ACCESS : Classe de manipulation des images

par Thierry GASPERMENT (arkham46.developpez.com/)

Date de publication : 18/03/08

Dernière mise à jour : 18/03/08

Ce document a pour objectif de décrire les propriétés et fonctions de la classe ClImage. La classe quant à elle a pour objectif de permettre à n'importe quel développeur de dessiner dans un contrôle image standard sans connaissance des API de dessin.

I - Que peut-on peut faire avec cette classe?

- On peut tout d'abord **dessiner** : des **lignes**, des **polygones**, des **cercles**, du **texte**.
- On peut également **insérer des petites images** dans l'image.
- Et on peut définir **des régions qui réagissent au survol ou au click de la souris**.
- Il y a bien sûr beaucoup d'autres fonctions qui seront détaillées dans la suite.

Un exemple est toujours plus parlant :

Les départements sont hachurés au passage de la souris, et on peut les sélectionner en cliquant dessus.
Le fond du formulaire est transparent pour ne laisser voir que la carte et les contrôles.

Les aiguillent des horloges se déplacent en fonction de l'heure.
Si on tire sur une aiguille l'heure change sur les deux horloges en même temps.

Consultez le tutoriel

II - Remarques préalables

II-A - Remarques

- Sauf exceptions mentionnées dans cette documentation, **les unités sont exprimées en Twips**.
(Les twips sont l'unité de mesure standard de VBA, par exemple : Me.Width est exprimé en Twips)
- **Spéficiez au maximum la taille des images que vous chargez** pour réduire la mémoire utilisée.
- **N'abusez pas des images dans vos applications** ; Access n'est pas fait pour ça, et ça nuit souvent à la lisibilité des formulaires.
- La classe est développée sur Access 2003; j'essaye d'assurer la meilleure compatibilité avec les versions antérieures.
- **N'oubliez de libérer la classe** au plus tard à la fermeture du formulaire :

```
If Not Climg Is Nothing Then Set Climg = Nothing
```

- Dans tous mes exemples je nomme **ClImg l'instance de la classe** et **Image0 le contrôle image**.
- Il est souvent mentionné le terme "contrôle image" qui peut être un contrôle image ou un formulaire (le support du formulaire étant limité...).
- Si le retour de la fonction n'est pas mentionné, elle retourne **Vrai s'il n'y a pas eu d'erreur**, **Faux dans le cas contraire**.
- Les **paramètres optionnels** sont notés : **[NomduParamètre] = ValeurParDéfaut**

II-B - Compatibilité

Le développement est fait sous Access 2003 (au format Access 2000) et Windows XP Home.
La compatibilité Access 97 n'est pas totale, notamment à cause de limitations graphiques d'Access.
Cependant la majeure partie des fonctionnalités fonctionnent sans problème.
(Voir "Limitations" ci-dessous.)

II-C - Limitations

- **DragForm** ne fonctionne que sur un formulaire indépendant.
- Avec Access 97, le chargement d'une image Jpeg de taille importante (> 1024 pixels) peut mener à un affichage de moindre qualité, voir une absence d'image à l'écran pour les tailles plus élevées (> 1900 pixels environ).
C'est apparemment une limitation du filtre Jpeg d'Access 97.
- Sous Windows 98, les images transclucides peuvent faire boguer l'affichage de l'image.
(Bug dans une API de Windows 98).
- Si l'image a une bordure, certaines fonctions font apparaîtrent des décalages de l'image. --> supprimez la bordure de l'image et ajouter un rectangle autour de l'image à la place - Sur Access 97, la fonction **Repaint** avec **pTemporary** à Vrai ne fonctionne pas (décalage) lorsqu'on déplace l'ascenceur vertical.

III - Les fonctions de base

III-A - SetImgCtrl

Initialise le contrôle Image (ou le formulaire) sur lequel on va dessiner.
C'est la première fonction à utiliser.
Une fois le contrôle initialisé on peut alors dessiner dessus.
On ne peut définir qu'un contrôle par classe.
Paramètres :

Paramètre	Type	Explication
Ctrl	Object	Contrôle ou formulaire
[pDelRegions]=True	Boolean	Si Vrai alors les régions définies ne sont pas supprimées. (Utile pour conserver les régions au niveau formulaire avant un SetFormRegion)
[pWidth]=0	Largeur de l'image	Si non renseigné => largeur du contrôle
[pHeight]=0	Hauteur de l'image	Si non renseigné => hauteur du contrôle
[pNoRepaint]=False	Boolean	Si Vrai alors l'image n'est pas redessinée à l'écran durant cette fonction.

Exemple :

```
lReturn = Climg.SetImgCtrl(Me.Image0)
```

III-B - Repaint

Redessine l'image dans le contrôle.
A l'exception de LoadFromFile, les fonctions dessinent en mémoire.
L'image n'est actualisée à l'écran qu'à l'appel de la fonction **Repaint**.
Paramètres :

Paramètre	Type	Explication
[pTemporary]=False	Boolean	Dessin direct à l'écran sans mise à jour de l'image dans le contrôle. L'affichage est beaucoup plus rapide mais il est temporaire.
[pResizeBefore]=False	Boolean	Si Vrai, l'image est redimensionnée à la taille du contrôle avant d'y être injecté.

III-C - LoadFromFile

Charge une image dans le contrôle image.
L'image est directement actualisée à l'écran.
Paramètres :

Paramètre	Type	Explication
pFichier	String	Chemin d'un fichier image (testé sur des fichiers jpg,gif,bmp)
[pWidth] = 0	Long	Largeur de l'image
[pHeight] = 0	Long	Hauteur de l'image
[pAntiAliase] = Faux	Boolean	Appliquer ou non un filtre antialising sur l'image
[pThumbNail] = Faux	Boolean	Lecture de la miniature intégrée à un Jpeg (Nécessite GDI+)

Si une des dimensions est nulle alors elle est calculée par rapport à l'autre en conservant le rapport hauteur/largeur.
Si les deux dimensions sont nulles alors l'image garde sa taille originale.
La fonction renvoie dans les paramètres pWidth et pHeight les dimensions de l'image chargée
Exemple :

```
lReturn = Climg.LoadFromfile("c:\test.jpg", Image0.Width)<br/>
```

III-D - KeepImgData

Sauvegarde une image en mémoire.
Paramètres :

Paramètre	Type	Explication
[pNom] = "BACKUP"	String	Nom de la sauvegarde
pX1 pY1 pX2 pY2	Long	Coordonnées du rectangle à sauvegarder

III-E - RefreshImgData

Charge une image préalablement sauvegardée en mémoire par la fonction KeepImgData.
Paramètres :

Paramètre	Type	Explication
[pNom] = "BACKUP"	String	Nom de la sauvegarde
pX1 pY1 pX2 pY2	Long	Coordonnées du rectangle à redessiner. A priori celles utilisées à l'appel de KeepImgData.
[pTranspColor] = -1	Long	Couleur de transparence
[pPercent] = 100	Long	Pourcentage de transclucidité

Utilisez pTranspColor et pPercent pour superposer plusieurs images.

III-F - DelImgData

Supprime de la mémoire une image préalablement sauvegardée par la fonction KeepImgData.
Paramètres :

Paramètre	Type	Explication
[pNom] = "BACKUP"	String	Nom de la sauvegarde

IV - Les fonctions de modification de l'image

IV-A - ChangeColor

Paramètres :

Remplace une couleur de l'image par une autre.

Paramètres :

Paramètre	Type	Explication
OldColor	Long	Ancienne couleur
NewColor	Long	Nouvelle couleur

Exemple :

Remplace la couleur de fond de l'image par celle du formulaire

```
lReturn = Climg.ChangeColor(10289151, Me.Section(acDetail).BackColor)
```

IV-B - FillColor

Rempli l'image d'une couleur unie.

