

# PureBasic et GDI+

par Denis

Date de publication : 16/03/08

Dernière mise à jour : 06/02/09

GDI+ succède à GDI, *Graphic Device Interface*, interface pour périphérique graphique. GDI+ constitue une couche placée entre l'application et les dispositifs graphiques (écrans, imprimantes, etc) et permet de manipuler facilement le 2D.

I - Avant-propos.....	3
II - Introduction.....	3
III - GdiPlus version 1.0 et 1.1.....	4
IV - Configuration requise.....	4
V - Les conventions du wrapper et de la doc GDI+ 1.0.....	5
V-1 - Les fonctions GDI+.....	5
V-2 - Les couleurs GDI+.....	5
VI - Les identificateurs GUID.....	5
VI-1 - Les GUID.....	5
VI-2 - Property Set Identifiers.....	6
VI-3 - Image Frame Dimensions.....	6
VI-4 - Image File Format.....	6
VI-5 - Image Encoder.....	7
VI-6 - Effect GUIDs.....	8
VII - Les systèmes de coordonnées.....	9
VII-1 - Systèmes de coordonnées.....	9
VII-2 - Transformations et systèmes de coordonnées.....	10
VIII - Liens.....	11
VIII-1 - Archive.....	11
VIII-2 - Aide en ligne.....	11
IX - Remerciements.....	11

## I - Avant-propos

Cet article est un aperçu de la documentation jointe à l'archive que vous pouvez télécharger dans la section [archive](#). Cette archive contient également plus de 600 exemples dans tous les domaines, et un fichier d'aide au format **chm** qui décrit environ 600 fonctions.

## II - Introduction

Les différents OS Windows utilisent GDI pour tout ce qui touche au graphisme. GDI est l'abréviation de **Graphics Device Interface**, et est une DLL avec de nombreuses fonctions. Avec le temps GDI a évolué.

L'arrivée de GDI+ version 1.0 avec Windows XP a modifié l'approche de la programmation du graphiste. GDI+ 1.0 est une DLL qui possède 609 fonctions. GDI+ constitue une couche placée entre l'application et les dispositifs graphiques (graphic devices).

GDI+ permet (entre autres) :

- Graphisme vectoriel en 2D avec gestion possible des coordonnées en nombres réels simple précision.
- Gestion de l'anti-crênelage (anti-aliasing) des images
- Nombreux formats d'images supportés (BMP, GIF, JPEG, EXIF, PNG, TIFF)
- Gestion de l'échelle des graphiques, on peut faire un zoom d'une image avec peu de code
- Gestion de la transparence
- Gestion différents modes de combinaison de couleur
- Gestion des textures, des brush (différents types de dégradés linéaires)
- Gestion des pen (gestion de différents formats de traits comme les pointillés etc.)
- Possibilité de définir les embouts des lignes soit avec des formes prédéfinies soit personnalisées.
- Gestion des régions pour obtenir les effets voulus (exclusion, intersection, union et d'autres)
- Gestion des Path (chemin) qui délimitent les contours des formes géométriques, ces formes pouvant être remplies avec des brush dédiées.
- Gestion des Container graphiques

GDI+ permet de manipuler facilement le 2D, sans avoir à sélectionner dans le dc, le pen, la font, la brush etc puis restitue le dc dans son état initial avant de retourner (c'est assez pénible avec GDI). On peut manipuler les images, les mettre à l'échelle, faire une rotation angulaire, une translation, cisailer ou mixer ces fonctions assez facilement. Les pen disposent de plusieurs fonctions pour faire des tirets etc. On peut définir des embouts prédéfinis pour les lignes comme par exemple des embouts de flèches ou créer des embouts personnalisés. Les brush sont assez nombreuses et diverses, on peut faire des dégradés très facilement en passant la couleur de départ et celle d'arrivée. On peut aussi définir une texture pour une brush à partir d'une image etc. Les régions peuvent être combinées pour obtenir les effets voulus (exclusion, intersection, union et d'autres). Différents types de formes sont réalisables facilement, (courbes de Bézier, camembert, polygones), ces formes pouvant être remplies avec tous les types de brush etc. Il y a aussi la gestion de la transparence avec le mode d'interpolation, composite, de qualité, toutes ces fonctions permettent d'agir sur le rendu final.

GDI+ est orienté objet et Microsoft ne donne presque aucune information sur les différentes fonctions appelées Flat API. Microsoft a développé un wrapper en C++ et aussi pour .Net avec des fonctions propres aux wrapper, la documentation Microsoft est dédiée à ces wrapper.

