EXTRACTION ET CONVERSION AVEC DIRECTSHOW ET DSPACK 234 SOUS DELPHI

Table des matières

I. INTRODUCTION....2
II. DESIGN DE LA FICHE....2
III. Etude du graphe d'extraction du son et de conversion....3
IV . Code de l'application....4
1.Ajout des filtres au graphe....4
a. Ajout du filtre source....4
b. Ajout des filtres intermédiaires....4
c. Ajout du filtre FILE WRITER....5
2.Connections des filtres....5
3.Jouer le graphe....6
4.Prise en charge de notre ProgressBar :....7
V. CONCLUSION....8

I. INTRODUCTION

Nous allons voir avec ce petit tutoriel comment il est possible de réaliser l'extraction de la bande sonore d'un fichier .wmv pour ensuite le convertir en fichier .wav. Nous allons dans un premier temps voir la conception de notre petit module de conversion qui peut aisément être rajouté au media player que nous avons déjà réalisé lors de notre premier tutoriel. Nous allons dans un second temps analysé le graphe permettant de réaliser cette conversion, puis nous en viendrons au codage de ce petit module.

II. DESIGN DE LA FICHE

Voici un aperçu de la fiche que l'on utilisera pour ce tutoriel :

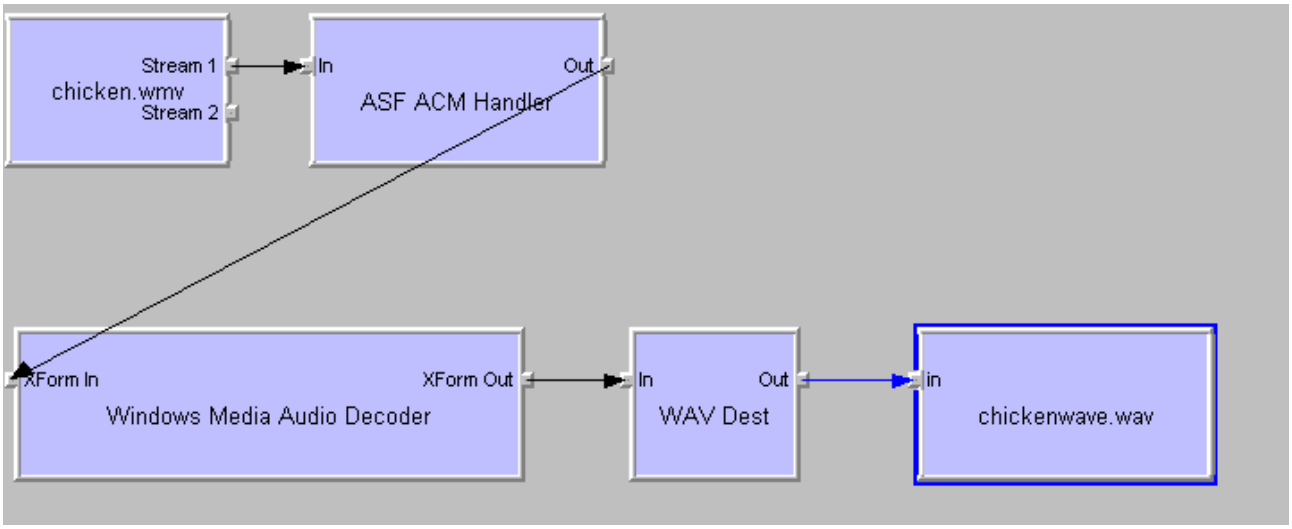
Donc vous allez créer une nouvelle fiche ressemblant à celle-ci. Cette fiche doit comporter les composants suivants :

- Un bouton permettant d'ouvrir la boîte de dialogue d'ouverture de fichiers multimédia.
- Un bouton permettant de sauvegarder le fichier wav ouvrant la boîte de dialogue de sauvegarde.
- Les deux boîtes de dialogue d'ouverture et de sauvegarde de fichier.
- Un bouton 'Démarrer la conversion' qui va nous permettre de faire jouer la graphe d'extraction et de conversion de fichiers wmv.
- Un composant ProgressBar pour voir l'état de la conversion.
- Un composant Timer1 qui va nous permettre de mettre à jour la position de notre progressBar1.

Nous allons donc maintenant étudier le graphe d'extraction et de conversion de fichiers wmv.

III. Etude du graphe d'extraction du son et de conversion

Voici le graphe que l'on va utiliser pour effectuer cette conversion.



Comme vous pouvez le constater nous n'utiliserons que la partie audio du signal, donc nous allons détailler ce graphe maintenant.

Nous allons utiliser comme filtre source le Windows Media Source Filter qui est présent dans la catégorie DirectShow Filters. C'est le filtre spécifique permettant d'ouvrir des fichiers Windows Media File.Ce filtre était utilisé avec le lecteur Windows Media Player 6,4. Ce filtre n'utilise pas le Windows Media Format SDK, donc le graphe ne peut pas inclure d'objet DMO. La partie audio doit toujours être reliée à un filtre ASF ACM Handler.

