オブジェクト指向とゲームプログラミング

▶ DirectX Audio編 － 第３回　オーディオファイルの読み込み ◀

オーディオファイルの読み込み

DirectMusicは、標準MIDIファイルとWaveファイルを読み込むメソッドIDirectMusicLoader8::LoadObjectFromFileを提供しています。Waveファイルの場合は、圧縮された形式(ADPCMやMP3)も読み込むことができます。このメソッドは、オーディオファイルを読み込み、それを管理するDirectMusicSegmentオブジェクトを生成します。

IDirectMusicSegment8::LoadObjectFromFileメソッド

－説明－

LoadObjectFromFileメソッドは、オーディオファイルを指定されたオブジェクトへ読み込み、そのインタフェースを返します。

－書式－

```
HRESULT LoadObjectFromFile(REFGUID rguidClassID, REFIID iidInterfaceID,
                           WCHAR* pwzFilePath, void** ppObject);
```

－パラメータ－

１つ目の引数(rguidClassID)は、生成するオブジェクトのクラス参照識別子です。DirectMusicSegmentオブジェクトの場合は、「CLSID_DirectMusicSegment」を指定します。

２つ目の引数(iidInterfaceID)は、取得するインタフェースの参照識別子です。IDirectMusicSegment8インタフェースを取得する場合は、「IID_IDirectMusicSegment8」を指定します。

３つ目の引数(pwzFilePath)は、読み込むファイル名をUNICODE文字列で指定します。UNICODEとは、すべての文字を２バイトで表す形式です。char配列で表される文字列は、半角文字を１バイト、全角文字を２バイトで表すマルチバイト文字列という形式です。マルチバイト文字列からUNICODE文字列への変換はMultiByteToWideChar関数で行います。なお、UNICODE文字列はWCHAR型の配列に格納します。

４つ目の引数(ppObject)は、インタフェースを受け取る変数のアドレスを指定します。

－戻り値－

成功した場合はS_OKまたはDMUS_S_PARTIALLOAD、失敗した場合はエラーコードを返します。

```
// オーディオファイルの読み込み
IDirectMusicLoader8*    pDMLoader;              // ローダオブジェクト(初期化済みとする)
IDirectMusicSegment8*   pDMSegment;             // データを管理するセグメントオブジェクト
char                    filne_name[MAX_PATH]; // オーディオファイル名

// ファイル名をUNICODEへ変換
WCHAR   wzFileName[MAX_PATH];
MultiByteToWideChar(CP_ACP, 0, filne_name, -1, wzFileName, MAX_PATH);

// ファイル読み込み
pDMLoader->LoadObjectFromFile(CLSID_DirectMusicSegment, IID_IDirectMusicSegment8,
                              wzFileName, (LPVOID*)&pDMSegment);
```

標準MIDIファイルの設定

読み込んだオーディオファイルが標準MIDIファイルの場合は、IDirectMusicSegment8::SetParamメソッドを呼び出し、正しく再生されるようにする必要があります。このとき、１つ目の引数に「GUID_StandardMIDIFile」を指定します。この処理は、Downloadメソッドを呼び出す前に行っておかなければなりません。

```
// 標準MIDIファイルの設定
pIDMSegment8->SetParam(GUID_StandardMIDIFile, 0xFFFFFFFF, 0, 0, NULL);
```

ダウンロード

DirectMusicSegmentオブジェクトは、DirectMusicPerformanceオブジェクトにダウンロードすることによって再生できるようになります。ダウンロードすると、音色データやウェーブデータがオーディオデータを再生する機能であるシンセサイザに読み込まれます。
ダウンロードは、IDirectMusicSegment8::Downloadメソッドで行います。引数は、ダウンロードするDirectMusicPerformanceオブジェクトです。

```
// パフォーマンスへダウンロード
IDirectMusicPerformance8*    pDMPerformance;     // パフォーマンスオブジェクト(初期化済み)
pDMSegment->Download(pDMPerformance);
```

