18

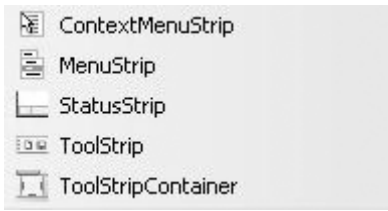
Barres de menu, d'état et de boutons

Dans ce chapitre, nous allons apprendre à créer et à manipuler :

- le menu de l'application (*menu strip* en anglais) ;
- le menu contextuel (*context menu strip* en anglais) ;
- une liste d'images, en préparation de la barre de boutons (*tool strip container* en anglais) ;
- la barre de boutons (*tool strip* en anglais) ;
- la barre d'état (*status strip* en anglais).

La plupart de ces composants ont été considérablement revus en version 2, ce qui les rend à la fois plus élaborés et plus simples à utiliser.

Figure 18-1



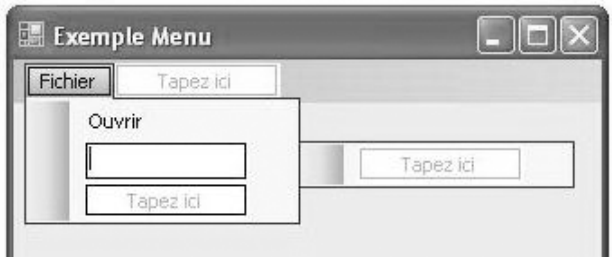
18.1 Le menu

18.1.1 Construire un menu

Créer le menu de l'application est l'enfance de l'art :

- Cliquez sur l'icône `MenuStrip` dans la boîte à outils (catégorie `Menu` et `barres d'outils`).
- Cliquez dans la fenêtre de développement.
- Un objet de la classe `MenuStrip` est créé. Par défaut, il s'appelle `menuStrip1` (propriété `Name`). Une ébauche de menu s'affiche dans la fenêtre de développement (voir figure 18-2). À tout moment, Visual Studio est prêt à créer des articles à gauche et au-dessous de l'article courant. Il suffit de compléter les cases qui s'ajoutent au fur et à mesure que le menu se construit (cliquez sur `Tapez ici` et confirmez la création d'un article de menu par `ENTREE`). Pour déplacer un article, procédez par glisser-déposer. Pour placer une ligne de séparation, signalez – (tiret) comme seule lettre de libellé (également possible : clic droit sur un article → `Insérer` → `Séparateur`, le séparateur étant alors inséré devant l'article).

Figure 18-2



Il est possible d'ajouter (clic droit sur un article → `Insérer`) une zone d'édition ou une boîte combo dans le menu, mais comme cela ne se rencontre pratiquement jamais dans un menu, nous réserverons ces opérations pour les barres d'outils et d'état.

18.1.2 Les classes de menu et d'articles

Le menu lui-même est un objet de la classe `MenuStrip` tandis que les articles sont des objets de la classe `ToolStripMenuItem` (et `ToolStripSeparator` pour un séparateur). La classe `MenuStrip`, ainsi que les classes `ContextMenuStrip` et `StatusStrip` dont il sera question dans ce chapitre, sont dérivées de la classe `ToolStrip`. Ceci est normal car il y a beaucoup de points communs entre le menu et les différentes barres. La classe `MenuStrip` agit comme conteneur pour des articles (objets `ToolStripMenuItem`).

Pour faire apparaître les propriétés d'un article : clic droit sur l'article → `Propriétés`. Empressez-vous de modifier son nom interne (propriété `Name`) pour que le programme reste lisible (par exemple `miFichier` pour l'article `Fichier` plutôt que `fichierToolStripMenuItem1` proposé par défaut). Une technique simple pour modifier tous

les noms internes et d'autres caractéristiques d'articles : clic droit sur une partie inoccupée de la barre de menu → `Modifier les éléments`.

Nous savons que la barre de menus est un objet de la classe `MenuStrip`. La propriété `MainMenuStrip` de la fenêtre donne accès à la barre de menus. Vous pouvez changer la couleur de fond (`BackColor`) et la police d'affichage des articles (`Font`). `MainMenuStrip.Items` donne accès aux articles de niveau supérieur (ceux qui sont toujours affichés).

Pour chaque article (de niveau supérieur ou non), modifiez éventuellement ses propriétés.

