L'IMPRESSION SOUS VISUAL BASIC

Par J-M RABILLOUD

Préambule

Le présent article va concerner l'utilisation de l'objet printer. Dans la première partie de cet article nous étudierons l'objet, ses propriétés et ses méthodes. Dans la seconde, nous verrons plusieurs méthodes pour simplifier son utilisation.

L'objet printer minus

Peu d'objet de visual basic crée une telle répulsion que l'objet printer. De par le fait de sa simplicité, et parce que l'impression est véritablement propre à chaque application, on ne trouve que très peu d'exemple d'utilisation de cet objet. Dans l'imaginaire, l'objet printer évoque des codes interminables, impossible à maintenir et demandant des heures de mises au point. Ceci est tellement vrai que l'on rencontre beaucoup de contrôles payants, souvent assez chers, censés simplifier la tâche. Quelques utilisateurs ont contourné le problème en détournant le DataReport pour réaliser leurs impressions.
Je vais tenter de démontrer dans cet article, que la réputation sulfureuse de cet objet est infondée et qu'il s'agit d'un objet puissant, aisé à utiliser, permettant de réaliser des impressions de grandes qualités.

J'imprime vers…

Avant de pouvoir imprimer, il va falloir utiliser un outil pour configurer l'imprimante et l'impression. Bien que les méthodes soient nombreuses, je préconise d'utiliser la boite de dialogue d'impression des CommonDialogs. Elle est simple à utiliser, et couvre largement les besoins classiques. A noter toutefois qu'il faut utiliser la propriété Flags pour la configurer au mieux.
Quelques points à noter toutefois :

- Il faut que la propriété **PrinterDefault soit vraie** pour que le choix de l'utilisateur se répercute correctement.
- Changer l'imprimante par défaut, n'est valide que pour la durée du programme.
- On valorise la propriétés Flags=&H40& pour appeler la boite de configuration de l'imprimante.

Comme il existe beaucoup d'exemples pour les CommonDialogs, je n'irai pas plus loin.

L'objet printer à terre

En lui-même, cet objet est très simple. Il est membre de la collection Printers, qui représente l'ensemble des imprimantes disponibles sur le système.
Regardons d'abord ses propriétés en les rangeant en deux catégories. Celles que l'on utilise en configuration, et celles dont on se sert au cours de l'impression. La notation (*RcdImp*) signifie qu'on valorise la propriété en général lors de la récupération de la boite de dialogue impression. Cette liste n'est pas exhaustive.
N.B : Dans cet article nous utiliserons le terme **méthode graphique** pour toutes les impressions qui ne sont pas du texte. (Ex : Circle, Line, PaintPicture…).

Configuration

- ColorMode : Défini si une imprimante **couleur** imprime ou non en couleur.
- Copies (*RcdImp*) : Nombre de copies
- Duplex : Impression recto verso
- Height, Width : Servent à connaître les dimensions de la feuille. Utilisez les seulement en lecture, afin de ne pas modifier la propriété PaperSize implicitement, ce qui risquerait de provoquer une erreur.
- Orientation (*RcdImp*) : Portrait ou paysage.
- PaperSize : Taille du papier. Dans cet article, tout sera construit autour du format A4.
- PrintQuality : Qualité de l'impression, va de brouillon à haute.
- ScaleMode : Unité de l'échelle. Par habitude j'utilise toujours 6 – Millimeter. Cette propriété est très importante, car elle va permettre de contrôler la position exacte de l'impression. Il est donc fortement conseillé de la définir au début et de ne plus la modifier.
- ScaleLeft, ScaleTop, ScaleHeight, ScaleWidth, sont des propriétés utilisées pour connaître la zone imprimable, comme nous le verrons dans les exemples.