DirectMusicSegmentオブジェクトの解放

DirectMusicSegmentオブジェクトの解放は、ほかのオブジェクトと同じようにReleaseメソッドで行います。このとき、DirectMusicSegmentオブジェクトが再生されていたり、パフォーマンスにダウンロードされていたりする場合もあるので、まずこれらを解除します。
また、DirectMusicLoaderオブジェクトがDirectMusicSegmentオブジェクトのデータをキャッシュしている場合もあり、これも解放する必要があります。解放しない場合、オブジェクトへの参照がキャッシュ内に残り、解放したはずのオブジェクトが存在し続けます。これは、メモリを消費するだけでなく、オブジェクトが特定の状態情報を保持する可能性があり、以降に読み込まれたファイルがキャッシュから取得されると、開始ポイントとループポイントが以前のオブジェクトを破棄したときと同じ状態になります。

```
// セグメントオブジェクトの解放
pDMPerformance->StopEx(pDMSegment, 0, 0);              // 再生を停止する
pDMSegment->Unload(pDMPerformance);                    // パフォーマンスからアンロードする
pDMLoader ->ReleaseObjectByUnknown(pDMSegment);        // ローダのキャッシュ参照を解放する
pDMLoader ->CollectGarbage();                          // キャッシュのクリーンアップ
pDMSegment->Release();                                 // セグメントオブジェクトを解放
```

検索フォルダの設定

オーディオファイルを読み込むフォルダは、デフォルトでは実行ファイルと同じフォルダです。これを変更するには、IDirectMusicLoader8::SetSearchDirectoryメソッドを呼び出します。このメソッドもフォルダ名の指定にUNICODE(WCHAR型配列)を使用します。

```
// 検索フォルダの設定
WCHAR   wzDir[_MAX_DIR];          // 検索フォルダ名
pDMLoader->SetSearchDirectory(GUID_DirectMusicAllTypes, wzDir, FALSE);
```

課　題

CDXAudio8クラスに、オーディオファイルを読み込む機能を追加しましょう。

(1)セグメントをカプセル化したクラスを作成します。

DirectMusicSegmentオブジェクトは、IDirectMusicSegment8インタフェースをとおして操作を行います。このインタフェースは、サウンドデバイスが無い環境やセグメントオブジェクトの生成に失敗した場合、NULLを保持します。ゲームプログラムでは、リズムアクションのような特別なジャンルを除けば、音が無くても支障はありません。つまり、セグメントオブジェクトの生成に失敗しても、プログラムをそのまま実行し続けて構わないということになります。

この場合、IDirectMusicSegment8インタフェースがNULLの場合とそうでない場合が考えられます。NULLは具体的なインスタンスを参照しておらず、そもそも読むことも書くこともできない領域を指しているので、NULLを介してメソッドを呼び出すと、Windowsは例外を発生させてプログラムを強制終了します。例外が発生しないようにするためには、そのインタフェースがNULLかどうかを調べなければなりません。

```
    // pDMSegmentは、IDirectMusicSegment8*型の変数(DirectMusicSegmentのインタフェース)
    if(pDMSegment != NULL) {
        // pDMSegmentを使った処理(再生や停止など)
    }
```

しかし、プログラムによってはこのようなチェックが非常に多くなってしまい、コードが煩雑になってしまう危険性があります。そこで、デザインパターンのNull Object(Null Device)パターンを適用します。

Null Objectパターンは、インタフェースやポインタがインスタンスを参照していないことを表現するのに、NULLを用いるのではなく、明示的に「何もしないオブジェクト」(Null Object)を使うようにするのです。そのNull Objectは、呼び出される可能性のあるメソッドはすべて持ちますが、すべて「何もしない」処理として実装します。このようなNull Objectを導入するすると、NULLチェックを省くことができ、簡潔なコードになります。

Null Objectパターンを適用したCMusicSegmentクラスのヘッダファイル(MusicSegment.hpp)を以下のように作成しましょう。

―MusicSegment.hpp―

```
/*
================================================================================================
                                オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
================================================================================================

  【対象ＯＳ】
    Microsoft Windows2000/XP

  【コンパイラ】
    Microsoft Visual C++ 2005

  【プログラム】
    MusicSegment.hpp
                ミュージックセグメントクラスヘッダ

  【履歴】
    * Version    1.00       2005/03/dd  hh:mm:ss

================================================================================================
*/

#pragma once

/*********************************************************************************************/
/*                                      インクルードファイル                                 */
/*********************************************************************************************/
#include <dmusici.h>

/*********************************************************************************************/
/*                                ミュージックセグメントインタフェース定義                   */
/*********************************************************************************************/
class IMusicSegment {
public:
    virtual ~IMusicSegment() {}
};

/*********************************************************************************************/
/*                                   ミュージックセグメントクラス定義                        */
/*********************************************************************************************/
class CMusicSegment : public IMusicSegment {
public:
    CMusicSegment(IDirectMusicLoader8* pDMLoader, IDirectMusicPerformance8* pDMPerformance,
                  IDirectMusicSegment8* pDMSegment);
    virtual ~CMusicSegment();

private:
    IDirectMusicLoader8*        m_pDMLoader;
    IDirectMusicPerformance8*   m_pDMPerformance;
    IDirectMusicSegment8*       m_pDMSegment;
};

/*********************************************************************************************/
```