Nous relions donc ensuite la sortie de notre filtre source avec l'entrée du filtre intermédiaire ASF ACM Handler. Il est utilisé de même pour garantir la compatibilité avec le lecteur Windows Media Player 6.4. Les nouvelles applications auront plutôt recours à un filtre comme le WM ASF Reader. Mais celui ci ne gère pas les flux (stream). D'ailleurs Microsoft préconise l'utilisation du Windows Media Player SDK pour la gestion des flux.

La sortie du filtre ASF ACM Handler doit être relié ensuite à un filtre qui va décoder les informations du fichier de type Windows Media Audio, celui-ci se nomme Windows Media Audio Decoder. Ce filtre convertit les données au format compressé wma au format non compressé (PCM 16 bits).

Pour pouvoir convertir le format wma vers le format WAV, il nous faut ensuite un filtre de destination permettant de convertir le fichier vers le format wav, celui ci est le fitre WAV DEST. Il prend donc en entrée un flux audio qui dans notre cas était à l'origine au format wma.

Il nous reste plus qu'à enregistrer notre fichier wav sur disque ceci est réalisé grâce au filtre FILE WRITER.

Donc nous en avons fini avec l 'étude de notre graphe nous allons passer maintenant à la partie

codage de notre petite application.

IV . Code de l'application

1. Ajout des filtres au graphe

Comme nous l'a montré l'étude du graphe que l'on vient de réaliser, nous avons vu que nous avions cinq filtres à ajouter au graphe, c'est ce que nous allons réaliser maintenant.

a. Ajout du filtre source

Nous allons commencé par le filtre qui va nous permettre d'ouvrir un fichier au format Windows Media File. Nous avons vu qu'il s'agissait du filtre Windows Media Source Filter.
Le nom du fichier à ouvrir est obtenu par l'intermédiaire de notre composant OpenDialog. Donc voici le code permettant d'ajouter notre filtre à notre graphe.

```
SourceFilter := AjoutFiltre('DirectShow Filters', 'Windows Media source filter', grph);

SourceFilter.QueryInterFace(IID_IFileSOurceFilter, fileSourceFilter);
string2wchar:=stringtowidechar(OpenDialog1.FileName,buffer1,255);
fileSourceFilter.load(string2wchar, nil);
```

Je vous rappelle que la fonction ***AjoutFiltre*** est une fonction que j'ai déjà décrite lors de mon tutoriel sur la gestion des fichiers WMV.
Je vous rappelle que l'on est obligé de convertir notre chaîne de caractères renfermant le nom du fichier en type PwideChar ce que fait la fonction stringtowidechar.
String2wchar étant de type PwideChar.

b. Ajout des filtres intermédiaires

Le premier filtre intermédiaire est le filtre ASF ACM Handler. L'ajout de ce filtre se fait par l'intermédiaire du code suivant :

```
IIntermediateFilter3 := AjoutFiltre('DirectShow Filters', 'ASF ACM Handler', grph);
```

Ce filtre se trouve toujours dans la catégorie DirectShow Filters.

Le deuxième filtre à ajouter est le filtre 'Windows Media Audio Decoder' qui se trouve toujours dans la catégorie DirectShow Filters :

```
IIntermediateFilter2 := AjoutFiltre('DirectShow Filters',
                                    'Windows Media Audio Decoder', grph);
```

Le troisième filtre à ajouter est le filtre 'WAV DEST', nous le faisons grâce au code suivant :

IIntermediateFilter := AjoutFiltre('DirectShow Filters', 'WAV Dest', grph);

c. Ajout du filtre FILE WRITER

Il nous plus qu'à insérer dans notre graphe notre filtre de destination, c'est-à-dire celui qui va nous permettre d'enregistrer notre fichier sur disque. Le code qui permet d'ajouter 'FILE WRITER' est le suivant :

IDestFilter := AjoutFiltre('DirectShow Filters', 'File writer', grph);

Ceci ne suffit pas pourtant, il faut pouvoir renseigner le nom du fichier de sauvegarde à notre filtre. Ceci s'effectue en implémentant l'interface IfileSinkFilter2, comme suit :

Filter.QueryInterFace(IID_IFileSinkFilter2, fileSinkFilter);

fileSinkFilter est de type IfileSinkFilter2.

L'interface IfileSinkFilter2 contient une fonction permettant d'associer le nom du fichier à notre objet que l'on vient de déclarer, celle-ci est SetFileName. Voici le code qui permet de réaliser cette opération :

```
string2wchar2:=stringtowidechar(SaveDialog1.FileName,buffer2,255);
fileSinkFilter.setfileName(string2wchar2, nil);
```

Maintenant que nous avons inséré les filtres dans notre graphe, il nous reste à les connecter entre eux. Ceci est l'objet de la prochaine partie.

2. Connections des filtres

Tout d'abord il nous faut connecter la broche de sortie de notre filtre source qui se nomme 'Stream1' avec la broche d'entrée de notre filtre intermédiaire (ASF ACM Handler) qui se nomme 'In'. Voici le code nécessaire :

ConnectionFiltres('Stream 1', 'In', SourceFilter, IintermediateFilter3);

Il nous faut ensuite connecter la sortie de notre premier filtre intermédiaire qui se nomme 'Out' avec l'entrée de notre deuxième filtre intermédiaire (Windows Media Audio Decoder) qui se nomme 'Xform In' :

ConnectionFiltres('Out', 'In', IIntermediateFilter3, IintermediateFilter2);

Il faut ensuite relier la broche 'Out' de notre deuxième filtre intermédiaire avec l'entré 'In' de notre troisième filtre intermédiaire (WAV Dest).

