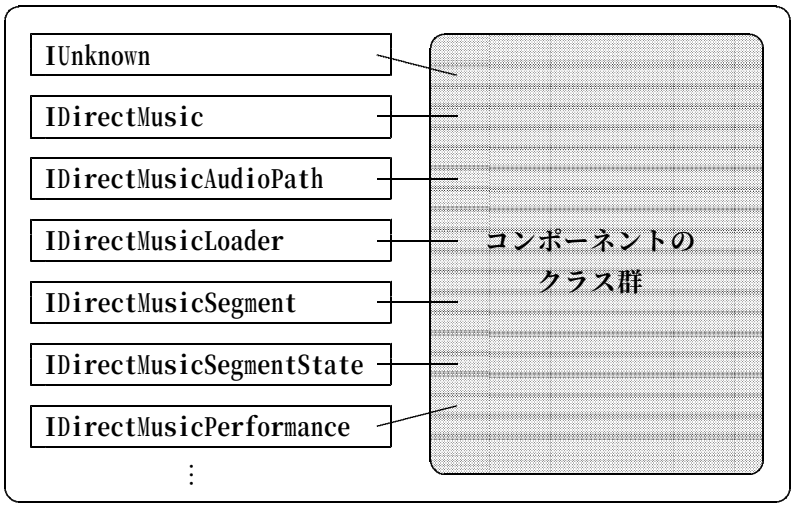
オブジェクト指向と ゲームプログラミング

▶ DirectX Audio編 － 第２回　DirectMusicオブジェクトの生成と解放 ◀

DirectMusicコンポーネント

DirectMusicの機能を使用するには、DirectMusicコンポーネントのインスタンス(DirectMusicオブジェクト)を生成し、そのインタフェースを取得しなければなりません。
DirectMusicコンポーネントはdmusic.dllに格納されていて、セグメントやパフォーマンスなど複数のコンポーネントの集合として実装されています。



DirectMusicの基本的なコンポーネント

DirectMusicでは、さまざまなコンポーネントを使用してオーディオファイルを再生します。ファイルを読み込んで再生する場合、以下のコンポーネントのオブジェクトが必要になります。

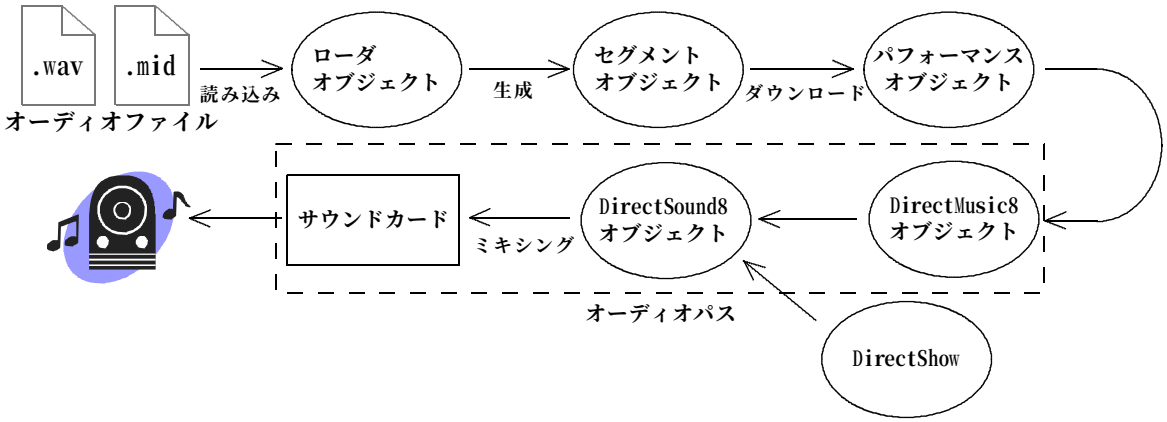
DirectMusicコンポーネント
DirectMusicの最も基本的なコンポーネントで、バッファ、ポート、マスタクロックを管理します。内部的に作成され、暗黙的に使用されるので、たいていの場合、このオブジェクトを生成する必要はありません。