Impression

- ➔ DrawMode, DrawStyle, DrawWidth sont des propriétés de traçage des méthodes graphiques telles que l'épaisseur du trait, pointillés etc…
- ➔ FillColor et FillStyle sont des propriétés de remplissage des méthodes graphiques.
- ➔ Font, FontBold, FontItalic, FontStrikethru, FontUnderline, FontSize, FontName sont les propriétés de police du texte.
- ➔ CurrentX, CurrentY sont les propriétés de position d'impression. Elles définissent le **coin supérieur gauche** du début de l'impression. Elles sont fondamentales à l'écriture d'un gestionnaire d'impression réussie, mais j'y reviendrai largement dans la deuxième partie. CurrentX augmente de la gauche vers la droite, et CurrentY du haut vers le bas.

Nous allons voir maintenant les méthodes, celles ci s'utilisent toujours au cours de l'impression.

➔ *printer*.**Circle** [**Step**] **(***x, y***),** *radius,* [*color, start, end, aspect*]
Trace un cercle, un arc de cercle ou une ellipse. Le mot clé Step précise si les coordonnées du centre sont relatives (à la position currentX, currentY) ou absolu ; *color* permet de forcer une couleur différente de celle définit dans *FillColor* ; *start, end* servent pour les arcs, et *aspect* pour définir s'il s'agit d'un cercle ou d'une ellipse.
➔ *printer*.**Line** [**Step**] **(***x1***,** y*1***)** [**Step**] - **(***x2***,** *y2***),** [*color*]**,** [**B**][**F**]
Trace une ligne ou un rectangle. Le premier mot clé Step définit si les coordonnées **(***x1***,** y*1***)** sont relatives au point (currentX, currentY) le deuxième si les coordonnées **(***x2***,** *y2***)** sont relatives au point **(***x1***,** y*1***).** [**B**] annonce le tracé d'un rectangle en fonction des coordonnées. [**F**] annonce que ce rectangle est rempli par la même couleur que le contour. S'il est omis, le rectangle est rempli selon les valeurs des propriétés FillStyle et FillMode.
N.B : Les propriétés CurrentX et currentY prennent les coordonnées **(***x2***,** *y2***)** après le tracé de la ligne.
➔ *printer*.**PSet** [**Step**] **(***x***,** *y***),** [*color*]
Trace un point dont la taille est définit par la propriété DrawMode.
➔ *printer*.**PaintPicture** *picture, x, y*
Imprime un graphique. *Picture* doit correspondre à la propriété picture d'un PictureBox, d'une form ou d'un objet particulier. (Nous verrons cela plus loin).

- ➔ TextHeight et TextWidth permettent de déterminer la hauteur ou la largeur dans le système d'unité ScaleMode. Elle attend la chaîne à imprimer en paramètre.
- ➔ EndDoc met fin au travail d'impression et envoie le document vers le périphérique.
- ➔ KillDoc annule l'impression en cours, rien n'est envoyé au périphérique.
- ➔ NewPage avance jusqu'à la page suivante de l'imprimante et redéfinit la position d'impression dans **l'angle supérieur gauche** de la nouvelle page, elle incrémente de 1 la propriété **Page**. **Attention** après l'appel de méthode, il n'est plus possible de modifier ou de revenir sur la page précédente.
- ➔ Scale, ScaleX et ScaleY permettent de redéfinir le système d'échelle.

Et enfin la méthode **Print**. Elle est bizarrement documentée. Pour trouver comment elle fonctionne, il faut rechercher "printer.print" dans l'aide car vous ne la trouverez pas dans les méthodes de l'objet printer(??). Pour vous évitez cette recherche son modèle est le suivant :
[*object*.]**Print** [*outputlist*] [{ ; | , }]
outputlist contient donc une liste de chaînes séparées par des' ,' ou des ';' .
Sachez enfin qu'elle provoque une erreur si elle est placée dans un bloc With.
L'objet printer ne gère pas d'événement.
Comme vous le voyez, il n'y a pas beaucoup de propriétés et de méthodes, et pourtant nous allons réaliser des impressions complètes avec uniquement cet objet.

Le système de coordonnées.

Il faut connaître quelques astuces pour ne pas se faire berner par l'objet printer.

