Dessiner sous MS Access - AccessPaint

par Charles A.

Date de publication : 25/10/2005

Dernière mise à jour : 25/10/2005

Réalisation pas à pas d'un formulaire de dessin. Catégorie d'article : Aller plus loin avec Access Niveau : Confirmé Compétences : . mise en oeuvre d'API . gestion des événements souris . gestion du presse-papier et des fichiers

I - Introduction
II - Rendu final de l'application
III - Principes
IV - Construction du formulaire
    IV-A - La conception du formulaire
    IV-B - Mission impossible ?
    IV-C - Mais comment ça marche ?
V - Nos amis les API
    V-A - Comment déclarer une API ?
    V-B - L'utilisation dans le code
    V-C - Comment trouver la bonne API ?
VI - Dessinons !
    VI-A - Le code
    VI-B - Les objets graphiques
    VI-C - Fonctionnements des figures
        VI-C-1 - Carré Plein ou Vide
        VI-C-2 - Disque ou Cercle
        VI-C-3 - La ligne brisée
    VI-D - Gestion de la souris
VII - Sauvegardons
VIII - Conclusion

I - Introduction

Cet article répond à deux questions posées sur le forum : comment dessiner dans un formulaire Access ou comment modifier une photo.

MS Access n'est pas du tout destiné à révéler les futurs artistes qui sommeillent en vous, tout simplement parce qu'aucun de ses objets n'est approprié à la création graphique.

> ***Pour réaliser rapidement une application sommaire de dessin, je vous recommande fortement d'utiliser Visual Basic (et VB.net aussi) avec lesquels ils vous sera possible de réaliser en très peu de lignes de codes ce que nous parvenons à faire ici à grand peine.***

Un autre choix pertinent pourrait être de réaliser un OCX dans VB, pour pouvoir l'utiliser par la suite dans MS Access.

Mais, nous sommes ici pour dire que rien ne fait peur à notre SGBD favori, alors tentons l'impossible et allons exprimer notre sensibilité picturale sur nos formulaires si familiers.

II - Rendu final de l'application

Qu'allons-nous réaliser ici ?

Une application de type paint préhistorique à l'intérieur d'un formulaire, regardez ce que cela peut donner ...

une très belle oeuvre d'un artiste ayant souhaité garder l'anonymat.

III - Principes

Comment dessiner ?

Voici les principes fondateurs de l'application :

. **Capter les déplacements de la souris** sur l'application

. **Détecter l'état des boutons de la souris**

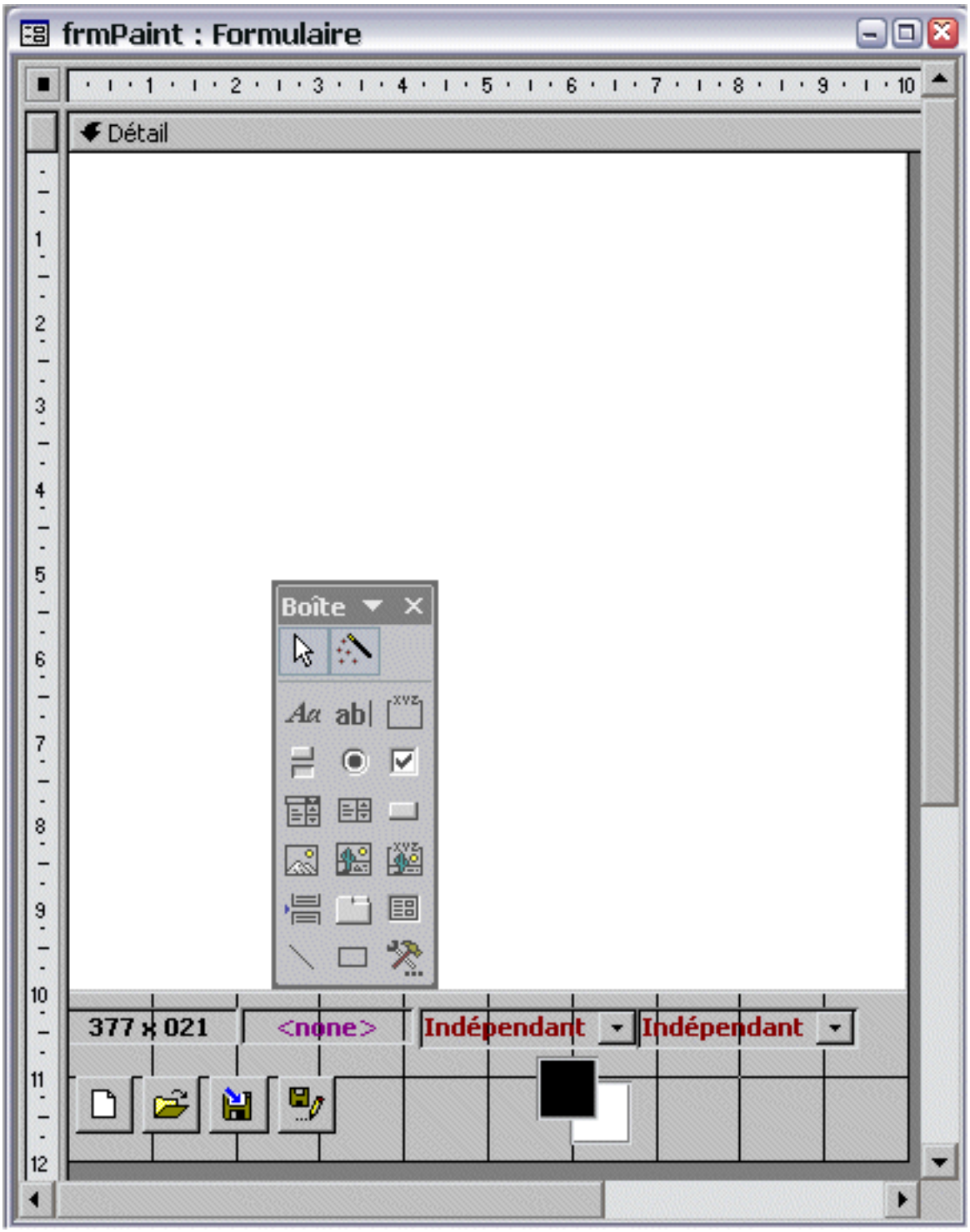
. Utiliser des **API pour dessiner des pixels** à l'endroit où se trouve la souris

. **Utiliser une API** pour afficher la boîte de dialogue de **choix de couleur**

. **Utiliser une API** pour sauvegarder notre **dessin**.