La documentation GDI+ que j'ai écrite pour PureBasic (fichier chm) fournie avec les exemples est dédiée aux Flat API, c'est-à-dire l'appel direct des fonctions de la DLL, ce que Microsoft déconseille mais qui fonctionne bien sous réserve d'initialiser correctement GDI+, de détruire les objets puis de terminer l'utilisation de GDI+.

Il est obligatoire d'initialiser au moins une fois GDI+ dans son programme (on peut le faire plusieurs fois et c'est même recommandé de le faire avec des threads ou des fonctions encapsulées dans une DLL). A partir du moment où l'on passe le pointeur d'initialisation à la fonction de "fermeture" de GDI+, les objets deviennent inaccessibles.

Reportez-vous à la section d'initialisation de GDI+ du fichier d'aide.

### Quelques sites dédiés à GDI+

 [La page Microsoft dédié à GDI+](#)

 [Le site de José Roca en anglais](#) (excellent site avec diverses rubriques dont GDI+, orienté PowerBasic)

 [Le site de bob powell](#)

### III - GdiPlus version 1.0 et 1.1

GDI+ version 1.0 est arrivé avec Windows XP et a été mis en place sur les plates-formes Windows Server 2003. La version 1.0 est redistribuable et peut être installée sur des versions antérieures à XP (Windows NT 4.0 SP6, Windows 2000, Windows 98 et Windows Me.).

Gdi+ a évolué suite à une faille de sécurité et en 2004 une nouvelle version corrigeant le problème a vu le jour, mais reste néanmoins la version 1.0.

Voici le lien de la page sur la sécurité de Gdi+ 1.0 (à lire) :

 <http://msdn2.microsoft.com/en-us/library/ms995328.aspx>

Et en français ici :

 <http://www.microsoft.com/france/technet/securite/ms04-028.msp>

La sortie de Vista en 2007 a introduit la version 1.1 **non redistribuable**. Cette version possède 630 fonctions au lieu de 609 pour Gdiplus 1.0.

Cette version introduit des effets sur les images comme la saturation, la correction des yeux rouge (photos), la balance des couleurs etc.

Bien que la version 1.1 soit présente avec Vista, la version 1.0 est également présente et c'est cette version 1.0 qui est utilisée (présente dans le dossier system32 de Vista au moins pour la version Premium) lorsque l'on utilise Gdiplus (utilisation des imports ou utilisation de la fonction PureBasic OpenLibrary()). En faisant une recherche sur le disque ou est installé Vista, on trouve gdiplus.dll avec une version de fichier 6xxxxxxx qui indique que c'est la version 1.1 (il faut mettre le pointeur de la souris et attendre l'affichage des infos du fichier).

Je ne sais pas si c'est sur toutes les versions de Vista mais c'est ce que j'ai avec la Version Familiale Premium. J'ai tenté une copie de la version 1.1 dans le dossier system32, j'ai le bien un message que cela s'est bien passé, mais au final, la version reste la 1.0 même avec les droits en écriture.

Cette version ne semble (mais est-ce juste ?) être distribuée qu'avec des applications MS ou avec .Net 3.0 ou le wrapper MS en C pour Gdi+.

Pour ceux qui aurait un système plus ancien que Vista, il est possible d'obtenir la version 1.1 en installant la visionneuse PowerPoint 2003 (PowerPoint Viewer 03), voir le post ici <http://www.xtremevb.com/t98285.html>

### IV - Configuration requise

GDI+ version 1.0 est arrivé avec Windows XP et a été mis en place sur les plates-formes Windows Server 2003. La version 1.0 est redistribuable et peut être installée sur des versions antérieures à XP (Windows NT 4.0 SP6, Windows 2000, Windows 98 et Windows Me.).

Si vous ne l'avez pas, installez-la depuis le site Microsoft :

 <http://www.microsoft.com/downloads/>

Pour PureBasic, j'ai commencé avec la version 4.00 jusqu'à la version 4.10 bêta 3 aujourd'hui (octobre 2007).

Les exemples fonctionnent sans problème sous les version 4.10 bêta 2 et 3 (je n'ai plus les versions antérieures) sous XP et VISTA avec quelques réserves sous VISTA pour certaines fonctions et un comportement différent pour certaines fonctions des métafichiers (voir la rubrique des problèmes et incertitudes).

Le fichier d'installation installe les fichiers **dans une arborescence qu'il faut conserver** sinon les fichiers ne compileront pas. Il détecte la présence des macros de Freak pour l'accès aux COM (Component Object Model) et vous demandera de les télécharger et de les installer si ces fichiers sont absents (message du fichier d'installation après extraction des fichiers). Ces fichiers sont nécessaires au fonctionnement du wrapper.

Il n'est possible d'extraire les fichiers qu'une seule fois, pour les extraire à nouveau, il faut relancer le fichier d'installation. Les fichiers extraits écrasent les fichiers portant le même nom s'ils ne sont pas protégés en écriture. Il est possible de créer la liste des fichiers extraits et de la coller dans le presse-papier.

