オブジェクト指向と ゲームプログラミング

▶ DirectX Audio編 － 第１４回　リスナ ◀

リスナ

リスナとは、音を聞くオブジェクトのことです。DirectSound3Dで出力される最終的なサウンドは、3Dサウンドバッファとリスナのパラメータによって決定されます。

リスナの取得

リスナは、DirectSound3DListenerオブジェクトで管理します。このオブジェクトのインタフェースは、プライマリバッファから取得します。プライマリバッファの生成時にDSBCAPS_CTRL3Dフラグを指定すると、生成後にQueryInterfaceメソッドで取得できるようになります。

```
// プライマリバッファ生成(pDSoundは、初期化済みのDirectSound8オブジェクト)
IDirectSoundBuffer*   pDSBPrimary;

DSBUFFERDESC   dsbd;
ZeroMemory(&dsbd, sizeof(dsbd));
dsbd.dwSize        = sizeof(dsbd);
dsbd.dwFlags       = DSBCAPS_PRIMARYBUFFER | DSBCAPS_CTRLVOLUME | DSBCAPS_CTRL3D;
dsbd.dwBufferBytes = 0;
dsbd.lpwfxFormat   = NULL;
pDSound->CreateSoundBuffer(&dsbd, &lpDSBPrimary, NULL);

// リスナインタフェースの取得
IDirectSound3DListener8*   pDS3DListener;   // リスナオブジェクトへのインタフェース
pDSBPrimary->QueryInterface(IID_IDirectSound3DListener8, (LPVOID*)&pDS3DListener);
```

リスナの設定

リスナは、向き、位置、速度、ロールオフ係数、ドップラー乗数、距離係数といったパラメータを設定することができます。これらの設定値と3Dバッファの設定値をもとに、スピーカから出力される最終的なサウンドが計算されます。なお、3Dでない通常のサウンドバッファは、これらの影響を受けません。

向きの設定

3Dサウンドの効果を正しく計算するには、リスナの向きを設定する必要があります。リスナの向きはIDirectSound3DListener8::SetOrientationメソッドで設定します。このとき、Direct3Dのカメラと同じように「リスナの前方ベクトル」と「リスナの上方ベクトル」を設定します。この２つのベクトルは直角でなければならず、DirectSound3Dは前方ベクトルを設定した後、必要に応じて上方ベクトルを調整します。

```
// 向きの設定
pDS3DListener->SetOrientation(0.0f, 0.0f, 1.0f,     // リスナの前方を示す方向ベクトル
                              0.0f, 1.0f, 0.0f,     // リスナの上方を示す方向ベクトル
                              DS3D_IMMEDIATE);
```

現在の向きは、GetOrientationメソッドで取得することができます。

位置の設定

リスナの位置はIDirectSound3DListener8::SetPositionメソッドで設定します。また、GetPositionメソッドで現在の位置を取得することができます。

```
// 位置の設定(左からｘ，ｙ，ｚ位置)
pDS3DListener->SetPosition(0.0f, 0.0f, 0.0f, DS3D_IMMEDIATE);
```

速度の設定

リスナの速度の設定はIDirectSound3DListener8::SetVelocityメソッドで行います。また、GetVelocityメソッドで現在の速度を取得することができます。

```
// 速度設定(左からｘ，ｙ，ｚの速度ベクトル)
pDS3DListener->SetVelocity(0.0f, 0.0f, 1.0f, DS3D_IMMEDIATE);
```

この速度は、DirectSound3Dがドップラー効果を計算するための値であり、実際にリスナが移動するわけではありません。

ロールオフ係数

ロールオフ係数とは、リスナとサウンドバッファの距離による音量減衰の度合いのことです。ロールオフ係数は、IDirectSound3DListener8::SetRolloffFactorメソッドで設定します。値の範囲は「DS3D_MINROLLOFFFACTOR」(ロールオフなし)から「DS3D_MAXROLLOFFFACTOR」(実世界の10倍の効果)です。デフォルト値はDS3D_DEFAULTROLLOFFFACTORで、実世界と同じ減衰量になっています。

```
// ロールオフ係数の設定
pDS3DListener->SetRolloffFactor(2.0f, DS3D_IMMEDIATE);       // 実世界の２倍
```

ドップラー乗数

ドップラー効果は、リスナと3Dサウンドバッファの相対的な速度により決定されますが、IDirectSound3DListener8::SetDopplerFactorメソッドでドップラー乗数を設定することにより、効果を調整することができます。設定できる範囲は、「DS3D_MINDOPPLERFACTOR」(ドップラー効果なし)から「DS3D_MAXDOPPLERFACTOR」(実世界の10倍の効果)です。デフォルト値はDS3D_DEFAULTDOPPLERFACTORで、実世界と同じ乗数になっています。

```
// ドップラー乗数の設定
pDS3DListener->SetDopplerFactor(1.5f, DS3D_IMMEDIATE);       // 実世界の１．５倍
```

