オブジェクト指向と
ゲームプログラミング

▶ DirectX Audio編 － 第６回　サウンドバッファ ◀

サウンドバッファ

サウンドバッファとは、DirectSoundが扱うサウンドのイメージを保持しておくメモリ領域です。DirectSoundでは、Waveファイル(.wav)を直接再生する機能も、読み込む機能も提供していません。DirectSoundでサウンドを扱うには、サウンドバッファを生成し、そこにサウンドのイメージを転送しておかなければなりません。
サウンドバッファには、プライマリバッファとセカンダリバッファの２種類あります。

プライマリバッファは、DirectSoundの初期化時に自動的に生成されるサウンドバッファです。このバッファは、OSおよびドライバの仕様により動作が異なります。
Windows98SE/Me/2000/XPでは、ドライバがWDM(Windows Driver Model)という規格で作成されています。WDMでは、プライマリバッファは実際には存在せず、すべての入力はOSが提供するミキサ(Kmixer)に直接出力されます。Kmixerは、現在再生されているすべてのサウンドの中で、もっともサンプリング周波数およびビットレートの高いもの(ただし、ハードウェアの上限を超えることはできない)にあわせて変換し、ミキシングして出力します。つまり、WDMドライバが使用されている時には、プライマリバッファのフォーマットを設定しても、意味をなさないということになります。
一方、Windows95/98などの古いOSでは、ドライバがMME(MultiMedia Extensions)という規格で作成されています。MMEでは、Kmixerがないので、DirectSound内部のミキサが動作します。これが本来のプライマリバッファです。この場合、DirectSoundで再生したすべてのサウンドは、プライマリバッファのフォーマットに合わせて変換され、ミキシングして出力されます。デフォルトのフォーマットは、サンプリング周波数22.05kHz，ビットレート8bit(CDの半分。FMラジオ程度の音質)ですが、協調レベルが標準レベル以外の場合は、変更することができます。
プライマリバッファは、通常DirectSoundが管理し、直接ふれることはありませんが、DirectSound全体の音量を変更したり、3Dサウンドを制御する場合には必要となります。また、プライマリバッファに直接データを書き込むことにより、セカンダリバッファ経由では難しいサウンド効果を実現することができます。ただし、この場合はセカンダリバッファを使ったミキシング再生などは行えなくなります。

セカンダリバッファは、サウンドのイメージを格納しておく領域です。セカンダリバッファ１つにつき１つのサウンドを制御することができます。このバッファは、サウンドカード上のメモリまたはシステムメモリに作成することができ、複数生成することができます。複数のセカンダリバッファを同時に再生した場合、それぞれの内容がミキシングされてプライマリバッファに書き込まれます。このとき、プライマリバッファと異なるフォーマットでも、セカンダリバッファごとに違うフォーマットでも、正常にミキシングされます。
セカンダリバッファは、同時に再生したい数だけ生成する必要がありますが、複数のセカンダリバッファで同じメモリを共有することができるようになっています。

セカンダリバッファの生成

セカンダリバッファの生成は、DSBUFFERDESC構造体に生成するサウンドバッファの情報を設定し、IDirectSound8::CreateSoundBufferメソッドに渡すことで行います。DSBUFFERDESC構造体には、以下のメンバがあります。

メンバ	データ型	説　　明
dwSize	DWORD	この構造体のサイズ(バイト数)
dwFlags	DWORD	バッファの能力を示すフラグ
dwBufferBytes	DWORD	バッファのサイズ。最小値DSBSIZE_MIN、最大値DSBSIZE_MAX
dwReserved	DWORD	予約済み。必ず0でなければならない
lpwfxFormat	LPWAVEFORMATEX	バッファのフォーマットを指定するWAVEFORMATEX構造体のアドレス
guid3DAlgorithm	GUID	DirectSound3Dが使うアルゴリズムの識別子。３Ｄサウンドを使用しない場合は、GUID_NULLでなければならない

バッファの能力を示すフラグは、主に以下のものを組み合わせて指定します。

DSBCAPS_CTRLVOLUME	ボリュームを変更することができます
DSBCAPS_CTRLPAN	パンを変更することができます。DSBCAPS_CTRL3Dと組み合わせることはできません