Les marges

Tout d'abord, une imprimante ne peut pas imprimer partout sur la feuille. Elle possède un système de marge propre à chaque imprimante. Ceci fait que le point de coordonnées (0,0) de l'imprimante n'est pas celui de la feuille. Pour déterminer par exemple la coordonnée absolue du point(0,0) il faut faire :
Xdemar=(Printer.Width-Printer.ScaleWidth)/2
Ydemar=(Printer.Heigth-Printer.ScaleHeigth)/2
On a toujours intérêt à définir une marge supérieure à la "marge imprimante". Par habitude j'utilise toujours dans mes impressions portrait des marges gauche/droite égales à 20 (en mm) et des marges haute/basse égales à 15.

Donc dans mon programme on trouvera toujours le code suivant :
```
Printer.TrackDefault = True
Printer.ScaleMode = 6
Printer.Orientation = 1
Printer.PaperSize = 9
Printer.PrintQuality = -4
Printer.ColorMode = 1
DecalX =20-(Printer.Width-Printer.ScaleWidth)/2
DecalY =15-(Printer.Heigth-Printer.ScaleHeigth)/2
```

L'impression d'une chaîne

Ce qui m'a le plus perturbé, la première fois que j'ai écrit un gestionnaire d'impression, c'est la façon dont fonctionne les propriétés currentX et currentY. En effet, globalement cela fonctionne comme une machine à écrire. L'impression d'une ligne ou d'un cadre donne à ces propriétés les valeurs de leurs points de fin **(***x2***,** *y2***).**
L'impression d'une chaîne dépend de la syntaxe. Pour faire simple je vais utiliser un exemple :
Si j'écris `Printer.Print MaChaine` la propriété currentX prend la valeur 0, et currentY augmente de la valeur TextHeight du texte plus 0.1 mm, bref c'est un retour à la ligne. Ce qui nous montre aussi que la valeur TextHeight n'est pas stricto sensu la hauteur du texte, mais la hauteur du texte plus un interligne.
Par contre si j'écris `Printer.Print MaChaine;` la propriété currentX augmente de la valeur TextWidth, et CurrentY reste identique. Ceci fait que le simple fait d'encadrer un texte juste après l'avoir écris ressemble rapidement à un travail de polytechnicien.

Le changement de page

L'impression d'une chaîne en dehors des marges intrinsèques de l'objet printer déclenche l'appel de la méthode NewPage, et tout ou partie de la chaîne sera imprimée sur une nouvelle page. Par contre, pour les méthodes graphiques, tout ce qui sortira des marges pourra être considéré comme définitivement perdu. Nous verrons plus loin comment éviter ces inconvénients.

La valorisation des propriétés

Les méthodes graphiques ainsi que la méthode Print utilise les valeurs des propriétés pour définir leurs impressions. Ainsi si dans mon code je place `Printer.FontSize=15` chaque appel suivant de la méthode Print imprimera la chaîne en police de taille 15. Si ceci se voit vite lors du test d'impression, il peut y avoir des effets pervers si on cherche à connaître les dimensions de la chaîne avant de changer la taille de la police. Par exemple le code suivant est différent selon que je mets ou non en commentaire la ligne `'Printer.FontBold = False` :
```
Printer.FontBold = True
Printer.FontSize = 14
Printer.CurrentX = 10
Printer.CurrentY = 10
Printer.Print "Titre"
Printer.FontSize = 12
'Printer.FontBold = False
Machaine = "Sous-Titre"
Largeur = Printer.TextWidth(Machaine)
```
Voilà, nous avons passé en revue les méthodes et propriétés de l'objet Printer, ainsi que les fonctionnements dont il faudra se méfier. Nous allons donc aborder dans cette seconde partie les façons de programmer efficacement cet objet

L'imprime à l'emploi

Remarque : n'oubliez pas de placer vos feuilles et PictureBox dans le même système d'unités que votre imprimante, afin de pas compliquer à loisir les conversions d'échelle.

