UTILISATION DE LA BIBLIOTHEQUE WIAAUT.DLL AVEC VISUAL BASIC .NET 2005

Table des matières

UTILISATION DE LA BIBLIOTHEQUE WIAAUT.DLL AVEC VISUAL BASIC .NET 2005

I. INTRODUCTION

La bibliothèque de fonctions wiaaut.dll fournit des fonctions permettant de :

- Utiliser différents types de périphériques tels que webcams, scanners et appareils photos numériques.
- Avoir des informations sur les périphériques utilisés en intégrant les boîtes de dialogue standard de Windows XP de gestion des images.
- Prendre des photos à partir de caméras vidéos type webcams.
- Paramétrer l'impression des photos.

Il est à noter que cette bibliothèque de fonctions n'est utilisable que sous XP.

Ce que je vous propose de réaliser au cours de ce tutoriel est une petite application qui permet d'acquérir une série de photos par l'intermédiaire d'une caméra vidéo. Dans un premier temps on va acquérir une seule photo, puis dans un second temps nous allons prendre une série de photos à intervalle réguliers en intégrant un timer. Nous verrons de même comment utiliser les différentes boîtes de dialogue standard dans un programme. Avant de commencer le développement nous allons nous pencher sur l'installation de cette bibliothèque.

II. INSTALLATION

Cette bibliothèque est téléchargeable sur le site de Microsoft à l'adresse suivante :

mettre adresse internet

Une fois téléchargé et dézippé le fichier correspondant, vous devez copié le fichier wiaaut.dll dans le répertoire C:\Windows\System32

Une fois cette opération réalisée, il vous suffit de taper la commande MS-DOS suivante :

regsvr32 wiaaut.dll

Voilà, maintenant vous êtes prêt à utiliser cette bibliothèque, il vous reste plus qu'à réaliser votre première application, en utilisant les nombreuses fonctions mises à votre disposition.

III. PRELIMINAIRES

Tout d'abord nous allons commencer par créer notre fichier projet. Pour ce faire Appuyer sur Fichier->Nouveau Projet, une fenêtre apparaît à l'écran et vous permet de choisir le type d'application que vous voulez créer. Choisissez le type d'application Application Windows

Donnez lui le nom que vous voulez et validez.

La solution se crée. La fenêtre principale s'ouvre et montre une fiche vide préconstruite.

Nous allons maintenant personnaliser cette fiche afin quelle ressemble à ce résultat :

Donc vous aurez besoin des composants suivants :

- Un composant mainMenu pour le Menu principal de l'application;
- un composant PictureBox pour afficher les images acquises;
- Un composant StatusBar pour afficher quelques états sur le déroulement de l'application;

En ce qui concerne le menu Fichier de l'application, vous devriez avoir les commande suivantes :

En ce qui concerne le menu Périphériques de l'application vous devriez avoir les commandes suivantes :

Le menu Image ne nous intéresse pas pour le moment et fera l'objet d'un nouveau tutoriel.

Donc nous voilà prêt pour commencer à implémenter le code de l'application.

Avant toute chose il faut référencer dans le code de l'application la bibliothèque wiaaut.dll :

```
imports WIA
```

Nous allons donc voir comment utiliser les fonctions mises à notre disposition.

IV. SELECTION D'UN PERIPHERIQUE

Tout d'abord avant de pouvoir prendre une photo, il est nécessaire de choisir un périphérique.

La bibliothèque nous fournit la boîte de dialogue de sélection de périphériques d'acquisition d'image.

Pour utiliser cette boîte de dialogue vous devez écrire le code suivant :

```
Dim d As Device

d = Nothing

class1 = CreateObject("WIA.CommonDialog")

    Try
       d = class1.ShowSelectDevice(WiaDeviceType.VideoDeviceType, True, False)
    Catch
       MessageBox.Show("Aucun périphérique n'a pu ou n'a été sélectionné... Veuillez
                        vérifier qu'un périphérique d'acquisition d'images est
                        branché",
                        "Avertissement", MessageBoxButtons.OK)


    End Try
```

Tout se passe par l'intermédiaire d'un objet Device, celui-ci étant instancié grâce au résultat retourné par la sélection du périphérique, et ceci par l'intermédiaire de la boîte de dialogue de sélection d'un périphérique.

