ACCESS : Tutoriel d'utilisation simple de la classe ClImage

par Thierry GASPERMENT (arkham46.developpez.com/)

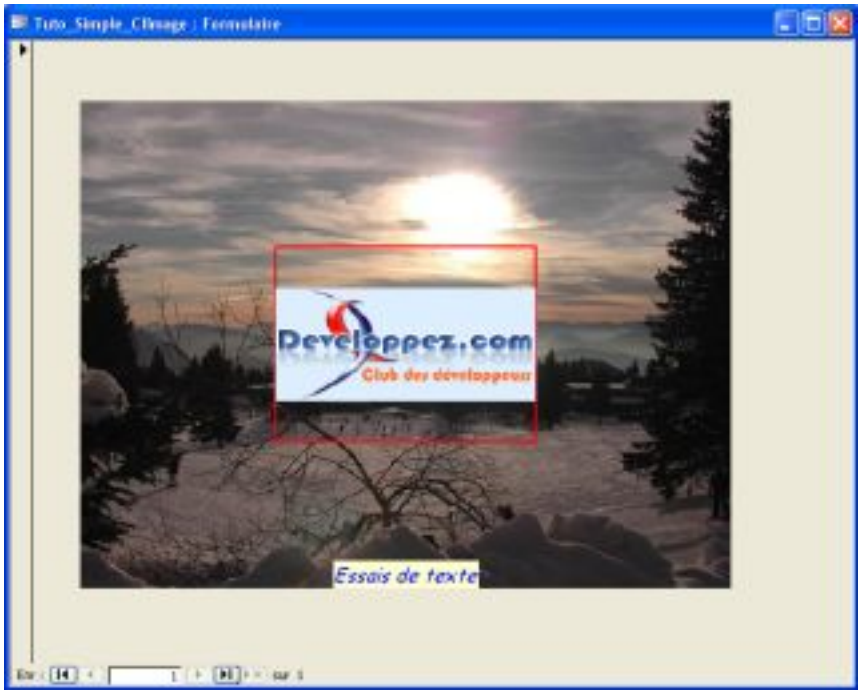
Date de publication : 22/01/06

Dernière mise à jour : 25/07/07

Ce document a pour objectif d'expliquer les bases de l'utilisation de la classe ClImage.
La classe ClImage apporte des fonctions pour dessiner sur un contrôle image standard et permet de rendre l'image interactive par l'ajout de régions sensibles aux actions de la souris.
Attention : la classe clImage n'est plus maintenue.
Pour gérer les images avec gdi, utilisez la nouvelle classe **clGdi32**.

I - Introduction

L'objectif est de proposer un exemple simple d'utilisation de la classe.
On va :
- charger une image de fond
- dessiner du texte
- dessiner un rectangle
- dessiner une image dans ce rectangle par dessus l'image de fond
- détecter le survol du rectangle
- détecter le click sur le rectangle



Voici le résultat que l'on obtiendra à la fin du tutoriel

Consultez la documentation des fonctions et des propriétés

II - Création du formulaire et du contrôle image

Créez un formulaire et placez-y un **contrôle image** de n'importe quelle taille.
Il n'est pas utile d'intégrer une image dans le contrôle car on va dessiner dessus.
Choissisez une image quelconque pour créer le contrôle puis dans les propriétés du contrôle dans l'onglet **Format**, effacez la propriété **Image**.
Définissez le **mode d'affichage à Zoom**. L'image sera redimensionnée en conservant ses proportions.
Vérifiez le **nom du contrôle** dans l'onglet **Autres**, changez le si-besoin en **Image0** (c'est le nom qu'on va utiliser dans ce tutoriel)

III - Création de la classe

Téléchargez la classe au format texte *(HTTP)*

Créez un **nouveau module de classe**.
Collez-y le contenu du fichier texte.
Sauvegardez le module avec le nom **CIImage**.
Attention : le nom sous lequel vous sauvegardez le module est important.

IV - Déclaration de la classe et initialisation

On va écrire notre code dans le module du formulaire. Cliquez sur **Affichage** --> **Code** pour ouvrir ce module.
Vérifiez que vous avez l'instruction **Option Explicit** en haut du module.
Sinon rajoutez le pour imposer la déclaration de toutes les variables, cela évite les étourderies.

