ゲームプログラミング

▶　DirectX Audio－第３回　オーディオファイルの読み込み　◀

DirectX Auidoは、オーディオファイルを読み込むためのメソッドが提供されているので、簡単に読み込むことができます。

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は、ローダオブジェクトのLoadObjectFromFileメソッドで行います。このメソッドは、オーディオファイルを読み込み、それを管理するセグメントオブジェクトを生成します。このメソッドは、Waveファイル(.wav)やMIDIファイル(.mid)といった形式ファイルを読み込むことができます。Waveファイルは、圧縮された形式(ADPCMやMP3)も読み込むことができます。

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形式のオブジェクトへ読み込み、そのインタフェースを取得します。

－パラメータ－
１つ目の引数は、取得するオブジェクトの形式の指定です。DirectMusicSegmentを取得する場合は、"CLSID_DirectMusicSegment"を指定します。

２つ目の引数は、取得するオブジェクトのインタフェース参照識別子です。DirectMusicSegment8型のオブジェクトのインタフェースを取得する場合は、"IID_IDirectMusicSegment8"を指定します。

３つ目の引数は、読み込むファイル名をUNICODE文字列で指定します。UNICODEは、すべての文字を２バイトで表す形式です。Ｃ言語で使用しているchar配列で表される文字列は、半角文字は１バイト、全角文字は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APIの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つ目の引数は、生成されるオブジェクトのインタフェースを受け取る変数のアドレスをLPVOID*型にキャストして指定します。

－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の原因をエラーコードで返します。

```
IDirectMusicLoader8*    pIDMLoader8;              // ローダオブジェクト(初期化済みとする)
IDirectMusicSegment8*   pIDMSegment8 = NULL;      // データを読み込むセグメントオブジェクト
char                    szFileName[MAX_PATH];     // オーディオファイル名

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

// ファイル読み込み
pIDMLoader8->LoadObjectFromFile(CLSID_DirectMusicSegment, IID_IDirectMusicSegment8,
                                wszFileName, (LPVOID*)&pIDMSegment8);
```

標準MIDIデータの設定

読み込んだオーディオデータが標準MIDIデータ(.MID)の場合は、セグメントオブジェクトのSetParamメソッドを呼び出し、１つ目の引数に"GUID_StandardMIDIFile"を指定する必要があります。こうすることにより、標準MIDIファイルを正しく再生できるようになります。この処理は、ダウンロードの前に行う必要があります。

```
IDirectMusicSegment8*   pIDMSegment8;        // 標準MIDIデータが読み込まれたセグメントオブジェクト
pIDMSegment8->SetParam(GUID_StandardMIDIFile, 0xFFFFFFFF, 0, 0, NULL);
```

ダウンロード

セグメントオブジェクトは、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シンセサイザ(オーディオデータを演奏するプログラム)に読み込まれます。
ダウンロードはセグメントオブジェクトのDownloadメソッドで行います。引数は、対象となるパフォーマンスオブジェクトです。

```
IDirectMusicPerformance8*   pIDMPerformance8;     // ダウンロード対象のパフォーマンスオブジェクト
IDirectMusicSegment8*       pIDMSegment8;         // セグメントオブジェクト

pIDMSegment8->Download(pIDMPerformance8);
```

セグメントオブジェクトの解放

セグメント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、セグメントオブジェクトは再生中であったり、パフォーマンスオブジェクトにダウンロードされていたりする場合があるので、まず最初にこれらを解除します。また、ローダオブジェクトがセグメントオブジェクトをキャッシュしている場合があるので、これも解放する必要があります。

```
// pIDMLoader8はローダオブジェクト、pIDMPerformance8はパフォーマンスオブジェクト
// pIDMSegment8は解放するセグメントオブジェクト
pIDMPerformance->StopEx(pIDMSegment8, 0, 0);          // 再生を停止する
pIDMSegment8->Unload(pIDMPerformance8);               // パフォーマンスからアンロードする
pIDMLoader8 ->ReleaseObjectByUnknown(pIDMSegment8);   // ローダのキャッシュを解放する
pIDMSegment8->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(.exe)と同じフォルダです。これを変更するには、ローダオブジェクトのSetSearchDirectoryメソッドで行います。このメソッドもフォルダ名の指定はUNICODE(WCHAR型)で行います。