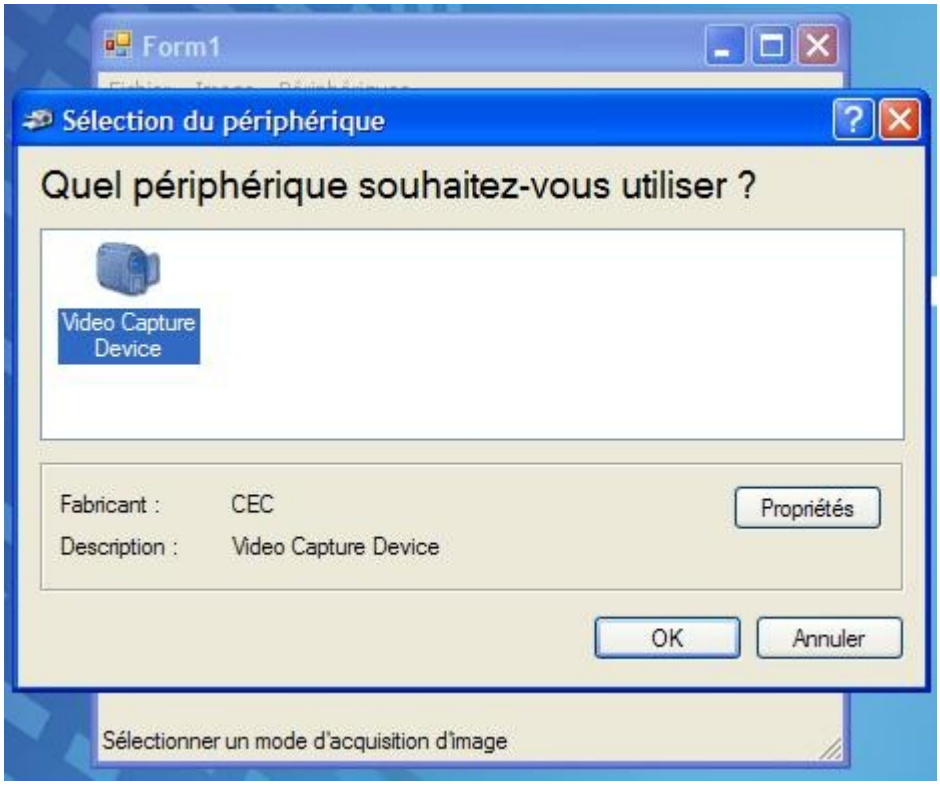
Une boîte de dialogue est créée par l'intermédiaire de la ligne de code :

```
class1 = CreateObject("WIA.CommonDialog")
```

La sélection s'opère grâce à ligne de code suivante :

```
d = class1.ShowSelectDevice(WiaDeviceType.VideoDeviceType, True, False)
```

Ce code va nous permettre de voir le résultat suivant à l'écran :



Il est à noter que la boîte de dialogue de sélection accepte différentes options qui permettent de sélectionner le type de périphériques que l'on veut utiliser. Il en existe quatre :

- WiaDeviceType.VideoDeviceType pour sélectionner une caméra vidéo;
- WiaDeviceType.CameraDeviceType pour sélectionner un appareil photo;
- WiaDeviceType.ScannerDeviceType pour sélectionner un scanner;
- WiaDeviceType.UnspecifiedDeviceType pour ne spécifier aucun type donc pour afficher tous les périphérique d'acquisition connectés à votre ordinateur.

Voilà en ce qui concerne la sélection d'un périphérique nous allons voir maintenant comment prendre une photo.

IV. ACQUISITION D'UNE PHOTO

Voici le code nécessaire pour réaliser l'acquisition d'une photo. Il suffit de taper la ligne de code suivante :

```
Item = d.ExecuteCommand(CommandID.wiaCommandTakePicture)
```

L'instance d'objet d est toujours celle que nous avions créé lors de la sélection du périphérique, elle est de type device. Voici maintenant la fonction permettant de gérer l'acquisition de la photo et son affichage sur la PictureBox.

```
Dim jpegGUID As String

    Dim jpegKey As Microsoft.Win32.RegistryKey = _
                Microsoft.Win32.Registry.ClassesRoot.OpenSubKey("" & _
                 "CLSID\{D2923B86-15F1-46FF-A19A-DE825F919576}\SupportedExtension\.jpg")
    jpegGUID = CType(jpegKey.GetValue("FormatGUID"), String)
    Dim fichier As FileInfo
    imagefile = CType(Item.Transfer(jpegGUID), WIA.ImageFile)
    imagefile = Item.Transfer(FormatID.wiaFormatJPEG)

    fichier = New FileInfo(fileName)
    If fichier.Exists Then


       fileName = "Temp000" + nbFichierTemp.ToString + ".jpg"
       fichier = Nothing
       fichier = New FileInfo(fileName)
       If fichier.Exists Then

          fichier.Delete()

          fichier = Nothing
       End If
       nbFichierTemp = nbFichierTemp + 1
    End If
    Text = fileName

    imagefile.SaveFile(fileName)

    img = Image.FromFile(fileName)
    PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    PictureBox1.Image = img
    class1 = CreateObject("WIA.CommonDialog")

    'class1.ShowItemProperties(Item, False)
    Item = Nothing
    'img = Nothing
    'imagefile = Nothing
```

