



eIGdiplus

Tutoriel : Premiers pas **Module VBA de gestion d'image**

par Thierry GASPERMENT (arkham46.developpez.com/)

Date de publication : 24/04/08

Dernière mise à jour : 24/04/08

Ce document a pour objectif d'expliquer les bases de l'utilisation de la classe eIGdiPlus. La classe eIGdiPlus 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.

I - Introduction.....	3
II - Création du formulaire et du contrôle image.....	3
III - Création de la classe.....	3
IV - Déclaration de la classe et initialisation.....	3
V - Chargement d'une image de fond.....	4
VI - Dessin du texte.....	5
VII - Dessin du rectangle.....	6
VIII - Affichage d'une petite image dans le rectangle.....	7
IX - La définition d'une région.....	9
X - Détecter le clic sur une région.....	10
XI - Détecter le survol d'une région.....	10
XII - Corriger les clignotements.....	11
XIII - Le code complet.....	11
XIV - Conclusion.....	12
XV - Téléchargements.....	12

I - Introduction

Ce tutoriel a été rédigé avec Access 2003.

Nous allons au cours de ce tutoriel :

- 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**

 **Lien vers la librairie en téléchargement sur Microsoft.com pour Windows autre que XP**

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.

Si nécessaire, choisissez 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 **CIGdiPlus**.

Attention : le nom sous lequel vous sauvegardez le module est important.

Ce module de classe VBA utilise la librairie `gdiplus.dll` de Microsoft.

Si vous utilisez une version de Windows autre que XP, télécharger la librairie et placez la dans le même répertoire que le fichier Access.

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.

Déclaration de la classe

```
Private gGdip As clGdiPlus
```

Première chose à faire : il faut **créer une instance de l'objet classe**.

(Pour l'instant gGdip a pour valeur **Nothing**).

On écrit ce code dans l'événement **Sur Chargement** du formulaire.

Dans les propriétés du formulaire, définissez [**Procédure événementielle**] dans l'événement **Sur Chargement**.

Cliquez sur les trois petits points [...] pour générer l'événement dans le code.

Initialisation du contrôle

```
Private Sub Form_Load()  
' Initialisation de la classe de dessin  
Set gGdip = New clGdiPlus  
End Sub
```

Deuxième chose à faire : **pensez à libérer la classe dès qu'elle n'est plus utile, au plus tard donc 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 gGdip Is Nothing Then Set gGdip = Nothing  
End Sub
```

V - Chargement d'une image de fond

Si vous visualisez le formulaire, vous ne voyez rien.

C'est normal car on n'a encore rien fait...

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.

La fonction utilisée pour charger le fichier est **OpenFile**.

Une fois le fichier ouvert, il faut ensuite mettre à jour le contrôle pour voir le résultat à l'écran.

(La fonction **OpenFile** ouvre le fichier en mémoire mais n'affiche rien).

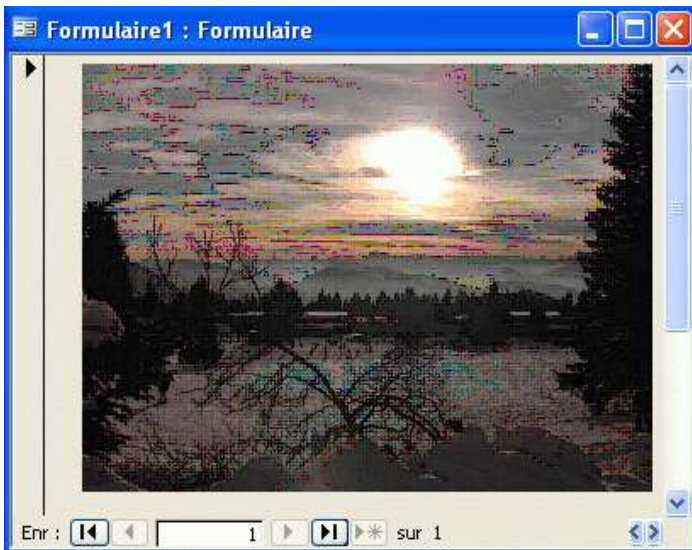
L'affichage à l'écran s'effectue en mettant à jour la propriété **PictureData** de l'image grâce à la fonction **RepaintControl**.

Chargement d'une image de fond