Paramètres :

Paramètre	Type	Explication
pColor	Long	Couleur de remplissage
[pX1] = 0 [pY1] = 0 [pX2] = -1 [pY2] = -1	Long	pX1,pY1,pX2,pY2 forment le rectangle qui doit être rempli. Si aucune coordonnée n'est définie alors l'image est entièrement remplie. Si pX2 est omis l'image est remplie jusqu'au bord droit. Si pY2 est omis l'image est remplie jusqu'en bas.
[pColor2]=-1	Long	Couleur pour dégradé Le dégradé va de la couleur pColor vers la couleur pColor2
[pHorizGradient]=Faux	Boolean	Faux : Gradient de gauche à droite Vrai : Gradient de haut en bas

Exemple :

Rempli de blanc le quart de l'image en bas à droite

```
lReturn = Climg.FillColor(vbWhite,Image0.Width/2,Image0.Height/2)
```

IV-C - ImgTransform

Applique une transformation à l'image.

---> Voir la section Valeurs des paramètres pour la liste des transformations

Paramètres :

Paramètre	Type	Explication
pTransform	Variant	Liste des transformations. On peut définir une transformation unique ou plusieurs transformations. Pour définir plusieurs transformations passer un tableau en paramètre.
[pParam]	Variant	Liste des paramètres des transformations. Chaque transformation peut éventuellement accepter un paramètre. Il doit y avoir autant de paramètres que de transformations, même si certaines transformations demandées n'en requièrent pas.
[pTranspcolor] = vbBlack	Long	Couleur utilisée pour remplir les zones sans images (après une rotation par exemple)
[pProgressBar] = Nothing	Object	Contrôle image qui va servir de barre de progression
[pProgressBackColor] = vbWhite	Long	Couleur de fond de la barre de progression
[pProgressBarColor] = vbRed	Long	Couleur de la barre de progression

Exemple :

Effectuer une rotation de 30° de l'image

```
' On rempli les zones noires par la couleur de fond du formulaire
' On utilise un contrôle image ImgProgress pour visualiser la progression du traitement
lReturn = climg.ImgTransform("ROTATE", 30, Me.Section(acDetail).BackColor, ImgProgress)
```

Effectuer une rotation de 30° de l'image puis appliquer un filtre 'niveau de gris'

```
' On rempli les zones noires par la couleur de fond du formulaire
lReturn = climg.ImgTransform(Array("ROTATE","GRAY"), Array(30,0), Me.Section(acDetail).BackColor)
```

IV-D - ImgResize

Redimensionne le contrôle contenant l'image.
Utile pour se donner un peu d'espace si on a déjà utilisé toute la surface du contrôle.
Paramètres :

Paramètre	Type	Explication
pX	Long	Nouvelle largeur
pY	Long	Nouvelle hauteur
[pImgSizeMode] = acOLESizeClip	Integer	Type de positionnement de l'image après redimensionnement
[pImgPictureAlignment] = 0	Integer	Position de l'image après redimensionnement (en haut à gauche par défaut)
[pColor] = vbBlack	Long	Couleur de remplissage de la zone agrandie
[pNoCtrlResize] = False	Boolean	Si Vrai, le contrôle n'est pas redimensionné (seule l'image est redimensionnée).

Par défaut l'image est placée en haut à gauche sans redimensionnement.

Exemple :

Double la taille du contrôle, laisse l'image inchangée en haut à gauche

```
' On multiple les dimensions par deux
' On laisse le repositionnement par défaut
' On colorie en blanc l'espace ajouté en bas et à droite
climg.ImgResize Image0.Width*2, Image0.Height*2, , , RGB(255, 255, 255)
```

V - Les fonctions de dessin

V-A - DrawPixel

Dessine un pixel.
Paramètres :

Paramètre	Type	Explication
pX	Long	Position horizontale du pixel
pY	Long	Position verticale du pixel
pColor	Long	Couleur du pixel

Exemple :

Dessiner un pixel rouge au milieu du contrôle

```
lReturn = climg.DrawPixel(Image0.Width/2,Image0.Height/2,vbRed)
```

V-B - DrawLine

Dessine une ligne.
Paramètres :

Paramètre	Type	Explication
pX1,pY1	Long	Position du premier point
pX2,pY2	Long	Position du deuxième point
[pPenColor] = vbBlack	Long	Couleur de la ligne
[pPenWidth] = 1	Long	Epaisseur de la ligne
[pPenStyle] = 0	Long	Style du trait ---> Voir la section Valeurs des paramètres pour la liste des types de traits
[pArrowLength] = 0	Long	Taille des traits de la flèche Si > 0, affiche une flèche

Exemple :

Dessiner une ligne bleue qui traverse le contrôle en diagonal

```
lReturn = climg.DrawLine(0,0,Image0.Width,Image0.Height,vbBlue)
```

V-C - DrawRectangle

Dessine un rectangle.
Paramètres :

Paramètre	Type	Explication
pX1,pY1,pX2,pY2	Long	Coordonnées du rectangle
[pBackColor] = -1	Long	Couleur de remplissage Laisser -1 pour un rectangle transparent.
[pPenColor] = vbBlack	Long	Couleur de la ligne
[pPenWidth] = 1	Long	Epaisseur de la ligne
[pPenStyle] = 0	Long	Style du trait ---> Voir la section Valeurs des paramètres pour la liste des types de traits.
[pRegion] = 0	String	Nom de la région à créer automatiquement

Exemple :

Dessiner un rectangle aléatoire

```
' pBackColor = -1 siginifie qu'on dessine uniquement la bordure sans le fond
' pPenColor = vbBlue pour dessiner en rouge
' Si l'image est affichée en mode Zoom ou Découpage il se peut que l'image ne remplisse
'   pas entièrement le contrôle
' On utilise donc les propriétés ImgX1,ImgY1,ImgX2,ImgY2 pour s'assurer que
'  les coordonnées du rectangle sont dans l'image visible
    climg.DrawRectangle climg.ImgX1 + CLng(Rnd * (climg.ImgX2 - climg.ImgX1)), climg.ImgY1 + CLng(Rnd *
 (climg.ImgY2 - climg.ImgY1)), _
                        climg.ImgX1 + CLng(Rnd * (climg.ImgX2 - climg.ImgX1)), climg.ImgY1 + CLng(Rnd *
 (climg.ImgY2 - climg.ImgY1)), _
                        -1, vbBlue
```

V-D - DrawEllipse

Dessine une ellipse.
Paramètres :

Paramètre	Type	Explication
pX1,pY1,pX2,pY2	Long	Position de l'ellipse
[pType] = 0	Integer	Type de positionnement Si Ptype = 0 : On passe un rectangle en paramètre, l'ellipse remplit ce rectangle pX1,Y1 : Point Haut-Gauche du rectangle pX2,Y2 : Point Bas-Droite du rectangle Si Ptype = 1 : on passe le centre et les rayons de l'ellipse en paramètre pX1,pY1 : Centre de l'ellipse pX2 : Rayon horizontal pY2 : Rayon vertical
[pBackColor] = -1	Long	Couleur de remplissage Laisser -1 pour une ellipse transparente
[pPenColor] = vbBlack	Long	Couleur de la ligne
[pPenWidth] = 1	Long	Epaisseur de la ligne
[pPenStyle] = 0	Long	Style du trait ---> Voir la section Valeurs des paramètres pour la liste des types de traits.
[pArcArray]	Variant	Tableau de 4 coordonnées formant une droite limitant l'ellipse
[pRegion] = 0	String	Nom de la région à créer automatiquement

Exemple :

Dessiner une ellipse au milieu du contrôle

```
' Dessin d'une ellipse rouge d'épaisseur 2, centrée au mileu du contrôle, remplie de vert,
'   et de rayons 1/5è de la taille du contrôle
lReturn = climg.DrawEllipse(Image0.Width / 2, Image0.Height / 2,Image0.Width / 5
 ,Image0.Height / 5, 1, vbGreen, vbRed, 2)
```

V-E - DrawPolygon

Dessine un polygone.
Paramètres :

Paramètre	Type	Explication
pPoints	Variant	Tableau de points formant le polygone : Array(X1,Y1,X2,Y2,X3,Y3,...)
[pBackColor] = -1	Long	Couleur de remplissage Laisser -1 pour un rectangle transparent.
[pPenColor] = vbBlack	Long	Couleur de la ligne
[pPenWidth] = 1	Long	Epaisseur de la ligne
[pPenStyle] = 0	Long	Style du trait ---> Voir la section Valeurs des paramètres pour la liste des types de traits.
[pRegion] = 0	String	Nom de la région à créer automatiquement

V-F - DrawPolyBezier

Dessine une courbe de Bézier.
Pour tracer une courbe de Bézier il faut au moins 4 points.
Le nombre total de points doit être : 4 + Nb * 3
On dessine alors (Nb+1) courbes de béziers.
Paramètres :

Paramètre	Type	Explication
pPoints	Variant	Tableau de points formant le polygone : Array(X1,Y1,X2,Y2,X3,Y3,...)
[pPenColor] = vbBlack	Long	Couleur de la ligne
[pPenWidth] = 1	Long	Epaisseur de la ligne
[pPenStyle] = 0	Long	Style du trait ---> Voir la section Valeurs des paramètres pour la liste des types de traits.