Ce code vous permet d'acquérir une photo au format JPEG, comme vous pouvez le constater il est nécessaire de lui spécifier la clé de registre correspondante grâce à la déclaration suivante :

```
    Dim jpegKey As Microsoft.Win32.RegistryKey = _
                Microsoft.Win32.Registry.ClassesRoot.OpenSubKey("" & _
                 "CLSID\{D2923B86-15F1-46FF-A19A-DE825F919576}\SupportedExtension\.jpg")
    jpegGUID = CType(jpegKey.GetValue("FormatGUID"), String)
```

Ensuite il faut créer un objet qui va nous permettre de prendre en charge la création de l'image en mémoire, cela se fait grâce à un objet de type ImageFile :

```
    imagefile = CType(Item.Transfer(jpegGUID), WIA.ImageFile)
    imagefile = Item.Transfer(FormatID.wiaFormatJPEG)
```

Ce code permet de spécifier le type de l'image,c'est à dire le JPEG dans notre cas.
Item.Transfer permet de spécifier le fait qu'il n'y aura qu'un seu fichier à transférer.

Après on spécifie un nom de fichier que l'on va attribuer à la photo, pour ce faire on va passer par l'intermédiaire d'un objet FileInfo qui va nous permettre de vérifier si le fichier existe.
Pour utiliser ce type d'objet il est nécessaire d'indiquer la référence suivante dans la section imports du code :

```
Imports System.IO.FileAttributes
```

Le programme utilise un système de fichier temporaire avec un index qui s'incrémente.

```
    If fichier.Exists Then

       fichier.Delete()

       fichier = Nothing
    End If
    nbFichierTemp = nbFichierTemp + 1
```

Il suffit ensuite de sauvegarder cette image dans le fichier temporaire par l'intermédiaire de l'objet imagefile

```
      imagefile.SaveFile(fileName)
```

Une fois cette sauvegarde effectuée vous pouvez l'afficher dans la PictureBox par l'intermédiaire de ce code :

```
    Dim img As Image


    img = Image.FromFile(fileName)
    PictureBox1.SizeMode = PictureBoxSizeMode.StretchImage
    PictureBox1.Image = img
```

Il suffit de créer un objet Image que l'on va associer au fichier que l'on vient de créer physiquement, et ce par l'utilisation de la fonction LoadFromFile qui permet de l'ouvrir. Il ne reste plus qu'à le « coller » sur la PictureBox, en utilisant la propriété Image de cet objet.

Voilà vous savez maintenant comment il est possible d'acquérir une image en utilisant un périphérique sélectionné par rapport à un type précis.

Maintenant nous allons améliorer cette fonction afin de pouvoir acquérir une série de photos prises à un intervalle de temps précis...

V. ACQUISITION AUTOMATISEE DE PHOTOS

Nous avons déjà l'ossature puisque nous avons déjà la fonction qui permet de prendre la photo ainsi que la fonction permettant de gérer l'acquisition d'une photo. Dans notre cas précis on veut juste ajouter la fonction de prise de photos automatisée à l'aide d'un timer. Et ensuite nous allons

récupérer toutes les photos acquises et les visualiser à l'écran.

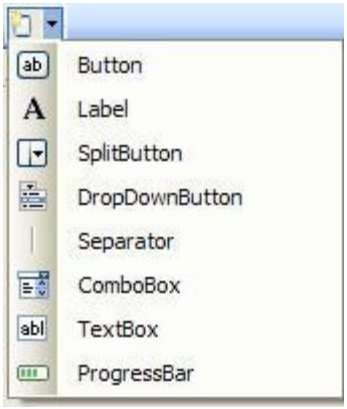
Afin de réaliser cette fonction il faut déposer sur votre fiche, en mode conception, le composant timer. Ce composant va nous permettre donc ces acquisitions à intervalle fixe. Et pour gérer l'acquisition de ce timer nous allons placer sur notre fiche une barre d'outil contenant deux boutons permettant de démarrer et d'arrêter le timer.

A vous de personnaliser cette barre d'outils comme bon vous semble ensuite.

Pour réaliser cette barre d'outils de base, il vous suffit d'ajouter à votre fiche le composant ToolStrip.