Utilisation générale

Le centrage

Pour centrer le texte sur la feuille ou dans un cadre, il suffit d'une simple opération mathématique. Par exemple pour centrer sur la feuille horizontalement
```
Printer.CurrentX=Printer.Width/2 - Printer.TextWidth(Machaine)/2
```

Encadrer un texte

Comme je l'ai dit plus haut, il vaut mieux éviter d'encadrer à la volée, mais pour les passionnés, sachez que la syntaxe est la suivante (pour un cadre situé à 1.5 mm autour du texte) :
machaine = "Essai d'encadrement"
Printer.Print machaine;
Printer.Line Step(-Picture1.TextWidth(machaine) - 1.5, -1.5)-Step(Picture1.TextWidth(machaine) + 3, Printer.TextHeight(machaine) + 3), , B
Vous noterez juste l'emploi des deux mots clés Step afin de travailler en coordonnées relatives.

Dessiner une case cochée

Là c'est très simple si on utilise pleinement le "dynamisme" des propriétés CurrentX et CurrentY :
Printer.Line -Step(4, 4), , B
Printer.Line -Step(-4, -4)
Printer.Line Step(4, 0)-Step(-4, 4)
Trace une case cochée de 4 mm de coté.

Insérer une image

Le plus simple est que cette image se trouve dans un contrôle image (pas forcément visible d'ailleurs) sur la feuille :
Printer.PaintPicture Image1.Picture, PositionX , PositionY
A noter que la position doit être en coordonnées **absolues**.

L'impression géométrique

Celle-ci est la plus simple. C'est une version un peu plus élaborée de la méthode PrintForm. Elle convient particulièrement pour les impressions d'un formulaire sur une page.
L'idée générale est de placer sur sa feuille, l'ensemble des contrôles qui devront être imprimés, dans une zone virtuelle (ou un picturebox). Eventuellement, pour gagner en qualité, on donnera un rapport hauteur/largeur à cette zone sensiblement égale à celui d'une page A4 dans le sens de l'orientation désiré (le rapport hauter/largeur d'une feuille A4 en mode portrait est de $\sqrt{2}$)
On calcule dans le code les coefficients d'expansions de cette zone vers la zone imprimable (et non vers la feuille) en X et en Y, par exemple :
FacX=(Printer.Width – 2*MargeX)/LargeurZone
FacY=(Printer.Heigth – 2*MargeY)/HauteurZone
On utilisera le plus petit de ces facteurs pour l'appliquer à la police.
Après cela on va parcourir la collection Controls de la feuille, tester la position ou le container, puis le type de contrôle afin de définir les actions.
Remarque : Placer ces contrôles dans un PictureBox simplifie la mise en place des contrôles sur la feuille, n'oubliez pas de mettre sa propriété ScaleMode égale à celle de l'imprimante. N'utilisez surtout pas de Frame, l'ensemble des coordonnées et dimensions de vos contrôles repasserait en twips.
Cette technique convient mieux à une impression en mode paysage, qui est le mode écran.
Exemple :
Ma zone imprimable en paysage fait 257*180. Je place mes contrôles à imprimer dans un PictureBox. J'ai besoin qu'il est une largeur de 200, je lui donne une hauteur de 140. Bien sur, je respecte la proportion dans ce cas ce qui me permettra de n'utiliser qu'un seul facteur.
Dans cet exemple je complique pour montrer la conversion, car vous n'êtes pas sans remarquer que mon PictureBox tiendrait sur la feuille avec un facteur 1.