距離係数

距離係数は、1 ベクトル単位におけるメートル数のことです。DirectSound3Dの「1.0」が、何メートルに相当するのかを表します。たとえば、バッファの速度が(2.0, 0.0, 0.0)である場合、音源はｘ軸に沿って、２メートル／秒で移動しているとみなされます。

距離係数の設定は、IDirectSound3DListener8::SetDistanceFactorメソッドで行います。設定できる範囲は、「DS3D_MINDISTANCEFACTOR」(floatの最小値)から「DS3D_MAXDISTANCEFACTOR」(floatの最大値)です。デフォルト値はDS3D_DEFAULTDISTANCEFACTORで、1.0メートルに設定されています。

```
// 距離係数の設定
pDS3DListener->SetDistanceFactor(0.1f, DS3D_IMMEDIATE);      // 0.1メートルに設定
```

すべてのパラメータの設定

IDirectSound3DListener8::SetAllParametersメソッドは、リスナに関するすべてのパラメータの設定を行うことができます。また、GetAllParametersメソッドですべてのパラメータを取得することができます。これらのメソッドを呼び出す前に、引数で渡すDS3DLISTENER構造体のdwSizeメンバに構造体のサイズを設定しておく必要があります。

```
// すべてのパラメータの設定
DS3DLISTENER   ds3dl;
ZeroMemory(&ds3dl, sizeof(ds3dl));
ds3dl.dwSize           = sizeof(ds3dl);
ds3dl.vPosition        = D3DXVECTOR3(0.0f, 0.0f, 0.0f);        // 位置
ds3dl.vVelocity        = D3DXVECTOR3(0.0f, 0.0f, 0.0f);        // 速度ベクトル
ds3dl.vOrientFront     = D3DXVECTOR3(0.0f, 0.0f, 1.0f);        // 前方ベクトル
ds3dl.vOrientTop       = D3DXVECTOR3(0.0f, 1.0f, 0.0f);        // 上方ベクトル
ds3dl.flDistanceFactor = 1.0f;                                 // 距離係数
ds3dl.flRolloffFactor  = DS3D_DEFAULTROLLOFFFACTOR;            // ロールオフ係数
```

```
    ds3dl.flDopllerFactor  = DS3D_DEFAULTDOPPLERFACTOR;                // ドップラー乗数
    pDS3DListener->SetAllParameters(&ds3dl, DS3D_IMMEDIATE);
```

課　　題

CDXAudio8クラスにリスナを追加しましょう。

(１)DirectSound3DListenerオブジェクトをカプセル化したクラスCSoundListenerを作成します。

DirectSound3DListenerオブジェクトは、IDirectSound3DListener8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3Dサウンドをサポートとしない場合、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、CSoundListenerクラスもNull Objectパターンを適用します。

CSoundListenerクラスのヘッダファイル(SoundListener.hpp)を以下のように作成しましょう。

－SoundListener.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】
    SoundListener.hpp
              サウンドリスナクラスヘッダ

  【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>

/*********************************************************************************************/
/*                                  サウンドリスナインタフェース定義                         */
/*********************************************************************************************/
class ISoundListener {
public:
    virtual ~ISoundListener() {}

    virtual void SetPosition(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z) = 0;
    virtual void SetVelocity(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z) = 0;
    virtual void SetOrientation(const D3DVECTOR& inFront, const D3DVECTOR& inTop) = 0;
    virtual void SetRolloffFactor(const D3DVALUE inRolloff) = 0;
    virtual void SetDopplerFactor(const D3DVALUE inDoppler) = 0;
    virtual void SetDistanceFactor(const D3DVALUE inDistance) = 0;
    virtual void SetAllParameters(const DS3DLISTENER& inListener) = 0;

    virtual void Initialize() = 0;
};

/*********************************************************************************************/
/*                                  サウンドリスナクラス定義                                 */
/*********************************************************************************************/
class CSoundListener : public ISoundListener {
public:
```

```
    CSoundListener(IDirectSound3DListener8* pDS3DListener);
    virtual ~CSoundListener() { m_pDS3DListener->Release(); }

    virtual void SetPosition(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z);
    virtual void SetVelocity(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z);
    virtual void SetOrientation(const D3DVECTOR& inFront, const D3DVECTOR& inTop);
    virtual void SetRolloffFactor(const D3DVALUE inRolloff);
    virtual void SetDopplerFactor(const D3DVALUE inDoppler);
    virtual void SetDistanceFactor(const D3DVALUE inDistance);
    virtual void SetAllParameters(const DS3DLISTENER& inListener);

    virtual void Initialize();

private:
    IDirectSound3DListener8*   m_pDS3DListener;
};