```
Private Sub Form_Load()  
' Initialisation de la classe de dessin  
Set gGdip = New clGdiPlus  
' Chargement d'une image de fond  
gGdip.OpenFile CurrentProject.Path & "\DSCN1099.JPG"  
' Affichage de l'image dans le contrôle  
gGdip.RepaintControl Me.Image0  
End Sub
```

On obtient alors l'image dans le formulaire.

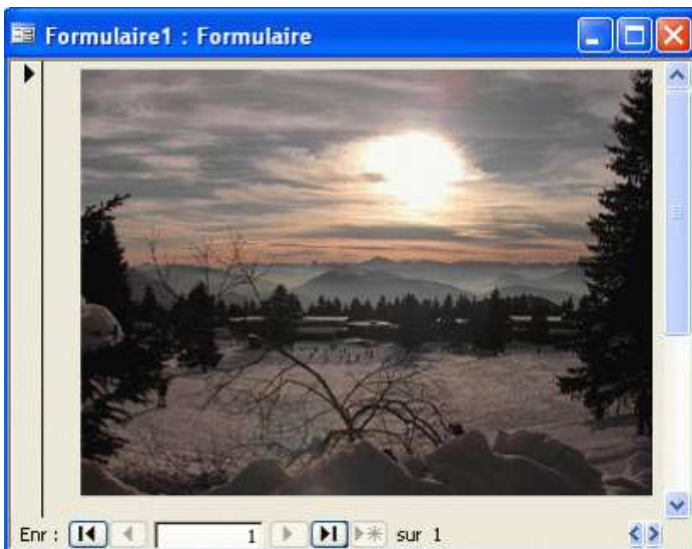
Il peut parfois y avoir un problème d'affichage au redimensionnement de l'image, généralement lors de l'affichage d'une image plus grande que le contrôle qui la contient.



Pour résoudre ce problème, passez le paramètre **pUseEMF** de la fonction **RepaintControl** à **Vrai**. Le problème d'affichage ne se produit pas pour un type d'image EMF.

Affichage avec format EMF

```
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0, , , True
```



VI - Dessin du texte

Nous souhaitons maintenant dessiner du texte sur l'image :

- Texte : Essais de texte
- Taille : 16 points
- Police : Comic sans MS
- Position : En bas, centré horizontalement
- Couleur du texte : Bleu
- Couleur du fond : Jaune pâle
- Italique : Oui
- Souligné : Non

- Barré : Non

- On passe 4 coordonnées en paramètres, ce sont les coordonnées d'un rectangle dans lequel on va dessiner le texte.

On utilise les coordonnées (0,0,gGdip.ImageWidth, gGdip.ImageHeight) qui sont celles de l'image complète.

- 1 et 2 sont les alignements du texte : centré horizontalement et positionné en bas dans le rectangle défini précédemment.

- **VbBlue** est la couleur du texte : on écrit le texte en bleu.

- **RGB(255, 255, 200)** correspond à une couleur de fond jaune pastel.

- **True** dans le paramètre italic pour afficher le texte en italique.

La taille du texte pour la fonction **DrawText** est exprimée en pixel sur l'image.

Pour l'exemple nous souhaitons que le texte affiché à l'écran soit de la même taille qu'un contrôle de taille de police égal à 16.

Cette valeur 16 est exprimée en points.

La fonction **FontSizeToPixel** va convertir les twips (ou points) en pixels sur l'image.

Pour tester j'ai placé un contrôle étiquette pour vérifier que la taille du texte correspond.

Dessine le texte