DSBCAPS_CTRLFREQUENCY	再生周波数を変更することができます
DSBCAPS_CTRLFX	エフェクト処理を加えることができます
DSBCAPS_CTRL3D	３Ｄサウンドです。DSBCAPS_CTRLPANと組み合わせることはできません。モノラル以外のバッファにも設定できません
DSBCAPS_GETCURRENTPOSITION2	再生カーソル位置を正確に取得できます
DSBCAPS_LOCDEFER	ボイス管理を行うことができます
DSBCAPS_MUTE3DATMAXDISTANCE	サウンドは、最大距離で無音になります。最大距離を超えるとバッファの再生が停止するので、CPUを消費しなくなります。ソフトウェアバッファに対してのみ適用されます。

バッファのフォーマットを指定するWAVEFORMATEX構造体は、以下のメンバがあります。

メンバ	データ型	説　　明
wFormatTag	WORD	フォーマットタイプ。セカンダリバッファでは、WAVE_FORMAT_PCMにする必要がある
nChannels	WORD	チャンネル数。モノラルデータは1，ステレオデータは2
nSamplesPerSec	DWORD	サンプリング周波数(１秒間に行われるサンプリング数)
nAvgBytesPerSec	DWORD	平均データ速度(１秒間に必要なバイト数)
nBlockAlign	WORD	ブロックアラインメント。データの最小構成単位
wBitsPerSample	WORD	１サンプルあたりのビット数。セカンダリバッファでは8または16
cbSize	WORD	追加フォーマット情報のサイズ(バイト数)

IDirectSound8::CreateSoundBufferメソッド

－説明－

CreateSoundBufferメソッドは、サウンドバッファを管理するDirectSoundBufferオブジェクトを生成し、IDirectSoundBufferインタフェース(バージョン１)を返します。

－書式－

```
HRESULT CreateSoundBuffer(LPCDSBUFFERDESC pcDSBufferDesc, LPDIRECTSOUNDBUFFER* ppDSBuffer,
                          LPUNKNOWN pUnkOuter);
```

－パラメータ－

１つ目の引数(pcDSBufferDesc)は、生成するサウンドバッファの情報が格納されたDSBUFFERDESC構造体変数のアドレスです。

２つ目の引数(ppDSBuffer)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です。LPDIRECTSOUNDBUFFER型またはIDirectSoundBuffer*型の変数のアドレスを指定します。

３つ目の引数(pUnkOuter)は、将来の互換性のためにあり、現時点ではNULLにしなければなりません。

－戻り値－

成功した場合はDS_OK、要求した３Ｄアルゴリズムを利用できず、ステレオパンが代用された場合はDS_NO_VIRTUALIZATIONを返します。それ以外はエラーコードを返します。

CreateSoundBufferメソッドが返すインタフェースは、IDirectSoundBuffer8ではなくIDirectSoundBufferです。IDirectSoundBuffer8インタフェースを取得するには、IDirectSoundBufferインタフェースのQueryInterfaceメソッドを呼び出します。１つ目の引数にIDirectSoundBuffer8インタフェースの参照識別子「IID_IDirectSoundBuffer8」、２つ目の引数にインタフェースを受け取る変数を指定します。

```
// セカンダリバッファ生成(22.05kHz, 8ビット, ステレオ(２チャンネル), 4秒)
// フォーマット設定
WAVEFORMATEX   wfx;
ZeroMemory(&wfx, sizeof(wfx));
wfx.wFormatTag      = WAVE_FORMAT_PCM;
wfx.nSamplesPerSec  = 22050;
wfx.wBitsPerSample  = 8;
wfx.nChannels       = 2;
wfx.nBlockAlign     = wfx.wBitsPerSample / 8 * wfx.nChannels;
wfx.nAvgBytesPerSec = wfx.nBlockAlign * wfx.nSamplesPerSec;
wfx.cbSize          = 0;

// セカンダリバッファ能力設定
DSBUFFERDESC   dsbd;
ZeroMemory(&dsbd, sizeof(dsbd));
dsbd.dwSize          = sizeof(dsbd);
```

```
dsbd.dwFlags         = DSBCAPS_LOCDEFER | DSBCAPS_CTRLVOLUME | DSBCAPS_CTRLPAN |
                       DSBCAPS_GETCURRENTPOSITION2;
dsbd.dwBufferBytes   = wfx.nAvgBytesPerSec * 4; // ４秒
dsbd.lpwfxFormat     = &wfx;
dsbd.guid3DAlgorithm = GUID_NULL;

// セカンダリバッファ生成(pDSoundは、初期化済みのDirectSound8オブジェクト)
IDirectSoundBuffer*   pdsb;
pDSound->CreateSoundBuffer(&dsbd, &pdsb, NULL);

// IDirectSoundBuffer8インタフェース取得
IDirectSoundBuffer8*   pDSBuffer;
pdsb->QueryInterface(IID_IDirectSoundBuffer8, (LPVOID*)pDSBuffer);

pdsb->Release();        // IDirectSoundBufferインタフェースは不用なので解放する
```