IV - Construction du formulaire

IV-A - La conception du formulaire



Nous dessinons notre formulaire avec :

. deux contrôles **ComboBox** (Taille et forme du pinceau)

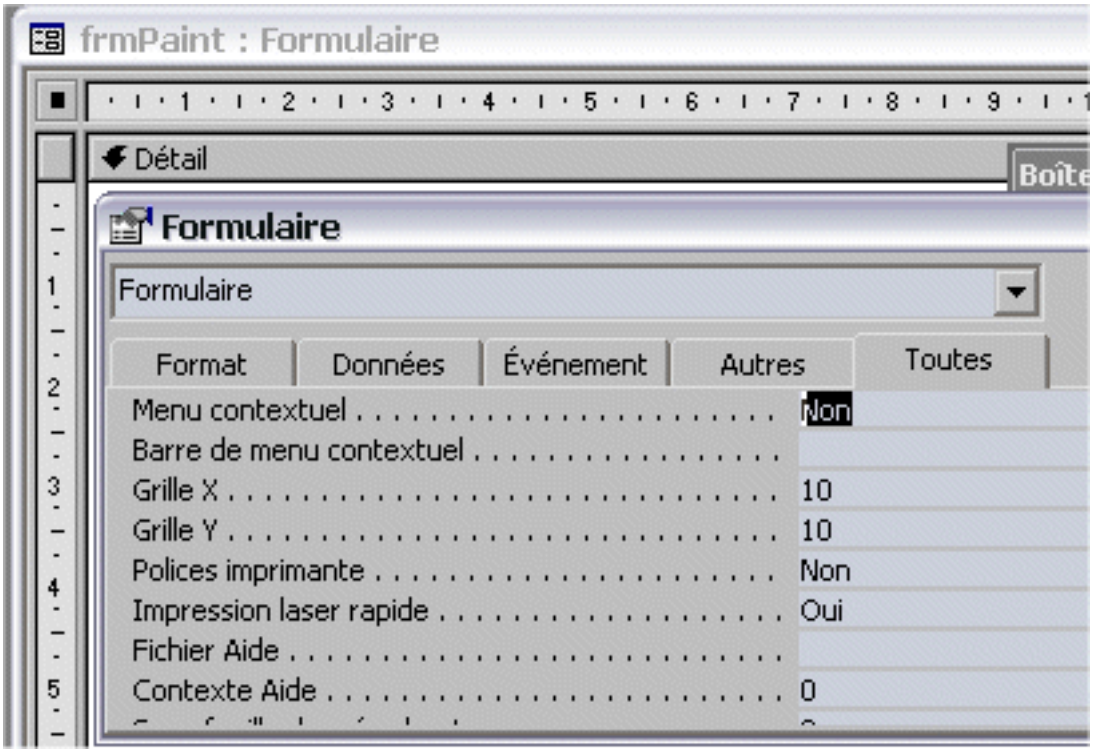
. deux contrôles **Label** (position de la souris et état des boutons)

. deux contrôles **Box** (couleur clic gauche et couleur clic droit)

. deux contrôles **Button** (Enregistrer et Enregistrer sous)

. un contrôle **Image** (zone de dessin).

> ***Toute action de clic droit sur un formulaire active l'apparition d'un menu contextuel, nous allons donc désactiver cette fonction.***
>
> *Il suffit de mettre "Non" sur la propriété Menu Contextuel du formulaire.*



Notre formulaire est maintenant prêt, mais il reste à coder les événements souris et bien sûr le tracé de points.

IV-B - Mission impossible ?

Il est, dans la pratique, très compliqué de dessiner dans un formulaire Access.

> ***Access ne dispose pas de fonctions natives pour dessiner**, hormis Pset() mais cette fonction n'est disponible que dans les **états**.*
>
> ***Pset()** est la fonction idéale pour tracer des lignes verticales dans les états, problème souvent abordé sur le forum.*

Sans fonctions natives, nous allons joyeusement utiliser l'extensibilité de VB / VBA et appeler des fonctions de l'interface de programmation (Application Programming Interface) qu'on appelle affectueusement API.

L'API que nous allons massivement utiliser est GDI (Graphic Device Interface), elle est dédiée à l'affichage.

Là où VB et VBA divergent, et ce, avant même l'avènement de VB.net c'est dans l'accessibilité des formulaires et des contrôles.

Dans Visual Basic, les forms et les contrôles sont accessibles par une poignée (handle) : un entier (de type long en 32 bits) qui permet à windows d'accéder directement à cet objet.

Cet entier est renvoyé par la propriété **.hWnd**

Ce n'est malheureusement pas le cas pour Access, du moins pour les versions XP et inférieures. La propriété **.hWnd** n'existe que pour les formulaires mais pas pour les contrôles.

Nous pourrons donc dans le meilleur des cas, dessiner dans le formulaire et donc pas seulement dans une zone d'image.

