▶　DirectX Audio－第６回　サウンドバッファ　◀

DirectSoundでは、Waveデータを保持しておく領域をサウンドバッファと呼びます。

サウンドバッファ

サウンドバッファとは、DirectSoundが扱うWaveデータを保持しておくメモリ領域です。ディスクに保存されているWaveデータは、DirectSoundで直接扱うことができません。また、Waveデータが必要になるたびにディスクから読み込んでいたのでは、再生まで時間がかかり、映像とずれてしまいます。DirectSoundが扱うWaveデータは、サウンドバッファに格納しておく必要があります。
サウンドバッファは、「プライマリバッファ」と「セカンダリバッファ」があり、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、サウンドカードと密接に関わっています。このバッファに書き込まれているデータがそのまま再生されます。DirectSoundの協調レベルが標準レベル以外の場合は、プライマリバッファのフォーマットを設定することができます。しかし、サウンドドライバがWDM(Windows Driver Model)の場合は、最終的にその能力に合わせて出力されるため、ほとんど意味がありません。
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は必要になります。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、再生したいWaveデータを格納しておく領域です。このバッファは、複数生成することができ、個別に制御することができます。複数のセカンダリバッファが同時に再生された場合、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。プライマリバッファと異なるフォーマットや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、最低でも同時に再生したい数を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、この構造体をDirectSoundオブジェクトの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)

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バージョン１オブジェクトを生成し、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、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型の変数のアドレスを指定します。

３つ目の引数は、将来の互換性のためにあり、現時点ではNULLにしないとエラーになります。

－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。それ以外はエラーの原因をエラーコードで返します。

CreateSoundBufferメソッドが生成するのは、DirectSoundBufferオブジェクト(バージョン１)です。DirectSoundBuffer8オブジェクトは、QueryInterfaceメソッドで取得します。１つ目の引数にDirectSoundBuffer8オブジェクトのインタフェース参照識別子"IID_IDirectSoundBuffer8"、２つ目の引数に生成されるサウンドオブジェクトのインタフェースを受け取る変数(LPDIRECTSOUNDBUFFER8型)のアドレスをLPVOID*型にキャストして指定します。

```
// 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(lpDSound8は、初期化済みのDirectSound8オブジェクト)
LPDIRECTSOUNDBUFFER   lpdsb = NULL;
lpDSound8->CreateSoundBuffer(&dsbd, &lpdsb, NULL);

// DirectSoundBuffer8オブジェクト取得
LPDIRECTSOUNDBUFFER8   lpDSBuffer8 = NULL;
lpdsbd->QueryInterface(IID_IDirectSoundBuffer8, (LPVOID*)lpDSBuffer8);

lpdsb->Release();      // DirectSoundBufferバージョン１は不要なので解放する
```

プライマリバッファの生成

プライマリバッファを生成すると、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もCreateSoundBufferメソッドで生成しますが、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にする必要があります。なお、プライマリバッファはDirectSoundBuffer8オブジェクトを取得することはできません。

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

LPDIRECTSOUNDBUFFER   lpDSBPrimary = NULL;
lpDSound8->CreateSoundBuffer(&dsbd, &lpDSBPrimary, NULL);
```

バッファの解放

プライマリバッファとセカンダリバッファは、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、保持していたバッファも解放されます。

サウンド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。同じ音を同時に再生したいときでも、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。このような場合、複数のセカンダリバッファで同じサウンドメモリを共有することができます。サウンドメモリを共有するセカンダリバッファを生成するには、DirectSound8オブジェクトのDuplicateSoundBufferメソッドを使います。

DuplicateSoundBufferメソッド
－説明－
DuplicateSoundBufferメソッドは、サウンドメモリを共有するDirectSoundBufferバージョン１オブジェクトを生成し、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、共有元のサウンドバッファです。DirectSoundBufferオブジェクトまたはDirectSoundBuffer8オブジェクトを指定します。

２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です。LPDIRECTSOUNDBUFFER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、それ以外はエラーの原因をエラーコードで返します。

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、共有元バッファ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。
サウンドメモリは、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。

DuplicateSoundBufferメソッドが生成するDirectSoundBufferオブジェクトは、バージョン１です。DirectSoundBuffer8オブジェクトは、QueryInterfaceメソッドで取得します。

```
// セカンダリバッファ複製(lpDSBuffer8は、初期化済みのDirectSoundBuffer8オブジェクト)
LPDIRECTSOUNDBUFFER   lpdsb = NULL;
lpDSound8->DuplicateSoundBuffer(lpDSBuffer8, &lpdsb);

