Utilisation de la bibliothèque de fonction DirectShowLib.dll
avec Delphi .Net (fonctionne avec la version Turbo Explorer).

Table des matières

I. Introduction

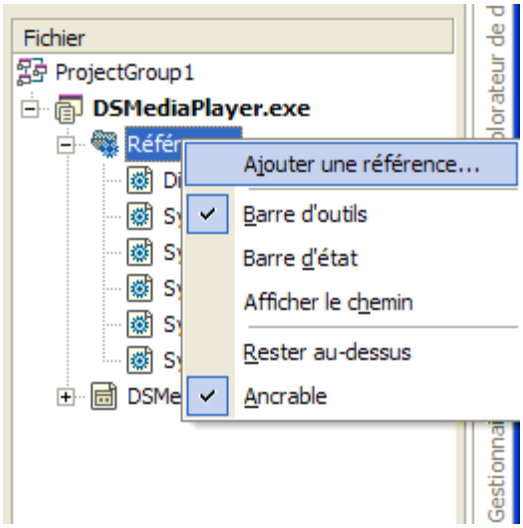
Je vais vous montrer par l'intermédiaire de ce tutoriel qu'il est possible de travailler sous l'environnement .Net avec la technologie ***DirectShow***, et ce grâce à la bibliothèque de fonctions DirectShowLib.dll. Vous verrez que l'on retrouvera les notions de graphes et de filtres ainsi que de DMO, comme sous l'environnement Win32. Cette technique est beaucoup plus élaborée que l'utilisation de la nouvelle bibliothèque de fonctions ***AudioVideoPlayback***. Nous allons donc pouvoir construire un media Player de base en utilisant les mêmes connaissances que nous avions sous Win32. Cette bibliothèque est un projet ***OpenSource*** de la communauté ***SourceForge***.

II. Préliminaires

Avant de pouvoir commencer à l'utiliser, il va falloir la télécharger, et pour ce faire rendez vous à l'adresse suivante : http://sourceforge.net/project/showfiles.php?group_id=136334

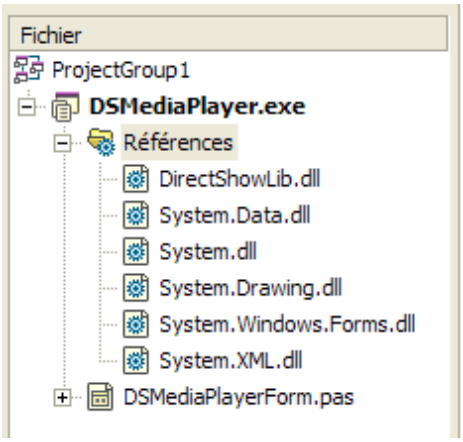
Une fois la bibliothèque dézippée, et installée sur votre machine, il ne vous reste plus qu'à ouvrir votre application Turbo Delphi.Net Explorer, de créer un nouveau projet que vous nommerez selon votre convenance. Ajoutez la dll dans le répertoire de votre projet.

Pour pouvoir l'utiliser dans votre projet, il va falloir ajouter une nouvelle référence de cette façon :



Dans la nouvelle fenêtre, cliquez sur le bouton Parcourir, et positionnez vous dans le répertoire de votre projet et sélectionnez le fichier DirectShowLib.dll puis appuyer sur OK.

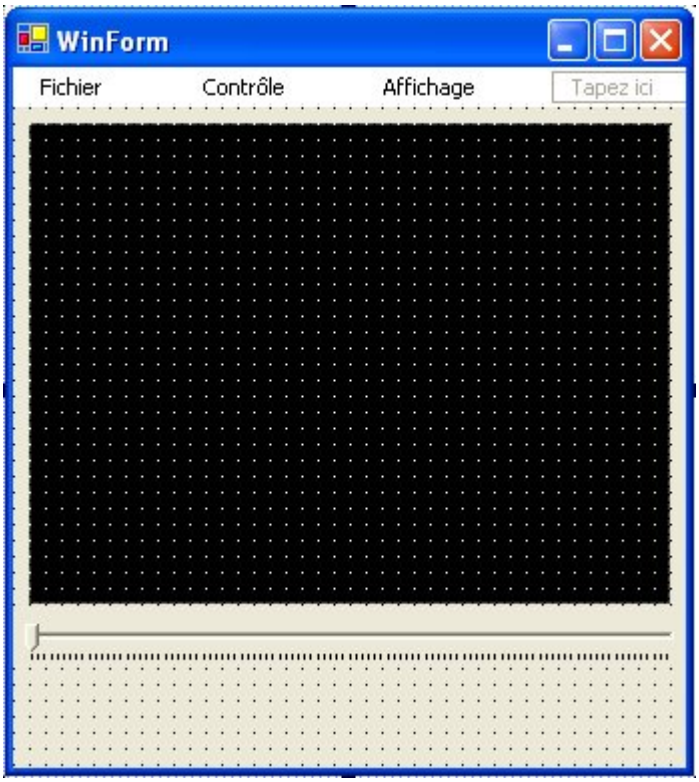
Une nouvelle référence est ajoutée dans l'onglet Gestionnaire de projet. Comme suit :