Mais il y a pire :

> ***Access ne dispose pas de propriété qui renvoie le handle du Device Context (Contexte de périphérique).***
>
> *Pour dessiner en utilisant les API, la fonction SetPixelV a besoin d'un Device Context, et là encore VB dispose de cette intéressante propriété **.hDC***

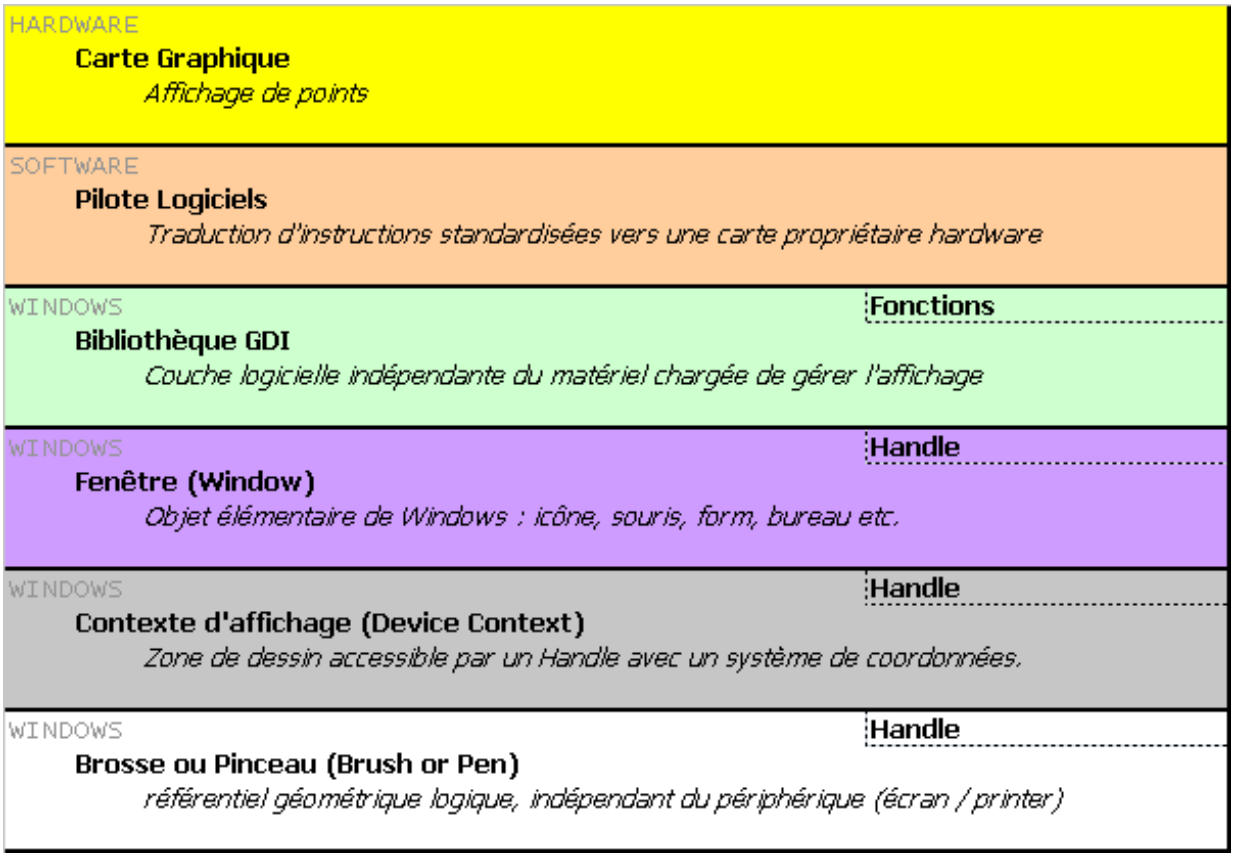
Il va falloir, là aussi, utiliser des API pour déterminer le handle du device contexte du formulaire puis l'ouvrir à nos fonctions de dessins.

Nous allons utiliser la fonction **GetDC()** de l'API **user32.dll**.

IV-C - Mais comment ça marche ?

Bon résumons, les API de GDI ne sont pas faites que pour dessiner sur Access, c'est pourquoi il faut respecter un certain principe de fonctionnement propre à Microsoft Windows.

Le diagramme ci-dessous permet de visualiser sommairement les différentes couches nécessaires au tracé d'un pixel sous Windows.



Il faut invoquer toutes les couches "Windows" pour parvenir à tracer des points sur l'écran.

Dans notre cas :

. Trouver le **handle** du formulaire

. Trouver le **device context** du formulaire

. Créer un **pinceau** pour dessiner d'une couleur avec un épaisseur définies

. Créer une **brosse** pour remplir des formes d'une couleur définie.

. Appeler une **fonction de traçage** de GDI.

V - Nos amis les API

L'utilisation d'une API se fait en deux étapes :

. **déclaration** pour inclure au projet VBA une fonction qui va pointer sur la DLL

. **appel dans le code** pour l'utilisation proprement dite.

Pour une information plus complète encore sur les API et VB, n'hésitez pas à lire l'excellent article de Bidou : Les API en Visual Basic

V-A - Comment déclarer une API ?

Attardons-nous un peu sur la déclaration d'une API qui sert à déterminer le Contexte d'Affichage (Device Context)

Déclaration d'API

```
'''""""""""""""""""""""""""""""""""""""""""""""'''
'''
'''              API user32.dll
'''
'''""""""""""""""""""""""""""""""""""""""""""""'''
'    retourne le handle (poignée) d'un contexte d'affichage ou device context
Private Declare Function apiGetDC Lib "user32" _
    Alias "GetDC" _
    (ByVal hwnd As Long) _
    As Long
```

Mot clés	***Arguments***	***Commentaires***
Declare Function	apiGetDC	Ce nom est libre, je préfixe api pour le retrouver facilement dans le code.
Lib	"user32"	Le nom de la bibliothèque, il est inutile de rajouter .dll
Alias	"GetDC"	Le vrai nom de la fonction incluse dans la DLL. Attention aux majuscules : les dll sont écrites en C / C++ ou autre et ne tolèrent pas d'approximation sur la casse.
(	ByVal hwnd As Long	Arguments de la fonction, ici le Handle d'une "Window"
)	As Long	Type de sortie

V-B - L'utilisation dans le code

Sur activation du formulaire

```
    hDCDest = apiGetDC(CtlhWnd(Me.imgPaint))
```

Nous constatons que la fonction est appelée exactement comme n'importe quelle fonction de VBA.