Dans ma feuille je déclare une constante de conversion
Private Const FacC! = 1.285 'facteur de conversion
Et la fonction Impression sera
Private Sub Impression()
Dim DecalY as single, DecalX as Single, ContImprim as Control

```
Printer.ScaleMode = 6
Printer.Orientation = 2
Printer.PaperSize = 9
Printer.PrintQuality = -4
Printer.ColorMode = 1
DecalX = 20 - (Printer.Width - Printer.ScaleWidth) / 2
DecalY = 15 - (Printer.Heigth - Printer.ScaleHeigth) / 2
'utilisez le décalage pour que le point(0,0) du PictureBox soit le point (0,0) de la zone imprimable
For Each ContImprim In Me.Controls
  If TypeOf ContImprim Is Label Then
    With Printer
      .FontName = ContImprim.FontName
      .FontBold = ContImprim.FontBold
      .FontSize = Int(FacEch * ContImprim.FontSize)
      .CurrentX = ContImprim.Left * FacEch
      .CurrentY = ContImprim.Top * FacEch
    End With
    Printer.Print ContImprim.Caption
  ElseIf TypeOf ContImprim Is TextBox Then
    With Printer
      .FontName = ContImprim.FontName
      .FontBold = ContImprim.FontBold
      .FontSize = Int(FacEch * ContImprim.FontSize)
      .CurrentX = ContImprim.Left * FacEch
      .CurrentY = ContImprim.Top * FacEch
    End With
    Printer.Print ContImprim.Text
  ElseIf TypeOf ContImprim Is CheckBox Then
    Printer.Line (ContImprim.Left * FacEch, ContImprim.Top * FacEch)-Step(7, 7), , B
    If ContImprim.Value = 1 Then
      Printer.Line (ContImprim.Left * FacEch, ContImprim.Top * FacEch)-Step(7, 7)
      Printer.Line ((ContImprim.Left * FacEch) + 7, ContImprim.Top * FacEch)-Step(-7, 7)
    End If
    With Printer
      .FontName = ContImprim.FontName
      .FontBold = ContImprim.FontBold
      .FontSize = Int(ContImprim.FontSize * FacEch)
      .CurrentX = ContImprim.Left * FacEch + 10
      .CurrentY = ContImprim.Top * FacEch + 2
    End With
    Printer.Print ContImprim.Caption
  ElseIf TypeOf ContImprim Is MSFlexGrid Then
    With Printer
      .CurrentX = ContImprim.Left * FacEch
      .CurrentY = ContImprim.Top * FacEch
    End With
    Printer.PaintPicture ContImprim.Picture, Printer.CurrentX, Printer.CurrentY
  End If
Next
```

End sub
Bien sûr, il s'agit d'un exemple pour quelques types de contrôles, mais le principe peut facilement s'étendre. Pour les polices, n'oubliez pas de prendre des polices TrueType et arrondissez le résultat.

Impression d'un ListBox.

Nous allons prendre le cas d'une liste à sélection unique, peu importe si tous ses éléments sont visibles.

```
Dim RappL As Single, AligH As Single, MaChaine As String, LigSel As Integer, compteur As Long
AligH = Printer.CurrentX
For compteur = 0 To List1.ListCount - 1
   If List1.Selected(compteur) Then LigSel = compteur
   MaChaine = List1.List(compteur)
   RappL =Iif (Printer.TextWidth(MaChaine) > RappL, Printer.TextWidth(MaChaine), RappL)
   Printer.Print MaChaine
   Printer.CurrentX = AligH
Next compteur
'encadrement de la liste
Printer.Line Step(-1, 0)-Step(RappL + 2, -1 * Printer.TextHeight(String$(List1.ListCount - 1, vbCrLf))), , B
'sur lignage de la sélection
Printer.Line Step(-1 * RappL - 2, Printer.TextHeight(String$(LigSel, vbCrLf)))-Step(RappL + 2, -1 *
Printer.TextHeight("")), QBColor(8), B
```

