ゲームプログラミング

▶　DirectX Audio－第４回　セグメントの再生と停止　◀

セグメントに読み込まれているオーディオデータの再生や停止は、簡単に行うことができます。

セグメントの再生

セグメントに読み込まれているオーディオデータの再生は、パフォーマンスオブジェクトのPlaySegmentExメソッドで行います。

－説明－
PlaySegmentExメソッドは、セグメントの再生を開始します。

－パラメータ－
１つ目の引数は、再生を開始するセグメントオブジェクトのインタフェースです。

２つ目の引数は、実装されていないので必ずNULLを指定します。

３つ目の引数は、テンプレートセグメントを指定します。通常は使用しないのでNULLにします。

４つ目の引数は、動作フラグです。おもに以下のフラグを指定します。

０	プライマリセグメントで再生。再生優先度がもっとも高いので、演奏時間が長いデータは必ずこのモードを使用。ただし、プライマリとして指定できるのは、１つのパフォーマンスにつき１つのセグメントのみ。
DMUS_SEGF_SECONDARY	セカンダリセグメントで再生。複数のセグメントを同時に再生可能。ただし、演奏時間が長いデータは不具合が起こる場合がある。
DMUS_SEGF_QUEUE	現在再生されているプライマリセグメントのデータが終了次第、即座に演奏を開始。

通常は、ＢＧＭなどの大きなデータはプライマリ、効果音などの小さなデータはセカンダリで演奏します(データによっては、同時に演奏できない場合があります)。

５つ目の引数は、再生が開始されるまでの時間です。０を指定するとできる限り早く開始されます。

６つ目の引数は、セグメント状態を取得。使用しない場合はNULLを指定します。

７つ目の引数は、再生開始時に再生を停止するセグメントオブジェクトまたはオーディオパスオブジェクトのインタフェースを指定します。使用しない場合はNULLを指定します。

８つ目の引数は、再生に使うオーディオパスオブジェクトのインターフェイスを指定します。デフォルトオーディオパスで再生する場合はNULLを指定します。

－戻り値－
成功した場合はS_OK、それ以外はエラーの原因をエラーコードで返します。

```
IDirectMusicPerformance8*   pIDMPerformance8;     // パフォーマンスオブジェクト(初期化済みとする)
IDirectMusicSegment8*       pIDMSegment8;         // 再生を開始するセグメントオブジェクト
pIDMPerformance8->PlaySegmentEx(pIDMSegment8, NULL, NULL, DMUS_SEGF_SECONDARY, 0, NULL,
                                NULL, NULL);      // セカンダリセグメントとして再生
```

セグメントごとに再生のループ回数を設定することが出ます。設定は、セグメントオブジェクトのSetRepeatsメソッドで行います。引数はループ回数で、DMUS_SEG_REPEAT_INFINITEで無限ループ、０はループなし、１以上はその回数だけループします。指定回数ループすると再生は停止します。

```
pIDMSegment8->SetRepeats(DMUS_SEG_REPEAT_INFINITE);   // 無限ループ
```

セグメントの停止

セグメントの再生停止は、パフォーマンスオブジェクトのStopExメソッドで行います。

StopExメソッド
－説明－
StopExメソッドは、セグメントの再生を停止します。

－パラメータ－
　１つ目の引数は、再生を停止するセグメントオブジェクトのインタフェースです。NULLを指定するとすべてのセグメントの再生が停止されます。

　２つ目の引数は、停止するまでの時間です。０を指定すると即座に停止されます。

　３つ目の引数は、停止フラグです。通常は０を指定します。

－戻り値－
　成功した場合はS_OK、失敗した場合はE_POINTERを返します。

```
IDirectMusicPerformance8*   pIDMPerformance8;     // パフォーマンスオブジェクト(初期化済みとする)
IDirectMusicSegment8*       pIDMSegment8;         // 再生を停止するセグメントオブジェクト
pIDMPerformance8->StopEx(pIDMSegment8, 0, 0);
```

ボリュームの設定

　パフォーマンス全体のボリュームの設定は、パフォーマンスオブジェクトのSetGlobalParamメソッドで行います。１つ目の引数に"GUID_PerfMasterVolume"、２つ目の引数にボリューム値を格納した変数のアドレス、３つ目の引数に２つ目の引数で指定した変数のサイズ(バイト数)を指定します。ボリューム値は+2000～-20000の範囲で指定でき、０が元のボリューム値です。しかし、実際に使用できる範囲は+1000～-10000です。このメソッドではセグメントごとのボリュームを設定することはできません。

```
IDirectMusicPerformance8*   pIDMPerformance8;     // パフォーマンスオブジェクト(初期化済みとする)
int    nVolume = -1000;                           // ボリュームを下げる
pIDMPerformance8->SetGlobalParam(GUID_PerfMasterVolume, (LPVOID)&nVolume, sizeof(nVolume));
```