Ici, nous imbriquons deux fonctions : **CtlhWnd()** et **apiGetDC()**

Le résultat est d'obtenir le Handle du Device Context de notre formulaire.

V-C - Comment trouver la bonne API ?

A la question : "Vous êtes trop forts, comment faites-vous pour connaître par coeur toutes les API ?"

Je suis tenté de répondre par l'affirmative, mais ce serait bien trop exagéré.

En effet, nul besoin de connaître sur le bout du clavier toutes les API Windows, il faut simplement disposer des bons outils de recherche.

Je vous conseille de faire vos recherches sur l'Internet en plus de la visionneuse API qui est fournie avec Visual Basic ou sur le réseau AllAPI.

VI - Dessinons !

VI-A - Le code

code du formulaire de dessin

```
Option Compare Database
Option Explicit

'''""""""""""""""""""""""""""""""""""""""""""""'''
'''
'''  définition des types utilisés par les API
'''
'''""""""""""""""""""""""""""""""""""""""""""""'''
Private Type apiPOINT
    x As Long
    y As Long
End Type

Private Type apiRECT
  left As Long
  top As Long
  right As Long
  bottom As Long
End Type

'''""""""""""""""""""""""""""""""""""""""""""""'''
'''
'''                 API user32.dll
'''
'''""""""""""""""""""""""""""""""""""""""""""""'''
'   retourne le handle (poignée) d'un contexte d'affichage ou device context
Private Declare Function apiGetDC Lib "user32" _
    Alias "GetDC" _
    (ByVal hWnd As Long) _
    As Long

'   libère un contexte d'affichage
Private Declare Function apiReleaseDC Lib "user32" _
    Alias "ReleaseDC" _
    (ByVal hWnd As Long, _
    ByVal hdc As Long) _
    As Long

'   renvoie les dimensions d'une fenêtre
Private Declare Function apiGetWindowRect Lib "user32" _
    Alias "GetWindowRect" _
    (ByVal hWnd As Long, _
    lpRect As apiRECT) _
    As Long

'   renvoie la position de la souris
Private Declare Function apiSetCursorPos Lib "user32" Alias "SetCursorPos" ( _
        ByVal x As Long, ByVal y As Long) As Long

'   fixe la position de la souris
Private Declare Function apiGetCursorPos Lib "user32" Alias "GetCursorPos" ( _
        lpPoint As apiPOINT) As Long

'''""""""""""""""""""""""""""""""""""""""""""""'''
'''
'''                  API gdi32.dll
'''
'''""""""""""""""""""""""""""""""""""""""""""""'''

'   sélectionne un pixel qui aura le statut de "current"
Private Declare Function apiMoveToEx Lib "gdi32.dll" Alias "MoveToEx" ( _
        ByVal hdc As Long, ByVal x As Long, ByVal y As Long, _
        lpPoint As apiPOINT) As Long

'   dessine une ligne jusqu'au pixel désigné X Y
Private Declare Function apiLineTo Lib "gdi32" Alias "LineTo" ( _
        ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long

'   dessine une ellipse sur un device context
```

code du formulaire de dessin

```
Private Declare Function apiEllipse Lib "gdi32" Alias "Ellipse" ( _
        ByVal hdc As Long, ByVal X1 As Long, _
        ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long

'   dessine un rectange sur un device context
Private Declare Function apiRectangle Lib "gdi32" Alias "Rectangle" ( _
        ByVal hdc As Long, ByVal X1 As Long, _
        ByVal Y1 As Long, ByVal X2 As Long, _
        ByVal Y2 As Long) As Long

'   crée un crayon, permet de mettre une couleurs aux ellipses
Private Declare Function apiCreatePen Lib "gdi32" Alias "CreatePen" ( _
        ByVal nPenStyle As Long, ByVal nWidth As Long, _
        ByVal crColor As Long) As Long

'   crée une brosse afin de remplir une forme
Private Declare Function apiCreateSolidBrush Lib "gdi32" Alias "CreateSolidBrush" ( _
        ByVal crColor As Long) As Long

'   sélectionne un objet Windows, dans notre cas un(e) pinceau(brosse) associé à un device context
Private Declare Function apiSelectObject Lib "gdi32" Alias "SelectObject" ( _
        ByVal hdc As Long, ByVal hObject As Long) As Long

'   efface un Pen ou autre objet afin de libérer la mémoire
Private Declare Function apiDeleteObject Lib "gdi32" Alias "DeleteObject" ( _
        ByVal hObject As Long) As Long

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'''
'''     Déclarations nécessaires au formulaire
'''
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' poignée (handle) du device context sur lequel nous allons dessiner
Private hDCDest As Long
' poignées (handles) du pinceau et de la brosse
Private hPen As Long
Private hhPen As Long
Private hBrush As Long
Private hhBrush As Long

' options de dessin
' booléen pour connaître l'état du bouton de souris (relaché / appuyé)
Private blnButtonHold As Boolean
Private intThickIndex As Integer
Private intShapeIndex As Integer
Private lngCurrentLeftColor As Long
Private lngCurrentRightColor As Long
Private PreviousPoint As apiPOINT

' dimensions en pixels du cadre à l'intérieur duquel nous dessinons
Private xMax As Long
Private xMin As Long
Private yMax As Long
Private yMin As Long

' chemin du fichier de sauvegarde
Private strCurrentFile As String

Private Sub boxLeftCol_Click()

' affichage du dialogue de choix de couleur
Dim lngReturn As Boolean
    lngReturn = aDialogColor(Me.boxLeftCol.Properties("BackColor"))
    lngCurrentLeftColor = Me.boxLeftCol.BackColor

End Sub

Private Sub boxRightCol_Click()

' affichage du dialogue de choix de couleur
Dim lngReturn As Boolean
    lngReturn = aDialogColor(Me.boxRightCol.Properties("BackColor"))
    lngCurrentRightColor = Me.boxRightCol.BackColor
End Sub

Private Sub cmbShape_BeforeUpdate(Cancel As Integer)
```