Voici maintenant les composants à ajouter sur la fiche pour pouvoir continuer notre projet :

- Une barre de menu ***TmainMenu*** pour la gestion du menu principal;
- Un contrôle ***TPanel*** qui va nous permettre de visualiser notre vidéo avec la propriété ***BackColor*** configuré sur la couleur noire;
- Un objet ***TTrackBar*** permettant de visualiser la progression de la lecture du clip ;
- Un objet ***TOpenFileDialog*** qui va nous permettre d'ouvrir notre fichier multimédia ;
- Un objet ***Ttimer*** qui va nous permettre de générer un timer Windows.

Voilà à quoi devrait ressembler votre fiche principale :



Maintenant que nous avons notre fiche qui est configuré, nous pouvons passer à la prochaine étape, le code de l'application.

III. Code de l'application

1. Ouverture d'un fichier

Tout d'abord il faut que vous définissiez dans votre ***TMainMenu*** un nouvel élément de menu comme ceci :

- Dans la zone « Tapez ici », écrivez « Fichier » par exemple ;
- Dans la case du dessous, toujours à la place de « Tapez ici », écrivez « Ouvrir » par exemple
- Double-cliquez sur cet élément et dans la fenêtre de code apparaît le squelette de votre nouvelle fonction

Ce squelette va nous permettre de coder la partie affichage et lecture de notre vidéo.

Dans un premier temps nous allons définir quelques variables locales qui vont nous servir ensuite dans l'écriture de notre fonction :

```
var fg2 : Ifiltergraph2;
    fg  : FilterGraph;
    ec : EventCode;
```

la variable fg va nous permettre de définir notre objet ***FilterGraph*** de base, c'est donc à partir de cet objet que nous allons construire notre graphe de lecture de fichier multimédia. En fait comme avec ***DSPACK*** c'est le système qui gère la création du graphe pour peu qu'il ait toutes les informations à sa disposition, sinon vous devrez passer par la « méthode manuelle ».
L'objet ***fg2*** de type ***iFilterGraph2*** va nous permettre de d'effectuer le rendu du clip multimédia.
La variable ***ec*** va nous permettre d'interagir avec des événements se produisant lors de la lecture du fichier.

Ensuite comme cette fonction a besoin de la boîte de dialogue d'ouverture de fichiers pour lui donner le nom du fichier à lire, il va falloir que l'utilisateur aura choisi est validé son choix de fichiers.
Il est à noter que la procédure d'ouverture d'un fichier avec ***le framework .Net*** diffère quelque peu de celle utilisée avec ***Win32***. En effet sous l'environnement ***Win32***, il suffit juste d'écrire le traitement à effectuer en utilisant le test suivant :

```
If OpenFileDialog1.Execute then
   Begin
   End ;
```

Alors qu'avec ***le framework .Net*** nous devons utiliser ce test :

```
if OpenFileDialog1.ShowDialog = System.Windows.Forms.DialogResult.OK then
  begin
  end ;
```

A l'intérieur de ce test nous allons donc ajouter le code permettant d'ouvrir et de lire notre fichier.
Tout d'abord créons nos instances d'objet fg et fg2 comme ceci :

```
fg := Nil;
fg := FilterGraph.Create;
fg2 := fg as IFilterGraph2;
```

Tout d'abord on s'assure qu'il n'existe pas d'instance de l'objet déjà créée ***(fg := Nil)***.
L'objet fg est créé en appelant le constructeur de la classe ***FilterGraph***. Ensuite on peut créer l'instance d'objet ***fg2*** par l'intermédiaire de l'interface IFilterGraph2, à laquelle on associe l'objet ***fg*** que l'on vient de créer précédemment.