V-G - DrawNewFont

Paramètres :

Définit la police de caractères

Paramètre	Type	Explication
pHeight	Long	Taille du texte (12, 20, ...)
[pAngle] = 0	Long	Angle d'inclinaison en degrés
[pWeight] = 400	Long	Epaisseur (400 = normal ; 700 = gras)
[pItalic] = False	Boolean	Italique Oui/Non
[pUnderscore] = False	Boolean	Souligné Oui/Non
[pStrikeOut] = False	Boolean	Barré Oui/Non
[pFontName] = "Arial"	String	Nom de la police (Arial, ...)

La police de caractères définie est utilisée jusqu'à ce qu'on en crée une autre.
Il n'y qu'une seule police de caractères définie à la fois.
Exemple :

Définition d'une nouvelle police de caractère

```
' Taille de 30 / Gras / Italique / Comic sans MS
climg.DrawNewFont 30, 0, 700, True, False, False, "Comic sans MS"
```

V-H - DrawResetFont

Rétablit la police de caractère d'origine.
C'est la police de caractère par défaut de windows.

V-I - DrawText

Dessine du texte.
Paramètres :

Paramètre	Type	Explication
pText	String	Texte à écrire Peut contenir des retours à la ligne (vbCrLf)
pX1,pY1,[pX2],[pY2]	Long	Position du texte. Si toutes les coordonnées sont renseignées alors on place le texte dans le rectangle défini par les deux points pX1,pY1 et pX2,pY2 Si pX2 et pY2 sont nuls alors on place le texte sur le point pX1,pY1 Si pY2 =0 alors on place le texte par rapport à pX1,pY1 sur une largeur égale à pX2 Si on a défini un angle pour la police de caractères alors on ne peut que positionner le texte sur le point pX1,pX2
[pBackColor] = -1	Long	Couleur de fond (-1 si transparent)
[pPenColor] = 0	Long	Couleur du texte
[pAlignHoriz] = 0	Integer	Alignement horizontal - 0 : Centre - 1 : Gauche - 2 : Droite
[pAlignVert] = 0	Integer	Alignement vertical - 0 : Centre - 1 : Haut - 2 : Bas
[pWordBreak] = Faux	Boolean	Retour à la ligne si le texte est trop grand Ne fonctionne que si on a défini un rectangle pour positionner le texte
[pPercent] = 100	Integer	Pourcentage pour affichage translucide (0 : invisible ; 100:normal)

VI - Les fonctions pour gérer la liste d'images

VI-A - ImageListAdd

Ajoute une image à la liste d'image.
Paramètres :

Paramètre	Type	Explication
pName	String	Identifiant de l'image
[pFile]=""	String	Chemin du fichier de l'image Si aucun fichier n'est passé en paramètre alors on sauvegarde l'image du contrôle dans la liste d'image
[pWidth]=0 [pHeight]=0	Long	Taille de l'image Si pWidth et pHeight sont nulles toutes les deux l'image conserve sa taille Si seulement une des deux tailles est nulles alors l'autre est calculée en en conservant les proportions de l'image d'origine.
[pTransform]	Variant	Liste des transformations. On peut définir une transformation unique ou plusieurs transformations. Pour définir plusieurs transformations passer un tableau en paramètre.
[pParam]	Variant	Liste des paramètres des transformations. Chaque transformation peut éventuellement accepter un paramètre. Il doit y avoir autant de paramètres que de transformations, même si certaines transformations demandées n'en requièrent pas.
[pTranspcolor] = -1	Long	Couleur de transparence Cette couleur sera utilisée pour remplir les zones sans image après transformation. Par exemple après une rotation de 30° il y a des zones qui n'ont pas de couleurs.
[pThumbNail] = Faux	Boolean	Lecture de la miniature intégrée à un Jpeg (Nécessite GDI+)

VI-B - ImageListExists

Teste si l'image existe déjà dans la liste.
La fonction renvoie Vrai si l'image est déjà dans la liste
Paramètres :

Paramètre	Type	Explication
pName	String	Identifiant de l'image

VI-C - ImageListDel

Supprime une image de la liste d'image.
Paramètres :

Paramètre	Type	Explication
pName	String	Identifiant de l'image

VI-D - PaintImage

Affiche une image de la liste d'images.
Paramètres :

Paramètre	Type	Explication
pName	String	Identifiant de l'image
pX1,pY1,[pX2],[pY2]	Long	Position de l'image Si pX2 et pY2 sont nuls alors on conserve la taille de l'image Si pX2 = 0 ou pY2 =0 alors on calcule la dimension manquante par rapport à l'autre en conservant le rapport hauteur/largeur. Si pImgSizeMode = acOLESizeAutoSize : pX1,pY1 : Position de l'image pX2 : Largeur de l'image pY2 : Hauteur de l'image Si pImgSizeMode = acOLESizeZoom/ acOLESizeStretch/acOLESizeClip : pX1,Y1 : Point Haut-Gauche du rectangle contenant l'image pX2,Y2 : Point Bas-Droite du rectangle contenant l'image
[pTranspcolor] = -1	Integer	Couleur de transparence : les points de cette couleur ne seront pas dessinés
[pImgSizeMode] = acOLESizeStretch	Integer	Type d'affichage de l'image (Echelle par défaut)
[pImgPictureAlignment] = 2	Integer	Position de l'image (centrée par défaut)
[pFactX]=0 [pFactY]=0	Single	Facteurs d'agrandissement/Réduction de l'image (0.5 , 2 , ...)
[pTransform]=Null	Variant	Tableau de transformations
[pParam]=Null	Variant	Tableau de paramètres des transformations
[pPercent]=100	Integer	Pourcentage pour affichage transclucide (0:invisible;100:normal)
[pRegion]=""	String	Region à ajouter automatiquement
[pTranspColorRegion] = -1	Long	Couleur à éventuellement utiliser pour déterminer la région Si -1 : la région est un rectangle de coordonnées pX1,pY1,pX2,pY2 Si >= 0 : la région est définie par les points de couleur différente de ce paramètre

VI-E - ImageListChangeColor

Change une couleur d'une image de la liste d'images.
Paramètres :

Paramètre	Type	Explication
pName	String	Identifiant de l'image
pOldColor	Long	Ancienne couleur
pNewColor	Long	Nouvelle couleur

VI - Les fonctions pour gérer les régions

VI-A - AddEllipticRegion

Ajoute une région elliptique.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region
pX1,pY1,pX2,pY2	Long	Position de l'ellipse
[pType] = 0	Long	Type de positionnement Si Ptype = 0 : On passe un rectangle en paramètre, l'ellipse remplit ce rectangle pX1,Y1 : Point Haut-Gauche du rectangle pX2,Y2 : Point Bas-Droite du rectangle Si Ptype = 1 : on passe le centre et les rayons de l'ellipse en paramètre pX1,pY1 : Centre de l'ellipse pX2 : Rayon horizontal pY2 : Rayon vertical

VI-B - AddPolygonRegion

Ajoute une région polygonale.
Pensez à fermer le polygone pour créer une région.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region
pPoints	Variant	Tableau de points formant le polygone Array(X1,Y1,X2,Y2,X3,Y3,...)