プライマリバッファの取得

プライマリバッファを取得すると、DirectSound全体の音量を変更したり、プライマリバッファのフォーマットを変更することができます。プライマリバッファの取得もCreateSoundBufferメソッドで行いますが、DSBUFFERDESC構造体のdwFlagsメンバに「DSBCAPS_PRIMARYBUFFER」を指定し、dwBufferBytesメンバを０、lpwfxFormatメンバをNULLに設定する必要があります。なお、プライマリバッファはIDirectSoundBuffer8インタフェースを取得することはできません。

```
// プライマリバッファ取得
DSBUFFERDESC   dsbd;
ZeroMemory(&dsbd, sizeof(dsbd));
dsbd.dwSize          = sizeof(dsbd);
dsbd.dwFlags         = DSBCAPS_PRIMARYBUFFER | DSBCAPS_CTRLVOLUME;
dsbd.dwBufferBytes   = 0;
dsbd.lpwfxFormat     = NULL;

IDirectSoundBuffer*   pDSBPrimary;
pDSound->CreateSoundBuffer(&dsbd, &pDSBPrimary, NULL);
```

バッファの解放

プライマリおよびセカンダリバッファオブジェクトは、ほかのオブジェクトと同じようにReleaseメソッドで解放します。解放すると、保持していたイメージとメモリ領域も解放されます。

```
// セカンダリバッファ解放
pDSBuffer->Release();
```

メモリを共有するバッファ

セカンダリバッファは、同時に再生したい数だけ必要になります。同じサウンドを同時に再生するときでも、新たにセカンダリバッファを作成しなければなりません。この場合、同じサウンドでありながら、それぞれ独立したメモリ領域が割り当てられ、同じイメージを保持することになります。DirectSoundでは、複数のセカンダリバッファで同じメモリを共有することができます。メモリを共有するセカンダリバッファの生成は、IDirectSound8::DuplicateSoundBufferメソッドで行います。

IDirectSound8::DuplicateSoundBufferメソッド

－説明－

DuplicateSoundBufferメソッドは、指定されたDirectSoundBufferオブジェクトとメモリを共有する新たなDirectSoundBufferオブジェクトを生成し、IDirectSoundBufferインタフェース(バージョン１)を返します。

－書式－

```
HRESULT DuplicateSoundBuffer(LPDIRECTSOUNDBUFFER pDSBufferOriginal,
                             LPDIRECTSOUNDBUFFER* ppDSBufferDuplicate);
```

－パラメータ－
　１つ目の引数(pDSBufferOriginal)は、共有元オブジェクトのインタフェースです。IDirectSoundBufferインタフェースまたはIDirectSoundBuffer8インタフェースを指定します。

　２つ目の引数(ppDSBufferDuplicate)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です。LPDIRECTSOUNDBUFFER型またはIDirectSoundBuffer*型の変数のアドレスを指定します。

－戻り値－
　成功した場合はDS_OK、それ以外はエラーコードを返します。

－備考－
　DSBCAPS_CTRLFXフラグを使って生成されたサウンドバッファは、共有することができません。このメソッドで生成されたバッファは、共有元と同じパラメータで初期化されますが、各バッファのパラメータは個別に変更でき、互いに影響を及ぼすことなく再生と停止を行うことができます。
　共有メモリは、共有するオブジェクトがすべて破棄されたとき、解放されます。

```
// セカンダリバッファの複製(pDSBufferは、初期化済みのDirectSoundBuffer8オブジェクト)
IDirectSoundBuffer*   pdsb;
pDSound->DuplicateSoundBuffer(pDSBuffer, &pdsb);

// IDirectSoundBuffer8インタフェース取得
IDirectSoundBuffer8*   pDSBuffer2;
pdsb->QueryInterface(IID_IDirectSoundBuffer8, (LPVOID*)pDSBuffer2);

pdsb->Release();        // DirectSoundBufferバージョン１は不用なので解放する
```