Toujours en utilisant cet objet nous allons définir un objet COM de type ***IMediaControl mc*** qui dérive de l'interface ***IMediaControl***, comme ceci :

```
mc := fg as IMediaControl;
```

De la même façon nous allons définir un objet un objet COM de type ***mEv*** qui dérive de l'interface ***IMediaEventEx*** :

```
mEv := fg as IMediaEventEx;
```

Pour effectuer le rendu du fichier multimédia nous allons utiliser la fonction ***RenderFile,*** comme ceci :

```
fg2.RenderFile(OpenFileDialog1.FileName, nil);
```

Elle prend en argument le nom du fichier et un autre argument qui correspond à une playlist dans notre cas nous lui avons fournit le mot clé nil car nous lisons qu'un seul fichier.

Ensuite il faut définir un conteneur pour visualiser notre fichier vidéo. Vous vous rappelez que nous avions déposé un objet ***TPanel*** à cet effet. Nous allons donc utiliser celui-ci. Mais dans un premier temps il nous faut créer un objet ***VideoWindow*** en utilisant l'interface ***IVideoWindow*** avec l'objet ***fg***:

```
vw := fg as IVIdeoWindow;
```

Une fois effectué, on va déclarer l'objet Panel1 en tant que conteneur de la vidéo comme ceci :

```
vw.put_Owner(Panel1.Handle);
```

Il faut lui demander de diffuser le clip sur notre objet :

```
vw.put_WindowStyle(WindowStyle.Child);
```

Il est ensuite possible d'ajuster la video aux dimensions de notre objet conteneur :

```
vw.SetWindowPosition(0, 0, 320, 240);
```

Maintenant que nous avons configuré la lecture de notre fichier multimédia, nous pouvons réaliser la lecture proprement dite, ce que nous verrons dans la partie qui suit.

2. Lecture d'un fichier

Nous avons configuré la lecture de notre application afin qu'elle puisse s'effectuer dans notre objet conteneur, c'est-à-dire notre objet Panel1. Il nous faut maintenant effectuer la lecture via notre objet COM ***mc***. Ceci se fait simplement en utilisant la fonction ***Run*** de l'interface ***IMediaControl*** :

```
mc.Run
```

Il faut ensuite s'assurer que la diffusion s'arrêtera bien à la fin du fichier multimédia en utilisant le code suivant :

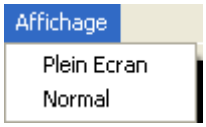
```
if mEv = nil then
      exit;
   while mEv.GetEvent(ec, p1, p2, 0) = 0 do
    begin
      mEv.FreeEventParams(ec, p1, p2); // p1 et p2 de type integer
      if ec = EventCode.Complete then // Déclarer var ec : EventCode
      begin
        mc.Stop;

      end;
    end;
```

Donc comme vous pouvez le constater on utilise les deux instances ***mc*** et ***mEv*** que nous avons définie dans la partie précédente.
C'est par l'intermédiaire de l'instance ***mEv*** que nous allons vérifier la fin de la diffusion de notre clip et en fin de compte stopper la diffusion en utilisant la méthode Stop de notre objet ***mc***.
Voilà maintenant nous avons un lecteur multimédia de base qui nous permet juste de lire notre fichier dans sa totalité. Nous allons maintenant définir quelques contrôles qui vont nous permettre de réaliser quelques opérations comme mettre en pause notre lecture, stopper la lecture, etc...

3. Contrôle de la lecture

Tout d'abord voyons les contrôles de base que nous pouvons introduire dans notre application multimédia. Ceux-ci vont nous permettre d'interagir avec la lecture.
Toutes ces commandes vont être accessibles par une commande de menu bien spécifique comme le montre l'image suivante :

Nous verrons de même comment afficher une vidéo en mode plein écran et en mode normal.
Cela se fera par l'intermédiaire des commandes « plein Ecran » et « Normal » du menu Affichage.