Impression de longues chaînes

En général, les chaînes sont coupées dans les contrôles par VbCrLf et l'impression se passe bien. Mais par exemples, lors de l'impression d'une chaîne qui n'apparaît pas sur le formulaire, ou dans un TextBox multilignes, ce caractère n'est pas présent dans la chaîne. L'imprimante risque alors de déborder du cadre ou de la feuille. On utilise alors la fonction suivante pour découper la chaîne.
N.B : Cette fonction gère le cas de très longues chaînes, elle peut être optimisée si l'on est sûr de ne jamais rencontrer de chaîne de plus de 3 ou 4 lignes.

```
Private Function DecoupeChaine(ByVal TextLong As String, ByVal TailleMax As Single) As String()
Dim TabSplit() As String, TabRetour() As String, compteur1 As Long, compteur2 As Long
Dim TabMot() As String, ChaineInter As String
   TabSplit = Split(TextLong, vbCrLf)
   ReDim TabRetour(0 To 0)
   For compteur1 = 0 To UBound(TabSplit)
      If Printer.TextWidth(TabSplit(compteur1)) > TailleMax Then
         TabMot = Split(TabSplit(compteur1), " ")
         For compteur2 = 0 To UBound(TabMot)
            If Printer.TextWidth(ChaineInter & " " & TabMot(compteur2)) > TailleMax Then
               TabRetour(UBound(TabRetour)) = ChaineInter
               ChaineInter = TabMot(compteur2)
               ReDim Preserve TabRetour(0 To UBound(TabRetour) + 1)
            Else
               ChaineInter = IIf(Len(ChaineInter) > 0, ChaineInter & " " & TabMot(compteur2),
TabMot(compteur2))
            End If
         Next compteur2
         If Len(ChaineInter) > 0 Then
            TabRetour(UBound(TabRetour)) = ChaineInter
            ChaineInter = ""
            ReDim Preserve TabRetour(0 To UBound(TabRetour) + 1)
         End If
      Else
         TabRetour(UBound(TabRetour)) = TabSplit(compteur1)
         ReDim Preserve TabRetour(0 To UBound(TabRetour) + 1)
      End If
   Next compteur1
   ReDim Preserve TabRetour(0 To UBound(TabRetour) - 1)
   DecoupeChaine = TabRetour
End Function
```

De manière générale il est bon de traiter ces chaînes de façon à ne pas laisser dedans de caractères "VbCrLf" ce qui ramènerait currentX contre la marge de gauche, sauf si c'est évidemment le but rechercher.

Imprimer un recordset ou une grille

Je regroupe ces deux cas car ils sont identiques. En effet un recordset peut être imaginer comme une grille de colonne champs et de ligne enregistrement. J'ai mis dans cet exemple deux modes d'encadrement différents, avant ou après l'écriture de la chaîne.

```
Private Sub cmdPrint_Click()
   Dim TabLong() As Single, compteur As Long, Total As Single
   Dim NomChamp As String, ReqLong As New ADODB.Recordset, strSQL As String
   Dim XFixe As Single, YMobile As Single, prnstring As String, ConnTemp As New ADODB.Connection
   ConnTemp.ConnectionString = Adodc1.ConnectionString
   ConnTemp.Open
   With Printer
      .TrackDefault = True
      .ScaleMode = 6
      .Orientation = 1
      .PaperSize = 9
      .PrintQuality = -4
      .ColorMode = 1
   End With
   XFixe = 20 - (210 - Printer.ScaleWidth) / 2
   YMobile = 15 - (297 - Printer.ScaleHeight) / 2
   ReDim TabLong(ReqGrille.Fields.Count - 1)
   With ReqGrille
      For compteur = 0 To .Fields.Count - 1
         NomChamp = .Fields(compteur).Name
         strSQL = "SELECT MAX(len([" & NomChamp & "])) AS LeMax FROM LaTable"
         ReqLong.Open strSQL, ConnTemp, adOpenStatic, adLockReadOnly
         ReqLong.MoveFirst
         TabLong(compteur) = IIf(ReqLong!lemax > Len(NomChamp), ReqLong!lemax, Len(NomChamp))
         ReqLong.Close
         TabLong(compteur) = Printer.TextWidth(String(TabLong(compteur), "X")) + 4
         Total = Total + TabLong(compteur)
      Next compteur
      .MoveFirst
      Total = XFixe
      For compteur = 0 To .Fields.Count - 1
         Printer.Line (Total, YMobile)-Step(TabLong(compteur), 8), QBColor(6), BF
         prnstring = CStr(.Fields(compteur).Name)
         Printer.CurrentX = Total + TabLong(compteur) / 2 - Printer.TextWidth(prnstring) / 2
         Printer.CurrentY = (YMobile + 4) - Printer.TextHeight(prnstring) / 2
         Printer.Print prnstring
         Total = Total + TabLong(compteur)
      Next compteur
      YMobile = YMobile + 8
      While Not .EOF
         Total = XFixe
         For compteur = 0 To .Fields.Count - 1
            prnstring = CStr(.Fields(compteur).Value)
            Printer.CurrentX = Total + TabLong(compteur) / 2 - Printer.TextWidth(prnstring) / 2
            Printer.CurrentY = (YMobile + 4) - Printer.TextHeight(prnstring) / 2
            Printer.Print prnstring
            Total = Total + TabLong(compteur)
            Printer.Line (XFixe, YMobile)-Step(Total - XFixe, 8), , B
         Next compteur
         .MoveNext
         YMobile = YMobile + 8
      Wend
   End With
   Printer.EndDoc
End Sub
```