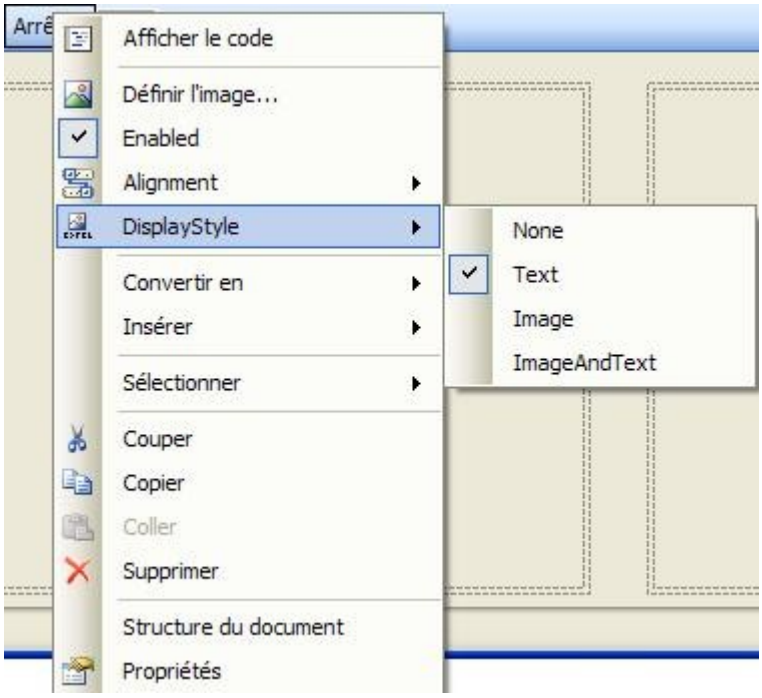
Afin d'ajouter des boutons à cette barre il suffit de cliquer sur la flèche de la boîte combo (combobox) :

Plusieurs choix s'offre à vous :



Dans notre cas nous avons besoin de deux boutons donc choisissez l'option Button.

Pour personnaliser chaque bouton afin qu'il n'affiche que du texte il faut mettre à jour le style d'affichage (Display Style), en cliquant droit sur le bouton en question. Un menu contextuel apparaît alors il vous suffit de choisir la commande DisplayStyle qui fait ensuite apparaître une série d'option, de la façon suivante :



Il vous suffit de choisir l'option Text. Vous pouvez constater que les barres d'outils sous Visual Studio 2005 sont personnalisables suivant vos souhaits. Vous pourrez donc facilement mettre à jour cette fonctionnalité.

Donc maintenant que la structure de l'application est mise en place, il va falloir passer à l'implémentation du code.

Occupons nous tout d'abord de la fonction associée au bouton démarrer de l'application qui va lancer le timer et donc les acquisitions successives de photos.

Le code de celle-ci est fort simple :

```
Timer1.Start()
```

Il se contente d'appeler la fonction Start de l'objet Timer qui met en marche le timer.

Les acquisitions vont être acquises toutes les x ms. La propriété qui gère cette période est appelée Interval. A vous de lui donner la valeur qui vous paraît la mieux adaptée, me concernant je l'ai initialisé à la valeur 5000, ce qui veut dire que les acquisitions auront lieu toutes les 5 secondes.

La fonction associée au bouton Arrêter est très simple aussi puisqu'elle se contente d'appeler la fonction Stop de l'objet Timer :

```
Timer1.Stop()
```

Il manque tout de même l'essentiel, c'est à dire la fonction qui permet de gérer les acquisitions successives. Celle-ci est géré par la fonction Timer1_Tick du timer. Pour y accéder directement, il suffit de double-cliquer sur le composant Timer. Cette fonction ne fait qu'appeler la fonction que nous avons commenté à la section précédente :

```
Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
        Acquisition()
End Sub
```

Donc toutes x millisecondes, une acquisition est effectuée, mais le problème c'est qu'avec cette méthode les photos sont affichés les unes après les autres, ce qui n'est pas très utile si l'on veut comparer les photos ou tout simplement les visualiser toutes ensemble. Elles sont stockées physiquement dans la mémoire du périphérique. Et si vous regarder bien elles sont stockées de même dans le répertoire bin de votre projet....

VI. IMPORTATIONS DES IMAGES

Deux méthodes s'offre à vous pour les visualiser, soit par l'intermédiaire d'une boîte de dialogue standard de récupération de d'images, soit par programmation en utilisant les fonctions mises à notre disposition. Nous allons nous penché sur la première qui semble la plus abordable.