Propriétés de la classe `ToolStripMenuItem`

`ToolStripMenuItem` ← `ToolStripDropDownItem` ← `ToolStripItem` ← `Component`

`Checked`	T/F	Indique que l'article est coché (coche en forme d'oiseau en vol).
`Enabled`	T/F	Indique que l'article est activable (cliquer dessus déclenche une action).
`Image`		Image associée à l'article et affichée à gauche du libellé.
`DropDownItems`	coll	Collection des articles du sous-menu de cet article.

`ShortcutKeys`	Shortcut	Touche de raccourci (encore appelée d'accélération) pour l'article (pour le raccourci ALT+une lettre du libellé, voir la propriété `Text`). `ShortcutKeys` peut prendre l'une des valeurs suivantes de l'énumération `Keys` (signification évidente d'après le nom de la touche de fonction ou le nom du raccourci) :
		`F1 à F12` `ShiftF1 à ShiftF12`
		`CtrlF1 à CtrlF12` `AltF1 à AltF12`
		`CtrlShiftF1 à CtrlShiftF12`
		`CtrlA à CtrlZ`
		`CtrlShiftA à CtrlShiftZ`
		`ShiftIns` `ShiftDel`
		`Ins` `Del`
		`CtrlIns` `CtrlDel`
		`AltBksp`

`ShowShortcutKeys`	T/F	Indique si le libellé de l'article doit mentionner la touche ou combinaison de raccourci (à droite dans le libellé, avec justification à droite). L'affichage de la touche de raccourci est alors automatique.
`Text`	str	Libellé de l'article. Dans ce libellé, vous pouvez préfixer une lettre du caractère & pour que la combinaison ALT+cette lettre joue le rôle d'accélérateur.
`Visible`	T/F	Indique si l'article est visible.

Vous pouvez modifier le look du menu en jouant sur la propriété `RenderMode`. Par défaut, ce mode est `ManagerRenderMode`, le mode `System` étant plus sobre (trop même au goût de certains).

Pour générer la fonction de traitement d'un article du menu, il suffit d'un double-clic sur l'article dans la fenêtre de développement. Ou bien passez par les propriétés de l'article et accédez à ses événements (ici l'événement `Click`).

18.1.3 Modification de menu par programme

La propriété `Items` de la barre de menu donne accès aux articles de niveau supérieur, ceux qui sont toujours affichés. La propriété `DropDownItems` d'un article permet d'accéder à ses sous-articles.

Des articles peuvent être ajoutés en cours d'exécution de programme. Commencez par ajouter dans le programme une fonction de traitement, ou utilisez une fonction déjà créée, ou utilisez la technique des fonctions anonymes. Par exemple (ici, ajout d'un article à la fin du sous-menu `Fichier`) :

```
private void mi_Click(object sender, EventArgs e)
{
 .....
}
```

Pour ajouter, par programme, l'article `Fermer` en fin de menu `Fichier` (article de la barre de menu avec `miFichier` comme nom interne) et faire traiter cet article `Fermer` par la fonction `mi_Click`, il suffit d'écrire (le deuxième argument se rapporte à l'image associée à l'article, inexistante ici, d'où la valeur `null`) :

```
ToolStripMenuItem mi = new ToolStripMenuItem("Fermer", null, mi_Click);
miFichier.DropDownItems.Add(mi);
```

Dans cette fonction `mi_Click`, on détecte l'origine du clic (détection indispensable si une même fonction de traitement est associée à plusieurs articles) en écrivant (on aurait pu tester les noms internes ou les `Tag`, ce qui est d'ailleurs nécessaire quand les libellés s'adaptent à la langue de l'utilisateur) :

```
ToolStripMenuItem m = (ToolStripMenuItem)sender;
if (m.Text == "Fermer") .....
```

On aurait aussi pu écrire (en version 2 de .NET uniquement), sans devoir écrire explicitement la fonction de traitement (..... désignant les instructions à exécuter en réponse à un clic sur l'article `Fermer`) :

```
ToolStripMenuItem m = new ToolStripMenuItem("Fermer");
m.Click += delegate { .....};
```

Pour insérer l'article `Fermer` en première position dans le sous-menu `Fichier` :

```
miFichier.DropDownItems.Insert(0, m);
```

Pour modifier dynamiquement l'image associée à l'article `miOuvrir` (`imgList` désignant un composant `Image`, voir la section 18.2, contenant plusieurs images, la deuxième étant maintenant associée à `miOuvrir`) :

```
miOuvrir.Image = imgList.Images[1];
```

Il pourrait également s'agir d'une image incorporée en ressource dans le programme (voir la section 13.2).