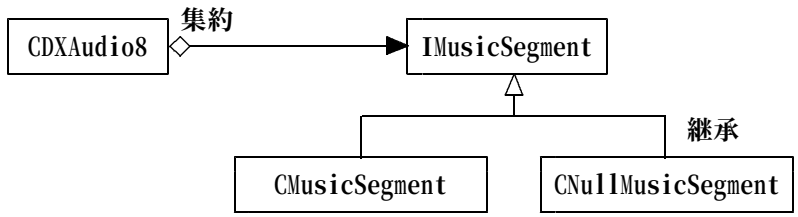
```
/*                                NULLミュージックセグメントクラス定義                                */
/*****************************************************************************************************/
class CNullMusicSegment : public IMusicSegment {
public:
    CNullMusicSegment() {}
    virtual ~CNullMusicSegment() {}
};
```

IMusicSegmentクラスは、CMusicSegmentクラスとCNullMusicSegmentクラスのインタフェースとして使用したいので、それらのクラスの基底クラスにし、publicな関数は基本的に純粋仮想関数として宣言します。CDXAudio8クラスがDirectMusicSegmentオブジェクトを生成しようとしたとき、その成否によってCMusicSegmentオブジェクトかCNullMusicSegmentオブジェクトを生成(するように設計)します。アップキャストにより、IMusicSegmentへのポインタには、そのどちらも代入することができます。IMusicSegmentポインタをとおせば、そのインスタンスがCMusicSegmentオブジェクトでもCNullMusicSegmentオブジェクトでも、それを意識することなく操作することができるというわけです。

CMusicSegmentクラスがDirectMusicSegmentオブジェクトをカプセル化したクラスになります。このクラスのメンバ変数は、IDirectMusicSegment8インタフェースだけで十分だと考えられますが、再生と停止を行うのにIDirectMusicPerformance8インタフェースが必要となります。また、DirectMusicSegmentオブジェクトを解放するのにIDirectMusicLoader8インタフェースが必要となるので、これらもメンバ変数として加えます。

CNullMusicSegmentクラスがNull Object役になります。今後、CNullMusicSegmentクラスには、IMusicSegmentクラスのpublicな関数をすべて「何もしない処理」として実装します。

最終的なクラス相関図は、以下のようになります。



(２)セグメントクラスのソースファイル(MusicSegment.cpp)を以下のように作成しましょう。

―MusicSegment.cpp―

```
/*
=====================================================================================================
                                オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
=====================================================================================================

  【対象ＯＳ】
    Microsoft Windows2000/XP

  【コンパイラ】
    Microsoft Visual C++ 2005

  【プログラム】
    MusicSegment.cpp
                ミュージックセグメントクラス

  【履歴】
    * Version    1.00       2005/03/dd  hh:mm:ss

=====================================================================================================
*/

/*****************************************************************************************************/
/*                                        インクルードファイル                                       */
/*****************************************************************************************************/
#include "MusicSegment.hpp"
#include <cassert>

/*****************************************************************************************************/
```

```
/*                                          コンストラクタ                                                    */
/*****************************************************************************************************/
CMusicSegment::CMusicSegment(IDirectMusicLoader8* pDMLoader, IDirectMusicPerformance8* pDMPerformance,
                             IDirectMusicSegment8* pDMSegment)
{
    assert(pDMLoader != NULL && pDMPerformance != NULL && pDMSegment != NULL);

    m_pDMLoader      = pDMLoader;       m_pDMLoader     ->AddRef();     // 参照カウンタインクリメント
    m_pDMPerformance = pDMPerformance;  m_pDMPerformance->AddRef();
    m_pDMSegment     = pDMSegment;      m_pDMSegment    ->AddRef();
}

/*****************************************************************************************************/
/*                                          デストラクタ                                              */
/*****************************************************************************************************/
CMusicSegment::~CMusicSegment()
{
    m_pDMPerformance->StopEx(m_pDMSegment, 0, 0);       // 停止
    m_pDMSegment->Unload(m_pDMPerformance);             // アンロード

    m_pDMLoader->ReleaseObjectByUnknown(m_pDMSegment);  // キャッシュ削除
    m_pDMLoader->CollectGarbage();                      // クリーンアップ

    m_pDMSegment    ->Release();                        // セグメント解放
    m_pDMPerformance->Release();                        // パフォーマンス参照解除
    m_pDMLoader     ->Release();                        // ローダ参照解除
}
```