VI-C - AddRectangleRegion

Ajoute une région rectangulaire.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region
pX1,pY1	Long	Point Haut-Gauche du rectangle
pX2,pY2	Long	Point Bas-Droite du rectangle

VI-D - CombineRegion

Combine des régions.
Combine les régions pRegion1 et pRegion2 : renvoie le résultat dans pRegionOut
Paramètres :

Paramètre	Type	Explication
pRegionOut	String	Identifiant de la region de destination
pRegion1	String	Identifiant de la region source 1
pRegion2	String	Identifiant de la region source 2
[pCombineType] = OR	String	Type de combinaison - OR : Ajout des régions - AND : Intersection des régions - DIFF : Différence (Region source 1) - (Region source 2) - XOR : Region composée des zones non communes aux 2 régions - COPY : Copie (Région Source 1)

VI-E - DelRegion

Supprime une région.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region à supprimer

VI-F - DelAllRegions

Supprime toutes les régions.

VI-G - FrameRegion

Dessine un trait qui encadre la région.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region
[pColor] = vbBlack	Long	Couleur du trait
[pWidth] = 1	Long	Epaisseur du trait

VI-H - FillRegion

Remplit une region d'une couleur ou de hachures.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region
[pColor] = vbBlack	Long	Couleur de remplissage ou des hachures
[Ptype] = 1	Long	Type de remplissage 0 : Hachures horizontales 1 : Hachures verticales 2 : Hachures diagonales à droite 3 : Hachures diagonales à gauche 4 : Hachures en croix 5 : Hachures en croix diagonales 6 : Couleur de remplissage

VI-I - GetMouseRegion

Identifie la region située sur un point.
Les paramètres sont de type Single car la fonction est utilisée principalement
à partir des événements de la souris qui renvoient des paramètres de type Single
Paramètres :

Paramètre	Type	Explication
pX	Single	Position X du point
pY	Single	Position Y du point
[BeginFromLast]=False	Boolean	Si Vrai, commence la recherche par la dernière région ajoutée.
[pExcludeRgn]	Variant	Tableau de régions à exclure de la recherche (ex : array("Rgn1",Rgn2")).
[pIncludeRgn]	Variant	Tableau de régions à inclure dans la recherche (ex : array("Rgn1",Rgn2")).

La fonction renvoie le nom de la région située sur le point précisé.
Exemple :

Recherche l'identifiant de la région sous le curseur de la souris

```
Private Sub Image0_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)<br/>
 Dim sRegion As String       ' Region survolée<br/>
 sRegion = climg.GetMouseRegion(X, Y)    ' Récupère l'identifiant de la région survolée<br/>
End Sub
```

VI-J - RegionInRegion

Teste l'intersection entre une région et une liste de régions.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region à tester
[pRegionArray]	Variant	Permet de limiter la recherche à un tableau de régions Si ce paramètre est omis le test à lieu avec toutes les régions sauf bien sûr la région pRegion
[pFull]=Faux	Boolean	- Vrai : Test si la région pRegion est totalement incluse dans une autre région. - Faux : Test si la région pRegion a une zone commune avec une autre région;

La fonction renvoie le nom de la région en collision avec pRegion.
Si plusieurs régions sont en collision avec pRegion le résultat est le nom de la première région dans l'ordre de création.
Exemple :

Test si sRegion est entrée en collision avec Region1 ou Region2

```
Dim lRegionCollision as string
lRegionCollision = climg.RegionInRegion(sRegion,Array(Region1,Region2))
```

VI-K - AddAutoRegion

Détermine une région définie par les points qui sont ou ne sont pas de la couleur spécifiée.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la region à créer
pColor	Long	Couleur déterminant la région
[pIncludeColor]=Faux	Boolean	- Vrai : La région contient les points de couleur pColor. - Faux : La région contient les points de couleur différente de pColor.
[pPointX,pPointY]	Long	Coordonnées d'un point : la région sera composée des points adjacents à celui-ci. (c'est l'équivalent du pot de peinture de Paint)

Exemple :

Création d'une région nommée 'RegionTest' qui inclue tous les points blancs de l'image

```
climg.AddAutoRegion "RegionTest",vbWhite,True
```

VI-L - SetFormRegion

Limite l'affichage de formulaire à une région.
Paramètres :

Paramètre	Type	Explication
pForm	Form	Formulaire à limiter.
[pColor] = vbBlack	Long	Couleur déterminant la région.
[pIncludeColor] = Faux	Boolean	- Vrai : La région contient les points de couleur pColor. - Faux : La région contient les points de couleur différente de pColor.
[pRegion] = ""	String	Utilise une région déjà définie. Il faut que cette région ait été définie avec la propriété gFormCoord = Vrai.

Exemple1 :

Comment limiter le formulaire à tous les points qui ne sont pas de couleur blanche

```
climg.SetFormRegion Me, vbWhite, False
```

Exemple2 :

Comment limiter le formulaire à une ellipse

```
' Dessin d'une ellipse de couleur rouge qui remplit le contrôle
climg.DrawEllipse 0, 0, climg.ImgCtrlWidth, climg.ImgCtrlHeight, , , 255, 2
' Passage en coordonnées formulaire pour la région
climg.FormCoord = True
' Ajoute une région formée par l'ellipse
```

Comment limiter le formulaire à une ellipse

```
climg.AddEllipticRegion "FormRegion", 0, 0, climg.ImgCtrlWidth, climg.ImgCtrlHeight
' On n'oublie pas de repasser dans le mode de coordonnées par défaut
climg.FormCoord = False
' On redessine l'image pour voir l'ellipse
climg.Repaint
' On limite le formulaire à la région formée par l'ellipse
climg.SetFormRegion Me, , , "FormRegion"
```

VI-M - ResetFormRegion

Rétablit l'affichage du formulaire complet.
Paramètres :

Paramètre	Type	Explication
pForm	Form	Formulaire à rétablir.

Remarque : Le formulaire est automatiquement rétabli lorsque la classe est libérée

VI-N - GetRectangleRegionHeight

Renvoit la hauteur en twips du rectangle contenant une région.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la région.

VI-O - GetRectangleRegionWidth

Renvoit la largeur en twips du rectangle contenant une région.
Paramètres :

Paramètre	Type	Explication
pRegion	String	Identifiant de la région.

VIII - Autres fonctions relatives à l'image

VIII-A - SaveToBmp

Sauvegarde l'image dans un fichier BMP.
Paramètres :

Paramètre	Type	Explication
pFichier	String	Chemin d'un fichier BMP.

VIII-B - SaveToJpg / SaveToGif/ SaveToPng / SaveToTif

Sauvegarde l'image dans un fichier JPG, GIF, PNG ou TIFF.
(Nécessite Gdi+)
Paramètres :

Paramètre	Type	Explication
pFichier	String	Chemin d'un fichier.
[pQuality]=80	Long	Qualité du fichier Jpeg (de 0 à 100)

VIII-C - SaveToEmf

Sauvegarde le bitmap dans un fichier EMF.
Paramètres :

Paramètre	Type	Explication
pFichier	String	Chemin du fichier. **Uniquement SaveToJpg**
[pTransparentColor]=-1	Long	Couleur de transparence

VIII-D - GetTextLength

Renvoie la taille d'un texte.
Utile pour connaître la taille du texte pour savoir si on assez de place pour l'écrire.
Paramètres :

Paramètre	Type	Explication
pText	String	Texte à mesurer.
pWidth	Long	Renvoie la largeur du texte.
pHeight	Long	Renvoie la hauteur du texte.
[pWordBreak]=Faux	Boolean	Retour à la ligne si texte trop long. Si pWordBreak = Vrai alors il faut passer pWidth en paramètre pour que Access sache sur quel largeur on compte écrire.