code du formulaire de dessin

```
    ' gestion de la combo de forme
        intShapeIndex = Me.cmbShape

    End Sub

    Private Sub cmbThick_BeforeUpdate(Cancel As Integer)

    ' gestion de la combo d'épaisseur
        intThickIndex = Me.cmbThick

    End Sub

    Private Sub cmdNew_Click()

    ' bouton de création de nouveau document
        strCurrentFile = vbNullString
        Me.imgPaint.Picture = vbNullString
        Me.Caption = "[Nouveau document]"

    End Sub

    Private Sub cmdOpen_Click()

    ' bouton de chargement d'une image
        strCurrentFile = OuvrirUnFichier(Me.hWnd, "Ouvrir ...", 1, "", "", CurrentProject.Path)
        If Len(strCurrentFile) > 0 Then
            Me.imgPaint.Picture = strCurrentFile
            Me.Caption = "[" & strCurrentFile & "]"
        End If
    End Sub

    Private Sub cmdSave_Click()

    ' bouton de sauvegarde d'une image
        If Len(strCurrentFile) > 0 Then
            ShiftImprimEcran strCurrentFile, _
                             xMin, _
                             yMin, _
                             xMax, _
                             yMax
            Me.Caption = "[" & strCurrentFile & "]"
            Me.imgPaint.Picture = strCurrentFile
        Else
            cmdSaveAs_Click
        End If

    End Sub

    Private Sub cmdSaveAs_Click()

    ' bouton enregistrer sous d'une image
        ' enregistrement d'un fichier temporaire au cas où la boîte
        '  de dialogue se superpose à notre dessin
        ShiftImprimEcran CurrentProject.Path & "\tmp.wip", _
                         xMin, _
                         yMin, _
                         xMax, _
                         yMax
        ' affichage de boîte de dialogue de sauvegarde
        strCurrentFile = EnregistrerUnFichier(Me.hWnd, "Enregistrer l'image sous", "*.bmp",
    CurrentProject.Path)
        If Len(strCurrentFile) > 0 Then
            If right(strCurrentFile, 4) <> ".bmp" Then
                strCurrentFile = strCurrentFile & ".bmp"
            End If
            ' copier coller du fichier temporaire en fichier sauvegardé
            CopierColler CurrentProject.Path & "\tmp.wip", strCurrentFile, 0
            Me.imgPaint.Picture = strCurrentFile
            Me.Caption = "[" & strCurrentFile & "]"
        Else
            ' l'utilisateur a annulé la sauvegarde, nous rechargeons le fichier temporaire
            Me.imgPaint.Picture = CurrentProject.Path & "\tmp.wip"
        End If

    End Sub

    Private Sub Form_Activate()
```

code du formulaire de dessin

```
    ' gestion des objets nécessaires au dessin via API
        ' obtention des dimensions du formulaire
        ' calcul de la taille de l'image
        Dim rectImg As apiRECT
        apiGetWindowRect Me.hWnd, rectImg

        ' les valeurs sont les marges dues au bords du formulaire
        xMin = rectImg.left + 6
        yMin = rectImg.top + 25
        xMax = Me.imgPaint.Width / 15 + xMin - 3
        yMax = Me.imgPaint.Height / 15 + yMin - 3
        intThickIndex = Me.cmbThick
        intShapeIndex = Me.cmbShape
        lngCurrentLeftColor = Me.boxLeftCol.BackColor
        lngCurrentRightColor = Me.boxRightCol.BackColor

        ' utilisation de l'API GetDC() pour récupérer le handle du device context
        hDCDest = apiGetDC(CtlhWnd(Me.imgPaint))
        Me.Caption = "[Nouveau document]"

    End Sub

    Private Sub Form_Deactivate()

        ' libération du device context
        apiReleaseDC CtlhWnd(Me.imgPaint), hDCDest

    End Sub

    Private Sub Form_Load()

        Me.lblMouseStatus.Caption = "<none>"
        blnButtonHold = False

    End Sub

    Private Sub imgPaint_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)

        Dim tampon As apiPOINT
        apiGetCursorPos tampon

        If Button = acLeftButton Or Button = acRightButton Then
            blnButtonHold = True
            Me.lblMouseStatus.Caption = IIf(Button = acLeftButton, "<left>", "<right>")
            Me.lblMousePos.Caption = Format(x / 15, "000") & " x " & Format(y / 15, "000")

            Select Case intShapeIndex
                Case 1 To 4
                    If intShapeIndex <= 2 Then
                        hBrush = apiCreateSolidBrush(IIf(Button = acLeftButton, lngCurrentLeftColor,
    lngCurrentRightColor))
                    Else
                        hBrush = apiCreateSolidBrush(IIf(Button = acLeftButton, lngCurrentRightColor,
    lngCurrentLeftColor))
                    End If
                    hPen = apiCreatePen(0, 1, IIf(Button = acLeftButton, lngCurrentLeftColor,
    lngCurrentRightColor))
                    hhPen = apiSelectObject(hDCDest, hPen)
                    hhBrush = apiSelectObject(hDCDest, hBrush)
                    If intShapeIndex = 1 Or intShapeIndex = 3 Then
                        apiRectangle hDCDest, tampon.x - intThickIndex / 2, tampon.y - intThickIndex /
    2, _
                                              tampon.x + intThickIndex / 2, tampon.y + intThickIndex / 2
                    Else
                        apiEllipse hDCDest, tampon.x - intThickIndex / 2, tampon.y - intThickIndex / 2,
    _
                                              tampon.x + intThickIndex / 2, tampon.y + intThickIndex / 2
                    End If
                    apiDeleteObject hhBrush
                    apiDeleteObject hBrush
                    apiDeleteObject hhPen
                    apiDeleteObject hPen

                Case 5
                    hPen = apiCreatePen(0, 1, IIf(Button = acLeftButton, lngCurrentLeftColor,
    lngCurrentRightColor))
                    hBrush = apiCreateSolidBrush(IIf(Button = acLeftButton, lngCurrentLeftColor,
    lngCurrentRightColor))
```