```
' Dessin du texte
gGdip.DrawText "Essais de texte", gGdip.FontSizeToPixel(16 , Me.Image0), _
                "Comic sans MS", 0, 0, gGdip.ImageWidth, gGdip.ImageHeight, _
                1, 2, vbBlue, , RGB(255, 255, 200), , True
```

Placez ce code avant l'instruction d'affichage dans le contrôle.

Une seule mise à jour de la propriété **PictureData** du contrôle sera nécessaire une fois tout le dessin effectué en mémoire.

Voilà ce que l'on obtient.



VII - Dessin du rectangle

On va **afficher un rectangle au centre de 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.

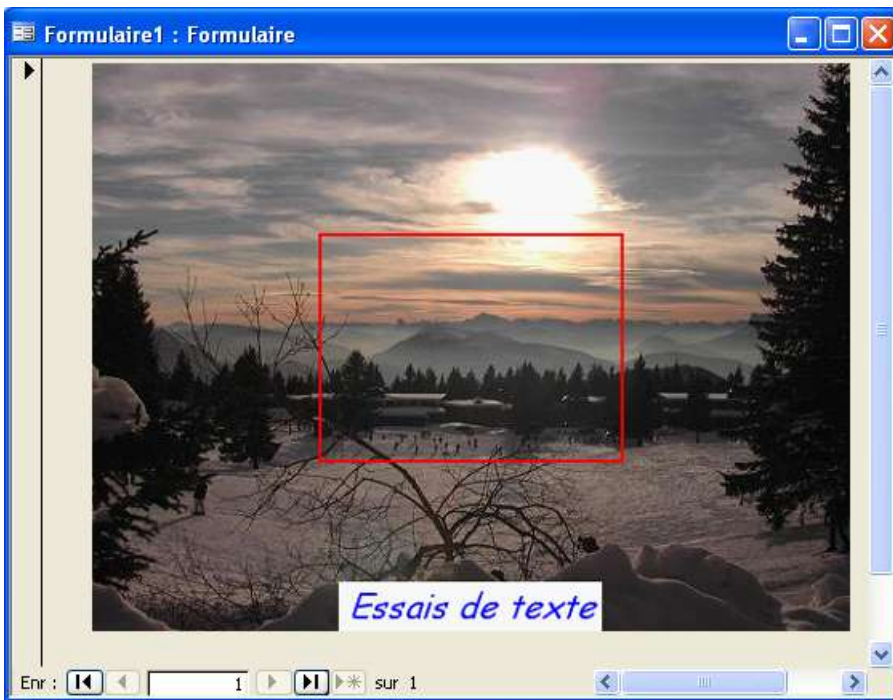
L'intérieur du rectangle est transparent; pour ajouter une couleur de remplissage, remplacez -1 par un code couleur.

Dessine un rectangle centré sur le contrôle

```
' Dessin du rectangle
gGdip.DrawRectangle gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
                    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
                    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
                    , vbRed, 2
```

Placez ce code avant l'instruction d'affichage dans le contrôle.

On obtient un rectangle au centre de l'écran.



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 avant l'instruction d'affichage dans le contrôle.

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 définit 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
gGdip.ImageListAdd "MonImage", CurrentProject.Path & "\logo.gif", 2 * gGdip.ImageWidth / 5
```

On va ensuite **dessiner l'image "MonImage"** dans le rectangle.

On utilise la fonction **DrawImage** qui permet de dessiner une image de la liste d'images.

On donne en paramètres de la fonction les mêmes coordonnées que pour le rectangle.

Puis on définit le **mode d'affichage** à **GdipSizeModezoom** et le **positionnement** à **GdipAlignCenter**.

L'image est dessinée dans le rectangle comme elle le serait dans un contrôle image avec les mêmes 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
gGdip.DrawImage "MonImage", gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
    -1, GdipSizeModezoom, GdipAlignCenter
```

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
gGdip.ImageListDel "MonImage"
```

On a fini de dessiner!

Voilà le code complet qui s'exécute au chargement du formulaire.