VIII-E - ImgToClipBoard

Transfert l'image principal vers le presse-papiers
C'est le bitmap en mémoire qui est transféré, pas le contenu de l'objet.
On peut donc transférer une image vers le presse-papiers sans la dessiner à l'écran.

VIII-F - CaptureScreen

Capture d'écran.
On peut capturer vers le presse-papiers, vers un fichier ou vers un contrôle image.
Paramètres :

Paramètre	Type	Explication
[pClipboard] = Faux	Boolean	Transfert vers le presse-papiers : oui/non
[pFile] = ""	String	Transfert vers un fichier : chemin du fichier
[pControl] = Nothing	Object	Transfert vers un contrôle : contrôle ou formulaire
[pX1],[pY1],[pX2],[pY2]	Long	Rectangle délimitant la zone à capturer Si on omet tous ces paramètres alors la fenêtre complète est capturée Si on omet pX2 alors on capture jusqu'au bord droit de la fenêtre Si on omet pY2 alors on capture jusqu'au bord bas de la fenêtre
[pWidth],[pHeight]	Long	Taille de l'image après capture Si pX2 et pY2 sont nuls alors on conserve la taille de la fenêtre Si pX2 = 0 ou pY2 =0 alors on calcule la dimension manquante en gardant les proportions de la fenêtre.
[pAntialiase] = Faux	Boolean	Appliquer un filtre anti-aliasing sur l'image Utile uniquement si on a définit pWidth et pHeight
[pActiveWindow] = Faux	Boolean	Capturer uniquement la fenêtre active Si Faux alors on capture tout l'écran.
[pJpeg] = Faux	Boolean	Capture dans un fichier JPEG si Vrai, sinon dans un fichier BMP (Nécessite Gdi+)
[pJpegQuality] = 80	Long	Qualité du fichier Jpeg (de 0 à 100)

VIII-G - GetMousePixelColor

Renvoie la couleur d'un point
Les paramètres sont de type Single car la fonction est utilisée principalement
à partir des événements de la souris qui renvoient des paramètres de type Single
Paramètres :

Paramètre	Type	Explication
pX,pY	Single	Coordonnées du point

la fonction renvoie la couleur du point dans un Long.

VIII-H - GetPixels

Renvoie un tableau de pixels avec les couleurs RGB
Paramètres :

Paramètre	Type	Explication
pPixels	Tableau de Byte	Tableau de points RGB
[pOneDimension]=False	Boolean	Si Vrai les données sont renvoyées dans un tableau à une dimensions.

		Peut être utile pour accélérer les traitements d'images.

Exemple :

Lire les couleurs des pixels

```
' Tableau pour recevoir les pixels
Dim lPixels() As Byte
Dim lCptX as long, lCptY as long
' Lecture des pixels de l'image
Climg.GetPixels lPixels
' On boucle sur les pixels
For lCptX = 1 to UBound(lPixels(), 2)
 For lCptY = 1 to UBound(lPixels(), 3)
  ' Composante bleue = lPixels(1, lCptX, lCptY)
  ' Composante verte = lPixels(2, lCptX, lCptY)
  ' Composante rouge = lPixels(3, lCptX, lCptY)
 Next
Next
```

VIII-I - SetPixels

Redessine l'image à partir un tableau de pixels avec les couleurs RGB
Paramètres :

Paramètre	Type	Explication
pPixels	Tableau de Byte	Tableau de points RGB
[pOneDimension]=False	Boolean	Si Vrai les données sont fournies dans un tableau à une dimensions. Peut être utile pour accélérer les traitements d'images.
[pWidth], [pHeigth]	Long	Si tableau à une dimensions, il est nécessaire de donner les dimensions de l'image.

Exemple :

Appliquer un filtre rouge sur l'image

```
' Tableau pour recevoir les pixels
Dim lPixels() As Byte
Dim lCptX as long, lCptY as long
' Lecture des pixels de l'image
Climg.GetPixels lPixels
' On boucle sur les pixels
For lCptX = 1 to UBound(lPixels(), 2)
 For lCptY = 1 to UBound(lPixels(), 3)
' On ne garde que la couleur rouge en mettant les deux autres composantes à zéro
  ' Annule la composante bleue
  lPixels(1, lCptX, lCptY) = 0
  ' Annule la composante verte
  lPixels(2, lCptX, lCptY) = 0
 Next
Next
' On réinjecte les couleurs dans l'image
climg.SetPixels lPixels
' On affiche l'image à l'écran
climg.Repaint
```

VIII-J - SetXPTheme

Active ou désactive le thème XP pour les contrôles.
Permet de réduire le scintillement de l'image (pour WinXP/Acc2003)
La fonction s'applique sur toute l'application, mais seulement sur les contrôles (les barres de menu, barres de titre, ... conservent le thème XP).
Voir la fonction **SetDoubleBufferXP** pour réduire les scintillements sans désactiver le thème XP.
Utilisez cette fonction **même si vous avez déjà désactivé le thème XP dans les options de la base de données**.
Paramètres :

Paramètre	Type	Explication
[pActive]=True	Boolean	Vrai pour activer. Faux pour désactiver.

Exemple :

Réduire les scintillements si la version d'Access est 2003

```
' Désactive le thème XP pour réduire les scintillements si la version d'Access est 2003
If SysCmd(acSysCmdAccessVer) = "11.0" Then climg.SetXPTheme False
```

VIII-K - SetDoubleBufferXP

Permet de réduire le scintillement de l'image (pour WinXP/Acc2003)
La fonction de double buffer est **gourmande en ressource**.
Envisagez l'utilisation de la fonction **SetXPTheme pour désactiver le thème XP**.
Paramètres :

Paramètre	Type	Explication
pForm	Access.Form	Formulaire dont on veut réduire le scintillement Privilégiez un formulaire de type indépendant (**Fen. Indépendante = Oui**). Si la fonction est appliquée sur un formulaire avec la propriété **Fen. Indépendante = Non**, alors le double buffer s'applique sur toute l'application et peut faire scintiller d'autres objets.
[pActive]=True	Boolean	Vrai pour activer. Faux pour désactiver.
[pMore]=False	Boolean	Si les scintillements persistent, essayer de mettre ce paramètre à Vrai.

Exemple :

Réduire les scintillements si la version d'Access est 2003

```
' Activer le double buffer pour réduire les scintillements si la version d'Access est 2003
If SysCmd(acSysCmdAccessVer) = "11.0" Then climg.SetDoubleBufferXP Me
```

IX - Les fonctions pour le formulaire

IX-A - DragForm

Permet le déplacement du formulaire
Cette fonction permet de déplacer le formulaire même s'il n'a pas de barre de titre.
Exemple :

Comment déplacer le formulaire en cliquant sur l'image

```
' Remarque : Ne fonctionne pas sur l'événement Click
Private Sub Image0_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    climg.DragForm
End Sub
```

IX-B - FormPos

Positionne le formulaire
Cette fonction permet de positionner le forumaire :
- soit par rapport à l'écran (uniquement formulaires indépendants)
- soit par rapport à le fenêtre de base de données
Paramètres :

Paramètre	Type	Explication
pForm	Form	Formulaire à positionner
[pImgPictureAlignment] = 2	Integer	Position du formulaire
[pTopMost] = Faux	Boolean	Vrai pour afficher le formulaire au premier plan (formulaire indépendant uniquement)
[pScreen] = Faux	Boolean	- Vrai : Positionnement par rapport à l'écran - Faux : Positionnement par rapport à la fenêtre base de données

Exemple :

Comment centrer le formulaire au chargement

```
Private Sub Form_Load()
 climg.FormPos Me, 2
End Sub
```

IX-C - FormSizeAble

Formulaire redimensionnable même sans barre de titre
Ajoute une fine bordure redimensionnable, mais pas de barre de titre
Paramètres :