L'impression sur plusieurs pages

Le cas se pose en général, pour l'impression de longues listes, de grands tableaux. J'envisage ici le cas d'une impression continue. En effet, l'impression de plusieurs fiches liées à un recordset par exemple n'est jamais qu'un cas d'impression de page unique répété à chaque changement de ligne du recordset.
Mais supposons pour reprendre le cas précédent, que je souhaite imprimer ma grille sans connaître au préalable son nombre de lignes. Je vais prendre un exemple que j'ai développé, il y a quelques années pour une association qui organisait un concours de pêche.
Le travail était simple, tout tenait dans une table. Il fallait pouvoir imprimer la liste des participants, puis le classement.
Sur ma page devait apparaître le logo de l'association, le titre, la date, le tableau et en bas le numéro de page.
Donc sur mon formulaire il y avait un contrôle image, deux labels et un MSFlexGrid (plus un dessin de poisson mais lui il n'y avait pas besoin de l'imprimer).
Je me suis basé sur les marges habituelles, il me fallait trois centimètres en haut pour le logo, et un centimètre en bas pour le numéro de page. Chaque ligne devait faire 8 mm de haut.
L'impression était en paysage afin de pouvoir visualiser toutes les colonnes.
Afin de connaître le nombre de page, pour pouvoir valoriser les propriétés FromPage et ToPage de la boite de dialogue d'impression, j'ai procédé comme suit :
Ma hauteur imprimable pour la grille était de 140 (210-2*15-30-10). Il rentrait donc 17 lignes de 8 mm par page soit 16 enregistrements plus la ligne de titre.
Ce préambule n'a pas pour but de faire de vous des experts en concours de pêche mais bien de vous montrer comment je me suis posé le problème. Allons-y!
Le code est le même que celui fait précédemment, sauf la différence d'orientation, dans la fonction ci dessous je mettrai juste les lignes différentes.

```
Private Sub Impression()
Printer.Orientation = 2
XFixe = 20 - (297 - Printer.ScaleWidth) / 2
YMobile = 45- (210 - Printer.ScaleHeight) / 2
NbLigne= ReqList.Recordcount
NbPage=Iif(NbLigne Mod 16 >0, NbLigne \ 16+1, NbLigne \ 16)

A la place de l'impression de la ligne de titre
Call EntetePied(TabLong)
YMobile = YMobile + 8

Et dans la boucle du recordset un compteur qui :
If Compt16=16 then
        Compt16=0
        Printer.NewPage
        Call EntetePied
        YMobile = 53 - (210 - Printer.ScaleHeight) / 2 '53 car 45+8 de la première ligne
End if
End Sub

Private Sub EntetePied(TabTaille() as Single)
Dim Xfixe!, Yfixe!, Total&, compteur&

XFixe = 20 - (297 - Printer.ScaleWidth) / 2
YFixe = 15 - (210 - Printer.ScaleHeight) / 2
YfirstLigne = 45 - (210 - Printer.ScaleHeight) / 2
Printer.PaintPicture imgLogo.Picture, XFixe, YFixe
Printer.CurrentX=Printer.Width/2 - Printer.TextWidth(Titre)/2
Printer.CurrentY=Yfixe+15- Printer.TextHeigth(Titre)/2
Printer.Print Titre
Printer.CurrentX=Printer.Width-20 - Printer.TextWidth(LaDate)
Printer.CurrentY=Yfixe+15- Printer.TextHeigth(LaDate)/2
Printer.Print LaDate
'Insère la ligne de titre
```

```
Total = XFixe
      For compteur = 0 To .Fields.Count - 1
         Printer.Line (Total, YMobile)-Step(TabTaille(compteur), 8), QBColor(6), BF
         prnstring = CStr(ReqGrille.Fields(compteur).Name)
         Printer.CurrentX = Total + TabTaille(compteur) / 2 - Printer.TextWidth(prnstring) / 2
         Printer.CurrentY = (YMobile + 4) - Printer.TextHeight(prnstring) / 2
         Printer.Print prnstring
         Total = Total + TabTaille(compteur)
      Next compteur
'Met Le pied De page
strPied="Page : " & Printer.Page & " de " & NbPage
Printer.CurrentX=Printer.Width/2 - Printer.TextWidth(strPied)/2
Printer.CurrentY=Yfixe+ Printer.ScaleHeight -20- Printer.TextHeigth(strPied)/2
Printer.Print strPied

End sub
```

