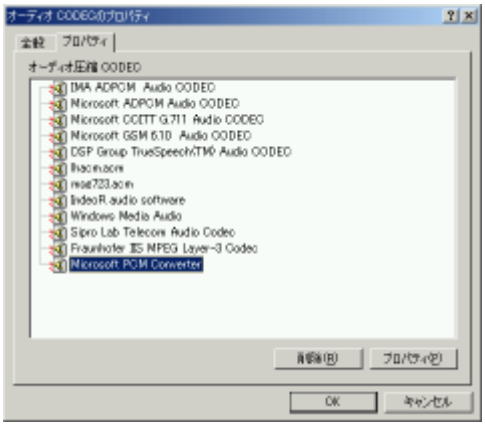
ゲームプログラミング

▶　DirectX Audio－第８回　ACMによるWaveファイルの読み込み　◀

ACMを利用すると、ADPCMやMP3などで圧縮されたWaveファイルを読み込むことができます。

ＡＣＭ

ACM(Audio Compression Manager)とは、音声データを圧縮および伸張するAPI群のことをいいます。利用可能な圧縮形式は環境により異なり、マルチメディアのプロパティで確認することができます。



ACMに対応したコーデック(CODEC:COmpression/DECompression[圧縮および伸張を行うプログラム])をインストールしたり、読み込んだりすることにより、扱える圧縮形式を増やすことができます。

圧縮されたWaveファイル

圧縮されたWaveファイルは、すべてのデータが圧縮されているのではなく、dataチャンクだけ圧縮されています。つまり、ヘッダなどは通常のWaveファイルで、Waveイメージだけ圧縮されているのです。圧縮されたWaveファイルを読み書きする場合は、ヘッダなどは通常どおり扱い、WaveイメージだけACMを利用します。

Waveイメージだけ圧縮したWaveファイルは、サウンドレコーダや音声編集ツールなどで作成することができます。

ＡＣＭを利用したWaveファイルの読み込み

DirectSoundは、圧縮されていないPCM形式(リニアPCM)しかサポートしていないため、圧縮されたWaveイメージをサウンドバッファに読み込んでも、正しく再生することはできません。そのため、リニアPCM形式に変換してからサウンドバッファに転送する必要があります。このとき、ACMを利用すると簡単にリニアPCM形式に変換することができます。ACMを利用した変換手順は、以下のようになります。

①変換後推奨フォーマットの取得	acmFormatSuggest関数
②ACMストリームを開く	acmStreamOpen関数
③変換後サイズの取得	acmStreamSize関数
④変換のためのヘッダの準備	acmStreamPrepareHeader関数
⑤目的の形式に変換	acmStreamConvert関数
⑥④のヘッダを破棄	acmStreamUnprepareHeader関数
⑦ACMストリームを閉じる	acmClose関数