L'ensemble extrait fait un peu plus de 15 Mo.

## V - Les conventions du wrapper et de la doc GDI+ 1.0

### V-1 - Les fonctions GDI+

Les fonctions de Gdiplus acceptent en général des paramètres en entrée qui sont définis pour chaque fonction dans ce fichier d'aide. Normalement j'ai essayé de suivre les conventions PureBasic pour les pointeurs et adresses. La presque totalité des fonctions retourne une valeur qu'il est indispensable de tester afin de s'assurer du bon déroulement de l'opération. Cette valeur de retour est un élément de l'énumération status. La valeur #Ok est la valeur qui permet de continuer normalement les opérations. Prenons par exemple la fonction suivante :

```
Resultat.l = GdiplusGetFontCollectionFamilyList(*fontCollection, numSought.l, *gpfamilies, @numFound.Long)
```

C'est la variable Resultat qui doit être testée.

**Les paramètres de la fonctions peuvent être :**

- des paramètres d'entrée, c'est-à-dire que le paramètre doit avoir une valeur avant l'appel de la fonction. Le paramètre d'entrée est défini dans la documentation par le sigle anglais in entre crochets [in]
- des paramètres de sortie, c'est-à-dire que le paramètre recevra une valeur lorsque la fonction s'exécutera. Le paramètre de sortie est défini dans la documentation par le sigle anglais out entre crochets [out]
- des paramètres d'entrée et/ou de sortie, c'est-à-dire que le paramètre peut avoir une valeur avant l'appel de la fonction et une valeur en retour. Le paramètre n'a pas obligatoirement les deux types, cela dépend de l'utilisation. Le paramètre d'entrée et/ou de sortie est défini dans la documentation par le sigle anglais in, out entre crochets [in, out]

Par exemple, la fonction **GdiplusGetFontCollectionFamilyList()** possède les trois types de paramètres.

### V-2 - Les couleurs GDI+

Sous Gdi, les couleurs sont des valeurs exprimées par les 3 composantes qui sont le rouge, vert et le bleu. Chaque composante est codée sur 8 bits (un octet) de sorte que chaque couleur peut avoir 256 valeurs (de 0 à 255). Les 3 couleurs sont regroupées sur 32 bits (Dword) avec les 8 bits de poids fort qui ne sont pas utilisés.

La couleur RVB a les 8 bits de poids faibles (b0 à b7) codant le rouge, les bits de b8 à b15 codant le vert et les bits b16 à b23 codant le bleu (appelé bgr). Gdi+ a introduit les couleurs avec la gestion de la transparence. Les 3 couleurs sont exprimées par les 24 premiers bits et la transparence par les 8 bits de poids fort (b24 à b31), appelée couche alpha. La couche alpha valant 0 indique une transparence complète et cette couche valant 255 indique une opacité complète. Sous Gdi+, les composantes rouge et bleue sont inversées par rapport à la convention Gdi.

Pour Gdi+, on peut soit coder les couleurs au format supporté par Gdi+ soit utiliser les couleurs Gdi (par exemple celles prédéfinies sous PB) dont les composantes rouges et bleues devront être inversées.

La procédure **Alpha(color.l)** retourne la valeur de la couche alpha de la couleur.

La procédure **ARGB(color, a.b = 255)** inverse les composantes rouges et bleues pour obtenir le format Gdi+, le deuxième paramètre étant la couche alpha.

La procédure **ARGBEX(color.l)** inverse les composantes rouges et bleues, la couche alpha est conservée. Cette procédure peut être utilisée pour retrouver une valeur Gdi à partir d'une couleur Gdi+ en conservant la couche alpha.

La macro **MAKECOLOR(a,r,g,b)** du fichier Gdiplus.pbi permet de passer les 4 composantes chacune étant un octet.

La procédure **SetAlpha(color, a.b = 255)** modifie la couche alpha de la valeur color sans inverser l'ordre des couleurs. Le second paramètre est la couche alpha qui sera mise à 255 (opacité complète) s'il est absent.

## VI - Les identificateurs GUID

### VI-1 - Les GUID

Le **GUID** est un nombre unique de 128 bits affecté à un objet lors de sa création.

Ce nombre est également utilisé pour identifier chaque composant O.L.E (objet Linking and Embedding). Les exemples fournis utilisent les Macros de Freak "**Framework For Creating COM Objects**" pour définir et utiliser les **GUID** Gdi+.

Les **GUID** Gdi+ peuvent être définis sous la forme de Datas en PureBasic, à vous de modifier les exemples si vous désirez utiliser les datas plutôt que les macros de Freak. Ces constantes peuvent être utilisées avec la structure **GUID** prédéfinie en PureBasic.