○ 課題 ○

(１)以下のプログラムは、セグメントの再生を開始するDXAPlaySegment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。

DXAPlaySegment関数
－説明－
　DXAPlaySegment関数は、指定されたセグメントの再生を開始します。

－パラメータ－
　const DXA_SEG dxaSeg…再生を開始するセグメントの指定。DXA_SEG列挙体を指定
　const DWORD dwSegFlag…セグメントの動作フラグ。主に以下のフラグを使用
　　０　　プライマリセグメントで再生。
　　DMUS_SEGF_SECONDARY　　セカンダリセグメントで再生。
　　DMUS_SEGF_QUEUE　　現在のプライマリセグメントのデータが終了次第、即座に再生を開始
　const int nLoop…ループ回数。以下のように指定
　　１以上　　指定回数ループする
　　０　　ループしない(１回だけ演奏する)
　　－１以下　　StopExメソッドが実行されるまで再生を続ける(無限ループ)

－戻り値－
　再生に成功した場合はtrue，それ以外はfalseを返します。

```
/*****************************************************************************************************/
/*                                          セグメント再生                                           */
/*****************************************************************************************************/
bool DXAPlaySegment(const DXA_SEG dxaSeg, const DWORD dwSegFlag, const int nLoop)
{
    if(NULL == g_pIDMPerformance8 || NULL == g_pIDMSeg8[dxaSeg]) {
        OutputDebugString("*** Error - パフォーマンス，セグメント未初期化(DXAPlaySegment)\n");
        return false;
    }

    g_pIDMPerformance8->StopEx(g_pIDMSeg8[dxaSeg], 0, 0);

    // ループ設定
    if(0 <= nLoop)
        ※ここは各自考えましょう
```

```
    else
        ※ここは各自考えましょう

    // 再生
    if(S_OK != ※ここは各自考えましょう(ヒント：dwSegFlagの使い方がポイント)) {
        OutputDebugString("*** Error - 再生失敗(DXAPlaySegment)\n");
        return false;
    }

    return true;
}
```

(２)以下のプログラムは、セグメントの再生を停止するDXAStop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。

DXAStopSegment関数
－説明－
DXAStopSegment関数は、指定されたセグメントの再生を停止します。

－パラメータ－
const DXA_SEG dxaSeg…再生を停止するセグメントの指定。DXA_SEG列挙体で指定

－戻り値－
なし

```
/*****************************************************************************************************/
/*                                          セグメント停止                                           */
/*****************************************************************************************************/
void DXAStopSegment(const DXA_SEG dxaSeg)
{

    if(NULL == g_pIDMPerformance8 || NULL == g_pIDMSeg8[dxaSeg]) {
        OutputDebugString("*** Error - パフォーマンス，セグメント未初期化(DXAStopSegment)\n");
        return;
    }

    ※ここは各自考えましょう(ヒント：セグメントの再生を停止します)
}
```

(３)以下のプログラムは、セグメント全体のボリューム(マスターボリューム)を設定するDXASetMaster Volume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。

```
/*****************************************************************************************************/
/*                                       マスターボリューム設定                                      */
/*****************************************************************************************************/
bool DXASetMasterVolume(int nVolume)
{
    if(NULL == g_pIDMPerformance8) {
        OutputDebugString("*** Error - パフォーマンス未初期化(DXASetMasterVolume)\n");
        return false;
    }

    // マスターボリューム設定
    if(S_OK != g_pIDMPerformance8->??????????????(??????????????????????,
                                                  (LPVOID)&nVolume, sizeof(nVolume))) {
        OutputDebugString("*** Error - マスターボリューム設定失敗(DXASetMasterVolume)\n");
        return false;
    }

    return true;
}
```

(４)以下のプログラムをTest関数に追加し、オーディオファイルが正しく読み込まれ、再生されるかを検証しましょう。

```
    DXACreateSegmentFromFile(DXA_SEG1, "MIDIまたはWaveファイル名");
    DXAPlaySegment(DXA_SEG1, 0, -1);
```