En-tête de module

```
Option Compare Database
Option Explicit
```

Pour pouvoir utiliser la classe il est nécessaire de la déclarer.
La classe est un objet dont le type est le nom sous lequel on a sauvegardé notre module de classe.
Elle se déclare comme n'importe quelle autre variable.
Comme c'est un objet on va utiliser le mot-clé **New** dans la déclaration, ainsi la classe sera initialisée automatiquement à la première utilisation.

Déclaration de la classe

```
Private ClImg As New ClImage
```

Première chose à faire : pensez à libérer la classe à la fermeture du formulaire.
La libération de la classe est importante car elle supprime tous les objets graphiques de la mémoire.
Dans les propriétés du formulaire, définissez **[Procédure événementielle]** dans l'événement **Sur Fermeture**.
Cliquez sur les trois petits points **[...]** pour générer l'événement dans le code.
A l'intérieur de la procédure **Form_Close** on va **libérer la classe** : il suffit de lui donner la valeur **Nothing** si elle n'a pas déjà cette valeur.

Libération de la classe

```
Private Sub Form_Close()
' Libération de la classe à la fermeture du formulaire
If Not ClImg is Nothing Then Set ClImg = Nothing
End Sub
```

Ensuite il faut **initialiser le contrôle image**. C'est à dire qu'on va préciser à la classe sur quel contrôle on veut dessiner.
On utilise la fonction **SetImgCtrl** en passant le contrôle image en paramètre.
On écrit ce code dans l'événement **Sur Chargement** du formulaire.

Initialisation du contrôle

```
Private Sub Form_Load()
' Initialisation du contrôle image
ClImg.SetImgCtrl Me.Image0
End Sub
```

A ce niveau on a fait tout ce qui était nécessaire et on peut maintenant commencer à dessiner.
Ce code est le strict minimum et est nécessaire dans tous les cas d'utilisation de la classe.

Code commum et nécessaire à toute utilisation de la classe

```
Option Compare Database
Option Explicit

' Déclaration de la classe
Private ClImg As New ClImage

Private Sub Form_Close()
' Libération de la classe à la fermeture du formulaire
If Not ClImg is Nothing Then Set ClImg = Nothing
```

Code commum et nécessaire à toute utilisation de la classe

```
End Sub

Private Sub Form_Load()
' Initialisation du contrôle image
ClImg.SetImgCtrl Me.Image0
End Sub
```