Les **GUID** Gdi+ commun à la version 1.0 et 1.1 sont déclarées dans le fichier GdiPlusImaging.h (Documentation MS SDK) et les définitions des **GUID** pour les macros Framework de Freak sont définies dans le fichier **gdiplus\_GUID.pbi**.

Les **GUID** Gdi+ spécifiques à la version 1.1 (pour les effets) sont déclarées dans le fichier Gdipluseffects.h (Documentation MS SDK)

## VI-2 - Property Set Identifiers

Les constantes suivantes représentent les **GUID** qui identifient les ensembles de propriétés des images.

Définitions au format "Framework For Creating COM Objects"

**(Gdiplus 1.0 et Gdiplus 1.1):**


```

; Identificateur FormatIDImageInformation
; Indique un ensemble de propriétés concernant les données de l'image.
DefineGUID(FormatIDImageInformation, $E5836CBE, $5EEF, $4F1D, $AC, $DE, $AE, $4C, $43, $B6, $08, $CE)

; Identificateur FormatIDJpegAppHeaders
; Indique un ensemble de propriétés pour les en-têtes des fichiers JPEG.
DefineGUID(FormatIDJpegAppHeaders, $1C4AFDCD, $6177, $43CF, $AB, $C7, $5F, $51, $AF, $39, $EE, $85)
    
```

## VI-3 - Image Frame Dimensions

Les formats graphiques **GIF** et **TIFF** permettent de stocker plusieurs images dans un fichier d'image. Les différentes images d'un fichier **GIF** sont utilisées pour l'animation, ainsi les images sont dites images de dimension temporelle. Les différentes images d'un fichier **TIFF** sont utilisées typiquement en tant que pages séparées, ainsi les images sont dites images de dimension paginée.

 *Gdi+ 1.0 ne supporte pas les gif à séquence d'images (images multiples).*

Les constantes suivantes représentent les **GUID** qui identifient les dimensions temporelles et paginées.

Définitions au format Framework For Creating COM Objects

**(Gdiplus 1.0 et Gdiplus 1.1)**

```

; Identificateur FrameDimensionPage
DefineGUID(FrameDimensionPage, $7462DC86, $6180, $4C7E, $8E, $3F, $EE, $73, $33, $A7, $A4, $83)

; Identificateur FrameDimensionResolution
DefineGUID(FrameDimensionResolution, $84236F7B, $3BD3, $428F, $8D, $AB, $4E, $A1, $43, $9C, $A3, $15)

; Identificateur FrameDimensionTime
DefineGUID(FrameDimensionTime, $6AEDBD6D, $3FB5, $418A, $83, $A6, $7F, $45, $22, $9D, $C8, $72)
    
```

## VI-4 - Image File Format

La fonction **GdiplusGetImageRawFormat()** retourne un identifiant global unique (**GUID**) qui indique le format de l'image. Les constantes suivantes représentent les **GUID** qui identifient ces formats d'image.

Définitions au format "Framework For Creating COM Objects"

**(Gdiplus 1.0 et Gdiplus 1.1)**

```

; Identificateur ImageFormatBMP
DefineGUID(ImageFormatWMF, $B96B3CAD, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatEMF
DefineGUID(ImageFormatEMF, $B96B3CAC, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatEXIF
DefineGUID(ImageFormatEXIF, $B96B3CB2, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatGIF
DefineGUID(ImageFormatGIF, $B96B3CB0, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatIcon
DefineGUID(ImageFormatIcon, $B96B3CB5, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatJPEG
DefineGUID(ImageFormatJPEG, $B96B3CAE, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatMemoryBMP
DefineGUID(ImageFormatMemoryBMP, $B96B3CAA, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatPNG
DefineGUID(ImageFormatPNG, $B96B3CAF, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatTIFF
DefineGUID(ImageFormatTIFF, $B96B3CB1, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatUndefined
DefineGUID(ImageFormatUndefined, $B96B3CA9, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)

; Identificateur ImageFormatWMF
DefineGUID(ImageFormatBMP, $B96B3CAB, $0728, $11D3, $9D, $7B, $00, $00, $F8, $1E, $F3, $2E)
    
```

## VI-5 - Image Encoder

Les fonctions **GdipSaveImageToFile()**, **GdipSaveImageToStream()**, **GdipSaveAddImage()** et **GdipSaveAdd()** utilisent un paramètre basé sur une structure **EncoderParameters** qui contient un Tableau de variables **EncoderParameter**. Chaque **EncoderParameter** a un membre de type **GUID** qui indique la catégorie du paramètre. Les constantes suivantes représentent les **GUID** qui identifient les différentes catégories de paramètre pour les codecs d'images (encodeur/décodeur).