Tout un menu peut être supprimé. Différentes techniques sont possibles pour cela :

```
MainMenuStrip.Dispose();         // la barre de menu disparaît
```

ou

```
MainMenuStrip.Items.Clear();
```

Avec la seconde solution, la barre de menus reste, mais les articles disparaissent.

Un menu peut être remplacé par un autre. Il suffit pour cela de créer différents `MenuStrip` (les différentes bandes de menus s'empilent alors dans la fenêtre de développement) et de jouer sur leur propriété `Visible` pour faire apparaître un menu plutôt qu'un autre.

18.1.4 Les événements liés au menu

L'événement `DropDownOpening` est signalé juste avant l'affichage d'un menu. Il vous donne l'occasion de griser des articles si ceux-ci, à ce moment, ne peuvent être invoqués. Pour griser l'article `Coller` (parce que rien n'est disponible dans le presse-papiers à ce moment), il suffit de :

- traiter l'événement `DropDownOpening` pour l'article de niveau supérieur (généralement l'article `Edition` dans la barre de menu) ;
- vérifier si le Presse-papiers contient du texte ou une image, en effectuant respectivement les tests (`res` prend la valeur `true` si du texte ou une image est présent dans le presse-papiers, voir l'exemple du chapitre 17) :

```
    bool res = Clipboard.GetDataObject().GetDataPresent("Text", true)
```

ou

```
    res = Clipboard.GetDataObject().GetDataPresent("Bitmap", true)
```

- modifier la propriété `Enabled` en conséquence (pour tester le programme, exécutez `Clipboard.Clear()` afin de vider le presse-papiers) :

```
    mnuColler.Enabled = res;
```

18.1.5 Les menus contextuels

Un menu contextuel a toutes les caractéristiques d'un menu : il est créé et traité comme un menu de la barre de menu mais peut être affiché n'importe où dans la fenêtre, généralement en réponse à un clic droit.

Un menu contextuel est un objet de la classe `ContextMenuStrip`. À partir de la boîte à outils, vous déposez l'icône `ContextMenuStrip` dans la fenêtre. L'icône de menu contextuel est affichée sous la fenêtre de développement, comme l'icône de menu.

Lorsque l'icône de menu contextuel est sélectionnée, le menu contextuel apparaît dans la fenêtre de développement, à l'emplacement du menu traditionnel (ce n'est pas le cas en cours d'exécution de programme). Modifiez les propriétés du menu contextuel (surtout sa propriété `Name` de manière à rendre le programme plus lisible), ajoutez des articles et

des fonctions de traitement, exactement comme nous venons de le faire pour un menu traditionnel.

Si `ctxtmnuRéserver` est le nom interne d'un menu contextuel, on affiche celui-ci au point (`X`, `Y`) de la fenêtre en exécutant :

```
ctxtmnuRéserver.Show(this, new Point(X, Y));
```

La touche d'échappement mais aussi tout clic en dehors du menu contextuel fait disparaître celui-ci. Tout clic sur un article du menu contextuel fait exécuter la fonction de traitement associée à cet article.

L'événement `Opening` est signalé juste avant l'affichage du menu contextuel. Traiter cet événement présente peu d'intérêt car les modifications dans le menu contextuel pouvaient être effectuées avant d'exécuter `Show`.

Il est possible de créer entièrement et dynamiquement un menu contextuel, juste avant l'affichage (par exemple dans la fonction qui traite le clic droit) :

```
ContextMenuStrip ctxMenu = new ContextMenuStrip();
ctxMenu.Items.Add("Garder", null, onGarder);
ctxMenu.Items.Add("Eliminer", null, onEliminer);
ctxMenu.Show(this, new Point(e.X, e.Y));
.....
private void onGarder(object sender, EventArgs e) { ..... }
private void onEliminer(object sender, EventArgs e) { ..... }
```

On aurait aussi pu écrire, grâce aux fonctions anonymes :

```
ContextMenuStrip ctxMenu = new ContextMenuStrip();
ToolStripMenuItem cmi = new ToolStripMenuItem("Garder");
cmi.Click += delegate { ..... };
ctxMenu.Items.Add(cmi);
cmi = new ToolStripMenuItem("Eliminer");
cmi.Click += delegate { ..... };
ctxMenu.Items.Add(cmi);
ctxMenu.Show(this, new Point(e.X, e.Y));
```

les ..... étant évidemment à remplacer par les instructions qui traitent respectivement les articles `Garder` et `Eliminer`.