CMusicSegmentクラスのコンストラクタでは、渡されたインタフェースのAddRefメソッドを呼び出し、参照カウンタを増やしています。COMのインタフェースを別の変数にコピーした場合、その期間が一時的でない場合は、このようにした方が安全です。このプログラムの場合は、CMusicSegmentオブジェクトが存在している間は、各インタフェースが参照しているオブジェクトが解放されなくなるからです。もちろん、このままでは参照カウンタが増えたままになり解放されなくなるので、CMusicSegmentオブジェクトがデストラクトされるときに、Releaseメソッドで参照カウンタを減らします。

(３)セグメントオブジェクトの解放漏れを防ぐため、生成されたすべてのセグメントオブジェクトは、CDXAudio8クラスが管理するようにします。
以下のメンバをCDXAudio8クラスに追加しましょう。

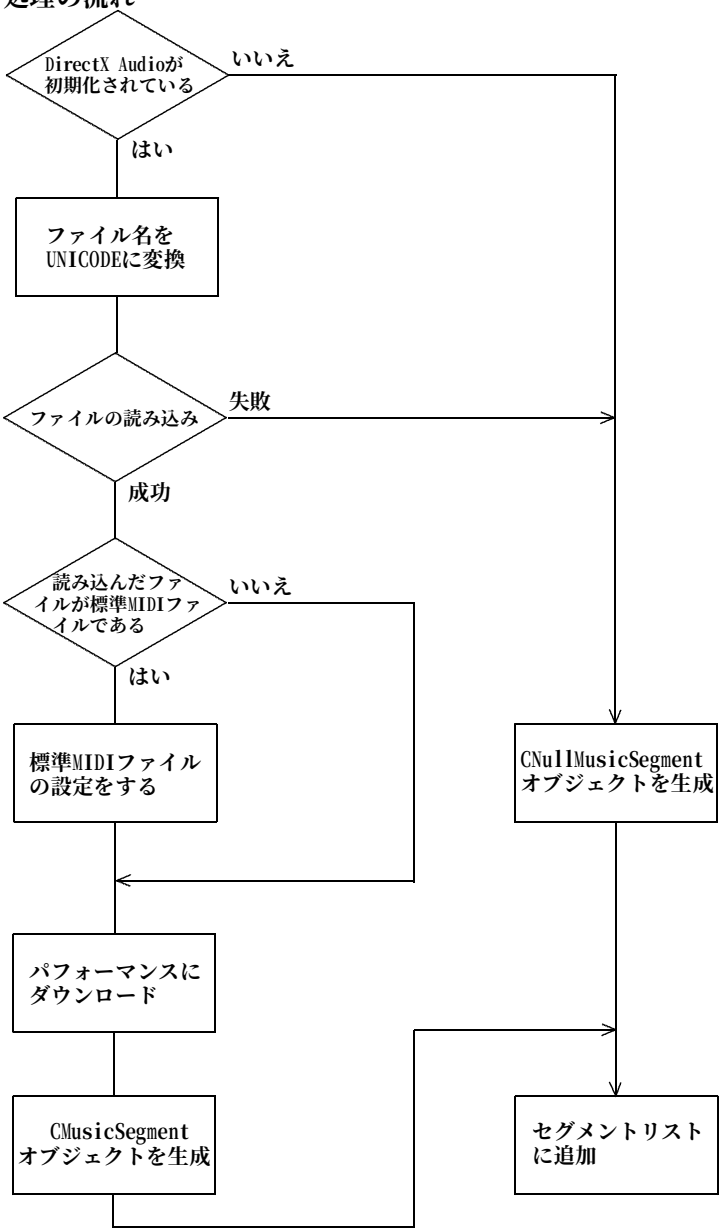
```
    std::list<IMusicSegment*>   m_SegmentList;      // セグメントリスト
```