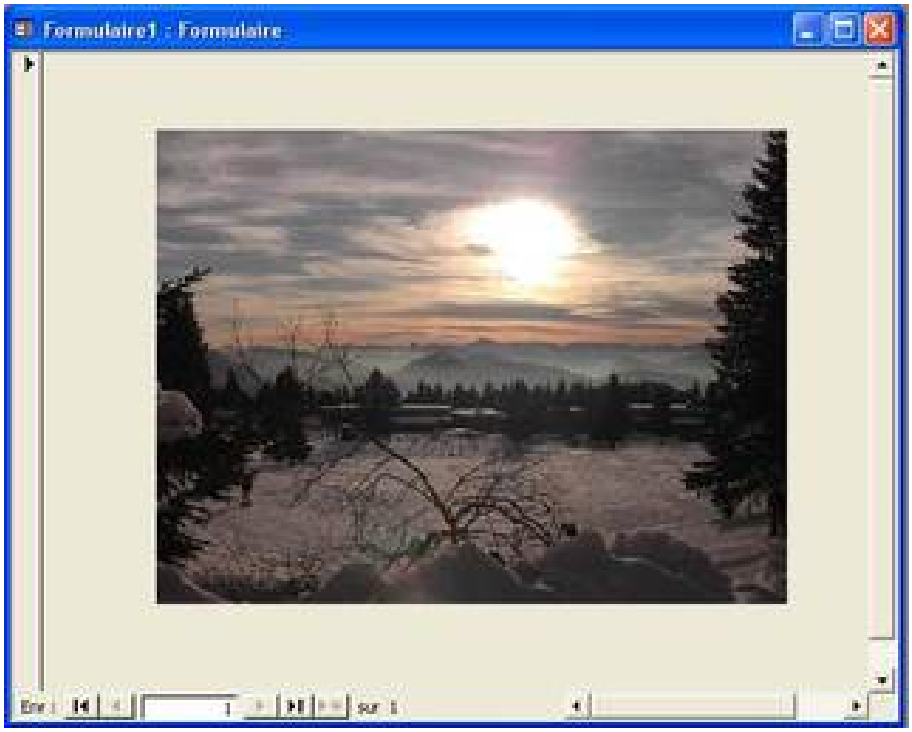
V - Chargement d'une image de fond

Si vous visualisez le formulaire vous remarquez que l'image est noire, alors qu'avant elle était transparente.
C'est parce que la fonction **SetImgCtrl** a créé une nouvelle image dont les points ont tous une couleur égale à 0 (couleur noire).
On va maintenant charger une image dans le contrôle.
Placez une image dans le même répertoire que votre base de données.
Mon image s'appelle *DSCN1099.JPG*, remplacez ce nom de fichier par le vôtre.
On utilise la propriété **ApplicationPath** pour récupérer le nom du répertoire de l'application.
(La propriété **ApplicationPath** est identique à **CurrentProject.Path** sur les versions à partir de Access 2000.)
La fonction utilisée pour charger le fichier est **LoadFromFile**; on précise la largeur de l'image à charger pour réduire la mémoire utilisée.

Chargement d'une image de fond

```
Private Sub Form_Load()
' Initialisation du contrôle image
ClImg.SetImgCtrl Me.Image0
' Chargement d'une image de fond
' On précise la largeur de l'image car il n'est pas nécessaire de charger une image plus large que le contrôle
' La hauteur sera automatiquement calculée en fonction de la largeur précisée et des proportions de l'image
ClImg.LoadFromfile ClImg.ApplicationPath & "DSCN1099.JPG", Me.Image0.Width
End Sub
```

On obtient alors l'image dans le formulaire.



Rien d'extraordinaire pour l'instant, si ce n'est une nette réduction de la mémoire utilisée si vous chargez une photo de vos vacances prise en 2048*1536 dans un contrôle 10 fois plus petit.

VI - Dessin du texte

On va maintenant dessiner du texte sur l'image.
Commençons par **définir la police de caractères**.
A la suite du code précédent on défini la nouvelle police :
- Taille : 24
- Angle : 0 (texte horizontal classique)
- Epaisseur : 400 (normal)
- Italique : Oui
- Souligné : Non
- Barré : Non
- Police : Comic sans MS (parce qu'elle rend bien...)

Définition de la police de caractères

```
ClImg.DrawNewFont 24, 0, 400, True, False, False, "Comic sans MS"
```

Puis **on dessine le texte** "Essais de texte".
- On passe 4 coordonnées en paramètres, ce sont les coordonnées d'un rectangle dans lequel on va dessiner le texte.

Les propriétés utilisées ici (**ImgX1**,**ImgY1**,**ImgX2**,**ImgY2**) sont les coordonnées de l'image visible dans le contrôle.
En effet le mode Zoom conserve les proportions de l'image d'origine.
Le contrôle n'est donc pas entièrement rempli par l'image si l'image et le contrôle n'ont pas le même rapport hauteur/largeur.
En utilisant ces propriétés on est sûr de dessiner sur l'image visible.
- **RGB(255, 255, 200)** correspond à une couleur de fond jaune pastel.
- **VbBlue** est la couleur du texte : on écrit le texte en bleu.
- **0** et **2** sont les alignements du texte : centré horizontalement et positionné en bas dans le rectangle défini précédemment.

Toutes les fonctions de dessin (seule la fonction LoadFromFile déroge à la règle) s'appliquent en mémoire et leur effet n'est pas immédiatement visible.
Pour voir à l'écran les modifications apportées à l'image, il faut le demander explicitement à l'aide de la fonction Repaint.

On utilise donc la fonction **Repaint** pour appliquer les modifications au contrôle.

Dessine le texte

```
' Dessin du texte
ClImg.DrawText "Essais de texte", ClImg.ImgX1, ClImg.ImgY1, ClImg.ImgX2,
 ClImg.ImgY2, RGB(255, 255, 200), vbBlue, 0, 2
' Dessin à l'écran
ClImg.Repaint
```

Voilà ce que l'on obtient.

C'est déjà plus intéressant...