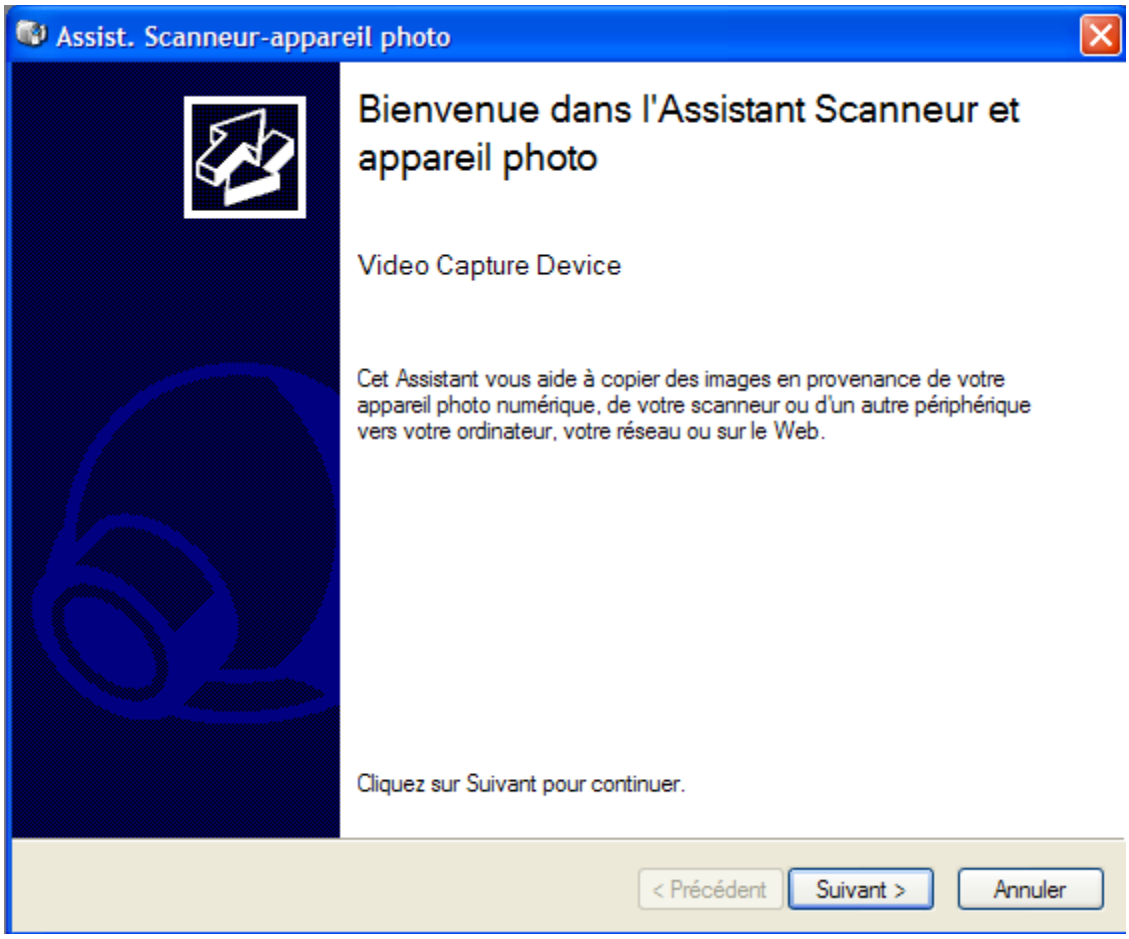
Nous allons appeler la boîte de dialogue dès l'arrêt du timer par exemple...

Donc nous allons compléter notre fonction pour gérer cette fonctionnalité.

Il suffit de rajouter les lignes de code suivantes :