DirectSoundコンポーネント
サウンドカードを制御するDirectSoundの最も基本的なコンポーネントです。DirectX Audioでは、オーディオデータの制御はDirectMusicだけで十分に行えますが、DirectShowとミキシングしたり、サウンドデータが格納されているメモリ領域(バッファ)にアクセスするような高度な操作を行う場合には必要となります。このオブジェクトも内部的に生成され、暗黙的に使用されます。

DirectMusicLoaderコンポーネント
オーディオファイルの検索や読み込みに使用するコンポーネントです。

DirectMusicPerformanceコンポーネント
再生の総合的な管理を行うコンポーネントです。オーディオデータの経路(オーディオパス)の設定、DirectMusicSegmentオブジェクトの再生状態などを管理します。

DirectMusicSegmentコンポーネント
DirectMusicSegmentコンポーネントのオブジェクトは、個々のオーディオデータを管理します。オーディオファイルは、ローダオブジェクトによってこのオブジェクトに読み込まれます。そして、これをパフォーマンスオブジェクトにダウンロードすることにより、再生できるようになります。



DirectMusicLoaderオブジェクトの生成

標準MIDIファイル(.mid)やWaveファイル(.wav)をDirectMusicに読み込む場合、DirectMusicLoaderオブジェクトが必要になります。
DirectMusicLoaderオブジェクトの生成は、CoCreateInstance関数で行います。１つ目の引数(クラスID)に「CLSID_DirectMusicLoader」、４つ目の引数(インタフェースID)に「IID_IDirectMusicLoader8」を指定します。

```
// DirectMusicLoaderオブジェクトの生成とインタフェースの取得
IDirectMusicLoader8*   pDMLoader = NULL;
CoCreateInstance(CLSID_DirectMusicLoader, NULL, CLSCTX_INPROC_SERVER,
                 IID_IDirectMusicLoader8, (LPVOID*)&pDMLoader);
```

関数が成功すると、５つ目の引数にインタフェースが格納されます。DirectMusicLoaderオブジェクトは、初期化する必要が無く、そのまますべての機能が使用できます。

オブジェクトの解放

生成または取得したCOMオブジェクトは、不用になったら解放しなければなりません。すべてのCOMオブジェクトには、解放するためのReleaseメソッドがあります。解放されたオブジェクトは再び初期化するまで使用できないので、インタフェースを格納していた変数にはNULLを代入し、無効なインタフェースであることを明示するようにします。こうしておくと、解放したインタフェースを再使用してしまい、不安定な動作になるという症状を防ぐことができます。

```
// DirectMusicLoaderオブジェクトの解放
if(pDMLoader != NULL) {      // ポインタチェック。NULLの場合は解放しない(できない)
    pDMLoader->Release();    // DirectMusicLoaderオブジェクト解放
    pDMLoader = NULL;        // NULLを代入し、無効なインタフェイスであることを示す
}
```

DirectMusicPerformanceオブジェクトの生成

DirectMusicでオーディオデータを再生するには、DirectMusicPerformanceオブジェクトが必要になります。
DirectMusicPerformanceオブジェクトの生成は、CoCreateInstance関数で行います。１つ目の引数(クラスID)に「CLSID_DirectMusicPerformance」、４つ目の引数(インタフェースID)に「IID_IDirectMusicPerformance8」を指定します。

```
// DirectMusicPerformanceオブジェクトの生成とインタフェースの取得
IDirectMusicPerformance8*   pDMPerformance = NULL;
CoCreateInstance(CLSID_DirectMusicPerformance, NULL, CLSCTX_INPROC_SERVER,
                 IID_IDirectMusicPerformance8, (LPVOID*)&pDMPerformance);
```

関数が成功すると、５つ目の引数にインタフェースが格納されます。

パフォーマンスの初期化

DirectMusicPerformanceオブジェクトは、初期化し、オーディオパスを設定しなければ再生することができません。
オーディオパスとは、オーディオデータがどのような経路をとおってスピーカーに到達するのかを定義するものです。たとえば、MIDIデータは外部に接続されているMIDI音源を使うのか、それともソフトウェアシンセサイザを使うのかを定義する必要があります。Waveデータはどのサウンドカードを使って再生するのかを決める必要があります。
オーディオパスはかなり詳細に定義できますが、通常はデフォルトのオーディオパスで十分です。デフォルトオーディオパスでは、MIDIデータは、ソフトウェアシンセサイザ「Microsoft GS Wavetable Synth」でWaveデータに変換されます。Waveデータは、DirectSoundによりほかのデータとミキシングされ、デフォルトのサウンドカードによってスピーカーに出力されます。
パフォーマンスの初期化とデフォルトオーディオパスをセットアップは、IDirectMusicPerformance8::InitAudioメソッドで行います。