VII - Dessin du rectangle

On va **afficher un rectangle au centre l'image**.
On donne les coordonnées du rectangle à dessiner à la fonction **DrawRectangle**.
On dessine ici un rectangle rouge dont la taille est égale à 2/5 de la taille de l'image visible.
L'intérieur du rectangle est transparent; pour ajouter une couleur de remplissage, remplacez -1 par un code couleur.

Dessine un rectangle centré sur le contrôle

```
' Dessin du rectangle
ClImg.DrawRectangle Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    -1, vbRed, 2
```

Placez ce code avant l'instruction **Repaint**.
Un seul **Repaint** est suffisant, il est inutile d'exécuter la fonction **Repaint** après chaque changement sur l'image.
On obtient un rectangle au centre de l'écran.

Là ça y est, on a commencé à dessiner! :o)

VIII - Affichage d'une petite image dans le rectangle

Maintenant on va afficher une petite image à l'intérieur du rectangle que l'on vient de dessiner.
On écrit toujours le code à la suite du précédent (mais avant le **Repaint**).
Pour dessiner une image il faut d'abord l'**ajouter à la liste d'images**.
Comme pour le chargement de l'image de fond on défini le chemin du fichier et on limite la taille de l'image chargée à la largeur du rectangle qui va contenir l'image.
On donne un nom à cette image : **MonImage**.

Ajoute l'image à la liste d'images

```
' Ajout de l'image à la liste d'images
ClImg.ImageListAdd "MonImage", ClImg.ApplicationPath & "logo.gif", 2 * (ClImg.ImgX2 - ClImg.ImgX1) / 5
```

On va ensuite **dessiner l'image "MonImage"** dans le rectangle.
On utilise la fonction **PaintImage** qui permet de dessiner une image de la liste d'images.
On donne en paramètres de la fonction les même coordonnées que pour le rectangle.
Puis on défini le **mode d'affichage** à **Zoom** et le **positionnement** à **2** (c-à-d centré).
L'image est dessinée dans le rectangle comme elle le serait dans un contrôle image avec les même propriétés de mode d'affichage et de positionnement.
Si votre image a une couleur de transparence mettez cette couleur de transparence à la place du -1 (vbWhite par exemple).

Dessine une image dans le rectangle rouge

```
' Dessine l'image dans le rectangle
' Mode Zoom centré
ClImg.PaintImage "MonImage", Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                             Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                             Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                             Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                             -1, acOLESizeZoom, 2
```

On n'a plus besoin de l'image donc on la supprime.

Supprime l'image de la liste d'images

```
' Supprime l'image de la liste
ClImg.ImageListDel "MonImage"
```

On a fini de dessiner!
Voilà le code complet qui s'éxecute au chargement du formulaire.

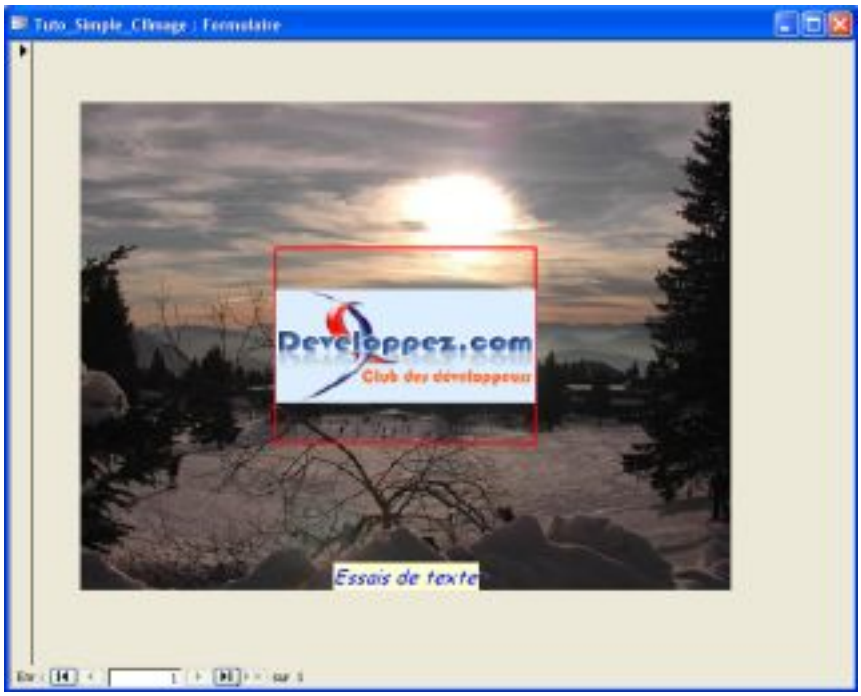
Code de dessin

```
Private Sub Form_Load()
' Initialisation du contrôle image
ClImg.SetImgCtrl Me.Image0
' Chargement d'une image de fond
' On précise la largeur de l'image car il n'est pas nécessaire de charger une image plus large que le contrôle
' La hauteur sera automatiquement calculée en fonction de la largeur précisée
ClImg.LoadFromfile ClImg.ApplicationPath & "DSCN1099.JPG", Me.Image0.Width
' Définition de la police de caractères
ClImg.DrawNewFont 24, 0, 400, True, False, False, "Comic sans MS"
' Dessin du texte
ClImg.DrawText "Essais de texte", ClImg.ImgX1, ClImg.ImgY1, ClImg.ImgX2,
 ClImg.ImgY2, RGB(255, 255, 200), vbBlue, 0, 2, False, 100
' Dessin du rectangle
ClImg.DrawRectangle Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
```

