オブジェクト指向と ゲームプログラミング

▶ DirectX Audio編 － 第１２回　DirectSound3Dと３Ｄバッファの生成 ◀

DirectSound3D

DirectSound3Dは、3Dグラフィックスと統合できるサウンド用の3D環境を提供します。DirectSound3Dの音源(3Dサウンドバッファ)は3D空間の座標を持ち、また、音を聞くオブジェクトを再現したリスナという概念も存在します。また、サウンドを移動させてドップラー効果を適用したり、距離に対するサウンドの減衰率のような物理的パラメータを制御したりすることができます。

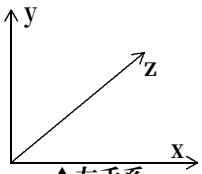
左右のスピーカから出力される音量は、DirectSound3Dが音源とリスナーの座標から計算します。そのため、ステレオデータは意味を持ちません。

音源、リスナともに位置と速度を持ちます。リスナが音源に近づくほど音量が大きくなります。速度が設定されている場合、ドップラー効果が自動的に計算されます。ただし、この速度は計算のための値であり、音源やリスナが自動で動くというわけではありません。位置はプログラムで適切な値を設定します。

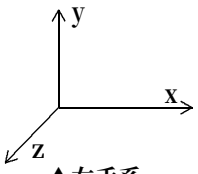
また、DirectSound3Dでは、D3DVALUEやD3DVECTORといったDirect3D(DirectX Graphics)でよく使われる型が使われるほか、座標系も同じ左手系であるため、Direct3Dと連携しやすくなっています。

左手座標

DirectSound3Dでは、3Dの座標を表すのに左手系が使われています。3D空間では、ｘ軸、ｙ軸、ｚ軸によって構成される座標系を用いますが、このとき、軸の方向付けによって２つの座標系が考えられます。ｘ軸とｙ軸で作られる平面が目の前にあると仮定したとき、ｘ-ｙ平面に対してｚ軸の方向が２種類あることが考えられます。つまり、平面からｚ軸が向こうを指している場合と、平面の手前を指している場合の２種類です。それらは、親指、人差し指、中指をそれぞれｙ軸、ｚ軸、ｘ軸に見立てたとき、左手で表されるか右手で表されるかによって、左手系あるいは右手系と呼ばれます。



▲左手系



▲右手系

３Ｄサウンドバッファの生成

3Dサウンドを制御するには、DirectSoundBuffer8オブジェクトとDirectSound3DBuffer8オブジェクトの２つが必要になります。DirectSound3DBuffer8オブジェクトは、3D空間上のバッファの制御を行うことができます。このオブジェクトでは、再生やボリュームの設定といったサウンドバッファの制御を行うことはできません。これらは、DirectSoundBuffer8オブジェクトが行います。

3Dサウンドバッファを生成するには、サウンドバッファの生成時にDSBCAPS_CTRL3Dフラグを指定します。このとき、パンの設定はDirectSound3Dが行うため、DSBCAPS_CTRLPANフラグは指定できません。また、guid3DAlgorithmメンバに3Dアルゴリズムを指定する必要があります。指定できるアルゴリズムは、以下のものです。なお、サウンドバッファのフォーマットのチャンネル数が２以上である場合、3Dサウンドバッファを生成することはできません。

DS3DALG_DEFAULT	デフォルトのアルゴリズム。現在は、DS3DALG_NO_VIRTUALIZATIONが選択されます。
DS3DALG_NO_VIRTUALIZATION	3Dサウンドをステレオパンとボリュームコントロールでエミュレートします。最も効率的なアルゴリズムですが、HRTF処理は供されません。
DS3DALG_HRTF_LIGHT	CPUにあまり負荷をかけないHRTF(頭部伝達関数)アルゴリズムで3Dサウンドが処理されます。
DS3DALG_HRTF_FULL	高品質なHRTFアルゴリズムで３Ｄサウンドが処理されます。

DS3DALG_HRTF_LIGHTとDS3DALG_HRTF_FULLは、サウンドカードがWDMドライバのとき使用できます。使用できない場合は、DS3DALG_NO_VIRTUALIZATIONで代用されます。