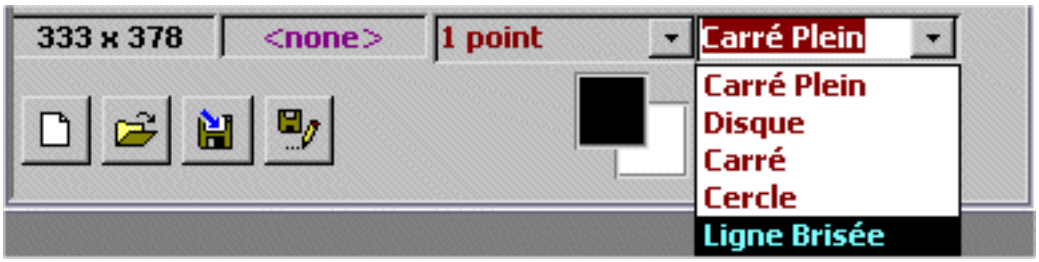
code du formulaire de dessin

```
                hhPen = apiSelectObject(hDCDest, hPen)
                hhBrush = apiSelectObject(hDCDest, hBrush)
                apiEllipse hDCDest, tampon.x - (intThickIndex / 2), tampon.y - (intThickIndex / 2),
_
                            tampon.x + (intThickIndex / 2), tampon.y + (intThickIndex / 2)
                apiDeleteObject hhBrush
                apiDeleteObject hBrush
                apiDeleteObject hhPen
                apiDeleteObject hPen
                PreviousPoint.x = tampon.x
                PreviousPoint.y = tampon.y

        End Select

    End If

End Sub

Private Sub imgPaint_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)

    Dim tampon As apiPOINT

    If blnButtonHold Then
        '   bornage de la position de la souris
        apiGetCursorPos tampon
        apiSetCursorPos Max(Min(tampon.x + Fix(intThickIndex / 2), xMax) - Fix(intThickIndex / 2),
xMin + Fix(intThickIndex / 2)), _
                        Max(Min(tampon.y + Fix(intThickIndex / 2), yMax) - Fix(intThickIndex / 2),
yMin + Fix(intThickIndex / 2))
        apiGetCursorPos tampon

        Select Case intShapeIndex
            Case 1 To 4
                If intShapeIndex <= 2 Then
                    hBrush = apiCreateSolidBrush(IIf(Button = acLeftButton, lngCurrentLeftColor,
lngCurrentRightColor))
                Else
                    hBrush = apiCreateSolidBrush(IIf(Button = acLeftButton, lngCurrentRightColor,
lngCurrentLeftColor))
                End If
                hPen = apiCreatePen(0, 1, IIf(Button = acLeftButton, lngCurrentLeftColor,
lngCurrentRightColor))
                hhPen = apiSelectObject(hDCDest, hPen)
                hhBrush = apiSelectObject(hDCDest, hBrush)
                If intShapeIndex = 1 Or intShapeIndex = 3 Then
                    apiRectangle hDCDest, tampon.x - intThickIndex / 2, tampon.y - intThickIndex /
2, _
                                          tampon.x + intThickIndex / 2, tampon.y + intThickIndex / 2
                Else
                    apiEllipse hDCDest, tampon.x - intThickIndex / 2, tampon.y - intThickIndex / 2,
_
                                          tampon.x + intThickIndex / 2, tampon.y + intThickIndex / 2
                End If
                apiDeleteObject hhBrush
                apiDeleteObject hBrush
                apiDeleteObject hhPen
                apiDeleteObject hPen

            Case 5
                If PreviousPoint.x <> 0 And PreviousPoint.y <> 0 Then
                    hPen = apiCreatePen(0, intThickIndex, IIf(Button = acLeftButton,
lngCurrentLeftColor, lngCurrentRightColor))
                    hhPen = apiSelectObject(hDCDest, hPen)
                    apiMoveToEx hDCDest, PreviousPoint.x, PreviousPoint.y, PreviousPoint
                    apiLineTo hDCDest, tampon.x, tampon.y
                    apiDeleteObject hhPen
                    apiDeleteObject hPen
                End If
                PreviousPoint.x = tampon.x
                PreviousPoint.y = tampon.y

        End Select


    Else
        Me.lblMousePos.ForeColor = vbBlack
        PreviousPoint.x = 0
        PreviousPoint.y = 0
```

code du formulaire de dessin

```
        End If
        Me.lblMousePos.Caption = Format(x / 15, "000") & " x " & Format(y / 15, "000")

    End Sub

    Private Sub imgPaint_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)

        Me.lblMouseStatus.Caption = "<none>"
        PreviousPoint.x = 0
        PreviousPoint.y = 0
        blnButtonHold = False

    End Sub

    Private Function Max(ByVal v1 As Long, ByVal v2 As Long) As Long

        If v1 >= v2 Then Max = v1 Else Max = v2

    End Function

    Private Function Min(ByVal v1 As Long, ByVal v2 As Long) As Long

        If v1 < v2 Then Min = v1 Else Min = v2

    End Function
```

Nous avons défini plusieurs formes dans la combo :



Une fois le Handle du Device Context récupéré nous pouvons tracer plusieurs formes

VI-B - Les objets graphiques

Nous utilisons l'API : **Rectangle**.

Pour le tracé de la forme, nous devons créer un pinceau (*Pen*) grâce à **CreatePen**.

Si nous souhaitons un carré plein nous devons créer une brosse (*Brush*) grâce à **CreateBrush**.