Code de dessin

```
                    -1, vbRed, 2
' Ajout de l'image à la liste d'images
ClImg.ImageListAdd "MonImage", ClImg.ApplicationPath & "logo.gif", 2 * (ClImg.ImgX2 - ClImg.ImgX1) / 5
' Dessine l'image dans le rectangle
' Mode Zoom centré
' On applique un filtre antialiasing
ClImg.PaintImage "MonImage", Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                             Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                             Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                             Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                             -1, acOLESizeZoom, 2
' Supprime l'image de la liste
ClImg.ImageListDel "MonImage"
' Dessin à l'écran
ClImg.Repaint
End Sub
```

Et le résultat obtenu.



On peut à présent passer à l'interactivité...

IX - La définition d'une région

Dessiner sur un contrôle c'est bien mais c'est encore mieux de rendre des zones sensibles et de réagir aux actions de l'utilisateur.
Pour faire ça il faut passer par la création de régions.
Pour créer une région on peut utiliser les fonctions dédiées :
AddEllipticRegion,**AddPolygonRegion**,**AddRectangleRegion** et **AddAutoRegion**.
Pour créer une région correspondant au rectangle rouge cela donnerait :

Crée une région correspondant au rectangle rouge

```
ClImg.AddRectangleRegion "MaRegion", Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                                     Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                                     Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                                     Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5
```

On voit que l'on serait obligé de passer à nouveau les même coordonnées en paramètres.
Il y a plus simple, il suffit d'utiliser le paramètre **pRegion** de la fonction **DrawRectangle** avec laquelle on a dessiné le rectangle.
On modifie alors le précédent appel à la fonction **DrawRectangle** comme suit :

Crée une région correspondant au rectangle rouge

```
' Dessin du rectangle
ClImg.DrawRectangle Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    -1, vbRed, 2, , "MaRegion"
```

On a simplement rajouté le paramètre **"MaRegion"** à la fin.
Cela va **créer automatiquement une région** nommée **"MaRegion"** définie par les coordonnées du rectangle dessiné.

X - Détecter le clic sur une région

On voudrait dans la suite afficher une boîte de message si l'utilisateur clique dans le rectangle rouge.
L'évenement **Sur clic** du contrôle image ne propose pas de paramètre donnant la position de la souris alors on va utiliser l'événement **Sur souris appuyée**.
La fonction **GetMouseRegion** nous renvoie le nom de la région située sous le curseur de la souris.
Si on a cliqué dans le rectangle rouge on reçoit alors la valeur **"MaRegion"**.
On peut à ce moment afficher une boîte de message.

Affiche un message si on clique dans le rectangle rouge

```
Private Sub Image0_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim lRegion As String
' On teste si la classe est initialisée
If ClImg Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = ClImg.GetMouseRegion(X, Y)
' Si on a cliqué dans le rectangle rouge alors on affiche un message
If lRegion = "MaRegion" Then MsgBox "Vous avez cliqué dans le rectangle rouge!"
End Sub
```

On obtient bien un message seulement si on clique dans le rectangle.
Mais on aimerait faire mieux parce qu'on n'a pas d'information qui nous indique qu'on survole une zone sensible de l'image...