/*****************************************************************************************************/
/*                                    NULLサウンドリスナクラス定義                                   */
/*****************************************************************************************************/
class CNullSoundListener : ※ここは各自考えましょう {
    ※このクラスの内容は、各自考えましょう
};
```

(２)リスナのパラメータを設定する各関数の仕様は、以下のとおりです。

SetPosition 設定

リスナの位置を設定します。

書式 `void SetPosition(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z);`

x	リスナのｘ座標
y	リスナのｙ座標
z	リスナのｚ座標

SetVelocity 設定

リスナの速度を設定します。

書式 `void SetVelocity(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z);`

x	ｘ方向の速度ベクトル
y	ｙ方向の速度ベクトル
z	ｚ方向の速度ベクトル

SetOrientation 設定

リスナの頭部の向きを設定します。

書式 `void SetConeOrientation(const D3DVECTOR& inFront, const D3DVECTOR& inTop);`

inFront	前方ベクトル
inTop	上方ベクトル

SetRolloffFactor 設定

距離に応じた減衰率を決定するロールオフ係数を設定します。

書式 `void SetRolloffFactor(const D3DVALUE inRolloff);`

inRolloff	ロールオフ係数

SetDopplerFactor 設定

ドップラー効果に対する乗数を設定します。

書式 `void SetDopplerFactor(const D3DVALUE inDoppler);`

inDoppler	ドップラー乗数

SetDistanceFactor 設定

距離係数(１ベクトル単位におけるメートル数)を設定します。

書式 `void SetDistanceFactor(const D3DVALUE inDistance);`

inDistance	距離係数

SetAllParameters 設定

リスナのすべてのパラメータを設定します。

書式 `void SetAllParameters(const DS3DLISTENER& inListener);`

inListener	リスナのパラメータが格納されたDS3DLISTENER構造体の変数

Initialize 初期化

リスナのすべてのパラメータを初期化します。

書void Initialize();

(３)関数の仕様をもとに、CSoundListenerクラスのソースファイル(SoundListener.cpp)を完成させましょう。

－SoundListener.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】
     SoundListener.cpp
                サウンドリスナクラス

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

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

/*****************************************************************************************/
/*                                      インクルードファイル                                 */
/*****************************************************************************************/
#include "SoundListener.hpp"
#include <cassert>

/*****************************************************************************************/
/*                                      コンストラクタ                                       */
/*****************************************************************************************/
CSoundListener::CSoundListener(IDirectSound3DListener8* pDS3DListener)
{
    assert(pDS3DListener != NULL);
    m_pDS3DListener = pDS3DListener;
    m_pDS3DListener->AddRef();          // 参照カウンタインクリメント
}

/*****************************************************************************************/
/*                                      位置設定                                             */
/*****************************************************************************************/
void CSoundListener::SetPosition(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z)
{
    m_pDS3DListener->※ここは各自考えましょう;
}

/*****************************************************************************************/
/*                                      速度設定                                             */
/*****************************************************************************************/
void CSoundListener::SetVelocity(const D3DVALUE x, const D3DVALUE y, const D3DVALUE z)
{
    m_pDS3DListener->※ここは各自考えましょう;
}

/*****************************************************************************************/
/*                                      向き設定                                             */
/*****************************************************************************************/
void CSoundListener::SetOrientation(const D3DVECTOR& inFront, const D3DVECTOR& inTop)
{
    m_pDS3DListener->??????????????(inFront.x, inFront.y, inFront.z,
                                    inTop.x, inTop.y, inTop.z, DS3D_IMMEDIATE);
}

/*****************************************************************************************/
```

```
/*                                          ロールオフ係数設定                                          */
/*******************************************************************************************************/
void CSoundListener::SetRolloffFactor(const D3DVALUE inRolloff)
{
    m_pDS3DListener->※ここは各自考えましょう;
}

/*******************************************************************************************************/
/*                                          ドップラー乗数設定                                          */
/*******************************************************************************************************/
void CSoundListener::SetDopplerFactor(const D3DVALUE inDoppler)
{
    m_pDS3DListener->※ここは各自考えましょう;
}

/*******************************************************************************************************/
/*                                          距離係数設定                                                */
/*******************************************************************************************************/
void CSoundListener::SetDistanceFactor(const D3DVALUE inDistance)
{
    m_pDS3DListener->※ここは各自考えましょう;
}

/*******************************************************************************************************/
/*                                          全パラメータ設定                                            */
/*******************************************************************************************************/
void CSoundListener::SetAllParameters(const DS3DLISTENER& inListener)
{
    m_pDS3DListener->※ここは各自考えましょう;
}

/*******************************************************************************************************/
/*                                          パラメータ初期化                                            */
/*******************************************************************************************************/
void CSoundListener::Initialize()
{
    DS3DLISTENER   Listener;
    ::ZeroMemory(&Listener, sizeof(Listener));
    Listener.dwSize           = sizeof(Listener);

    Listener.vPosition.x      = 0.0f;                           // 位置
    Listener.vPosition.y      = 0.0f;
    Listener.vPosition.z      = 0.0f;

    Listener.vVelocity.x      = 0.0f;                           // 速度
    Listener.vVelocity.y      = 0.0f;
    Listener.vVelocity.z      = 0.0f;

    Listener.vOrientFront.x   = 0.0f;                           // 前方ベクトル設定
    Listener.vOrientFront.y   = 0.0f;
    Listener.vOrientFront.z   = 1.0f;

    Listener.vOrientTop.x     = 0.0f;                           // 上方ベクトル設定
    Listener.vOrientTop.y     = 1.0f;
    Listener.vOrientTop.z     = 0.0f;

    Listener.flDistanceFactor = DS3D_DEFAULTDISTANCEFACTOR;     // 係数設定
    Listener.flRolloffFactor  = DS3D_DEFAULTROLLOFFFACTOR;
    Listener.flDopplerFactor  = DS3D_DEFAULTDOPPLERFACTOR;

    m_pDS3DListener->SetAllParameters(&Listener, DS3D_IMMEDIATE);
}
```