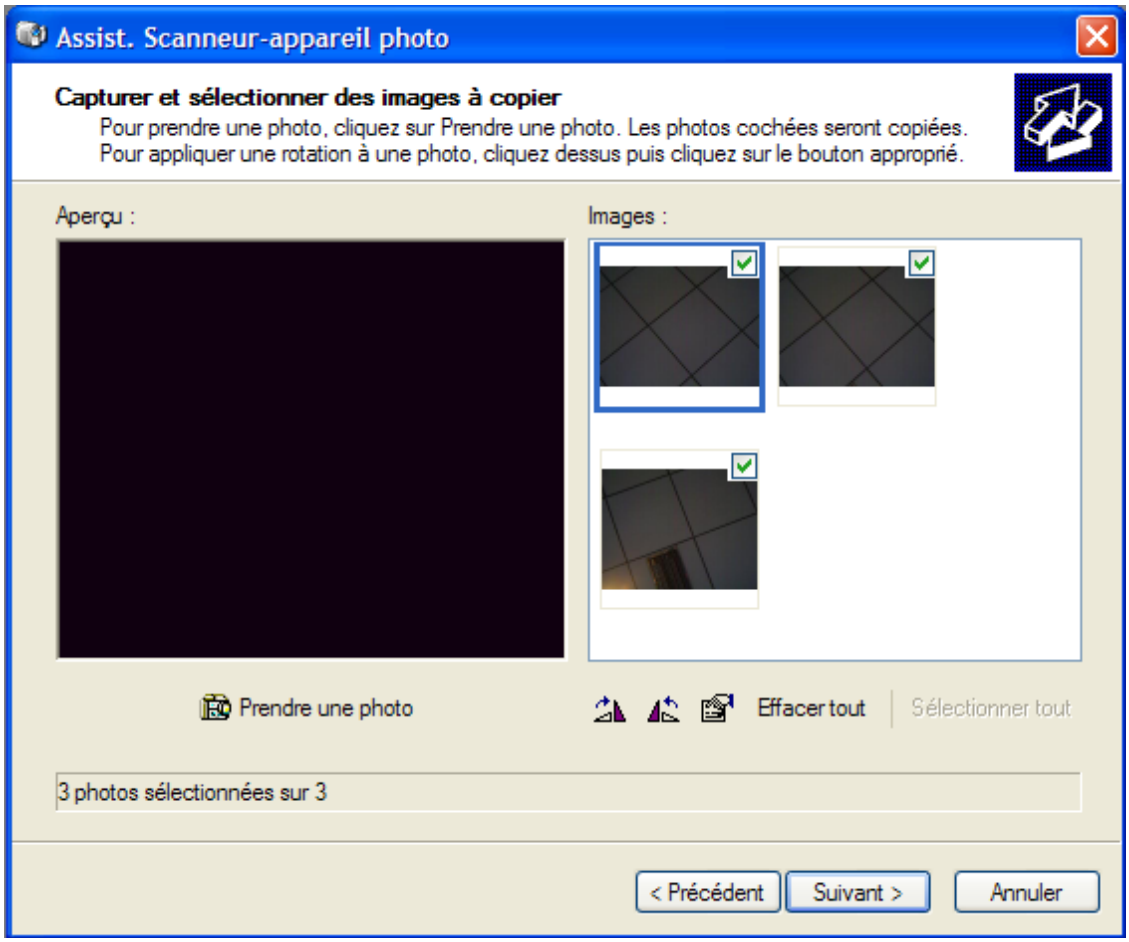
```
Dim class1 As WIA.CommonDialog
class1 = CreateObject("WIA.CommonDialog")
class1.ShowAcquisitionWizard(d)
```

Ce code a pour effet d'ouvrir les boîtes de dialogues suivante qui permettent d'importer vos photos à partir d'un appareil photo numérique ou d'un scanner.



Lorsque vous appuyez sur le bouton Suivant vous accéder à une nouvelle fenêtre permettant de

sélectionner les images que vous voulez importer :



Les étapes suivantes vous permettent de choisir le répertoire dans lequel vous voulez sauvegarder vos images. Le répertoire par défaut est bien entendu Mes images, libre à vous de changer de répertoire.

Il est donc fort simple de visualiser les photos que vous venez de prendre par l'intermédiaire des boîtes de dialogue communes fournies par Microsoft.

Voilà maintenant nous allons voir comment cette bibliothèque nous facilite la vie pour imprimer les photos importées.

VII. UTILISATION DE L'ASSISTANT D'IMPRESSION

Comme dans les cas précédents Microsoft fournit une boîte de dialogue standard commune qui permet d'imprimer ses photos acquises très simplement.

Il est possible de même d'imprimer une ou plusieurs photos en même temps. Dans le premier cas un simple objet Wia.Item suffira, alors que dans le deuxième cas, il faudra passer par un vecteur de fichiers image, Wia.Vector.

Afin de pouvoir sélectionner les fichiers images à imprimer, nous allons utiliser un objet OpenFileDialog, qui est une boîte de dialogue standard d'ouverture de fichier. Posez cet objet sur la feuille de conception de votre fiche.

Vous allez utiliser ce code qui va vous permettre de réaliser cette opération:

```
Dim class1 As WIA.CommonDialog
Dim vFiles As Object
vFiles = CreateObject("WIA.Vector")

class1 = CreateObject("WIA.CommonDialog")

OpenFileDialog1.Multiselect = True

OpenFileDialog1.Filter = "Images (*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*"
If OpenFileDialog1.ShowDialog = DialogResult.OK Then
For Each file As String In OpenFileDialog1.FileNames
        vFiles.Add(file)
Next



End If



class1.ShowPhotoPrintingWizard(vFiles)
```

Donc comme vous pouvez le constater l'appel d'une boîte de dialogue commune s'effectue de la même façon :

```
Dim class1 As WIA.CommonDialog
....
class1 = CreateObject("WIA.CommonDialog")
....
class1.ShowPhotoPrintingWizard(vFiles)
```

La fonction qui permet d'afficher l'assistant d'impression est donc ShowPrintingWizard qui prend en argument l'ensemble de fichiers que l'on a sélectionné.