XI - Détecter le survol d'une région

On va, pour terminer, déterminer si on survole notre rectangle et modifier le curseur pour indiquer que l'on peut cliquer sur cette zone.
Pour savoir le nom de la région survolée on va procéder de la même manière que pour le clic mais sur l'événement **Sur souris déplacée**.
Si on survole le rectangle on modifie le curseur avec la fonction **SetHandCursor**.
Sinon on rétablit le curseur par défaut avec la fonction **ResetCursor**.

Affiche une icône en forme de main lorsqu'on survole le rectangle

```
Private Sub Image0_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim lRegion As String
' On teste si la classe est initialisée
If ClImg Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = ClImg.GetMouseRegion(X, Y)
' Change le curseur lorsqu'on survole le rectangle
If lRegion = "MaRegion" Then ClImg.SetHandCursor Else ClImg.ResetCursor
End Sub
```

Voilà c'est fini on a bien notre curseur qui change au survol du rectangle.

XII - Le code complet

Voici le code complet de ce tutoriel :

Code complet

```
Option Compare Database
Option Explicit

' Déclaration de la classe
Private ClImg As New ClImage

Private Sub Form_Close()
' Libération de la classe à la fermeture du formulaire
If Not ClImg Is Nothing Then Set ClImg = Nothing
End Sub

Private Sub Form_Load()
' Initialisation du contrôle image
ClImg.SetImgCtrl Me.Image0
' Chargement d'une image de fond
' On précise la largeur de l'image car il n'est pas nécessaire de charger une image plus large que le contrôle
' La hauteur sera automatiquement calculée en fonction de la largeur précisée et des proportions de l'image
ClImg.LoadFromfile ClImg.ApplicationPath & "DSCN1099.JPG", Me.Image0.Width
' Définition de la police de caractères
ClImg.DrawNewFont 24, 0, 400, True, False, False, "Comic sans MS"
' Dessin du texte
ClImg.DrawText "Essais de texte", ClImg.ImgX1, ClImg.ImgY1, ClImg.ImgX2,
 ClImg.ImgY2, RGB(255, 255, 200), vbBlue, 0, 2, False, 100
' Dessin du rectangle
ClImg.DrawRectangle Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                    Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                    -1, vbRed, 2, , "MaRegion"
' Ajout de l'image à la liste d'images
ClImg.ImageListAdd "MonImage", ClImg.ApplicationPath & "logo.gif", 2 * (ClImg.ImgX2 - ClImg.ImgX1) / 5
' Dessine l'image dans le rectangle
' Mode Zoom centré
' On applique un filtre antialiasing
ClImg.PaintImage "MonImage", Image0.Width / 2 - (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                             Image0.Height / 2 - (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                             Image0.Width / 2 + (ClImg.ImgX2 - ClImg.ImgX1) / 5, _
                             Image0.Height / 2 + (ClImg.ImgY2 - ClImg.ImgY1) / 5, _
                             -1, acOLESizeZoom, 2
' Supprime l'image de la liste
ClImg.ImageListDel "MonImage"
' Dessin à l'écran
ClImg.Repaint
End Sub

Private Sub Image0_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim lRegion As String
' On teste si la classe est initialisée
If ClImg Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = ClImg.GetMouseRegion(X, Y)
' Si on a cliqué dans le rectangle rouge alors on affiche un message
If lRegion = "MaRegion" Then MsgBox "Vous avez cliqué dans le rectangle rouge!"
End Sub

Private Sub Image0_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
Dim lRegion As String
' On teste si la classe est initialisée
If ClImg Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = ClImg.GetMouseRegion(X, Y)
' Change le curseur lorsqu'on survole le rectangle
If lRegion = "MaRegion" Then ClImg.SetHandCursor Else ClImg.ResetCursor
```

Code complet

```
End Sub
```

XIII - Conclusion

On a donc dessiné sur un contrôle image et on a rendu une zone sensible pour donner de l'interactivité à l'image.
Le code est je pense assez simple, à la portée de tout le monde j'espère.
Merci à Xo, cafeine et Tofalu pour leur relecture.

XIV - Téléchargements

Télécharger la base Access de ce tutoriel au format ACCESS 2000 *(HTTP)*