(４)以下のメンバをCDXAudio8クラスに追加しましょう。

```
    ISoundListener*   m_pListener;      // リスナ
```

(５)CDXAudio8クラスの適切な場所に、リスナを取得する以下の関数を追加しましょう。

```
    ISoundListener* GetListener() const { return m_pListener; }
```

(６)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_pListener(NULL)
```

(７)CDXAudio8::Init関数にリスナを生成する処理を追加し、さらにtry～catchによる例外処理を行うように変更します。Init関数を以下のように変更し、足りない部分を補いましょう。

```
bool CDXAudio8::Init(const HWND hWnd)
{
    Release();

    try {
        // DirectSoundオブジェクトの生成とインタフェースの取得
        if(::※ここは各自考えましょう != S_OK)
            throw "*** Error - DirectSoundオブジェクト生成失敗(CDXAudio8_Init)\n";

        // DirectSoundオブジェクト初期化
        if(m_pDSound->※ここは各自考えましょう != DS_OK)
            throw "*** Error - DirectSoundオブジェクト初期化失敗(CDXAudio8_Init)\n";

        // DirectSound協調レベル設定
        if(※ここは各自考えましょう != DS_OK)
            throw "*** Error - DirectSound協調レベル設定失敗(CDXAudio8_Init)\n";

        // DirectMusicLoaderオブジェクトの生成とインタフェースの取得
        if(::※ここは各自考えましょう != S_OK)
            throw "*** Error - ローダオブジェクト生成失敗(CDXAudio8_Init)\n";

        // DirectMusicPerformanceオブジェクトの生成とインタフェースの取得
        if(::※ここは各自考えましょう != S_OK)
            throw "*** Error - パフォーマンスオブジェクト生成失敗(CDXAudio8_Init)\n";

        // パフォーマンスの初期化とデフォルトオーディオパスのセットアップ
        if(m_pDMPerformance->※ここは各自考えましょう != S_OK)
            throw "*** Error - パフォーマンス初期化失敗(CDXAudio8_Init)\n";

        // 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))
            throw "*** Error - プライマリバッファ取得失敗(CDXAudio8_Init)\n";

        // リスナ生成
        IDirectSound3DListener8*   pDS3DListener;
        if(m_pDSBPrimary->QueryInterface(IID_IDirectSound3DListener8, (LPVOID*)&pDS3DListener) == S_OK) {
            m_pListener = new ?????????????(pDS3DListener);
            pDS3DListener->Release();
        } else {
            ::OutputDebugString("*** Error - リスナ取得失敗(CDXAudio8_Init)\n");
            m_pListener = new CNullSoundListener();
        }
    } catch(LPCSTR ErrorString) {
        ::OutputDebugString(ErrorString);
        Release();

        // 初期化に失敗した場合は、リスナにNULLオブジェクトを設定
        m_pListener = new ??????????????????();

        return false;
    }

    return true;
}
```

(８)CDXAudio8::Release関数の適切な場所に、リスナを解放する以下のプログラムを追加しましょう。

```
// リスナ解放
delete m_pListener;
m_pListener = NULL;
```

(９)リスナが正しく動作するか確認しましょう。