ConnectionFiltres('Out', 'In', IIntermediateFilter2, IintermediateFilter);

Il nous reste plus qu'à relier la sortie de notre troisième filtre intermédiaire avec l'entrée 'In' de notre

filtre terminal (File writer).

ConnectionFiltres('Out', 'In', IIntermediateFilter, IdestFilter);

Maintenant que nos filtres sont connectés entre eux il nous reste plus qu'à lancer le processus d'extraction et de conversion de la partie audio de notre fichier wmv.

3. Jouer le graphe

Ceci commence par la création d'un objet de type IcaptureGraphBuilder2 en appelant l'interface IcaptureGraphBuilder2. On lui associe ensuite notre objet de type IfilterGraph par l'intermédiaire de la fonction SetFilterGraph :

```
hr := CoCreateInstance(CLSID_CaptureGraphBuilder2, nil,
                       CLSCTX_INPROC_SERVER, IID_ICaptureGraphBuilder2,
                       ICapGraph);
IcapGraph.SetFilterGraph(grph);
```

Il faut ensuite effectuer le rendu de notre graphe ceci s 'effectue par l'intermédiaire de la fonction ***RenderStream***, en lui associant les filtres source et de destination :

```
IcapGraph.RenderStream(nil,nil,SourceFilter, nil, IDestFilter);
```

Comme dans le tutoriel précédent lorsqu'un graphe est en train d'être joué, il monopolise toutes les ressources, et nous n'avons plus aucun contrôle sur notre application. Pour pallier ce problème, il faut rajouter le code suivant :

```
fElapsed := Now;
nCount := 0;
Timer1.Enabled :=True;
while nCount = 0 do
    begin
    if mEvent <> nil then
      begin
      mEvent.WaitForCompletion(100, nCount);

      Application.ProcessMessages;

      end
   else break;
end;
```

L'Ajout de ***Application.ProcessMessages*** permet d'avoir un contrôle sur les autres messages qui pourraient intervenir lors du rendu de notre graphe.

L'objet ***mEvent*** est obtenu en faisant appel à l'interface ***ImediaEvent*** comme suit :

```
grph.QueryInterface(IID_IMediaEvent, mEvent);
```

Maintenant que notre graphe est prêt à effectuer sa fonction de conversion nous allons effectuer la prise en charge de notre ***ProgressBar1***.

4. Prise en charge de notre ProgressBar :

La première chose à faire est de créer un objet de type ***ImediaSeeking*** en appelant l'interface ***ImediaSeeking***:

```
if failed(grph.QueryInterface(IID_IMediaSeeking, mMediaSeeking)) then
        ShowMessage('Can not get IMediaSeeking interface');
```

Il faut noter que toute les méthodes de notre tutoriel sont des fonctions qui renvoie un résultat, je les ai omis par facilité, mais elles permettent par exemple d'afficher un message d'avertissement si l'opération demandée ne s'est pas bien déroulée. Sachez cependant que ce modèle est utilisable pour toutes les autres méthodes.
Cet objet va nous permettre de recueillir la position courante dans le fichier, et la durée totale du fichier. Ces informations sont nécessaires pour mettre à jour la position de notre progressBar au fur et à mesure de la conversion de notre fichier. Bien entendu a mise à jour de la position se fera par l'utilisation de notre objet Timer.

A partir de notre objet, nous pouvons obtenir la durée totale de notre clip :

```
mMediaSeeking.GetDuration(dDuree);
```

dDuree étant de type int64.

Une fois cette information acquise, il nous faut acquérir l'information concernant la position actuelle dans le clip, et ceci à chaque appel de la méthode événement OnTimer de notre objet Timer.
Pour réaliser cette opération il suffit donc d'écrire le code suivant :

```
mMediaSeeking.GetCurrentPosition(dCurrentTime);
```

dCurrentTime étant aussi de type int64.

Ensuite pour mettre à jour la position de notre progressBar, il suffit de réaliser cette opération :

```
ProgressBar1.Position :=  Round((100 * dCurrentTime/dDuree));
```

Pour indication je vous donne le code complet de la méthode :

```
procedure TForm1.Timer1Timer(Sender: TObject);
var ftime : TDateTime;

begin
fTime := Now-fElapsed;

mMediaSeeking.GetCurrentPosition(dCurrentTime);
ProgressBar1.Position :=  Round((100 * dCurrentTime/dDuree));
end;
```

V. CONCLUSION

Nous venons de voir comment réaliser une opération d'extraction de l'audio à partir d'un clip audiovisuel au format wmv et de convertir cette partie sous le format wav. Cette opération somme toute assez simple, vous permettra de réaliser d'autres opérations de conversions assez simplement en vous appuyant sur cet exemple.