Ces fichiers sont sélectionnés par l'intermédiaire de la boîte de dialogue d'ouverture de fichiers. Nous allons paramétrer cette boîte de dialogue pour qu'elle puisse ouvrir plusieurs fichiers en même temps, et pour qu'elle puisse ouvrir seulement certains types d'images, en l'occurence des fichiers image.

```
OpenFileDialog1.Multiselect = True

OpenFileDialog1.Filter = "Images (*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*"
```

Celà peut se faire soit par programmation, soit en utilisant la fenêtre de propriétés de l'objet.

Dans notre cas il suffit d'initialisé la propriété MultiSelect du composant à True et d'initialiser la propriété Filter en utilisant la ligne de code précédente.

Une fois ceci effectué il suffit d'afficher la boîte et de valider les la sélection.

```
OpenFileDialog1.ShowDialog = DialogResult.OK
```

La sélection est effective seulement si l'utilisateur appuie sur le bouton OK. C'est pour cette raison que l'on utilise un bloc conditionnel If ... Then ... End If

Une fois la sélection effectuée il faut remplir un tableau avec tous les chemins d'accès aux fichiers

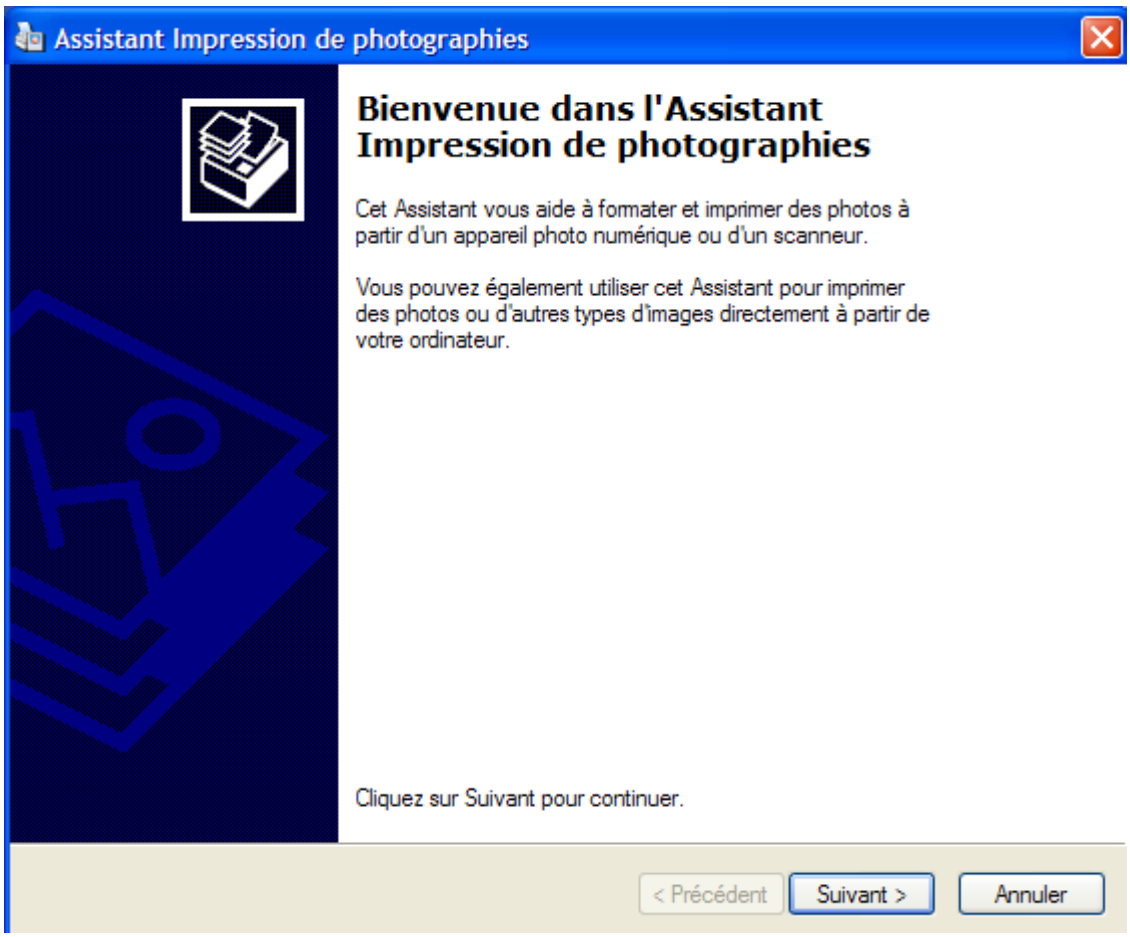
```
For Each file As String In OpenFileDialog1.FileNames
        vFiles.Add(file)
Next
```

Pour ce faire on utilise donc une boucle en utilisant un tableau vFiles que l'on va remplir en fonction du nombre total de fichiers sélectionnés. A noter que vFiles est bien une instance de l'objet WIA.Vector :

```
Dim vFiles As Object
vFiles = CreateObject("WIA.Vector")
```

Il ne reste plus qu'à passer en argument le vecteur à la fonction permettant l'affichage de l'assistant.