課　　題

　CDXAudio8クラスに、サウンドバッファの生成と解放を行う機能を追加しましょう。

(１)サウンドバッファをカプセル化したクラスを作成します。

　DirectSoundBufferオブジェクトは、IDirectSoundBuffer8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DirectSoundBufferオブジェクトの生成に失敗した場合、NULLを保持します。インタフェースがNULLかどうかをいちいち調べると、コードが煩雑になってしまうので、CSoundBufferクラスもNull Objectパターンを適用します。

　サウンドバッファクラスのヘッダファイル(SoundBuffer.hpp)を以下のように作成しましょう。

－SoundBuffer.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】
     SoundBuffer.hpp
                サウンドバッファクラスヘッダ

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

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

#pragma once

/*************************************************************************************************/
/*                                        インクルードファイル                                   */
```

```
/*****************************************************************************************************/
#include <dsound.h>

/*****************************************************************************************************/
/*                                    サウンドバッファインタフェース定義                                 */
/*****************************************************************************************************/
class ISoundBuffer {
public:
    virtual ~ISoundBuffer() {}
};

/*****************************************************************************************************/
/*                                      サウンドバッファクラス定義                                       */
/*****************************************************************************************************/
class CSoundBuffer : public ISoundBuffer {
public:
    CSoundBuffer(IDirectSoundBuffer8* pDSBuffer);
    virtual ~CSoundBuffer() { m_pDSBuffer->Release(); }

private:
    IDirectSoundBuffer8*    m_pDSBuffer;
};

/*****************************************************************************************************/
/*                                    NULLサウンドバッファクラス定義                                     */
/*****************************************************************************************************/
class CNullSoundBuffer : public ISoundBuffer {
public:
    CNullSoundBuffer() {}
    virtual ~CNullSoundBuffer() {}
};
```

ISoundBufferクラスは、CSoundBufferクラスとCNullSoundBuffer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DirectSoundBuffer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CSoundBufferオブジェクトかCNullSoundBufferオブジェクトを生成(するように設計)します。

CSoundBufferクラスがDirectSoundBufferオブジェクトをカプセル化したクラスです。CSoundNullBufferクラスがそのNull Objectです。今後、CNullSoundBufferクラスには、ISoundBuffer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。

(２)サウンドバッファクラスのソースファイル(SoundBuffer.cpp)を以下のように作成しましょう。

－SoundBuffer.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】
     SoundBuffer.cpp
                サウンドバッファクラス

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

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

/*****************************************************************************************************/
/*                                        インクルードファイル                                         */
/*****************************************************************************************************/
#include "SoundBuffer.hpp"
```

```
#include <cassert>

/*****************************************************************************************************/
/*                                          コンストラクタ                                           */
/*****************************************************************************************************/
CSoundBuffer::CSoundBuffer(IDirectSoundBuffer8* pDSBuffer)
{
    assert(pDSBuffer != NULL);
    m_pDSBuffer = pDSBuffer;
    m_pDSBuffer->AddRef();      // 参照カウンタインクリメント
}
```

(３)サウンドバッファ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サウンドバッファオブジェクトは、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<ISoundBuffer*>   m_BufferList;     // バッファリスト
```

(４)DirectSound全体の音量や3Dサウンドの制御のために、プライマリバッファを取得しておきます。
以下のメンバをCDXAudio8クラスに追加しましょう。

```
IDirectSoundBuffer*   m_pDSBPrimary;         // プライマリバッファ
```

(５)CDXAudio8クラスのコンストラクタに、以下の初期化子を追加しましょう。

```
m_pDSBPrimary(NULL)
```

(６)CDXAudio8::Initialize関数の適切な場所に、プライマリバッファを取得する以下のプログラムの足りない部分を補い、追加しましょう。

```
// プライマリバッファ取得
DSBUFFERDESC   dsbd;
::ZeroMemory(&dsbd, sizeof(dsbd));
dsbd.dwSize        = sizeof(dsbd);
dsbd.dwFlags       = ????????????????????? | DSBCAPS_CTRLVOLUME;
dsbd.dwBufferBytes = ※ここは各自考えましょう;
dsbd.lpwfxFormat   = ※ここは各自考えましょう;
if(FAILED(※ここは各自考えましょう)) {
    ::OutputDebugString("*** Error - プライマリバッファ取得失敗(CDXAudio8_Init)\n");
    Release();
    return false;
}
```

(７)CDXAudio8::Release関数の適切な場所に、プライマリバッファを解放する処理を追加しましょう。