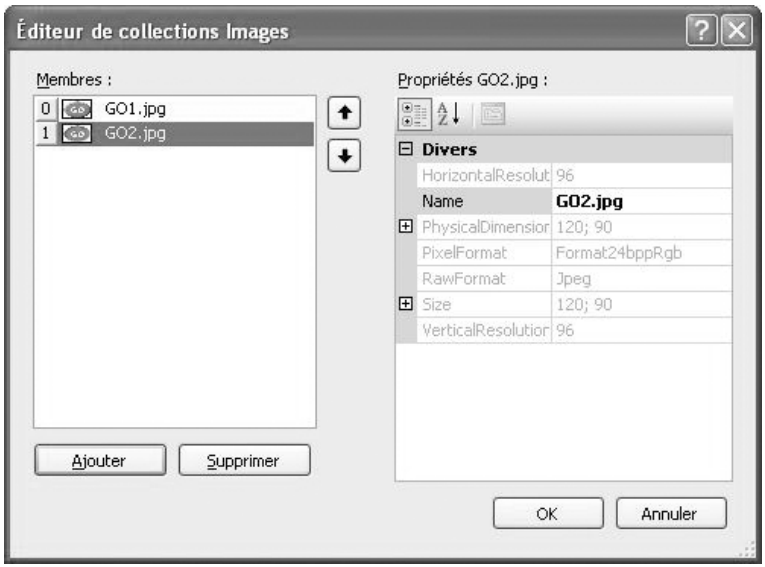
18.2 Les listes d'images

En préparation de la barre de boutons (constituée d'une série d'icônes, mais pas uniquement), il faut créer une liste d'images, toutes de même taille. Une liste d'images permet aussi de regrouper plusieurs images ayant un rapport entre elles. Les différentes images d'une liste d'images sont généralement affichées à la même taille, quelle que soit leur taille d'origine.

Une liste d'images (composant `ImageList`) contient essentiellement la propriété `Images` qui correspond à une collection d'images. Un clic sur les trois points de suspension

amène à l'avant-plan l'éditeur de collection d'images. Pour ajouter une image dans la collection, il suffit de cliquer sur le bouton d'ajout et de sélectionner (dans la boîte de sélection de fichier) un fichier d'extension `JPEG`, `ICO`, `GIF`, `BMP` et `PNG`. Pour modifier l'ordre des images, on sélectionne (dans la partie gauche de l'éditeur) l'image à déplacer, et on clique sur les flèches vers le haut et vers le bas (voir figure 18-3) :

Figure 18-3



Les propriétés de la classe `ImageList` sont peu nombreuses. La méthode `Draw` ne doit pas être utilisée dans le cas des boutons de commande (l'affichage de l'image est alors automatique) mais nous présenterons néanmoins ici les différentes variantes de `Draw` (voir le chapitre 13 consacré au GDI+).

Classe `ImageList`		
`ImageList` ← `Component`		
Propriétés de la classe `ImageList`		
`ColorDepth`	`enum`	Une des valeurs de l'énumération `ColorDepth` qui indique le nombre de couleurs dans l'image : `Depth4Bit` (image limitée à 16 couleurs), `Depth8Bit` (à 256 couleurs), `Depth16Bit` ou `Depth32Bit`. Il est important d'initialiser correctement ce champ dans le cas d'images avec zones transparentes.
`Images`	`coll`	Collection des images de la liste.
`ImageSize`	`Size`	Taille de chaque image (toutes les images de la collection doivent être de même taille). La taille des images introduites dans un `ImageList` est limitée à 255 x 255 pixels.
`TransparentColor`	`Color`	Couleur qui doit être traitée comme couleur de transparence. À l'emplacement des pixels de cette couleur, le fond de la fenêtre est préservé (initialiser correctement `ColorDepth`).