Paramètre	Type	Explication
pForm	Form	Formulaire à rendre redimensionnable
pSizeAble	Boolean	- Vrai pour rendre le formulaire redimensionnable - Faux pour annuler

Exemple :

Comment rendre le formulaire redimensionnable au chargement et sans barre de titre

```
Private Sub Form_Load()
 climg.FormSizeAble Me, True
End Sub
```

IX-D - RedimCtrlInitialize

Initialise la liste des contrôles pour un futur redimensionnement
Voir RedimCtrl pour un exemple.
Paramètres :

Paramètre	Type	Explication
pForm	Form	Formulaire contenant les contrôles

IX-E - RedimCtrl

Redimensionne les contrôles en fonction des dimensions d'origine et de la taille actuelle du formulaire
A tester sur des formulaires simples, le résultat risque d'être illisible
Paramètres :

Paramètre	Type	Explication
pForm	Form	Formulaire contenant les contrôles à redimensionnés

Exemple :

Comment redimensionner les contrôles d'un formulaire lorsqu'on redimensionne le formulaire

```
' Au chargement du formulaire
Private Sub Form_Load()
    ' On ajoute les contrôles à redimensionnés
    climg.RedimCtrlInitialize Me
End Sub
' Au redimensionnement du formulaire
Private Sub Form_Resize()
    ' Redimensionnement des contrôles en fonction de la nouvelle taille du formulaire
    climg.RedimCtrl Me
end sub
```

X - Les fonctions de conversion

X-A - CtrlToFormX et CtrlToFormY

Converti depuis les coordonnées contrôle vers les coordonnées formulaire
Utile pour déterminer la position d'un point du contrôle dans le formulaire afin d'y déplacer un contrôle
Paramètres :

Paramètre	Type	Explication
pX ou pY	Long	Abcisse ou ordonnées à convertir

La fonction retourne la valeur convertie en twips.
Exemple :

Dessiner un rectangle et y centrer une zone de texte

```
' Déclaration des coordonnées
Dim X1 As Long, Y1 As Long, X2 As Long, Y2 As Long
' Initialisation du contrôle image
climg.SetImgCtrl Me.Image0
' On rempli l'image de blanc
climg.FillColor vbWhite
' Coordonnées du rectangle
X1 = 1000
Y1 = 1000
X2 = 1000 + Étiquette0.Width * 2
Y2 = 1000 + Étiquette0.Height * 2
' Dessine le rectangle
climg.DrawRectangle X1, Y1, X2, Y2
' Conversion en coordonnées formulaire
X1 = climg.CtrlToFormX(X1)
Y1 = climg.CtrlToFormY(Y1)
X2 = climg.CtrlToFormX(X2)
Y2 = climg.CtrlToFormY(Y2)
' Centrage du contrôle dans l'emplacement
X1 = X1 + ((X2 - X1) - Étiquette0.Width) / 2
Y1 = Y1 + ((Y2 - Y1) - Étiquette0.Height) / 2
' Déplacement du contrôle
Étiquette0.Left = X1
Étiquette0.Top = Y1
' Redessine le contrôle
climg.Repaint
```

X-B - ImgToFormX et ImgToFormY

Converti depuis les coordonnées image vers les coordonnées formulaire
Utile pour déterminer la position d'un point de l'image dans le formulaire afin d'y déplacer un contrôle
Paramètres :

Paramètre	Type	Explication
pX ou pY	Long	Abcisse ou ordonnées à convertir **Exception : ces coordonnées sont exprimées en pixels!**
[pImgOrigCoord] = False	Boolean	Si Vrai convertir à partir du système de coordonnées de l'image d'origine

La fonction retourne la valeur convertie **en pixels**

Exemple :

Centrer une zone de texte dans un élément du dessin

```
    ' Déclaration des coordonnées
    Dim X1 As Long, Y1 As Long, X2 As Long, Y2 As Long
    ' Initialisation du contrôle image
    climg.SetImgCtrl Me.Image0
    ' Emplacement de l'étiquette en coordonnées image
    ' Ces coordonnées ont été déterminées sur l'image d'origine et sont donc en pixels

' Elles correspondent à une boîte rectangulaire dessinée sur l'image et dans laquelle on veux placer notre étiq
    X1 = 364: Y1 = 13: X2 = 522: Y2 = 55
    ' Conversion en coordonnées formulaire
    ' pImgOrigCoord = True car les coordonnées ont été déterminées sur l'image d'origine et en pixels
    X1 = climg.ImgToFormX(X1, True)
    Y1 = climg.ImgToFormY(Y1, True)
    X2 = climg.ImgToFormX(X2, True)
    Y2 = climg.ImgToFormY(Y2, True)
    ' Centrage du contrôle dans l'emplacement
    X1 = X1 + ((X2 - X1) - Étiquette0.Width) / 2
    Y1 = Y1 + ((Y2 - Y1) - Étiquette0.Height) / 2
    ' Déplacement du contrôle
    Étiquette0.Left = X1
    Étiquette0.Top = Y1
```

X-C - ImgToCtrlX et ImgToCtrlY

Converti depuis les coordonnées image vers les coordonnées contrôle
Paramètres :

Paramètre	Type	Explication
pX ou pY	Long	Abcisse ou ordonnées à convertir **Exception : ces coordonnées sont exprimées en pixels!**
[pDecalage] = True	Boolean	Calcul avec décalage (Position de l'image dans le contrôle) Définir à Vrai pour convertir une position, à Faux pour convertir une taille.
[pPixel] = True	Boolean	Si Vrai renvoie le résultat en pixels, sinon en twips (par défaut)
[pImgOrigCoord] = False	Boolean	Si Vrai convertir à partir du système de coordonnées de l'image d'origine

X-D - CtrlToImgX et CtrlToImgY

Converti depuis les coordonnées contrôle vers les coordonnées image
Paramètres :

Paramètre	Type	Explication
pX ou pY	Long	Abcisse ou ordonnées à convertir
[pDecalage] = True	Boolean	Calcul avec décalage (Position de l'image dans le contrôle) Définir à Vrai pour convertir une position, à Faux pour convertir une taille.
[pImgOrigCoord] = False	Boolean	Si Vrai convertir dans le système de coordonnées de l'image d'origine

La fonction retourne la valeur convertie **en pixels**

X-E - CtrlToImgArray

Converti un tableau depuis les coordonnées contrôle vers les coordonnées image
Paramètres :

Paramètre	Type	Explication
pPoints	Variant	Tableau de points (X1,Y1,X2,Y2,X3,Y3,...) en twips
[pImgOrigCoord] = False	Boolean	Si Vrai convertir vers le système de coordonnées de l'image d'origine

La fonction retourne un tableau de valeurs converties **en pixels**

X-F - ImgToCtrlArray

Converti un tableau depuis les coordonnées image vers les coordonnées contrôle
Paramètres :

Paramètre	Type	Explication
pPoints	Variant	Tableau de points (X1,Y1,X2,Y2,X3,Y3,...) **en pixel**
[pImgOrigCoord] = False	Boolean	Si Vrai convertir depuis le système de coordonnées de l'image d'origine

La fonction retourne un tableau de valeurs converties **en twips**

X-G - TwipsToPixelX et TwipsToPixelY

Converti les Twips en Pixels
Paramètres :

Paramètre	Type	Explication
pTwipsX pTwipsY	Long	Valeur à convertir (**en Twips)**

La fonction retourne la valeur convertie **en Pixels**

X-H - PixelToTwipsX et PixelToTwipsY

Converti les Pixels en Twips
Paramètres :

Paramètre	Type	Explication
pPixelsX pPixelsY	Long	Valeur à convertir (**en Pixels)**

La fonction retourne la valeur convertie **en Twips**

X-I - TwipsToCm

Converti les Twips en Cm

Paramètres :

Paramètre	Type	Explication
pTwips	Long	Valeur à convertir (**en Twips**)

La fonction retourne la valeur convertie **en Cm**

X-J - CmToTwips

Converti les Cm en Twips
Paramètres :

Paramètre	Type	Explication
pCm	Single	Valeur à convertir (**en Cm**)

La fonction retourne la valeur convertie **en Twips**

XI - Les autres fonctions utiles

XI-A - SetHandCursor

Applique un curseur en forme de main

XI-B - ResetCursor

Rétabli le curseur par défaut

XI-C - Split97

Split une liste en tableau
Utile pour access 97 qui n'a pas de fonction Split.
Paramètres :

Paramètre	Type	Explication
pStrg	String	Chaîne de caractère à diviser
pSep	String	Séparateur

La fonction retourne les données sous forme de tableau.