Maintenant nous avons plus qu'à détailler ces principales fonctions qui ont nous permettre de réaliser notre Media player de base.

a. Jouer le fichier

Pour jouer le fichier, nous l'avons vu précédemment il suffit d'utiliser notre objet MediaControl (mc), en lui associant sa méthode Run :

```
procedure TWinForm.miJouer_Click(sender: System.Object; e: System.EventArgs);
begin
   mc.Run;
end;
```

b. Mettre en Pause la lecture du Fichier

Pour mettre en pause la lecture d'un fichier multimédia, on utilise le même objet mais avec la méthode Pause :

```
procedure TWinForm.miPause_Click(sender: System.Object; e: System.EventArgs);
begin
   mc.Pause;
end;
```

c. Stopper la lecture du fichier

Pour stopper la diffusion d'un clip, il suffit d'utiliser la méthode Stop de cet objet :

```
procedure TWinForm.miStop_Click(sender: System.Object; e: System.EventArgs);
begin
   mc.Stop;
end;
```

d. Mode d'affichage

Comme nous l'avons dit plus haut dans cette partie il est possible de lire notre vidéo en mode plein écran et de basculer aussi en mode normal.
Ceci s'effectue grâce à la fonction ***put_FullScreenMode*** qui prend en argument un booléen permettant de changer de mode (***fullScreenMode***). Donc si nous voulons visualiser notre vidéo en mode Plein Ecran on va utiliser ce code :

```
vw.put_FullScreenMode(OABool.True);
```

Si nous voulons visualiser notre vidéo en mode Normal il suffit d'utiliser le code suivant :

```
vw.put_FullScreenMode(OABool.False);
```

4. Visualisation de la progression de la lecture

Si vous lancez l'application telle qu'elle, vous rendrez compte que la diffusion de notre fichier multimédia n'est pas optimale. En effet nous n'avons aucun moyen de contrôle sur le processus de diffusion. En effet la diffusion ne s'arrête qu'une fois que la totalité du media a été diffusé.
Pour ce faire on va utiliser notre ***Timer*** qui nous permet à chaque interruption d'avoir un accès à l'interface utilisateur, nous allons donc pouvoir effectuer certains contrôles sur la lecture de notre clip.
Avant le bloc de code précédent vous allez introduire la ligne suivante :

```
Timer1.Enabled := True;
```

Cela nous permet d'activer notre timer. Pour que notre objet soit fonctionnel il faut lui ordonner de faire quelques tâches spécifiques.

Tout d'abord nous avons besoin de connaître la position courante dans notre clip à un instant donné, pour ce faire il nous faut créer un objet COM ***mp*** qui dérive de l'interface ***IMediaPosition*** :

```
mp := fg as IMediaPosition;
```

Vous pouvez le faire au moment de la création de tous les objets COM que nous avons vu précédemment.

Une fois ceci effectué nous pouvons obtenir la position courante de notre clip ainsi que la durée totale de notre clip. Nous obtenons la durée totale de notre clip via la fonction ***get_Duration*** de l'interface ***IMediaPosition*** :

```
mp.get_Duration(pLength);
```

pLength étant de type ***double***.

Nous obtenons de même la position courante par l'intermédiaire de la fonction ***get_CurrentPosition*** de l'interface ***IMediaPosition*** :

```
mp.get_CurrentPosition(pTime);
```

pTime étant de type ***double***.

Maintenant que nous avons recueilli les informations dont nous avons besoin, nous pouvons calculer la position à la position courante de notre composant ***TTrackBar*** . Pour ce faire il suffit de calculer le delta :

```
DeltaH := 100 / pLength * pTime;
```

DeltaH étant de type ***double***.

Il suffit ensuite d'attribuer ce Delta à la valeur de notre ***TrackBar*** :

```
TrackBar1.Value := Convert.ToInt16(DeltaH);
```

En fonction de notre position on peut changer notre mode d'affichage. Par exemple si la position de notre TrackBar est égale à 100, c'est-à-dire que la diffusion de notre clip est terminée, on peut passer du mode plein écran au mode normal :

```
if Trackbar1.Value = 100 then
  begin
  vw.put_FullScreenMode(OABool.False);
  mp.put_CurrentPosition(0);
  Trackbar1.Value := 0;
  mc.Run;
  mc.Pause;
  end;
```

IV.Conclusion

Nous avons vu comment utiliser la bibliothèque de fonctions DirectShowLib avec Turbo Delphi .Net pour concevoir une petite application multimédia de base. Nous avons vu que l'on pouvait utiliser ces connaissances acquises avec DSPACK sous Win32, en effet vous pourrez reprendre sans trop de mal vos développements sous Win32 pour les adapter sous le framework .Net. Nous verrons prochainement comment utiliser les filtres avec DirectShowLib.