Méthodes de la classe `ImageList`	
`void Draw(Graphics, Point p, int n);`	Dessine la `n`-ième image de la liste à partir du point `p`. L'affichage est réalisé dans l'objet (généralement une fenêtre mais il peut s'agir d'un contrôle) sur lequel porte le premier argument.
`void Draw(Graphics, int x, int y, int n);`	Même chose mais les coordonnées du coin supérieur gauche sont données en deux arguments entiers plutôt que dans un objet `Point`.
`void Draw(Graphics, int x, int y, int w, int h, int n);`	Même chose mais la largeur et la hauteur de la zone d'affichage sont spécifiées respectivement dans `w` et `h`.

On crée souvent une liste d'images pour préparer la création de la barre des boutons. Nous n'aurons pas besoin des méthodes de la classe `ImageList` dans ce travail de préparation. Nous utiliserons cependant ces méthodes pour d'autres préparations que celle de la barre des boutons.

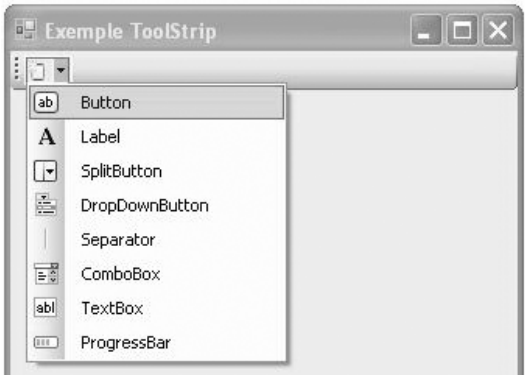
18.3 La barre d'outils

Une barre d'outils reprend en général, mais sous forme visuelle, les principaux articles du menu. Cette barre est en général accolée au bord supérieur de la fenêtre mais elle peut être accolée à n'importe quel bord ou même être rendue flottante. Il n'est pas rare qu'elle contienne des zones d'édition, des boîtes combo ou d'autres contrôles.

Après avoir amené le composant `ToolStrip`, la barre d'outils (ici encore vide, voir figure 18-4) est préfigurée dans la fenêtre de développement (ici dans la partie supérieure de la fenêtre, sous l'éventuel menu).

Si la barre d'outils se retrouve au-dessus de la barre de menu et que vous désirez changer cela : clic droit sur la barre de menu → `Mettre en arrière-plan`.

Figure 18-4



Dans la barre d'outils, il est possible de placer des boutons (`Button`, `SplitButton` ou `DropDownButton`), des étiquettes (`Label`), des boîtes combo, des zones d'édition et des barres de progression. Mais on peut aussi placer, par programme, n'importe quel composant.

Pour placer un composant entre deux autres déjà placés dans la barre d'outils : clic sur l'icône de droite dans la barre d'outils. Pour déplacer un composant : opération de glisser-déposer (ou passez par l'éditeur de la propriété `Items`). Pour supprimer ou insérer : clic droit sur le composant. Ce clic droit a d'autres possibilités, comme l'ajout d'une image à un bouton.

Remplir une barre d'outils est plus simple à faire qu'à expliquer. À chaque composant de la barre, donnez un nom interne significatif. Chaque composant de la barre d'outils est alors utilisé comme un composant normal, tant en phase de développement qu'en cours d'exécution.

18.3.1 Les différents types de boutons dans une barre d'outils

Un `ToolStripButton` (petit bouton avec image) est généralement affiché sans texte mais avec une bulle d'aide. Un libellé peut néanmoins être associé au bouton (voir les propriétés `DisplayStyle`, `Text`, `TextAlign`, `TextDirection` et `TextImageRelation`).

Les `ToolStripSplitButton` et `ToolStripDownButton` sont assez semblables : bouton avec flèche à droite pour faire apparaître un sous-menu. Dans le cas d'un `SplitButton`, il faut cliquer sur la flèche pour faire apparaître le sous-menu. Dans le cas d'un `DropDownButton`, on peut cliquer sur l'image ou sur la flèche pour faire apparaître le sous-menu. Dans les deux cas, le sous-menu se complète comme un menu (propriété `DropDownItems`).