// DirectSoundBuffer8オブジェクト取得
LPDIRECTSOUNDBUFFER8   lpDSBuffer2 = NULL;
lpdsbd->QueryInterface(IID_IDirectSoundBuffer8, (LPVOID*)lpDSBuffer2);

lpdsb->Release();      // DirectSoundBufferバージョン１は不要なので解放する
```

○ 課題 ○

プライマリバッファとセカンダリバッファを生成する処理を追加しましょう。

(１)DXAUtilsでは、セカンダリバッファ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、バッファごとに変数を与える方法、たとえば以下のように

```
LPDIRECTSOUNDBUFFER8   lpDSBuffer1;      // セカンダリバッファ１
LPDIRECTSOUNDBUFFER8   lpDSBuffer2;      // セカンダリバッファ２
LPDIRECTSOUNDBUFFER8   lpDSBuffer3;      // セカンダリバッファ３
LPDIRECTSOUNDBUFFER8   lpDSBuffer4;      // セカンダリバッファ４
```

とするよりも利点が多いためです。たとえば、すべてのバッファ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行うことができます。また、DXAUtils.hにバッファ指定用の列挙体を宣言します。これは、セカンダリバッファ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－

```
// サウンドバッファ指定用列挙体
enum DXA_SB {
    DXA_SB1,      // サウンドバッファ１
    DXA_SB2,      // サウンドバッファ２
    DXA_SB3,      // サウンドバッファ３
    DXA_SB4,      // サウンドバッファ４
    DXA_SB5,      // サウンドバッファ５
    DXA_SB6,      // サウンドバッファ６
    DXA_SB7,      // サウンドバッファ７
    DXA_SB8,      // サウンドバッファ８
    DXA_SB_MAX    // サウンドバッファ最大数
};
```

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、バッファオブジェクトのインタフェースを定義します。

－追加２－

```
static LPDIRECTSOUNDBUFFER    g_lpDSBPrimary          =  NULL;     // プライマリバッファ
static LPDIRECTSOUNDBUFFER8   g_lpDSBuf8[DXA_SB_MAX]  = {NULL};    // セカンダリバッファ
```

このように定義した場合、たとえばセカンダリバッファ１(添字0)を指定するには

```
g_lpDSBuf8[DXA_SB1]
```

または、

```
g_lpDSBuf8[0]
```

と記述します。

(２)プライマリバッファを生成する処理を、DirectX Audioの初期化時に行うようにします。以下のプログラムを完成させ、適切な場所に追加しましょう。

－追加３－

```
    // プライマリバッファ生成
    DSBUFFERDESC   dsbd;
    ZeroMemory(&dsbd, sizeof(dsbd));
    dsbd.dwSize          = ※ここは各自考えましょう
    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(※ここは各自考えましょう) {
        OutputDebugString("*** Error - プライマリバッファ生成失敗(DXAInit)\n");
        DXARelease();
        return false;
    }
```

(３)プライマリバッファを解放する処理を適切な場所に追加しましょう。

(４)以下のプログラムは、セカンダリバッファを生成するDXACreateBuffer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。なお、この関数は(５)で作成する関数も必要です。

DXACreateBuffer関数
－説明－
DXACreateBuffer関数は、セカンダリバッファを生成します。指定されたバッファがすでに生成されている場合は、エラーとせずに解放したあと生成します。

－パラメータ－
const DXA_SB dxaBuf…生成するバッファの指定。DXA_SB列挙体型を指定
LPWAVEFORMATEX lpwfxFormat…生成するバッファのフォーマットを格納したWAVEFORMATEX構造体変数のアドレス
const DWORD dwBufferBytes…生成するバッファのバイト数

－戻り値－
生成に成功した場合はtrue，それ以外はfalseを返します。

－追加４－
```
/*****************************************************************************************************/
/*                                        セカンダリバッファ生成                                       */
/*****************************************************************************************************/
bool DXACreateBuffer(const DXA_SB dxaBuf, LPWAVEFORMATEX lpwfxFormat, const DWORD dwBufferBytes)
{
    if(NULL == g_lpDSound8) {
        OutputDebugString("*** Error - DirectSound未初期化(DXACreateBuffer)\n");
        return false;
    }

    DXAReleaseBuffer(dxaBuf);