XI-D - fMax

Renvoie la valeur maxi entre a et b
Paramètres :

Paramètre	Type	Explication
a b	Variant	Valeurs à comparer

La fonction retourne la plus élevée des deux valeurs.

XI-E - fMin

Renvoie la valeur mini entre a et b
Paramètres :

Paramètre	Type	Explication
a b	Variant	Valeurs à comparer

La fonction retourne la moins élevée des deux valeurs.

XI-F - DialogColor

Boîte dialoge pour le choix d'une couleur
Paramètres :

Paramètre	Type	Explication
pHandle	Long	Handle du parent de la boîte de dialogue Utilisez Me.hWnd ou application.HwndAccessApp

La fonction retourne -1 si aucune couleur n'est choisie.

XI-G - LongToRGB

Conversion code couleur Long vers RGB
Paramètres :

Paramètre	Type	Explication
pLong	Long	Valeur de la couleur
pRed	Long	Composante rouge de la couleur
pGreen	Long	Composante verte de la couleur
pBlue	Long	Composante bleue de la couleur

XI-H - ColorBrightness

Change la luminosité d'une couleur
Paramètres :

Paramètre	Type	Explication
pLong	Long	Valeur de la couleur
pPercent	Integer	Pourcentage de luminosité

Exemple :

Transformer du rouge en rose pâle

```
dim lRose as long
' Avec 180% de luminosité le rouge devient rose
lRose = ClImg.ColorBrightness(vbRed, 180)
```

XI-I - RGBtoHLS

Conversion RGB vers HLS
Le système RGB donne les composantes Rouge/Vert/Bleue d'une couleur.
Le système HLS donne les composantes Teinte/Luminosité/Saturation d'une couleur.
Paramètres :

Paramètre	Type	Explication
pRed	Long	Composante Rouge
pGreen	Long	Composante Verte
pBlue	Long	Composante Bleue
pHue	Long	Teinte
pLightness	Long	Luminosité
pSaturation	Long	Saturation

XI-J - HLStoRGB

Conversion HLS vers RGB

Le système HLS donne les composantes Teinte/Luminosité/Saturation d'une couleur.
Le système RGB donne les composantes Rouge/Vert/Bleue d'une couleur.
Paramètres :

Paramètre	Type	Explication
pHue	Long	Teinte
pLightness	Long	Luminosité
pSaturation	Long	Saturation
pRed	Long	Composante Rouge
pGreen	Long	Composante Verte
pBlue	Long	Composante Bleue

XI-K - FontSizeToHeight

Converti de la taille de police pour la fonction CreateNewFont
Permet de convertir une taille de police (par exemple MonContrôle.FontSize) pour créer avec la fonction DrawNewFont une police de caractère de même taille que celle d'un contrôle.
Renvoie la valeur convertie pour CreateNewFont
Paramètres :

Paramètre	Type	Explication
pFontSize	Long	Valeur à convertir

Exemple :

Créer une police identique à celle d'une étiquette Etiqu1

```
ClImg.DrawNewFont ClImg.FontSizeToHeight(Etiqu1.FontSize), 0, Etiqu1.FontWeight, Etiqu1.FontItalic,
 Etiqu1.FontUnderline, False, Etiqu1.FontName
```

XII - Les propriétés

XII-A - FormWidthInit et FormHeightInit

Largeur et hauteur initiales du formulaire
Type Long en Lecture/Ecriture.

XII-B - ImgForm

Formulaire parent de l'image
Type Form en Lecture seule.

XII-C - ApplicationPath

Chemin de l'application
Type String en Lecture seule.
Utile pour access97 car Currentproject.Path n'existe pas

XII-D - ImgX1,ImgY1,ImgX2,ImgY2

Position de l'image dans le contrôle
Type Long en Lecture seule.
Dépendent des propriétés ImgSizeMode et ImgPictureAlignment du contrôle
Si l'image est affichée en mode Zoom ou Découpage il se peut qu'elle n'occupe pas tout le contrôle.
Ces propriétés permettent de connaître la position de l'image visible dans le contrôle.

XII-E - ImgOrigCoord

Travail dans le système de coordonnées de l'image d'origine
Type Boolean en Ecriture seule.
A utiliser si les coordonnées utilisée ont été récupérée sur l'image d'origine. C'est à dire en pixel sur l'image non redimensionnée.

XII-F - FormCoord

Travail dans le système de coordonnées du formulaire
Type Boolean en Ecriture seule.
A utiliser pour définir une région en passant les coordonnées formulaire en paramètre.
C'est à dire en twips, la position d'un contrôle par exemple.

XII-G - ImgWidth et ImgHeight

Taille de l'image en twips
Type Long en Lecture seule.
C'est la taille de l'image en mémoire avant redimensionnement par Access pour affichage d'après la propriété ImgSizeMode.

XII-H - ImgCtrlWidth et ImgCtrlHeight

Taille du contrôle en twips

Type Long en Lecture seule.
C'est la taille du contrôle ou du formulaire.

XII-I - CtrlTranspColor

Couleur de transparence du contrôle
Permet de voir le fond du formulaire à travers les parties transparentes de l'image.
Donnez la valeur -1 pour annuler la transparence.
Exemple :

Rendre la couleur blanche transparente

```
' Définit le blanc comme couleur transparente
climg.CtrlTranspColor = vbWhite
```

XII-J - Antialiase

Lissage du dessin.
Mettre à vrai pour activer l'antialising.

XIII - Valeurs des paramètres.

Les noms des énumérations sont donnés entre parenthèses (A partir d'Access 2000)
Paramètre **pImgSizeMode** : Mode d'affichage
Ce paramètre a les même valeurs que la propriété SizeMode d'un contrôle image :
- acOLESizeClip (ModeSizeClip) : Découpage
- acOLESizeStretch (ModeSizeStretch) : Echelle
- acOLESizeZoom (ModeSizeZoom) : Zoom
- acOLESizeAutoSize (ModeSizeAutoSize) : Placement sur un point sans redimensionnement

Paramètre **pImgPictureAlignment** : Alignement
Ce paramètre a les même valeurs que la propriété PictureAlignment d'un contrôle image :
- 0 (AlignTopLeft) : Supérieur gauche
- 1 (AlignTopRight) : Supérieur droit
- 2 (AlignCenter) : Centre
- 3 (AlignBottomLeft) : Inférieur gauche
- 4 (AlignBottomRight) : Inférieur droit

Paramètre **pPenStyle** : Style de trait
- 0 (PenSolid) : Trait plein
- 1 (PenDash) : Tirets
- 2 (PenDot) : Pointillés
- 3 (PenDashDot) : Tiret - point
- 4 (PenDashDotDot) : Tiret - point - point

Paramètre **pType** : Type de remplissage
- 0 (HatchHorizontal) : Hachures horizontales
- 1 (HatchVertical) : Hachures verticales
- 2 (HatchDiagRight) : Hachures diagonales à droite
- 3 (HatchDiagLeft) : Hachures diagonales à gauche
- 4 (HatchCross) : Hachures en croix
- 5 (HatchDiagCross) : Hachures en croix diagonales
- 6 (FillWithColor) : Couleur de remplissage

Paramètre **pEllipseType** : Type de coordonnées des ellipses
- 0 (EllipseRectangle) : Les coordonnées définissent un rectangle contenant l'ellipse.
- 1 (EllipseCenter) : Les coordonnées définissent le centre et les rayons de l'ellipse.