ACMの関数を使用する場合は、ヘッダファイル<mmreg.h>および<msacm.h>とライブラリ"msacm32.lib"が必要になります。

○ 課題 ○

以下のプログラムは、DXACreateBufferFromFile関数をACMを用いるように変更したものです。MSDNライブラリなどでACM関数について調べながら、プログラムを解読しましょう。

```
/*****************************************************************************************************/
/*                                    セカンダリバッファ生成(ファイル)                                 */
/*****************************************************************************************************/
bool DXACreateBufferFromFile(const DXA_SB dxaBuf, LPSTR lpszFileName)
{
    if(NULL == g_lpDSound8) {
        OutputDebugString("*** Error - DirectSound未初期化(DXACreateBufferFromFile)\n");
        return false;
    }

    DXAReleaseBuffer(dxaBuf);

    // ACMWAVE構造体定義
    struct ACMWAVE {
        HMMIO         hmioFile;      // マルチメディアファイルハンドル
        HACMSTREAM    hACMStream;    // ACMストリーム
        LPBYTE        lpbyFormat;    // フォーマット
        LPBYTE        lpbyImage;     // ウェーブイメージ

        // コンストラクタ
        ACMWAVE() : hmioFile(NULL), hACMStream(NULL), lpbyFormat(NULL), lpbyImage(NULL)
        {
        }

        // デストラクタ
        ~ACMWAVE()
        {
            delete[] lpbyImage;
            delete[] lpbyFormat;
            if(NULL != hACMStream)  acmStreamClose(hACMStream, 0);
            if(NULL != hmioFile)    mmioClose(hmioFile, 0);
        }
    } acmWave;

    // ファイルオープン
    acmWave.hmioFile = mmioOpen(lpszFileName, NULL, MMIO_READ | MMIO_ALLOCBUF);
    if(NULL == acmWave.hmioFile) {
        OutputDebugString("*** Error - ファイルオープン失敗(DXACreateBufferFromFile)\n");
        return false;
    }

    MMCKINFO   mmciParent, mmciChild;
    ZeroMemory(&mmciParent, sizeof(mmciParent));
    ZeroMemory(&mmciChild,  sizeof(mmciChild ));

    // 'WAVE'チャンクへ
    mmciParent.fccType = mmioFOURCC('W', 'A', 'V', 'E');
    if(MMSYSERR_NOERROR != mmioDescend(acmWave.hmioFile, &mmciParent, NULL, MMIO_FINDRIFF)) {
        OutputDebugString("*** Error - 'WAVE'チャンクが見つかりません(DXACreateBufferFromFile)\n");
        return false;
    }

    // 'WAVE'チャンクの'fmt 'チャンクへ
    mmciChild.ckid = mmioFOURCC('f', 'm', 't', ' ');
    if(MMSYSERR_NOERROR != mmioDescend(acmWave.hmioFile, &mmciChild, &mmciParent, MMIO_FINDCHUNK)) {
        OutputDebugString("*** Error - 'fmt 'チャンクが見つかりません(DXACreateBufferFromFile)\n");
        return false;
    }

    // フォーマット情報取得
    acmWave.lpbyFormat           = new BYTE[mmciChild.cksize];
    LPWAVEFORMATEX   lpwfxFormat = (LPWAVEFORMATEX)acmWave.lpbyFormat;
    if(mmciChild.cksize != (DWORD)mmioRead(acmWave.hmioFile, (HPSTR)lpwfxFormat, mmciChild.cksize)) {
        OutputDebugString("*** Error - フォーマット情報取得失敗(DXACreateBufferFromFile)\n");
        return false;
    }

    // 'fmt 'チャンク退出
```

```
if(MMSYSERR_NOERROR != mmioAscend(acmWave.hmioFile, &mmciChild, 0)) {
    OutputDebugString("*** Error - 'fmt 'チャンク退出失敗(DXACreateBufferFromFile)\n");
    return false;
}

// 'WAVE'チャンクの'data'チャンクへ
mmciChild.ckid = mmioFOURCC('d', 'a', 't', 'a');
if(MMSYSERR_NOERROR != mmioDescend(acmWave.hmioFile, &mmciChild, &mmciParent, MMIO_FINDCHUNK)) {
    OutputDebugString("*** Error - 'data'チャンクが見つかりません(DXACreateBufferFromFile)\n");
    return false;
}

// PCMに変換
WAVEFORMATEX   wfxDecode;
ZeroMemory(&wfxDecode, sizeof(wfxDecode));

// 推奨フォーマット取得
wfxDecode.wFormatTag = WAVE_FORMAT_PCM;
if(0 != acmFormatSuggest(NULL, lpwfxFormat, &wfxDecode, sizeof(wfxDecode),
                         ACM_FORMATSUGGESTF_WFORMATTAG)) {
    OutputDebugString("*** Error - 推奨フォーマット取得失敗(DXACreateBufferFromFile)\n");
    return false;
}

// ACMストリームオープン
if(0 != acmStreamOpen(&acmWave.hACMStream, NULL, lpwfxFormat, &wfxDecode, NULL,
                      0, 0, ACM_STREAMOPENF_NONREALTIME)) {
    OutputDebugString("*** Error - ACMストリームオープン失敗(DXACreateBufferFromFile)\n");
    return false;
}

// デコードサイズ取得
DWORD   dwDecodeSize;
if(0 != acmStreamSize(acmWave.hACMStream, mmciChild.cksize, &dwDecodeSize, ACM_STREAMSIZEF_SOURCE)) {
    OutputDebugString("*** Error - デコードサイズ取得失敗(DXACreateBufferFromFile)\n");
    return false;
}

// 変換元イメージ読み込み
acmWave.lpbyImage = new BYTE[mmciChild.cksize];
if(mmciChild.cksize != (DWORD)mmioRead(acmWave.hmioFile, (HPSTR)acmWave.lpbyImage, mmciChild.cksize)) {
    OutputDebugString("*** Error - 変換元イメージ読み込み失敗(DXACreateBufferFromFile)\n");
    return false;
}

// バッファ生成
if(false == DXACreateBuffer(dxaBuf, &wfxDecode, dwDecodeSize)) {
    OutputDebugString("*** Error - バッファ生成失敗(DXACreateBufferFromFile)\n");
    return false;
}

// バッファロック
LPVOID    lpDSBMem [2];
DWORD     dwDSBSize[2];
if(DS_OK != g_lpDSBuf8[dxaBuf]->Lock(0, 0, &lpDSBMem[0], &dwDSBSize[0], &lpDSBMem[1], &dwDSBSize[1],
                                     DSBLOCK_ENTIREBUFFER)) {
    OutputDebugString("*** Error - バッファロック失敗(DXACreateBufferFromFile)\n");
    DXAReleaseBufer(dxaBuf);
    return false;
}

// ACMストリームヘッダ設定
ACMSTREAMHEADER   acmsh;
ZeroMemory(&acmsh, sizeof(acmsh));
acmsh.cbStruct    = sizeof(acmsh);
acmsh.fdwStatus   = 0;
acmsh.pbSrc       = acmWave.lpbyImage;
acmsh.cbSrcLength = mmciChild.cksize;
acmsh.pbDst       = (LPBYTE)lpDSBMem[0];
acmsh.cbDstLength = dwDSBSize[0];
if(0 != acmStreamPrepareHeader(acmWave.hACMStream, &acmsh, 0)) {
    OutputDebugString("*** Error - ACMストリームヘッダ設定失敗(DXACreateBufferFromFile)\n");
    g_lpDSBuf8[dxaBuf]->Unlock(lpDSBMem[0], 0, lpDSBMem[1], 0);
    DXAReleaseBuffer(dxaBuf);
```

```
        return false;
    }

    // 変換
    const MMRESULT   mmr = acmStreamConvert(acmWave.hACMStream, &acmsh, 0);
    acmStreamUnprepareHeader(acmWave.hACMStream, &acmsh, 0);
    g_lpDSBuf8[dxaBuf]->Unlock(lpDSBMem[0], dwDSBSize[0], lpDSBMem[1], 0);
    if(0 != mmr) {
        OutputDebugString("*** Error - 変換失敗(DXACreateBufferFromFile)\n");
        DXAReleaseBuffer(dxaBuf);
        return false;
    }

    return true;
}
```