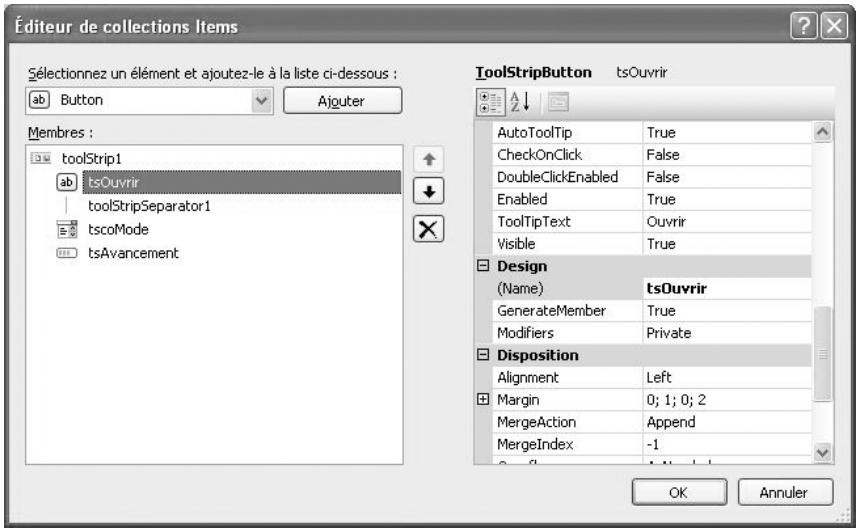
18.3.2 Les autres types de composants dans une barre d'outils

Les `ToolStripLabel` permettent d'afficher du texte ou de séparer divers éléments de la barre (laisser dans ce cas des espaces blancs dans la propriété `Text`, l'espace laissé libre étant proportionnel au nombre de caractères blancs dans `Text`). Ce composant peut jouer le rôle de lien (faire passer à `true` la propriété `IsLink` et traiter l'événement `Click`) ou contenir une image qui peut être modifiée en cours d'exécution de programme (propriété `Image`).

Les `ToolStripTextBox`, `ToolStripComboBox` et `ToolStripProgressBar` ne demandent aucune explication complémentaire par rapport aux zones d'édition, zones d'affichage et barres de progression.

La boîte de dialogue `Editeur de collections` (`smartag` → `Modifier les éléments`) vous permet de spécifier les caractéristiques de chaque élément sans même devoir passer par la fenêtre des propriétés. Associez un nom interne à chaque élément, spécifiez ses caractéristiques (notamment type de bouton et numéro d'image dans la liste des images, voir figure 18-5).

Figure 18-5



Présentons les rares propriétés propres à la barre d'outils (clic droit en un emplacement libre de la barre d'outils → `Propriétés`). Il s'agit ici de la barre d'outils dans son ensemble et non des composants individuels. N'insistons pas sur `BackColor`, `BackgroundColor` et `BackgroundImageLayout`, maintenant bien connus. Le mode `System` de `RenderMode` donne un aspect plus sobre. C'est la propriété `Items` qui est la plus importante car elle donne la collection des éléments de la barre.

N'importe quel composant peut être placé dans une barre d'outils, à condition de passer par un objet `ToolStripControlHost`. Ainsi, pour insérer une case à cocher dans la barre d'outils (instructions à placer dans la fonction traitant l'événement `Load`) :

```
ToolStripControlHost ch;
.....
void Form1_Load(object sender, EventArgs e)
{
 CheckBox cb = new CheckBox();
 cb.Text = "xyz"; cb.BackColor = Color.Transparent;
 ch = new ToolStripControlHost(cb);
 toolStrip1.Items.Add(ch);
}
```

Pour retrouver l'état de la case :

```
CheckBox c = (CheckBox)ch.Control;
bool bEtat = c.Checked;
```

Autre exemple : voyons comment insérer une animation dans la partie droite de la barre, comme le font notamment Internet Explorer, Netscape Navigator et FireFox, durant les temps de chargement, plus ou moins longs. Il faut d'abord créer les différentes figures de l'animation avec un logiciel approprié. Ces différentes images fixes seront affichées à

intervalles rapprochés, comme on le fait dans toute animation. Ces images sont insérées dans une liste d'images.