Code de dessin

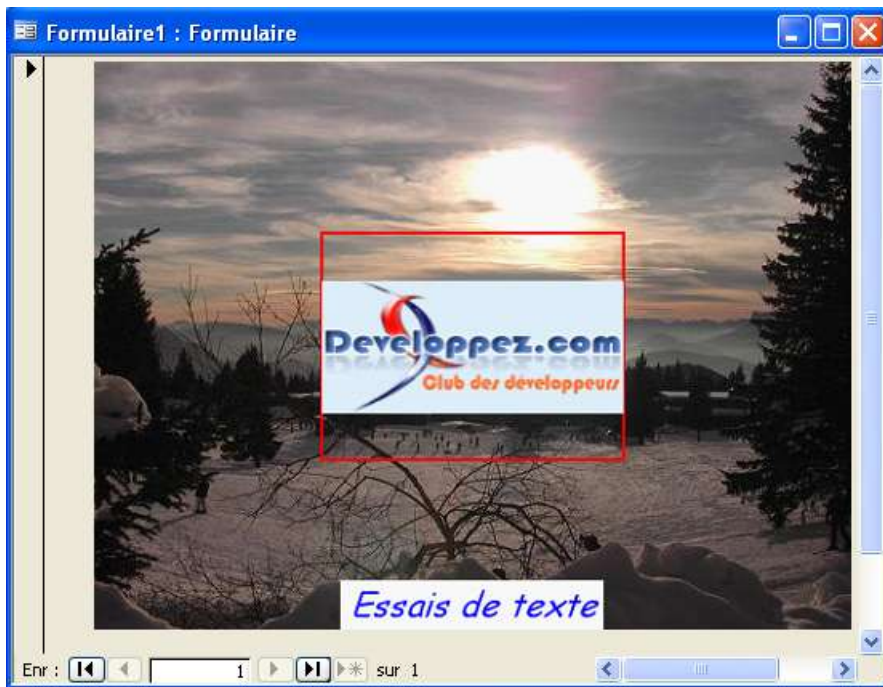
```
Option Compare Database
Option Explicit

' Déclaration de la classe
Private gGdip As New ClGdiPlus

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

Private Sub Form_Load()
' Initialisation de la classe de dessin
Set gGdip = New ClGdiPlus
' 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
gGdip.OpenFile CurrentProject.Path & "\DSCN1099.JPG"
' Dessin du texte
gGdip.DrawText "Essais de texte", gGdip.FontSizeToPixel(16, Me.Image0), _
    "Comic sans MS", 0, 0, gGdip.ImageWidth, gGdip.ImageHeight, _
    1, 2, vbBlue, , RGB(255, 255, 500), , True
' Dessin du rectangle
gGdip.DrawRectangle gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
    , vbRed, 2
' Ajout de l'image à la liste d'images
gGdip.ImageListAdd "MonImage", CurrentProject.Path & "\logo.gif", 2 * gGdip.ImageWidth / 5
' Dessine l'image dans le rectangle
' Mode Zoom centré
gGdip.DrawImage "MonImage", gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
    -1, GdipSizeModezoom, GdipAlignCenter
    gGdip.ImageListGetPixel "MonImage", 1, 1
' Supprime l'image de la liste
gGdip.ImageListDel "MonImage"
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0, , , True
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 :

CreateRegionRect, **CreateRegionPolygon** et **CreateRegionEllipse**.

Pour créer une région correspondant au rectangle rouge cela donne :

Crée une région correspondant au rectangle rouge

```
gGdip.CreateRegionRect "MaRegion", gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5
```

On voit que l'on serait obligé de passer à nouveau les mêmes 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
gGdip.DrawRectangle gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
    , 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'événement **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 **GetRegionXY** nous renvoie le nom de la région située sous le curseur de la souris.

Notez qu'on converti d'abord la position de la souris en pixel sur l'image avec les fonctions **CtrlToImgX** et **CtrlToImgY**. 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 gGdip Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = gGdip.GetRegionXY(gGdip.CtrlToImgX(X, Me.Image0), gGdip.CtrlToImgY(Y, Me.Image0))
' 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 le hachurer.

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 souhaite hachurer la région avec la fonction **HatchRegion**.

On utilise les fonctions de sauvegarde en mémoire (**KeepImage** et **ResetImage**) pour rétablir l'image avant de la hachurer.

Ainsi on repart à chaque fois de l'image d'origine.

Il faut donc exécuter la fonction **KeepImage** une fois l'image dessinée dans la procédure **Form_Load**

La variable Static **sRegion** nous est utile pour ne dessiner ou retirer les hachures que si la région survolée change.

Ajout dans Form_Load à la fin de la procédure

```
' Conserve l'image
gGdip.KeepImage
```

Hachure la région 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
Static sRegion As String
' On teste si la classe est initialisée
If gGdip Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = gGdip.GetRegionXY(gGdip.CtrlToImgX(X, Me.Image0), gGdip.CtrlToImgY(Y, Me.Image0))
' si changement de région
If sRegion <> lRegion Then
' Rétabli l'image originale
gGdip.ResetImage
' Rempli la région en bleu
gGdip.HatchRegion lRegion, vbBlue, , HatchStyleForwardDiagonal, 150
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0, , , True
End If
```

Hachure la région lorsqu'on survole le rectangle

```
' Conserve le nom de la dernière région survolée
sRegion = lRegion
End Sub
```

Voilà c'est fini on a bien notre région hachurée au survol du rectangle.