Définitions au format "Framework For Creating COM Objects"  
**(Gdiplus 1.0 et Gdiplus 1.1)**

```

; Identificateur CodecIImageBytes
; Identifie les réglages mémoire de l'encoder.
DefineGUID(CodecIImageBytes, $025D1823, $6C7D, $447B, $BB, $DB, $A3, $CB, $C3, $DF, $A2, $FC)

; Identificateur EncoderChrominanceTable
; Identifie les réglages de la table de chrominance de l'encoder.
DefineGUID(EncoderChrominanceTable, $F2E455DC, $09B3, $4316, $82, $60, $67, $6A, $DA, $32, $48, $1C)

; Identificateur EncoderColorDepth
; Identifie les réglages de la profondeur de couleur de l'encodeur.
DefineGUID(EncoderColorDepth, $66087055, $AD66, $4C7C, $9A, $18, $38, $A2, $31, $0B, $83, $37)

; Identificateur EncoderCompression
; Identifie les réglages de la compression de l'encodeur.
DefineGUID(EncoderCompression, $E09D739D, $CCD4, $44EE, $8E, $BA, $3F, $BF, $8B, $E4, $FC, $58)

; Identificateur EncoderLuminanceTable
; Identifie les réglages de la table de luminance de l'encoder.
DefineGUID(EncoderLuminanceTable, $EDB33BCE, $0266, $4A77, $B9, $04, $27, $21, $60, $99, $E7, $17)

; Identificateur EncoderQuality
; Identifie les réglages de la qualité de l'encoder.
DefineGUID(EncoderQuality, $1D5BE4B5, $FA4A, $452D, $9C, $DD, $5D, $B3, $51, $05, $E7, $EB)
    
```

```

; Identificateur EncoderRenderMethod
; Identifie les réglages des méthodes de rendu de l'encoder, y compris si une image devrait être affichée en mode
DefineGUID(EncoderRenderMethod, $6D42C53A, $229A, $4825, $8B, $B7, $5C, $99, $E2, $B9, $A8, $B8)

; Identificateur EncoderSaveFlag
; Identifie les réglages du drapeau de sauvegarde de l'encoder.
DefineGUID(EncoderSaveFlag, $292266FC, $AC40, $47BF, $8C, $FC, $A8, $5B, $89, $A6, $55, $DE)

; Identificateur EncoderScanMethod
; Identifie les réglages du mode de scan de l'encoder, y compris si une image est entrelacée ou non entrelacée.
DefineGUID(EncoderScanMethod, $3A4E2661, $3109, $4E56, $85, $36, $42, $C1, $56, $E7, $DC, $FA)

; Identificateur EncoderTransformation
; Identifie les réglages de transformation de l'encodeur.
DefineGUID(EncoderTransformation, $8D0EB2D1, $A58E, $4EA8, $AA, $14, $10, $80, $74, $B7, $B6, $F9)

; Identificateur EncoderVersion
; Identifie la version logiciel de l'encodeur.
DefineGUID(EncoderVersion, $24D18C76, $814A, $41A4, $BF, $53, $1C, $21, $9C, $CC, $F7, $97)
    
```

### (Gdiplus 1.1 uniquement):

```

; Identificateur EncoderColorSpace
DefineGUID(EncoderColorSpace, $AE7A62A0, $EE2C, $49D8, $9D, $7, $1B, $A8, $A9, $27, $59, $6E)

; Identificateur EncoderImageItems
DefineGUID(EncoderImageItems, $63875E13, $1F1D, $45AB, $91, $95, $A2, $9B, $60, $66, $A6, $50)

; Identificateur EncoderSaveAsCMYK
DefineGUID(EncoderSaveAsCMYK, $A219BBC9, $A9D, $4005, $A3, $EE, $3A, $42, $1B, $8B, $B0, $6C)
    
```

## VI-6 - Effect GUIDs

Les constantes suivantes représentent les GUID qui identifient ces formats d'image.

Définitions au format "Framework For Creating COM Objects"