(４)CDXAudio8クラスの適切な場所に、セグメントオブジェクトを生成し、そのインタフェースを返す
以下のCreateSegmentFromFile関数の足りない部分を補い、追加しましょう。

```
/*****************************************************************************************************/
/*                                          セグメント生成                                            */
/*****************************************************************************************************/
IMusicSegment* CDXAudio8::CreateSegmentFromFile(LPCSTR inFileName)
{
    // MusicSegmentオブジェクト生成
    IMusicSegment*          pSegment   = NULL;
    IDirectMusicSegment8*   pDMSegment = NULL;
    try {
        if(m_pDMLoader == NULL)
            throw "*** Error - DirectX Audio未初期化(CDXAudio8_CreateSegmentFromFile)\n";

        // DirectMusicSegmentオブジェクト生成
        // ファイル名をUNICODEへ変換
        WCHAR   wzFileName[MAX_PATH];
        ::??????????????????????(CP_ACP, 0, inFileName, -1, wzFileName, MAX_PATH);

        // ファイル読み込み
        if(FAILED(m_pDMLoader->????????????????????(CLSID_DirectMusicSegment, IID_IDirectMusicSegment8,
                                                    wzFileName, (LPVOID*)&pDMSegment)))
            throw "*** Error - ファイル読み込み失敗(CDXAudio8_CreateSegmentFromFile)\n";

        // 標準MIDIファイル設定
        MUSIC_TIME   mt;
        pDMSegment->GetLength(&mt);
```

```
        if(mt != 1)
            pDMSegment->SetParam(GUID_StandardMIDIFile, 0xFFFFFFFF, 0, 0, NULL);

        // パフォーマンスにダウンロード
        if(pDMSegment->????????(m_pDMPerformance) != S_OK)
            throw "*** Error - ダウンロード失敗(CDXAudio8_CreateSegmentFromFile)\n";

        pSegment = new CMusicSegment(m_pDMLoader, m_pDMPerformance, pDMSegment);
    } catch(LPCTSTR ErrorString) {
        ::OutputDebugString(ErrorString);
        // 例外が発生した場合は、NULLオブジェクトを生成
        pSegment = new ????????????????();
    }
    SafeRelease(pDMSegment);

    // セグメントリストへ追加
    m_SegmentList.?????????(pSegment);

    return pSegment;
}
```

－処理の流れ－



CDXAudio8::CreateSegmentFromFile関数の仕様は、以下のとおりです。

CreateSegmentFromFile　生成

指定されたオーディオファイルを読み込み、それを管理するセグメントオブジェクトを生成します。

試IMusicSegment* CreateSegmentFromFile(LPCSTR inFileName);

Return	生成されたセグメントオブジェクトへのポインタ
inFileName	オーディオファイルの名前

※標準MIDIファイルの判定について

CDXAudio8::CreateSegmentFromFile関数では、標準MIDIファイルとWaveファイルを読み込むことができます。読み込んだファイルが標準MIDIファイルの場合は、SetParamメソッドで標準MIDIファイルの設定を行う必要があります。しかし、DirectX Audioには、読み込んだファイルが標準MIDIファイルであるかどうかを調べるメソッドがありません。そこで、Waveファイルを読み込んだセグメントは、セグメントのデータ長を取得するGetLengthメソッドが必ず１を返すという性質を利用して判定しています。

(５)以下のプログラムは、CDXAudio8::CreateSegmentFromFile関数が生成したセグメントオブジェクトを解放するReleaseSegment関数です。足りない部分を補い、適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                          セグメント解放                                           */
/*****************************************************************************************************/
void CDXAudio8::ReleaseSegment(IMusicSegment*& pSegment)
{
    m_SegmentList.??????(pSegment);      // セグメントリストから削除
    delete pSegment;                     // セグメントオブジェクト解放
    pSegment = NULL;
}
```

この関数にCDXAudio8::CreateSegmentFromFile関数が返したセグメントオブジェクトへのポインタを渡すと、セグメントリストから検索し、削除されます。

(６)以下のプログラムは、CDXAudio8クラスが管理しているすべてのセグメントオブジェクトを解放するReleaseAllSegments関数です。足りない部分を補い、適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                          全セグメント解放                                         */
/*****************************************************************************************************/
void CDXAudio8::ReleaseAllSegments()
{
    for(std::list<IMusicSegment*>::iterator it = m_SegmentList.?????(); it != m_SegmentList.???(); it++)
        delete *it;
    m_SegmentList.clear();
}
```

(７)CDXAudio8クラスの解放時に、すべてのセグメントが解放されるように、以下のプログラムを適切な場所に追加しましょう。

```
    // 全セグメント解放
    ReleaseAllSegments();
```

応用問題　セグメントオブジェクトにニックネームを付けられるようにし、ニックネームからセグメントオブジェクトを検索できる機能を追加しましょう。