La gestion d'un objet graphique comprend 3 étapes :

. Création : API **CreatePen**

. Sélection : API **SelectObject** avec un handle de Device Context

. Destructions : API **DeleteObject** du handle de la sélection, et du handle de l'objet

gestion des objets graphiques
```
                    ' Créations
                    hPen = apiCreatePen(0, 1, IIf(Button = acLeftButton, lngCurrentLeftColor,
   lngCurrentRightColor))
                    hBrush = apiCreateSolidBrush(IIf(Button = acLeftButton, lngCurrentLeftColor,
   lngCurrentRightColor))
                    ' Sélections
                    hhPen = apiSelectObject(hDCDest, hPen)
                    hhBrush = apiSelectObject(hDCDest, hBrush)
                    ' ici nous mettons la fonction Rectangle ou Ellipse

                    ' Destructions
                    apiDeleteObject hhBrush
                    apiDeleteObject hBrush
                    apiDeleteObject hhPen
                    apiDeleteObject hPen
```

Important : Penser à libérer les ressources

Si vous oubliez de détruire les objets vous allez rapidement saturer la mémoire d'affichage.

Il est toujours de la responsabilité du programmeur de prévoir la libération de la mémoire.

VI-C - Fonctionnements des figures

VI-C-1 - Carré Plein ou Vide

Nous utilisons l'API **Rectangle()** dont les arguments sont :

. le handle du device context

. les coordonnées du rectangle (gauche, haut, droite, bas)

La couleur du contour est définie par la couleur du Pinceau, la couleur du remplissage est définie par la couleur de la Brosse.

VI-C-2 - Disque ou Cercle

Nous utilisons l'API **Ellipse()** dont les arguments sont :

. le handle du device context

. les coordonnées du rectangle (gauche, haut, droite, bas) qui contiennent l'ellipse tangente

La couleur du contour est définie par la couleur du Pinceau, la couleur du remplissage est définie par la couleur de la Brosse.

VI-C-3 - La ligne brisée

Nous allons utiliser la fonction API **LineTo()** dont les arguments sont :

. le handle du device context

. les coordonnées du point vers lequel nous traçons la droite

Attention ! Le tracé se fait depuis le dernier point dit "courant", mais quel est-il ?

Dans la pratique c'est le dernier point sur lequel l'API GDI a travaillé, mais pour ne rien laisser au hasard nous allons nous-même le déterminer.

Nous le fixons grâce à une autre API **MoveToEx()** dont les arguments sont :

. le handle du device context

. les coordonnées du point courant.

VI-D - Gestion de la souris

Le dessin au clavier reste assez contre-intuitif voire peu ergonomique, c'est pourquoi nous allons gérer les événements souris afin de déclencher les actions de dessin.

événement souris	actions
bouton enfoncé	Tracé de pixels aux coordonnées courantes Stockage du bouton enfoncé : *acLeftButton* ou *acRightButton* Attributions des coordonnées courantes au *PreviousPoint*.
bouton relaché	Vidage du *PreviousPoint* Vidage du bouton enfoncé
souris déplacée	Tracé de pixels aux coordonnées courantes Tracé d'une ligne depuis le point actuel jusqu'au point précédent (cas de la ligne brisée)

événement souris	actions
	Attributions des coordonnées courantes au *PreviousPoint*.

VII - Sauvegardons

Nouvelle déconvenue : là aussi contrairement à VB, notre dessin en API n'est absolument pas stocké, nous avons dessiné sur l'écran mais pas à l'intérieur du contrôle image.

En cas de permutation d'application vous remarquerez que vous aurez tout perdu.

Le contrôle image que nous utilisons ne sait pas enregistrer sur le disque directement son contenu.

Pour effectuer une sauvegarde nous allons devoir faire :

. une copie d'écran dans le presse-papier.

. une écriture d'un fichier BMP à partir du presse-papier mais pour une zone délimitée précise.

module de sauvegarde d'une image

```
Option Compare Database
Option Explicit

'''""""""""""""""""""""""""""""""""""""""""""""""""""'''
'
'                    API kernel32.dll
'
'''""""""""""""""""""""""""""""""""""""""""""""""""""'''
'   copie de fichier
Private Declare Function apiCopyFile Lib "kernel32" Alias "CopyFileA" ( _
    ByVal lpExistingFileName As String, ByVal lpNewFileName As String, _
    ByVal bFailIfExists As Long) As Long

'   suppression de fichier
Private Declare Function apiDeleteFile Lib "kernel32" Alias "DeleteFileA" ( _
    ByVal lpFileName As String) As Long


'''""""""""""""""""""""""""""""""""""""""""""""""""""'''
'
'                     API gdi32.dll
'
'''""""""""""""""""""""""""""""""""""""""""""""""""""'''
'   sélection d'un objet Windows
Private Declare Function apiSelectObject Lib "gdi32" Alias "SelectObject" ( _
    ByVal hdc As Long, ByVal hObject As Long) As Long

'   création d'un device context
Private Declare Function apiCreateCompatibleDC Lib "gdi32" Alias "CreateCompatibleDC" ( _
    ByVal hdc As Long) As Long

Private Declare Function apiBitBlt Lib "gdi32.dll" Alias "BitBlt" ( _
    ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, _
    ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, _
    ByVal XSrc As Long, ByVal YSrc As Long, ByVal dwRop As Long) As Long

Private Declare Function apiGetDIBits Lib "gdi32" Alias "GetDIBits" ( _
    ByVal aHDC As Long, ByVal hBitmap As Long, _
    ByVal nStartScan As Long, ByVal nNumScans As Long, _
    lpBits As Any, lpBI As apiBITMAPINFO, ByVal wUsage As Long) As Long

Private Declare Function apiCreateDIBSection Lib "gdi32" Alias "CreateDIBSection" ( _
    ByVal hdc As Long, pBitmapInfo As apiBITMAPINFO, ByVal un As Long, _
    ByVal lplpVoid As Long, ByVal Handle As Long, ByVal dw As Long) As Long
```