A l'exécution vous devriez voir apparaître ceci :



Lorsque vous appuyer sur le bouton Suivant une autre fenêtre s'ouvre vous permettant de choisir un ou plusieurs des fichiers sélectionnés, et ainsi de suite...

VIII. INFORMATIONS SUR LES PERIPHERIQUES CONNECTES

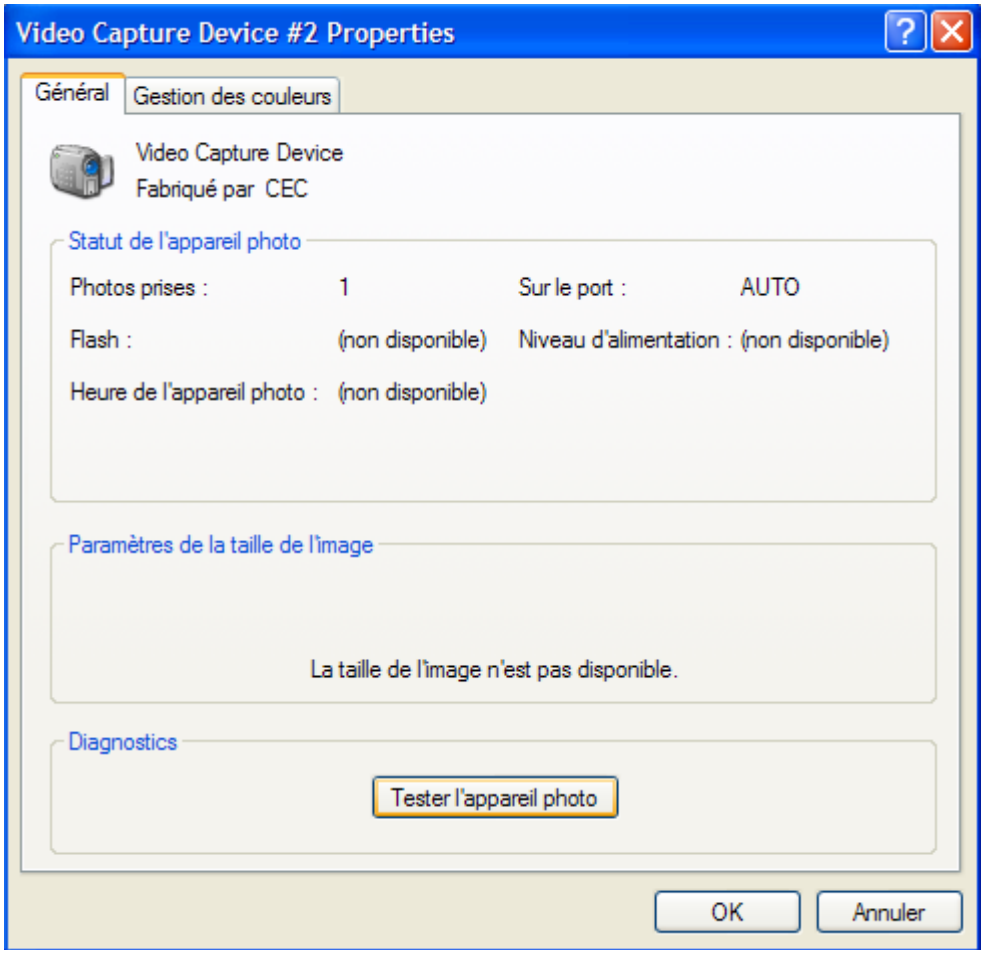
Comme d'habitude le code à fournir est basique, cet assistant est affichable par l'intermédiaire de la fonction ShowDeviceProperties

```
Dim class1 As WIA.CommonDialog
class1 = CreateObject("WIA.CommonDialog")
class1.ShowDeviceProperties(d)
```

Cette fonction prend en argument le périphérique tel que nous l'avions déclaré et instancié au moment de sa sélectionner :

```
Dim d As Device
```

Vous devriez voir apparaître ceci à l'écran :



Donc vous pouvez obtenir différentes informations concernant le périphérique, il vous est possible de tester votre appareil photo par aussi.

IX. CONCLUSION

Nous avons pu voir par l'intermédiaire de ce tutoriel, les fonctions de base permettant de gérer très facilement ses images acquises par l'intermédiaire d'un appareil photo numérique ou d'une caméra vidéo. La bibliothèque fournit tout un ensemble d'assistant prêt à l'emploi utilisable en peu de temps car nécessitant peu de code. Vous pourrez utiliser ce tutoriel de base par exemple pour construire une application de surveillance de base.