XII - Corriger les clignotements

Le changement d'image lors du survol de la souris peut produire un effet de clignotement.

Il semble que le couple Access 2003 + Windows XP soit le plus touché par cet effet.

Trois solutions pour essayer de se débarrasser de ces clignotements :

- Utiliser la fonction **SetXPTheme** pour désactiver le thème XP sur les contrôles de l'application.

Désactiver le thème XP pour faire disparaître les scintillements.

- Utiliser la fonction **SetDoubleBufferXP** pour activer le double tampon sur le formulaire.

Le double tampon élimine efficacement les scintillements de l'image, mais est gourmand en ressource système.

- Utiliser la fonction **FastRepaint** pour redessiner l'image directement sur le formulaire.

Au lieu de modifier la propriété **PictureData** du contrôle, dessiner directement l'image sur le formulaire.

Cette fonction dessine temporairement l'image.

XIII - Le code complet

Voici le code complet de ce tutoriel :

Code complet

```
Option Compare Database
Option Explicit

' Déclaration de la classe
Private gGdip As New ClGdiPLus

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

Private Sub Form_Load()
' Initialisation de la classe de dessin
Set gGdip = New ClGdiPLus
' 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
gGdip.OpenFile CurrentProject.Path & "\DSCN1099.JPG"
' Dessin du texte
gGdip.DrawText "Essais de texte", gGdip.FontSizeToPixel(16, Me.Image0), _
    "Comic sans MS", 0, 0, gGdip.ImageWidth, gGdip.ImageHeight, _
    1, 2, vbBlue, , RGB(255, 255, 500), , True
' Dessin du rectangle
gGdip.DrawRectangle gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
    , vbRed, 2, , , "MaRegion"
' Ajout de l'image à la liste d'images
gGdip.ImageListAdd "MonImage", CurrentProject.Path & "\logo.gif", 2 * gGdip.ImageWidth / 5
' Dessine l'image dans le rectangle
' Mode Zoom centré
gGdip.DrawImage "MonImage", gGdip.ImageWidth / 2 - gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 - gGdip.ImageHeight / 5, _
    gGdip.ImageWidth / 2 + gGdip.ImageWidth / 5, _
    gGdip.ImageHeight / 2 + gGdip.ImageHeight / 5, _
    -1, GdipSizeModezoom, GdipAlignCenter
gGdip.ImageListGetPixel "MonImage", 1, 1
```

Code complet

```

' Supprime l'image de la liste
gGdip.ImageListDel "MonImage"
' Conserve l'image
gGdip.KeepImage
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0,, , True
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 gGdip Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = gGdip.GetRegionXY(gGdip.CtrlToImgX(X, Me.Image0), gGdip.CtrlToImgY(Y, Me.Image0))
' 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
Static sRegion As String
' On teste si la classe est initialisée
If gGdip Is Nothing Then Exit Sub
' Récupère le nom de la région sur laquelle on a cliqué
lRegion = gGdip.GetRegionXY(gGdip.CtrlToImgX(X, Me.Image0), gGdip.CtrlToImgY(Y, Me.Image0))
' si changement de région
If sRegion <> lRegion Then
' Rétabli l'image originale
gGdip.ResetImage
' Rempli la région en bleu
gGdip.HatchRegion lRegion, vbBlue, , HatchStyleForwardDiagonal, 150
' Affichage de l'image dans le contrôle
gGdip.RepaintControl Me.Image0, , , True
End If
' Conserve le nom de la dernière région survolée
sRegion = lRegion
End Sub
    
```

XIV - Conclusion

On a donc dessiné sur un contrôle image et on a rendu une zone sensible pour donner de l'interactivité à l'image. Merci à l'équipe Office de developpez.com pour ses relectures, commentaires et encouragements!

XV - Téléchargements

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