IDirectMusicSegment8::InitAudioメソッド

－説明－
InitAudioメソッドは、パフォーマンスを初期化し、オプションとしてデフォルトオーディオパスをセットアップします。

－書式－
```
HRESULT InitAudio(IDirectMusic** ppDirectMusic, IDirectSound** ppDirectSound,
                  HWND hWnd, DWORD dwDefaultPathType, DWORD dwPChannelCount,
                  DWORD dwFlags, DMUS_AUDIOPARAMS* pParams);
```

－パラメータ－
１つ目の引数(ppDirectMusic)は、DirectMusicオブジェクトのインタフェースを指定または取得する場合に使用します。NULLを指定すると暗黙的に生成され、使用されます。

２つ目の引数(ppDirectSound)は、DirectSoundオブジェクトのインタフェースを指定または取得する場合に使用します。NULLを指定すると暗黙的に生成され、使用されます。

３つ目の引数(hWnd)は、DirectSoundの初期化に使用するメインウィンドウのハンドルです。NULLを指定すると、現在前面にあるウィンドウが使用されます。

４つ目の引数(dwDefaultPathType)は、オーディオパスのタイプを指定します。おもに以下のものを使用します。

DMUS_APATH_DYNAMIC_MONO	モノラル出力
DMUS_APATH_DYNAMIC_STEREO	ステレオ出力
DMUS_APATH_SHARED_STEREOPLUSREVERB	ステレオ出力とリバーブ効果
DMUS_APATH_DYNAMIC_3D	3Dサウンド

５つ目の引数(dwPChannnelCount)は、パスに割り当てるチャンネル数(同時に再生できる数)です。

６つ目の引数は、要求する機能を指定します。通常は「DMUS_AUDIOF_ALL」を指定し、すべての機能が使用できるようにします。

７つ目の引数は、シンセサイザ(データを再生する機能)に必要な機能を指定します。使用しない場合はNULLにします。

－戻り値－
成功した場合はS_OK、それ以外はエラーコードを返します。

```
// パフォーマンスの初期化とデフォルトオーディオパスのセットアップ
pDMPerformance->InitAudio(NULL, NULL, hWnd, DMUS_APATH_DYNAMIC_STEREO, 64,
                          DMUS_AUDIOF_ALL, NULL);    // hWndはメインウィンドウのハンドル
```

パフォーマンスオブジェクトの解放

パフォーマンスオブジェクトは、CloseDownメソッドでオブジェクトを閉じたあと、Releaseメソッドで解放します。

```
// パフォーマンスオブジェクトの解放
pDMPerformance->CloseDown();     // オブジェクトを閉じる
pDMPerformance->Release();       // オブジェクトを解放
```

課　　題

CDXAduio8クラスに、DirectX Audioの初期化、解放を行う処理を追加しましょう。

(１)CDXAduio8クラスに、ローダとパフォーマンスを管理する以下のメンバを追加しましょう。

```
IDirectMusicLoader8*        m_pDMLoader;          // ローダ
IDirectMusicPerformance8*   m_pDMPerformance;     // パフォーマンス
```

(２)CDXAduio8クラスのインスタンス生成時に、メンバ変数に初期値が設定されるようにします。コンストラクタを以下のように変更し、初期化子を追加しましょう。

```
CDXAudio8::CDXAudio8() : m_pDMLoader(NULL), m_pDMPerformance(NULL)
{
}
```

(３)DirectX Audioを初期化してCDXAduio8クラスを使用可能にするInitialize関数と、DirectX AudioおよびCDXAduio8クラスが使用していた資源を解放するRelease関数を作成します。
以下のプロトタイプを適切な場所に追加しましょう。

```
bool Initialize(const HWND hWnd);
void Release();
```