```
IDirectMusicLoader8*   pIDMLoader8;              // ローダオブジェクト(初期化済みとする)
WCHAR                  wszDir[MAX_PATH + 1];     // 検索フォルダ名
pIDMLoader8->SetSearchDirectory(GUID_DirectMusicAllTypes, wszDir, FALSE);
```

○ 課題 ○

オーディオファイルをセグメントに読み込む関数を作成しましょう。

(１)DXAUtilsでは、セグメントオブジェクトのインタフェースを配列にまとめて扱います。配列で管理するのは、セグメントごとに変数を与える方法、たとえば以下のように

```
IDirectMusicSegment8*   pIDMSegment1;        // セグメントオブジェクト1
IDirectMusicSegment8*   pIDMSegment2;        // セグメントオブジェクト2
IDirectMusicSegment8*   pIDMSegment3;        // セグメントオブジェクト3
IDirectMusicSegment8*   pIDMSegment4;        // セグメントオブジェクト4
```

とするよりも利点が多いためです。たとえば、すべてのセグメント対して同じ操作を行いたい場合、配列ならループを回すだけで処理することができます。また、DXAUtils.hにセグメント指定用の列挙体を宣言します。これは、セグメントオブジェクトの配列を指定するときの添字として使用します。

－追加１－

```
// セグメント指定用列挙体
enum DXA_SEG {
    DXA_SEG1,          // セグメント１
    DXA_SEG2,          // セグメント２
    DXA_SEG3,          // セグメント３
    DXA_SEG4,          // セグメント４
    DXA_SEG5,          // セグメント５
    DXA_SEG6,          // セグメント６
    DXA_SEG7,          // セグメント７
    DXA_SEG8,          // セグメント８
    DXA_SEG_MAX        // セグメント最大数
};
```

上記の定義に過不足がある場合は、各自調整してください。次に、DXAUtils.cppのグローバル変数として、セグメントオブジェクトのインタフェースを配列で定義します。

－追加２－
```
static IDirectMusicSegment8*   g_pIDMSeg8[DXA_SEG_MAX] = {NULL};    // DirectMusicセグメントオブジェクト
```

このように定義した場合、たとえばセグメント１(添字0)を指定するには

```
g_pIDMSeg8[DXA_SEG1]
```

と記述します(g_pIDMSeg8[0]と記述することもできます)。

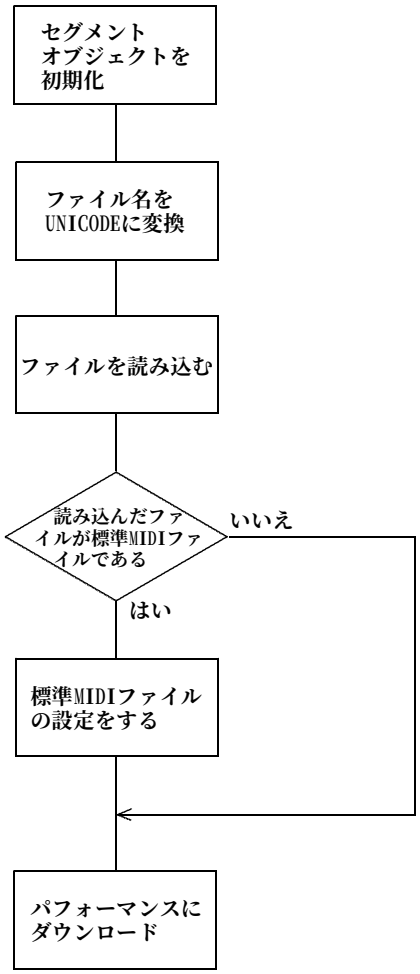
(２)以下のプログラムは、オーディオファイルをセグメントに読み込むDXALoadFromFile関数です。関数の仕様とフローチャートをよく読み、プログラムを完成させましょう。なお、この関数は(３)で作成する関数も必要です。

DXACreateSegmentFromFile関数
－説明－
DXACreateSegmentFromFile関数は、セグメントを生成し、オーディオデータを読み込みます。

－パラメータ－
const DXA_SEG dxaSeg…セグメントの指定。DXA_SEG列挙体を指定
LPCSTR lpszFileName…読み込むオーディオファイル名(.midまたは.wav)

－戻り値－
成功した場合はtrue，それ以外はfalseを返します。

－フローチャート－



－追加３－

```
/*****************************************************************************************************/
/*                                          セグメント生成                                           */
/*****************************************************************************************************/
bool DXACreateSegmentFromFile(const DXA_SEG dxaSeg, LPCSTR lpszFileName)
{
    if(NULL == g_pIDMLoader8 || NULL == g_pIDMPerformance8) {
        OutputDebugString("*** Error - ローダ，パフォーマンス未初期化(DXACreateSegmentFromFile)\n");
        return false;
    }