### (Gdiplus 1.1 uniquement):

```

; Identificateur BlurEffectGuid
DefineGUID(BlurEffectGuid, $633C80A4, $1843, $482B, $9E, $F2, $BE, $28, $34, $C5, $FD, $D4)

; Identificateur BrightnessContrastEffectGuid
DefineGUID(BrightnessContrastEffectGuid, $D3A1DBE1, $8EC4, $4c17, $9F, $4C, $EA, $97, $AD, $1C, $34, $3D)

; Identificateur ColorBalanceEffectGuid
DefineGUID(ColorBalanceEffectGuid, $537E597D, $251E, $48dA, $96, $64, $29, $CA, $49, $6B, $70, $F8)

; Identificateur ColorCurveEffectGuid
DefineGUID(ColorCurveEffectGuid, $DD6A0022, $58E4, $4A67, $9D, $9B, $D4, $8E, $B8, $81, $A5, $3D)

; Identificateur ColorLUTEEffectGuid
DefineGUID(ColorLUTEEffectGuid, $A7CE72A9, $0F7F, $40d7, $B3, $CC, $D0, $C0, $2D, $5C, $32, $12)

; Identificateur ColorMatrixEffectGuid
DefineGUID(ColorMatrixEffectGuid, $718F2615, $7933, $40E3, $A5, $11, $5F, $68, $FE, $14, $DD, $74)

; Identificateur HueSaturationLightnessEffectGuid
DefineGUID(HueSaturationLightnessEffectGuid, $8B2DD6C3, $EB07, $4D87, $A5, $F0, $71, $08, $E2, $6A, $9C, $5F)

; Identificateur LevelsEffectGuid
DefineGUID(LevelsEffectGuid, $99C354EC, $2A31, $4f3A, $8C, $34, $17, $A8, $03, $B3, $3A, $25)

; Identificateur RedEyeCorrectionEffectGuid
DefineGUID(RedEyeCorrectionEffectGuid, $74D29D05, $69A4, $4266, $95, $49, $3C, $C5, $28, $36, $B6, $32)

; Identificateur SharpenEffectGuid
    
```

```
DefineGUID(SharpenEffectGuid, $63CBF3EE, $C526, $402C, $8F, $71, $62, $C5, $40, $BF, $51, $42)
; Identificateur TintEffectGuid
DefineGUID(TintEffectGuid, $1077AF00, $2848, $4441, $94, $89, $44, $AD, $4C, $2D, $7A, $2C)
```

## VII - Les systèmes de coordonnées

### VII-1 - Systèmes de coordonnées

GDI+ utilise trois systèmes ou espaces de coordonnées :

- Les coordonnées **world** (World coordinates) ou coordonnées **universelles**.
- Les coordonnées de **page** (Page coordinates).
- Les coordonnées **device** (Device coordinates) ou coordonnées de périphérique.

**⚠** Par défaut, GDI+ utilise pour ces trois systèmes de coordonnées, une position zéro ( $x=0, y=0$ ) se trouvant dans le coin supérieur gauche de la surface à dessiner.

Les coordonnées universelles (world coordinates) sont les coordonnées que vous passez aux fonctions, par exemple les coordonnées d'une ligne. Ce mode n'est pas lié à la réalité de l'affichage ou de l'impression.

Les coordonnées de page se réfèrent au système de coordonnées utilisé par une surface de dessin, comme par exemple une fenêtre ou une ImageGadget en PureBasic. En règle générale, il s'agit de la zone client mais on peut dessiner sur la zone non client.

Les coordonnées de périphérique sont les coordonnées utilisées par le périphérique physique sur lequel on va dessiner, tel qu'un écran ou une imprimante.

Lorsque l'on utilise la fonction **GdipDrawLineI(\*graphics, \*pen, x1, y1, x2, y2)**, les points  $x1, y1, x2, y2$  passés à cette fonction se trouvent dans l'espace de coordonnées universelles. Avant que GDI+ ne puisse tracer la ligne à l'écran, les coordonnées subissent une série de transformations.

Par défaut, il n'y a aucune transformation et on peut considérer, pour un travail à l'écran par exemple, que les coordonnées sont exprimées en pixels.

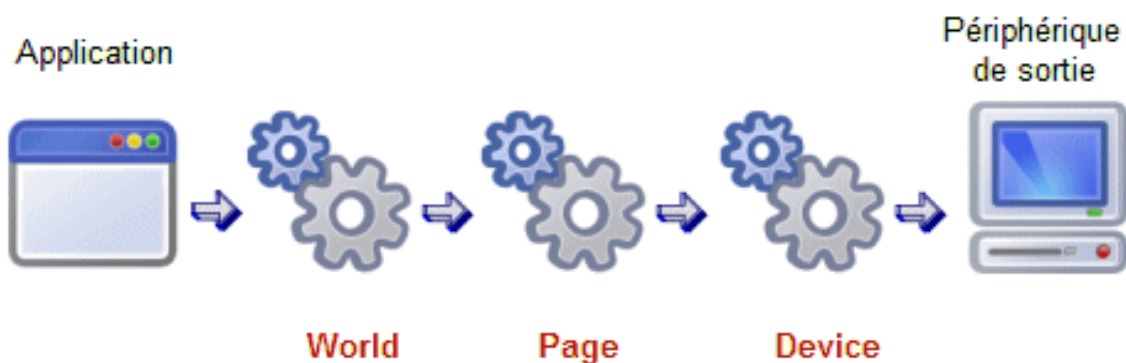
Mais les transformations peuvent être appliquées par le programmeur ou par GDI+.

