvent dans le dossier Presse papiers après installation des sources de l'ouvrage).

Ajoutez un contrôle `ContextMenuStrip` au projet et définissez les entrées de menu suivantes : Couper, Copier, Coller, Effacer et Quitter. Renommez ces commandes à l'image de leurs noms respectifs (la propriété `Name` de la commande Couper est initialisée à "Couper", la propriété `Nom` de la commande Coller est initialisée à "Coller", etc.). Pour rendre le menu opérationnel, vous devez modifier la propriété `ContextMenuStrip` des objets sur lesquels vous voulez l'utiliser. Sélectionnez `Form1` dans la fenêtre des propriétés et affectez la valeur `ContextMenuStrip1` à la propriété `ContextMenuStrip`.

Double-cliquez sur la commande Couper et complétez la procédure `Couper_Click()` comme suit :

```
Private Sub Couper_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Couper.Click
    Clipboard.SetDataObject(PictureBox1.Image)
    PictureBox1.Image = Image.FromFile("c:\vide.bmp")
End Sub
```

La première instruction place le contenu de l'image (`Picture1.Image`) dans le presse-papiers (`Clipboard.SetDataObject()`). La deuxième instruction affecte l'image blanche c:\vide.bmp au contrôle `PictureBox1`.

Double-cliquez sur la commande Copier et complétez la procédure `Copier_Click()` comme suit :

```
Private Sub Copier_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Copier.Click
    Clipboard.SetDataObject(PictureBox1.Image)
End Sub
```

Cette procédure ressemble à `Couper_Click()`. Mais ici, après la copie de l'image dans le presse-papiers, le contrôle `PictureBox` n'est pas effacé. Double-cliquez sur la commande Coller. La procédure `Coller_Click()` est légèrement plus complexe que les précédentes :

```
Private Sub Coller_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Coller.Click
    If (Clipboard.GetDataObject().GetDataPresent("Bitmap")) Then
        PictureBox1.Image = (Clipboard.GetDataObject().GetData("Bitmap"))
    End If
End Sub
```

L'instruction `if` teste le contenu du presse-papiers à l'aide de la fonction `GetDataObject().GetDataPresent("Bitmap")`.

```
If (Clipboard.GetDataObject().GetDataPresent("Bitmap")) Then
```

Si la valeur `True` est renvoyée par cette fonction, cela signifie que le contenu du presse-papiers est bien de type bitmap. Dans ce cas, la fonction `GetDataObject().GetData("Bitmap")` récupère le contenu du presse-papiers et l'affecte au contrôle `PictureBox` :

```
PictureBox1.Image = (Clipboard.GetDataObject().GetData("Bitmap"))
```

Double-cliquez sur la commande Effacer et complétez la procédure `Effacer_Click()` comme suit :

```
Private Sub Effacer_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Effacer.Click
    PictureBox1.Image = Image.FromFile("c:\vide.bmp")
End Sub
```

Cette procédure se contente de copier l'image blanche c:\vide.bmp dans le contrôle `PictureBox` afin de l'effacer.

Double-cliquez enfin sur la commande Quitter et ajoutez le mot `End` à la procédure `Quitter_Click()` :

```
Private Sub Quitter_Click(ByVal sender As System.Object, ByVal e As System
➥.EventArgs) Handles Quitter.Click
    End
End Sub
```

Voici le listing complet du programme. Vous trouverez les fichiers correspondants dans le dossier Presse papiers après installation des sources de l'ouvrage.

```
Public Class Form1

    Private Sub Couper_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Couper.Click
        Clipboard.SetDataObject(PictureBox1.Image)
        PictureBox1.Image = Image.FromFile("c:\vide.bmp")
    End Sub

    Private Sub Effacer_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Effacer.Click
        PictureBox1.Image = Image.FromFile("c:\vide.bmp")
    End Sub

    Private Sub Copier_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Copier.Click
        Clipboard.SetDataObject(PictureBox1.Image)
    End Sub

    Private Sub Coller_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Coller.Click
        If (Clipboard.GetDataObject().GetDataPresent("Bitmap")) Then
            PictureBox1.Image = (Clipboard.GetDataObject().GetData("Bitmap"))
        End If
    End Sub

    Private Sub Quitter_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Quitter.Click
        End
    End Sub
End Class
```

Accès au presse-papiers *via* .NET 4.0

Dupliquez le contenu du dossier Presse papiers dans le dossier Presse papiers 2. Ouvrez la solution Presse papiers.sln du dossier Presse papiers 2. Pour utiliser les procédures et fonction de .NET 4.0, vous allez modifier les procédures événementielles de l'application.

Lancez la commande Code dans le menu Affichage (ou appuyez sur la touche F7). Modifiez la procédure `Couper_Click()` comme suit :

```
Private Sub Couper_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Couper.Click
```

```
    My.Computer.Clipboard.SetImage(PictureBox1.Image)
    PictureBox1.Image = Image.FromFile("c:\vide.bmp")
End Sub
```

La première instruction place le contenu du contrôle `PictureBox` dans le presse-papiers à l'aide de la procédure `SetImage()`.

La seconde instruction copie le contenu du fichier c:\vide.bmp dans le contrôle `PictureBox`.

Vous allez maintenant modifier la procédure `Copier_Click()` comme suit :

```
Private Sub Copier_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Copier.Click
  My.Computer.Clipboard.SetImage(PictureBox1.Image)
End Sub
```

L'unique instruction de cette procédure copie le contenu du contrôle `PictureBox` dans le presse-papiers à l'aide de la procédure `SetImage()`.

La dernière action va consister à modifier la procédure `Coller_Click()` :

```
Private Sub Coller_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
➥Handles Coller.Click
  If (My.Computer.Clipboard.ContainsImage) Then
    PictureBox1.Image = My.Computer.Clipboard.GetImage()
  End If
End Sub
```

L'instruction `If` teste si le presse-papiers contient un objet graphique :

```
If (My.Computer.Clipboard.ContainsImage) Then
```

Dans l'affirmative, cet objet est affecté au presse-papiers :

```
PictureBox1.Image = My.Computer.Clipboard.GetImage()
```

Les procédures `Effacer_Click()` et `Quitter_Click()` ne changent pas, car elles utilisent des instructions Visual Basic traditionnelles.

Voici le listing complet de l'application. Les fichiers correspondants se trouvent dans le dossier Projects\Presse papiers 2 des sources de l'ouvrage.

```
Public Class Form1

    Private Sub Couper_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Couper.Click
        My.Computer.Clipboard.SetImage(PictureBox1.Image)
        PictureBox1.Image = Image.FromFile("c:\vide.bmp")
    End Sub
```

```
    Private Sub Copier_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Copier.Click
        My.Computer.Clipboard.SetImage(PictureBox1.Image)
    End Sub

    Private Sub Coller_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Coller.Click
        If (My.Computer.Clipboard.ContainsImage) Then
            PictureBox1.Image = My.Computer.Clipboard.GetImage()
        End If
    End Sub

    Private Sub Effacer_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Effacer.Click
        PictureBox1.Image = Image.FromFile("c:\vide.bmp")
    End Sub

    Private Sub Quitter_Click(ByVal sender As System.Object, ByVal e As System.
    ➥EventArgs) Handles Quitter.Click
        End
    End Sub
End Class
```