    DXAReleaseSegment(dxaSeg);  // セグメント初期化

    // ファイル名をUNICODEへ変換
    WCHAR   wszFileName[MAX_PATH];
    MultiByteToWideChar(CP_ACP, 0, lpszFileName, -1, wszFileName, MAX_PATH);
    // ファイル読み込み
    if(FAILED(※ここは各自考えましょう)) {
        OutputDebugString("*** Error - ファイル読み込み失敗(DXACreateSegmentFromFile)\n");
        return false;
    }

    // 標準MIDIデータ設定
    MUSIC_TIME   mt;
    g_pIDMSeg8[dxaSeg]->GetLength(&mt);
    if(1 != mt)
        g_pIDMSeg8[dxaSeg]->※ここは各自考えましょう;

    // パフォーマンスへダウンロード
    if(S_OK != ※ここは各自考えましょう) {
        OutputDebugString("*** Error - ダウンロード失敗(DXACreateSegmentFromFile)\n");
        DXAReleaseSegment(dxaSeg);
        return false;
    }

    return true;
}
```

※標準MIDIファイルの判定について
　DXACreateSegmentFromFile関数では、標準MIDIファイルとWaveファイルなどのオーディオ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。DXACreateSegmentFromFile関数では、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。

(３)セグメントを解放するDXAReleaseSegment関数の仕様を参考に、関数を完成させましょう。

DXAReleaseSegment関数
－説明－
　DXAReleaseSegment関数は、指定されたセグメントを解放します。

－パラメータ－
　const DXA_SEG dxaSeg…解放するセグメントの指定。DXA_SEG列挙体型を指定

－戻り値－
　なし

－追加４－

```
/*****************************************************************************************************/
/*                                          セグメント解放                                           */
/*****************************************************************************************************/
void DXAReleaseSegment(const DXA_SEG dxaSeg)
{
    if(NULL == g_pIDMSeg8[dxaSeg])
        return;

    // パフォーマンス参照解除
    if(NULL != g_pIDMPerformance8) {
```

```
        g_pIDMPerformance8->StopEx(g_pIDMSeg8[dxaSeg], 0, 0);    // 再生停止
        g_pIDMSeg8[dxaSeg]->※ここは各自考えましょう;              // アンロード
    }

    // ローダ参照(キャッシュ)解除
    if(NULL != g_pIDMLoader8) {
        g_pIDMLoader8->ReleaseObjectByUnknown(g_pIDMSeg8[dxaSeg]);
        g_pIDMLoader8->CollectGarbage();
    }

    g_pIDMSeg8[dxaSeg]->※ここは各自考えましょう;                  // 解放
    g_pIDMSeg8[dxaSeg] = NULL;
}
```

(４)－追加５－のたりない部分を補って適切な場所に追加し、DirectX Audio解放時にすべてのセグメントが正しく解放されるようにしましょう。

－追加５－

```
    // セグメント解放
    for(int dxaSeg = DXA_SEG1; dxaSeg < DXA_SEG_MAX; dxaSeg++)
        ????????????????((DXA_SEG)dxaSeg);
```

(５)以下のプログラムは、オーディオファイルを読み込むディレクトリを設定するDXASetSearchDirectory関数です。関数の仕様をよく読み、プログラムを完成させましょう。

DXASetSearchDirectory関数

－説明－

DXASetSearchDirectory関数は、ローダによって読み込まれるオーディオファイルを検索するディレクトリを設定します。

－パラメータ－

LPCSTR lpszDir…設定する検索ディレクトリ名

－戻り値－

関数が成功した場合はtrue，それ以外はfalseを返します。

－追加６－

```
/*****************************************************************************************************/
/*                                          検索ディレクトリ設定                                       */
/*****************************************************************************************************/
bool DXASetSearchDirectory(LPCSTR lpszDir)
{
    if(NULL == g_pIDMLoader8) {
        OutputDebugString("*** Error - ローダ未初期化(DXASetSearchDirectory)\n");
        return false;
    }

    // ディレクトリ名をUNICODEへ変換
    WCHAR   wszDir[MAX_PATH];
    ????????????????????(CP_ACP, 0, lpszDir, -1, wszDir, MAX_PATH);

    // 検索ディレクトリ設定
    if(S_OK != g_pIDMLoader8->????????????????????(GUID_DirectMusicAllTypes, wszDir, FALSE)) {
        OutputDebugString("*** Error - 検索ディレクトリ設定失敗(DXASetSearchDirectory)\n");
        return false;
    }

    return true;
}
```