Une des transformations, appelée **transformation universelle**, convertit des coordonnées universelles en coordonnées de page.

Une autre transformation, appelée **transformation de page**, convertit les coordonnées de page en coordonnées de périphérique.

On peut schématiser le cheminement de ces transformations avec le dessin suivant :

### Les transformations de coordonnées



Gdi+ permet de définir les coordonnées passées aux fonctions sous la forme de nombres entiers ou de nombres réels (simple précision). Beaucoup de fonctions admettent en paramètres les nombres entiers ou réels, mais il s'agit bien de fonctions différentes. Les fonctions se terminant par un I utilisent des nombres entiers.

La fonction **GdipDrawLineI(\*graphics, \*pen, x1, y1, x2, y2)** a les paramètres  $x1, y1, x2, y2$  qui sont des nombres entiers.

La fonction **GdipDrawLine(\*graphics, \*pen, x1, y1, x2, y2)** a les paramètres x1, y1, x2, y2 qui sont des nombres réels.

Selon les fonctions, Les coordonnées peuvent être transmises par des valeurs de type **Point**, **PointF**, **Rect** et **RectF**. Avec GDI+, on dessine principalement sur un objet appelé graphic. Le programmeur peut contrôler ce graphique avec plusieurs fonctions.

## VII-2 - Transformations et systèmes de coordonnées

Si on souhaite dessiner à partir de coordonnées d'origine différente de x=0, y=0, il faut appliquer une translation en x et y sachant que l'on peut avoir une translation différente pour x et y.

La fonction **GdipTranslateWorldTransform(\*graphics, dx.f, dy.f, order.I)** permet d'introduire cette translation (reportez-vous à la fonction pour le paramètre order).

**!** Avec cette fonction, la translation s'applique à tous les objets du graphique.

L'exemple utilise un graphique à partir du Hdc de la fenêtre.

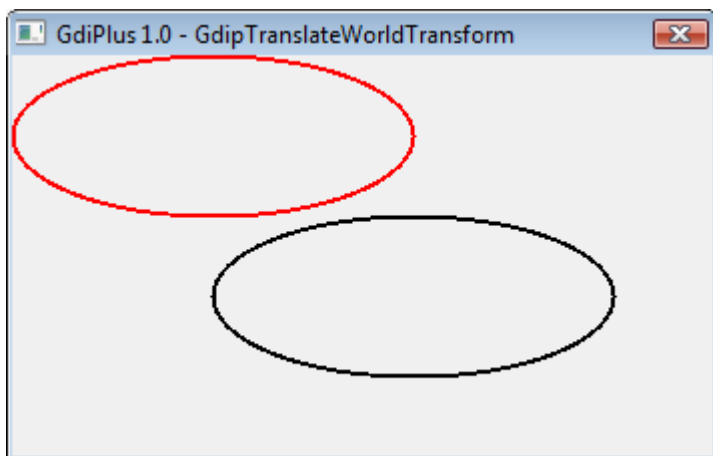
Une première ellipse est dessinée en rouge à la position x=0 et y = 0 sans introduire de translation.

Ensuite, on applique au graphique une translation en x de 100 et y de 80.

Une seconde ellipse est dessinée en noir à la position x=0 et y = 0. Cette ellipse est réellement positionnée en x = 100 et y = 80 du fait de la translation.

Puisque l'on a pas modifié l'unité du graphique, les valeurs sont en pixels.

**Voici ce que donne l'exemple de translation :**



*translation*

Les coordonnées des points définissant l'ellipse (l'ellipse est définie par les coordonnées du rectangle qui l'englobe) dans les trois espaces de coordonnées sont comme ceci :

Espaces de coordonnées	Coordonnées rectangle
World	0,0,200,80
Page	100, 80, 200, 80
Device	100, 80, 200, 80

Puisque l'unité de mesure est le pixel, les coordonnées device sont identiques aux coordonnées de page. Si vous utilisez une unité de mesure différente du pixel (par exemple le pouces (inch)), alors les coordonnées device seront différentes des coordonnées de page.

Les fonctions **GdipTransformPoints()** et **GdipTransformPointsI()** permettent de convertir un tableau de points d'un espace de coordonnée à un autre (la 1ère fonction pour les coordonnées en réels, la seconde pour les nombres entiers).

Les fonctions **GdipGetPageUnit()**, **GdipSetPageUnit()**, **GdipGetPageScale()** et **GdipSetPageScale()** permettent de manipuler les transformations de page. Deux fonctions en lecture seule, **GdipGetDpiX()** et **GdipGetDpiY()**, permettent d'examiner les points par pouce horizontaux et verticaux du périphérique d'affichage.