Soit `imgCube` le composant `ImageList`. Chargez dans `imgCube` les différentes images (ici quatre) de l'animation (cube en rotation). Adaptez la taille de chaque image (propriété `Size`) à la hauteur de la barre (25 pixels par défaut). Écrire (`tb` désignant la barre d'outils) :

```
int H = tb.Height;
imgCube.ImageSize = new Size(H, H);
.
```

Créons maintenant un `PictureBox` et accrochons-le dans le `ToolStripControlHost` (toujours dans la fonction traitant l'événement `Load`) :

```
PictureBox pbCube;
.....
void Form1_Load(object sender, EventArgs e)
{
 pbCube = new PictureBox();
 pbCube.Image = imgCube.Images[0];
 ToolStripControlHost ch = new ToolStripControlHost(pbCube);
 ch.Alignment = ToolStripItemAlignment.Right;
 tb.Items.Add(ch);
}
```

On génère un contrôle `Timer`, on fait passer sa propriété `Enable` à `true` et sa propriété `Interval` à `200` (ce qui donnera cinq images par seconde, ce qui est bien suffisant ici). Dans la fonction traitant l'événement `Tick` du `Timer`, on écrit :

```
int nImage=0;
private void timer1_Tick(object sender, System.EventArgs e)
{
 pbCube.Image = imgCube.Images[nImage];
 nImage++; if (nImage>3) nImage = 0;
}
```

18.4 La barre d'état

Une barre d'état (composant `StatusStrip`) est créée de manière semblable. Inutile donc de le répéter.

19. La barre d'état (encore vide de compartiments dans la figure 18-6) est préfigurée dans la fenêtre de développement.

Figure 18-6

Cliquez sur la barre d'état (pour la sélectionner), puis sur la flèche vers le bas dans la première icône pour créer, dans la barre d'état, des étiquettes (`StatusLabel`, qui peut être une zone d'affichage, une image ou un lien), une barre de progression (`ProgressBar`) ou un bouton (`DropDownButton` ou `SplitButton`). Pensez à donner des noms significatifs aux noms internes (propriété `Name`).

Dans le cas de zones d'affichage, modifiez `BorderStyle` avec ses valeurs de l'énumération `Border3DStyle` : `Flat` (aucun relief), `Raised` (ressort de l'écran), `Sunken` (rentre dans l'écran), `Etched` (rainure) et quelques variantes. Ces valeurs n'ont cependant d'effet que si `BorderSides` vaut `All` (ce sont effectivement les bords qui donnent l'effet de relief). En général, le dernier compartiment est de type `Spring`, ce qui lui permet d'occuper tout l'espace restant dans la barre d'état.

Pour afficher l'heure dans le compartiment (`StatusLabel`) dont le nom interne est `stHeure` :

```
stHeure.Text = DateTime.Now.ToLongTimeString();
```

ou encore (`st` désignant la barre d'état) :

```
st.Items["stHeure"].Text = DateTime.Now.ToLongTimeString();
```

`Items` peut en effet être indexé sur une position ou sur un nom interne de contrôle dans la barre d'état.

Programmes d'accompagnement

`MenuGraphique` Affichage d'un menu dont les articles sont des images.

Figure 18-7

`BarreBoutons` Barre de boutons avec animation dans un bouton toujours cadré à droite (cube dont chaque face représente le logo des éditions Eyrolles et qui est en rotation permanente). Un menu déroulant ainsi qu'une boîte combo sont également insérés dans la barre des boutons.

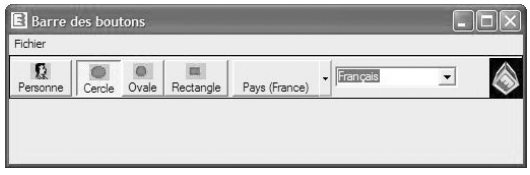


Figure 18-8

`BarreEtat`	Barre d'état avec affichage permanent de l'heure, affichage permanent de la position de la souris et affichage graphique dans un compartiment pour signaler la position du curseur (au-dessus de l'un des rectangles de couleur).

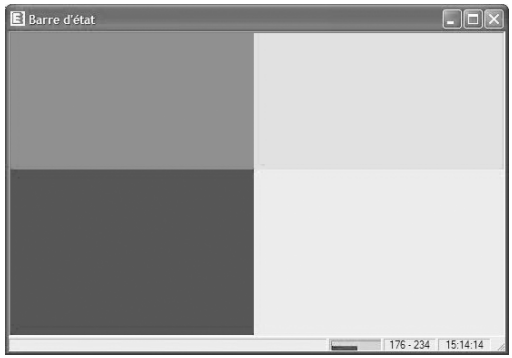


Figure 18-9