Vous pouvez noter que l'on peut très bien écrire le pied de page avant le corps du texte.

Prévisualisation

Bien sûr, vous vous voyez déjà poussant des wagonnets de feuilles vers l'imprimante afin de réaliser vos tests d'impression. Afin de sauver les forêts, je vais vous montrer comment fabriquer une feuille de visualisation.
Dans cet exemple, je reste dans un cas d'impression paysage, mais on peut faire un outil beaucoup plus complexe en gérant dynamiquement l'orientation de la feuille, faire une réduction de taille etc…
Tous les scalemode de la feuille et des picturebox sont en millimètres
Dans ma feuille je mets les contrôles suivant :
Un commandButton caption Fermer
Un PictureBox

AutoRedraw	True
Name	PctCont
Height	109.5
Width	153

Dans ce PictureBox, un HscrollBar sur toute la largeur

Name	HScroll1
LargeChange	148
SmallChange	15

Un VscrollBar sur toute la hauteur

Name	VScroll1
LargeChange	105
SmallChange	10

Et enfin un autre pictureBox (donc contenu dans pctCont)

AutoRedraw	True
Appearance	Flat
Name	PctPict
Height	210
Index	1
Left	0
Top	0
Width	153

Dans le code de cette feuille

```
Private Sub HScroll1_Change()
   pctCible(1).Left = HScroll1.Value * -1
End Sub

Private Sub VScroll1_Change()
   pctCible(1).Top = VScroll1.Value * -1
End Sub
```

L'astuce consiste à se dire qu'à peu de choses près l'objet printer et le PictureBox fonctionnent de la même façon. Donc on écrit une fonction ou dans le code on remplacera partout Printer par Cible. Pour les méthodes différentes pour les deux objets on fera un test avec la fonction Is

```
Public sub Impression (Cible as object)

        If Cible Is Printer Then
                With Printer
                        .TrackDefault = True
                        .ScaleMode = 6
                        .Orientation = 1
                        .PaperSize = 9
                        .PrintQuality = -4
                        .ColorMode = 1
                End With
        End if
        '
        '
        '
        If Cible Is Printer then
                Printer.EndDoc
        Else
                frmView.Show
        End If
End Sub
```

Le seul cas particulier étant pour NewPage

```
If Cible Is Printer then
        Printer.NewPage
Else
        Cible.Visible=False
        Load frmView.pctPict(Count+1)
        Set Cible= frmView.pctPict(Count)
        Cible.Visible=True
End if
```

Dans le cas d'une impression multi page, il convient de modifier le code d'évènements des scrollbars.
L'appel de la fonction se fait de la manière suivante

```
Call Impression(frmView.pctPict.Picture)
```

Conclusion

Il y aurait encore de nombreuses astuces possibles à décrire. Le véritable problème de l'impression est de bien analyser ce que l'on veut obtenir. L'impression est rarement anarchique, et en imaginant un modèle géométrique simple, on réalise un code d'impression simple.
J'espère avoir fait changer d'avis les plus récalcitrant d'entre vous.