Vous pouvez utiliser la fonction **GdipSetPageUnit()** pour spécifier une unité de mesure voulue au graphique, par exemple le millimètre au lieu du pixel (unité par défaut pour l'écran). Toutes les coordonnées seront transcrites en millimètre quel que soit le périphérique de sortie.

N'oubliez pas que les pen (par exemple) ont une unité qui peut être différente de celle du graphique, les résultats peuvent aussi varier en fonction du pen!

Il est également possible de modifier l'échelle du graphique.

La fonction **GdipSetPageScale()** permet de définir le facteur de mise à l'échelle pour la transformation de page du graphique.

Les transformations géométriques :

Il est possible d'effectuer des transformations géométriques sur le graphique (la translation (**translate**), la mise à l'échelle (**scale**), la rotation (**rotate**)). Nous avons abordé les deux premières (Gdi+ utilise des matrices pour réaliser les opérations).

La fonction **GdipRotateWorldTransform()** permet de faire une rotation angulaire du graphique avec un angle exprimé en degrés.

Les différentes fonctions vues ci-dessus permettent de s'affranchir de l'utilisation directe des matrices. Ces fonctions s'appliquent à l'ensemble du graphique et par conséquent à tous les objets qu'il contient.

Gdi+ permet d'appliquer des transformations géométriques sur un objet et de nombreux objets possèdent des fonctions de transformations simples qui ne s'appliquent qu'à eux-même.

- La transformation est dite **locale** lorsqu'elle n'affecte qu'un objet en particulier, par exemple une brush, un texte, une image etc. L'effet n'est pas appliqué au graphique complet.
- La transformation est dite **globale** lorsqu'elle affecte le graphique et tous les objets qu'il contient.
- La transformation est dite **composite** lorsqu'il s'agit de plusieurs transformations globales enchaînées

## VIII - Liens

### VIII-1 - Archive

Téléchargez l'**archive**.

Le fichier d'installation installe les fichiers **dans une arborescence qu'il faut conserver** sinon les fichiers ne compileront pas. Il détecte la présence des macros de Freak pour l'accès aux COM (Component Object Model) et vous demandera de les télécharger et de les installer si ces fichiers sont absents (message du fichier d'installation après extraction des fichiers). Ces fichiers sont nécessaires au fonctionnement du wrapper.

Il n'est possible d'extraire les fichiers qu'une seule fois, pour les extraire à nouveau, il faut relancer le fichier d'installation. Les fichiers extraits écrasent les fichiers portant le même nom s'ils ne sont pas protégés en écriture. Il est possible de créer la liste des fichiers extraits et de la coller dans le presse-papier.

L'ensemble extrait fait un peu plus de 15 Mo.

### VIII-2 - Aide en ligne

Vous pouvez également consulter l' **aide en ligne**.

## IX - Remerciements

**Je tiens à remercier particulièrement les membres suivants du forum français :**

**Erix14** qui a rédigé les exemples et la doc des sections Bitmap et Text (j'ai adapté un peu la doc).

Il a rédigé l'exemple de l'horloge qui illustre les fonctions de gestion mémoire sous GDI+.

Il a trouvé un problème de "fuite mémoire" avec la fonction **GdipDrawString()**.

Pour nos échanges sur GDI+.

Je suis parti de son wrapper et de celui de **Flype** et j'ai adapté le tout pour mon besoin. J'ai retenu (en majorité) l'utilisation des pointeurs du wrapper de Flype.

**Flype**, pour son wrapper qui m'a mis le pied à l'étrier.

Ses différents exemples.

**Fred** pour avoir créé PureBasic, très simple, pratique, complet (assez pour moi).

**Je tiens à remercier les membres suivants du forum anglais :**

**SFSxOI** pour avoir adapté mon wrapper pour Gdi+ 1.1.

Pour nos échanges sur différents codes.


**Einander** et **Netmaestro** pour leurs différents exemples. Il y a 3 exemples d'**Einander** qui fonctionnent sans le wrapper.

**Freak** pour avoir développé des macros pour l'accès aux COM (Component Object Model). Je n'ai utilisé que la gestion des GUID (définition et accès).

Voir ici  <http://www.purebasic.fr/english/viewtopic.php?t=22132>

**Je tiens à remercier tout particulièrement :**

**José ROCA** pour son site et informations sur Gdi+ (en anglais), orienté PowerBasic, pour nos échanges par mails auxquels il a toujours répondu.

 **Le site de José Roca en anglais** (excellent site avec diverses rubriques dont GDI+, orienté PowerBasic)

Je tiens également à remercier **ArHacKnIdE** pour sa relecture de l'article.