    // セカンダリバッファ能力設定
    DSBUFFERDESC   dsbd;
    ZeroMemory(&dsbd, sizeof(dsbd));
    dsbd.dwSize          = ※ここは各自考えましょう
    dsbd.dwBufferBytes   = ※ここは各自考えましょう
    dsbd.dwReserved      = ※ここは各自考えましょう
    dsbd.lpwfxFormat     = ※ここは各自考えましょう
    dsbd.dwFlags         = DSBCAPS_LOCDEFER | DSBCAPS_CTRLVOLUME | DSBCAPS_CTRLPAN |
                           DSBCAPS_GETCURRENTPOSITION2;
    dsbd.guid3DAlgorithm = GUID_NULL;

    // DirectSoundBuffer生成
    LPDIRECTSOUNDBUFFER   lpdsb;
    if(FAILED(※ここは各自考えましょう)) {
        OutputDebugString("*** Error - DirectSoundBuffer生成失敗(DXACreateBuffer)\n");
        return false;
    }

    // DirectSoundBuffer8取得
    const HRESULT   hr = lpdsb->?????????????(?????????????????????????, (LPVOID*)&g_lpDSBuf8[dxaBuf]);
    lpdsb->Release();
    if(S_OK != hr) {
        OutputDebugString("*** Error - DirectSoundBuffer8取得失敗(DXACreateBuffer)\n");
        return false;
    }

    return true;
}
```

(５)セカンダリバッファを解放するDXAReleaseBuffer関数の仕様を参考に、関数を完成させましょう。

DXAReleaseBuffer関数
－説明－
DXAReleaseBuffer関数は、指定されたセカンダリバッファを解放します。

－パラメータ－
const DXA_SB dxaBuf…解放するバッファの指定。DXA_SB列挙体型を指定

－戻り値－
なし

－追加５－
```
/*****************************************************************************************************/
/*                                          セカンダリバッファ解放                                     */
/*****************************************************************************************************/
void DXAReleaseBuffer(const DXA_SB dxaBuf)
{
    if(NULL != g_lpDSBuf8[dxaBuf]) {
        g_lpDSBuf8[dxaBuf]->Stop();
        ※ここは各自考えましょう;
        ※ここは各自考えましょう;
    }
}
```

(６)以下のDXAReleaseAllBuffers関数は、すべてのセカンダリバッファを解放する関数です。不足部分を補って関数を完成させましょう。

－追加６－
```
/*****************************************************************************************************/
/*                                          全セカンダリバッファ解放                                   */
/*****************************************************************************************************/
void DXAReleaseAllBuffers()
{
    for(int i = 0; i < DXA_SB_MAX; i++)
        ????????????????((DXA_SB)i);
}
```

この関数は、シーンチェンジのようなすべてのセカンダリバッファを解放したいときに使用すると便利です。

(７)－追加７－を適切な場所に追加し、DirectX Audio解放時にすべてのセカンダリバッファが正しく解放されるようにしましょう。

－追加７－
```
    // すべてのセカンダリバッファを解放
    DXAReleaseAllBuffers();
```

(８)以下のプログラムは、サウンドメモリを共有するセカンダリバッファを生成するDXADuplicateBuffer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。

DXADuplicateBuffer関数
－説明－
DXADuplicateBuffer関数は、サウンドメモリを共有するセカンダリバッファを生成します。指定されたバッファがすでに生成されている場合は、エラーとせずに解放したあとに生成します。

－パラメータ－
const DXA_SB dxaOriginal…共有元のバッファの指定。DXA_SB列挙体型を指定
const DXA_SB dxaDuplicate…生成するバッファの指定。DXA_SB列挙体型を指定

－戻り値－
生成に成功した場合はtrue，それ以外はfalseを返します。

－追加８－
```
/*****************************************************************************************************/
/*                                          セカンダリバッファ複製                                     */
/*****************************************************************************************************/
bool DXADuplicateBuffer(const DXA_SB dxaOriginal, const DXA_SB dxaDuplicate)
{
    DXAReleaseBuffer(dxaDuplicate);

    if(NULL == g_lpDSBuf8[dxaOriginal]) {
        OutputDebugString("*** Error - バッファ未生成(DXADuplicateBuffer)\n");
        return false;
    }
```

```
    // バッファ複製
    LPDIRECTSOUNDBUFFER   lpdsb;
    if(※ここは各自考えましょう) {
        OutputDebugString("*** Error - バッファ複製失敗(DXADuplicateBuffer)\n");
        return false;
    }

    // DirectSoundBuffer8取得
    const HRESULT   hr = ※ここは各自考えましょう
    lpdsb->Release();
    if(S_OK != hr) {
        OutputDebugString("*** Error - DirectSoundBuffer8取得失敗(DXADuplicateBuffer)\n");
        return false;
    }

    return true;
}
```