(４)Initialize関数およびRelease関数を実装します。以下のプログラムをソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                          初期化                                                   */
/*****************************************************************************************************/
bool CDXAduio8::Initialize(const HWND hWnd)
{
    Release();

    return true;
}

/*****************************************************************************************************/
/*                                          解　放                                                   */
/*****************************************************************************************************/
void CDXAduio8::Release()
{
}
```

(５)以下のプログラムをデストラクタの適切な場所に追加しましょう。

```
Release();
```

デストラクタに解放処理を記述しておけば、Release関数を呼び忘れたとしても、CDXAduio8クラスのインスタンスが破棄されるときにデストラクタが呼び出されるため、資源の解放漏れ(リソースリーク)を防ぐことができます。

(６)CDXAduio8クラスのコンストラクタの適切な場所に、COMライブラリの初期化を行うプログラムを追加しましょう。

(７)CDXAduio8クラスのデストラクタの適切な場所に、COMライブラリの解放を行うプログラム追加しましょう。

(８)ローダオブジェクトの生成を行うプログラムを追加します。Initialize関数の適切な場所に、以下のプログラムを完成させて追加しましょう。

```
// DirectMusicLoaderオブジェクトの生成とインタフェースの取得
if(::※ここは各自考えましょう != S_OK) {
    ::OutputDebugString("*** Error - ローダオブジェクト生成失敗(CDXAudio8_Initialize)\n");
```

```
        Release();
        return false;
    }
```

(９)パフォーマンスオブジェクトの生成と初期化を行うプログラムを追加します。Initialize関数の適切な場所に、以下のプログラムを完成させて追加しましょう。

```
    // DirectMusicPerformanceオブジェクトの生成とインタフェースの取得
    if(::※ここは各自考えましょう != S_OK) {
        ::OutputDebugString("*** Error - パフォーマンスオブジェクト生成失敗(CDXAudio8_Initialize)\n");
        Release();
        return false;
    }

    // パフォーマンスの初期化とデフォルトオーディオパスのセットアップ
    if(m_pDMPerformance->※ここは各自考えましょう != S_OK) {
        ::OutputDebugString("*** Error - パフォーマンス初期化失敗(CDXAudio8_Initialize)\n");
        Release();
        return false;
    }
```

(７)ローダオブジェクトとパフォーマンスオブジェクトの解放を行うプログラムを追加します。Release関数の適切な場所に、以下のプログラムを完成させて追加しましょう。

```
    // パフォーマンスオブジェクト解放
    if(m_pDMPerformance != NULL) {
        ※ここは各自考えましょう;
        ※ここは各自考えましょう;
        m_pDMPerformance = NULL;
    }

    // ローダオブジェクト解放
    if(m_pDMLoader != NULL) {
        ※ここは各自考えましょう;
        ※ここは各自考えましょう;
    }
```

(10)オブジェクトの解放処理は、DirectX Audioオブジェクト以外でもまったく同じ処理で行うことが多いので、テンプレートを作成しておきます。
以下のプログラムを適切な場所に追加しましょう。

```
template <class T> inline void SafeRelease(T& x) { if(x != NULL) { x->Release(); x = NULL; }}
```

テンプレート関数SafeReleaseにより、ローダオブジェクトの解放は以下のように記述できます。

```
SafeRelease(m_pDMLoader);       // ローダオブジェクトの解放
```

(11)ソースファイル"CGameApp.cpp"でCDXAudio8クラスが認識できるように、適切なヘッダファイルをインクルードしましょう。

(12)アプリケーションの初期化を行うCGameApp::Initialize関数の適切な場所に、以下の初期化１または初期化２を追加しましょう。

－初期化１－

```
// DirectX Audio初期化
if(DXAudio().Initialize(GetHWnd()) == false)
    return false;
```

－初期化２－

```
// DirectX Audio初期化
DXAudio().Initialize(GetHWnd());
```

(13)アプリケーションの解放を行うCGameApp::Release関数の適切な場所に、以下のプログラムを追加しましょう。

```
DXAudio().Release();  // DirectX Audio解放
```