サウンドバッファが生成できたら、IDirectSoundBuffer8::QueryInterfaceメソッドでDirectSound3DBuffer8オブジェクトのインタフェースを取得します。

```
// ３Ｄサウンドバッファ生成
// 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       = 1;
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_CTRLFREQUENCY
                       DSBCAPS_CTRLFX | DSBCAPS_CTRL3D | DSBCAPS_GETCURRENTPOSITION2;
dsbd.dwBufferBytes   = wfx.nAvgBytesPerSec * 4;      s// ４秒
dsbd.lpwfxFormat     = &wfx;
dsbd.guid3DAlgorithm = DS3DALG_HRTF_LIGHT;

// 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 = NULL;
pDSound->CreateSoundBuffer(&dsbd, &pdsb, NULL);

// DirectSoundBuffer8オブジェクト取得
IDirectSoundBuffer8*   pDSBuffer = NULL;
pdsb->QueryInterface(IID_IDirectSoundBuffer8, (LPVOID*)pDSBuffer);

pdsb->Release();        // DirectSoundBufferは不用なので解放する

// IDirectSound3DBuffer8インタフェース取得
IDirectSound3DBuffer8*   pDS3DBuffer = NULL;    // ３Ｄバッファオブジェクトへのインタフェース
pDSBuffer->QueryInterface(IID_IDirectSound3DBuffer8, (LPVOID*)&pDS3DBuffer);
```

DirectSound3DBuffer8オブジェクトの解放は、Releaseメソッドで行います。また、DirectSound3DBuffer8オブジェクトは、DirectMusicからも取得することができます。

課　　題

サウンドバッファにサウンド３Ｄバッファ機能を追加しましょう。

(１)3Dバッファをカプセル化したクラスを作成します。

DirectSound3DBufferオブジェクトは、IDirectSound3DBuffer8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、3Dバッファを必要としない場合や、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、3DバッファクラスもNull Objectパターンを適用します。

3Dバッファクラスは､通常のサウンドバッファに3Dバッファの機能を拡張したものと考えられるので、サウンドバッファクラスを継承して作成しますが、サウンドバッファに3Dバッファの機能を追加したものと考えることもできます。クラスを拡張する場合、継承のほかに、集約を用いても行うことができます。3Dバッファクラスは単独のクラスで作成し、CSoundBuffeクラスに集約させることにします。このように、継承より集約を重視したプログラミングを、アスペクト指向プログラミング(AOP:Aspect Oriented Programing)と呼びます。

以上を考慮した3Dバッファクラスのヘッダファイル(Sound3DBuffer.hpp)を次のように作成しましょう。

－Sound3DBuffer.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】
    Sound3DBuffer.hpp
                サウンド３Ｄ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３Ｄバッファインタフェース定義                       */
/*********************************************************************************************/
class ISound3DBuffer {
public:
    virtual ~ISound3DBuffer() {}
};

/*********************************************************************************************/
/*                                    サウンド３Ｄバッファクラス定義                           */
/*********************************************************************************************/
class CSound3DBuffer : public ISound3DBuffer {
public:
    CSound3DBuffer(IDirectSound3DBuffer8* pDS3DBuffer);
    virtual ~CSound3DBuffer() { m_pDS3DBuffer->Release(); }

private:
    IDirectSound3DBuffer8*   m_pDS3DBuffer;
};

/*********************************************************************************************/
/*                                  NULLサウンド３Ｄバッファクラス定義                         */
/*********************************************************************************************/
class CNullSound3DBuffer : public ISound3DBuffer {
public:
    CNullSound3DBuffer() {}
    virtual ~CNullSound3DBuffer() {}
};
```

ISound3DBufferクラスは、CSound3DBufferクラスとCNullSound3DBuffer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。サウンドバッファクラスがDirectSound3DBuffer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CSound3DBufferオブジェクトかCNullSound3DBufferオブジェクトを生成(するように設計)します。

CSound3DBufferクラスがDirectSound3DBufferオブジェクトをカプセル化したクラスです。CNullSound3DBuffer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)３Ｄバッファクラスのソースファイル(Sound3DBuffer.cpp)を次のように作成しましょう。

－Sound3DBuffer.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】
     Sound3DBuffer.cpp
                サウンド３Ｄ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 "Sound3DBuffer.hpp"
#include <cassert>

/*********************************************************************************************/
/*                                        コンストラクタ                                       */
/*********************************************************************************************/
CSound3DBuffer::CSound3DBuffer(IDirectSound3DBuffer8* pDS3DBuffer)
{
    assert(pDS3DBuffer != NULL);
    m_pDS3DBuffer = pDS3DBuffer;
    m_pDS3DBuffer->AddRef();        // 参照カウンタインクリメント
}
```