Paramètre **pAlignHoriz** : Alignement horizontal
- 0 (AlignCenter) : Centré.
- 1 (AlignLeft) : A gauche.
- 2 (AlignRight) : A droite

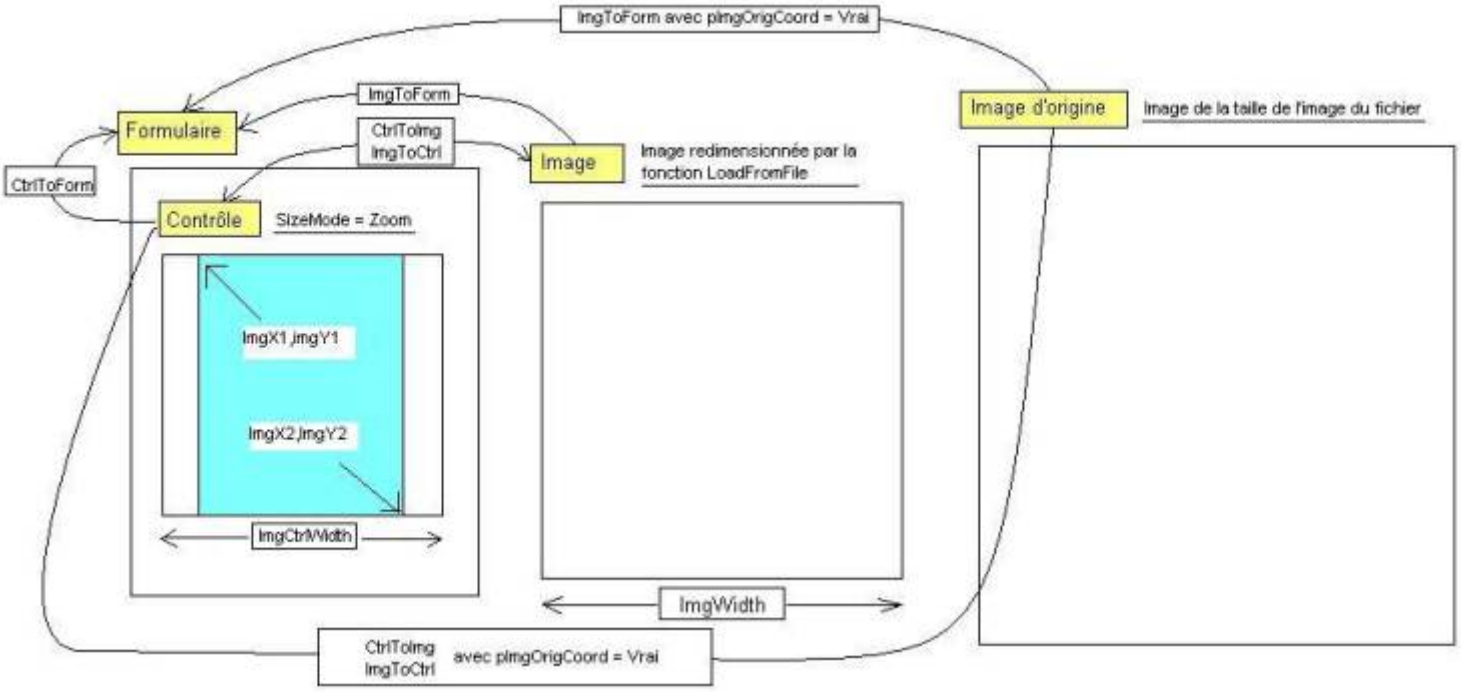
Paramètre **pAlignVert** : Alignement vertical
- 0 (AlignCenter) : Centré.
- 1 (AlignTop) : En haut.
- 2 (AlignBottom) : En bas.

Paramètres **pTransformation** et **pParam** : Transformations

Transformation	Paramètre	Description
BRIGHTNESS	Pourcentage 100 = normal	Luminosité de l'image
FLIPHORIZ	Aucun	Retourne l'image horizontalement
FLIPVERT	Aucun	Retourne l'image verticalement
ROTATELEFT	Aucun	Rotation 90° vers la gauche
ROTATERIGHT	Aucun	Rotation 90° vers la droite
ROTATE	Angle en degrés	Rotation d'un angle quelconque
FASTROTATE	Angle en degrés	Rotation rapide d'un angle quelconque Windows > 98 requis.
ANTIALIASE	Aucun	Applique un filtre antialiasing sur l'image
BLUR	Aucun	Effet de flou
INVERSE	Aucun	Inverse les couleurs (négatif)
GRAY	Aucun	Affichage l'image en niveaux de gris
SEPIA	Aucun	Affichage l'image couleur Sepia (=vieilles photos)
SMOOTH	Aucun	Adouci l'image
SHARPEN	Coëfficient (>=5)	Accentue l'image
EMBOSS	Aucun	Fait ressortir les contours

XIV - Les différents systèmes de coordonnées

Il y a 4 systèmes de coordonnées utilisés par la classe :
- **Coordonnées image d'origine**
Si on charge un fichier avec la fonction LoadFromFile, ce sont les coordonnées en pixels dans l'image contenue dans le fichier.
- **Coordonnées image**
Ce sont les coordonnées en pixels dans l'image en mémoire sur laquelle on dessine.
Si on a redimensionné l'image au chargement alors elles sont différentes des coordonnées image d'origine.
- **Coordonnées contrôle**
Ce sont les coordonnées dans le contrôle en twips.
L'image peut être redimensionnée (propriété SizeMode) ou positionnée (propriété PictureAlignment)
de telle sorte que les coordonnées du contrôle ne correspondent pas à celles de l'image.
- **Coordonnées formulaire**
Ce sont les coordonnées dans le formulaire



XV - C'est quoi GDI / GDI+ ??

GDI et GDI+ sont des librairies graphiques.
En d'autres termes ce sont des fonctions que l'on peut appeler depuis nos programmes en ayant pris soin auparavant de les déclarer : ce sont les fameuses API.
La librairie GDI est intégrée à toutes les versions de Windows, elle contient des fonctions de dessins basiques et ne sait gérer que le format Bitmap (BMP)
Pour pallier le manque de fonctionnalités de la librairie GDI, Microsoft a sorti une autre librairie : GDI+ (ou GdiPlus).
Cette seconde librairie est beaucoup plus puissante et reconnaît en autre le format Jpeg et les données Exif intégrées aux images Jpeg, notamment par les appareils photos numériques.
En ce qui concerne la classe ClImage, elle met à profit GDI+ pour :
- Enregistrer les images au format Jpeg/Gif/Png/Tiff;
- Lire les miniatures intégrées aux images Jpeg.

Cette dernière fonctionnalité est très utile pour afficher rapidement une petite miniature d'une photo; c'est d'ailleurs grâce à ces miniatures intégrées que les appareils photos (peu puissants) arrivent à afficher rapidement une prévisualisation des photos directement sur l'appareil.
Par contre GDI+ n'est pré-installée que sur Windows XP.
Il est possible de télécharger la librairie pour d'autres systèmes (Windows 2000; Windows 98; Windows ME; Windows NT; Windows XP).
Lien vers la libraire en téléchargement sur Microsoft.com
Elle s'appelle GdiPlus.dll et il suffit de la placer dans le répertoire de l'application ou dans le répertoire système.

XVI - Conclusion

Cette documentation référence les fonctions et les propriétés sans pour autant expliquer comment débuter pour utiliser la classe.
Pour des explications plus pratiques tournez vous vers les tutoriels.
Imprimez le PDF pour avoir la documentation papier à portée de main...
Un grand **MERCI** à tous ceux qui m'ont encouragé sur le forum et par MP.
Merci à Xo et à cafeine pour leur relecture.

XVII - Les téléchargements

Télécharger la classe au format texte *(HTTP)*
Ensuite insérez le contenu du fichier dans un module de classe.
Télécharger la base d'exemples au format ACCESS 2000 *(HTTP)*