module de sauvegarde d'une image

```
'   suppression d'un device context
Private Declare Function apiDeleteDC Lib "gdi32" Alias "DeleteDC" ( _
    ByVal hdc As Long) As Long

'   suppression d'un objet windows
Private Declare Function apiDeleteObject Lib "gdi32" Alias "DeleteObject" ( _
    ByVal hObject As Long) As Long


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
'                     API user32.dll
'
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'   récupération du handle du Desktop
Private Declare Function apiGetDesktopWindow Lib "user32.dll" Alias "GetDesktopWindow" () As Long

'   récupération du device context d'après un handle de window
Private Declare Function apiGetDC Lib "user32.dll" Alias "GetDC" ( _
    ByVal hwnd As Long) As Long


Private Type apiBITMAPINFO
    biSize As Long
    biWidth As Long
    biHeight As Long
    biPlanes As Integer
    biBitCount As Integer
    biCompression As Long
    biSizeImage As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
    biRUsed As Long
    biRImportant As Long
End Type

Private Type apiBITMAPFILEHEADER
    bfType As Integer
    bfSize As Long
    bfReserved1 As Integer
    bfReserved2 As Integer
    bfOffBits As Long
End Type

Private Const GHND = &H42
Private Const MAXSIZE = 4096
Private Const SRCCOPY = &HCC0020
Private Const DIB_RGB_COLORS = 0&
Private Const BI_RGB = 0&

' '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'
'              Sur une section précise
'                              de l'écran
'
' '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function ShiftImprimEcran(strNomDuFichier As String, X1 As Long, Y1 As Long, X2 As Long, Y2 As Long)

On Error GoTo Finally
Dim lngLargeur As Long, lngHauteur As Long
Dim lngHdc As Long
Dim lngHBmp As Long
Dim bmiBitmapInfo As apiBITMAPINFO
Dim bmfBitmapFileHeader As apiBITMAPFILEHEADER
Dim lngFnum As Integer
Dim pixels() As Byte
Dim bolOuvert As Boolean

lngHdc = apiCreateCompatibleDC(0)
If lngHdc = 0 Then
  GoTo Finally
End If
'Récupère les dimensions de l'écran
lngHauteur = Y2 - Y1
lngLargeur = X2 - X1
'Crée un bitmap vierge
```

module de sauvegarde d'une image

```
  With bmiBitmapInfo
    .biBitCount = 32
    .biCompression = BI_RGB
    .biPlanes = 1
    .biSize = Len(bmiBitmapInfo)
    .biHeight = lngHauteur
    .biWidth = lngLargeur
    .biSizeImage = (((((.biWidth * .biBitCount) + 31) \ 32) * 4 - _
      (((.biWidth * .biBitCount) + 7) \ 8)) * .biHeight
  End With
  lngHBmp = apiCreateDIBSection(lngHdc, bmiBitmapInfo, DIB_RGB_COLORS, _
                                ByVal 0&, ByVal 0&, ByVal 0&)
  If lngHBmp = 0 Then
    GoTo Finally
  End If
  If apiSelectObject(lngHdc, lngHBmp) = 0 Then
    GoTo Finally
  End If
  'Copie le contenu de l'ecran
  If apiBitBlt(lngHdc, 0&, 0&, lngLargeur, lngHauteur, _
               apiGetDC(apiGetDesktopWindow()), X1&, Y1&, SRCCOPY) = 0 Then
    GoTo Finally
  End If

  'Crée l'entête du fichier bmp
  With bmfBitmapFileHeader
    .bfType = &H4D42&
    .bfOffBits = Len(bmfBitmapFileHeader) + Len(bmiBitmapInfo)
    .bfSize = .bfOffBits + bmiBitmapInfo.biSizeImage
  End With
  'Lit les bits du bitmap et les place dans le tableau de pixels
  ReDim pixels(1 To 4, 1 To lngLargeur, 1 To lngHauteur)
  If apiGetDIBits(lngHdc, lngHBmp, 0, lngHauteur, pixels(1, 1, 1), _
                  bmiBitmapInfo, DIB_RGB_COLORS) = 0 Then
    GoTo Finally
  End If
  lngFnum = FreeFile
  'Crée le fichier
  Open strNomDuFichier For Binary As lngFnum
  bolOuvert = True
  'Ecrit l'entête
  Put #lngFnum, , bmfBitmapFileHeader
  'Ecrit les informations du bitmap
  Put #lngFnum, , bmiBitmapInfo
  'Ecrit les bits de l'image
  Put #lngFnum, , pixels
  Finally:
  'Ferme le fichier si ouvert
  If bolOuvert Then Close lngFnum
  'Supprime les objets
  If lngHBmp <> 0 Then apiDeleteObject lngHBmp
  If lngHdc <> 0 Then apiDeleteDC lngHdc

  End Function

  Function CopierColler(Source As String, Destination As String, _
    NepasEcraser As Long) As Long
  CopierColler = apiCopyFile(Source, Destination, NepasEcraser)
  End Function

  Function CouperColler(Source As String, Destination As String, _
    NepasEcraser As Long) As Long
  Dim r As Long
  r = apiCopyFile(Source, Destination, NepasEcraser)
  If r Then CouperColler = apiDeleteFile(Source)
  CouperColler = r
  End Function
```

Je remercie au passage Tofalu pour son aide sur cette fonction.

L'intérêt est de passer à cette fonction les bonnes coordonnées afin de ne sauvegarder que les dimensions de l'image et non pas le contour.

VIII - Conclusion

Le but de ce tutoriel est de repousser les limites d'Access, d'illustrer l'utilisation intensive d'API et d'expliquer le fonctionnement des routines d'affichage de GDI sous Windows.

Je ne développerai pas les API d'affichage de dialogues de choix de couleur, disponibles dans la FAQ.

Bons dessins sous MS Access ! Vous pouvez télécharger la base exemple ici.

Merci de m'avoir lu.