(３)CDXAudio8::CreateBufferFromFile関数を３Ｄバッファ対応に変更します。プロトタイプを以下のように変更しましょう。

```
ISoundBuffer* CreateBufferFromFile(LPSTR inFileName, const bool in3DBuffer = false);
```

in3DBufferは、通常のバッファを生成する場合はfalse，３Ｄバッファを生成する場合はtrueを渡します。省略するとfalseになります。

(４)CDXAudio8::CreateBufferFromFile関数を以下のように変更しましょう。

```
/*********************************************************************************************/
/*                                        バッファ生成                                         */
/*********************************************************************************************/
ISoundBuffer* CDXAudio8::CreateBufferFromFile(LPSTR inFileName, const bool in3DBuffer)
{
    (省略)

        // セカンダリバッファ生成
        ???????????   dsbd;
        ::ZeroMemory(&dsbd, sizeof(dsbd));
        dsbd.dwSize          = sizeof(dsbd);
        dsbd.dwBufferBytes   = ChildChunk.cksize;
        dsbd.dwReserved      = 0;
        dsbd.lpwfxFormat     = &Format;
        dsbd.dwFlags         = DSBCAPS_LOCDEFER | DSBCAPS_CTRLVOLUME | DSBCAPS_CTRLFREQUENCY |
                               DSBCAPS_CTRLFX | DSBCAPS_GETCURRENTPOSITION2;
        if(in3DBuffer == false) {
            dsbd.dwFlags        |= DSBCAPS_CTRLPAN;
            dsbd.guid3DAlgorithm = GUID_NULL;
        } else {
            dsbd.dwFlags        |= ※ここは各自考えましょう;
```

```
            dsbd.guid3DAlgorithm = DS3DALG_HRTF_LIGHT;
        }

    (省略)

    return pBuffer;
}
```

(５)ISound3DBufferクラスをCSoundBufferクラスに集約します。以下のメンバを適切な場所に追加しましょう。

```
    ISound3DBuffer*   m_p3DBuffer;      // ３Ｄバッファ
```

(６)CNullSoundBufferクラスは、「何もしない」クラスなので、「何もしない」３Ｄバッファを集約します。以下のメンバを適切な場所に追加しましょう。

```
    CNullSound3DBuffer   m_3DBuffer;    // 何もしない３Ｄバッファ
```

(７)サウンドバッファクラスに、３Ｄバッファへのポインタを取得する関数Get3DBufferを追加します。以下の「追加１～３」を適切な場所に追加しましょう。

－追加１－

```
virtual ISound3DBuffer* Get3DBuffer() = 0;
```

－追加２－

```
virtual ISound3DBuffer* Get3DBuffer() { return m_p3DBuffer; }
```

－追加３－

```
virtual ISound3DBuffer* Get3DBuffer() { return &m_3DBuffer; }
```

(８)CSoundBufferクラスのコンストラクタを以下のように変更しましょう。

```
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

    // ３Ｄバッファ設定
    IDirectSound3DBuffer8*   pDS3DBuffer;
    if(m_pDSBuffer->??????????????(IID_IDirectSound3DBuffer8, (LPVOID*)&pDS3DBuffer) == S_OK) {
        m_p3DBuffer = new ??????????????(pDS3DBuffer);
        pDS3DBuffer->Release();
    } else {
        m_p3DBuffer = new ????????????????????();
    }
}
```

応用問題　サウンドバッファクラスを継承して３Ｄバッファクラスを作成した場合、どのようなプログラムになるのか考察してみましょう。