ゲームプログラミング

▶　DirectShow－第３回　マルチメディアファイルの再生と停止　◀

マルチメディアファイルの再生や停止は、簡単に行うことができます。

マルチメディアファイルの再生と停止

マルチメディアストリームオブジェクトに読み込まれているファイルの再生や停止は、SetStateメソッドで状態を設定することにより行います。このメソッドの引数に"STREAMSTATE_RUN"を指定すると再生状態に、"STREAMSTATE_STOP"を指定すると停止状態になります。また、現在の状態はGetStateメソッドで取得することができます。

```
IAMMultiMediaStream*   piAMMStream;      // マルチメディアストリームオブジェクト(初期化済みとする)

piAMMStream->SetState(STREAMSTATE_RUN);      // 再生
piAMMStream->SetState(STREAMSTATE_STOP);     // 停止

STREAM_STATE   stState;
piAMMStream->GetState(&stState);             // 状態取得
```

再生位置は、GetTimeメソッドで取得、Seekメソッドで任意の位置に設定することができます。

```
piAMMStream->Seek(100);          // 再生位置を100に設定

STREAM_TIME   stTime;
piAMMStream->GetTime(&stTime); // 再生位置取得
```

サーフェイスの更新

ファイルを再生状態に設定すると再生が始まります。音声データはDirectSoundへ出力されます。映像データはサーフェイスに転送されますが、サーフェイスの内容は自動的に更新されません。DirectDrawストリームサンプルオブジェクトのUpdateメソッドを呼び出したときに更新されます。このメソッドを呼び出すと、再生時間に適した映像データ取り出され、DirectShowに接続されているサーフェイスへ転送されます。なお、このメソッドは新しいデータに更新されるまで待機するので、映像のフレームレートが低い場合は、プログラム全体のフレームレートもそれに合わせて落ちてしまいます。

```
// サーフェイス更新
IDirectDrawStreamSample*   piDDSSample;   // 初期化済みのDirectDrawストリームサンプルオブジェクト
piDDSSample->Update(0, NULL, NULL, 0);
```

Updateメソッドで更新されるサーフェイスは、DirectShowに接続されているという点を除けば、ほかのオフスクリーンサーフェイスとまったく同じ性質を持っているので、BltメソッドやBltFastメソッドで転送したり、カラーキーを設定することができます。もちろん、ロストする場合もあります。

ループ再生

DirectShowは、DirectX Audioのような自動的にループする機能がないので、プログラムで実現する必要があります。ループするタイミングは、Updateメソッドの戻り値で判断します。戻り値がS_OKの場合は、新たなデータが取得できたことを表し、再生が継続していることを表します。S_OK以外の場合は、データの最後まで再生したかエラーであることを表し、再生が停止されていることになります。ループ再生を行う場合は、戻り値がS_OK以外のとき、再生位置を任意の位置まで戻し、再び再生を行います。

```
// ループ再生
IAMMultiMediaStream*       piAMMStream; // 初期化済みのマルチメディアストリームオブジェクト
IDirectDrawStreamSample*   piDDSSample; // 初期化済みのDirectDrawストリームサンプルオブジェクト

if(S_OK != piDDSSample->Update(0, NULL, NULL, 0)) {
    piAMMStream->Seek(0);                  //先頭まで戻す
    piAMMStream->SetState(STREAMSTATE_RUN);
}
```

○ 課題 ○

マルチメディアファイルの再生と停止を行う関数を作成しましょう。

(１)ループ回数を保持する変数－追加１－を適切な場所に追加しましょう。

－追加１－
```
static int   g_nLoop = 0;
```

(２)以下のプログラムは、マルチメディアファイルの再生を開始するDSPlay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。

DSPlay関数
－説明－
DSPlay関数は、マルチメディアファイルの再生を先頭から開始します。

－パラメータ－
const int nLoop…ループ回数。０は１回だけ再生(ループなし)。負の場合は無限ループ

－戻り値－
再生に成功した場合はtrue，それ以外はfalseを返します。

－追加２－
```
/*****************************************************************************************************/
/*                                          再　　生                                                 */
/*****************************************************************************************************/
bool DSPlay(const int nLoop)
{
    if(NULL == g_piAMMStream) {
        OutputDebugString("*** Error - マルチメディアストリーム未初期化(DSPlay)\n");
        return false;
    }

    g_nLoop = nLoop;    // ループ回数の保存

    g_piAMMStream->※ここは各自考えましょう;    // 再生位置を先頭に設定
    g_piAMMStream->※ここは各自考えましょう;    // 状態を再生に設定

    return true;
}
```

(３)以下のプログラムは、DirectShowに接続されているサーフェイスを更新するDSUpdateSurface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。

DSUpdateSurface関数
－説明－
DSUpdateSurface関数は、DirectShowに接続されているサーフェイスを新しいサンプルで更新します。また、ループ処理も行います。

－パラメータ－
なし

－戻り値－
更新に成功した場合はtrue，再生終了またはエラーの場合はfalseを返します。

－追加３－
```
/*****************************************************************************************************/
/*                                        サーフェイス更新                                           */
/*****************************************************************************************************/
bool DSUpdateSurface()
{
#ifdef _DEBUG
    if(NULL == g_piDDSSample) {
        OutputDebugString("*** Error - DirectDrawストリームサンプル未初期化(DSUpdateFrame)\n");
```

```
        return false;
    }
#endif

    // サンプル更新
    if(S_OK != g_piDDSSample->??????(0, NULL, NULL, 0)) {
        // ループカウンタ処理
        if(※ここは各自考えましょう)
            return false;   // 再生終了
        if(0 < g_nLoop)
            ※ここは各自考えましょう;

        DSPlay(g_nLoop);
        if(S_OK != g_piDDSSample->Update(0, NULL, NULL, 0)) {
            OutputDebugString("*** Error - ループ失敗(DSUpdateFrame)\n");
            return false;
        }
    }

    return true;
}
```

(４)以下のプログラムは、マルチメディアファイルの再生を停止するDSStop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。

DSStop関数
－説明－
DSStop関数は、マルチメディアファイルの再生を停止します。

－パラメータ－
なし

－戻り値－
なし

－追加４－

```
/*****************************************************************************************************/
/*                                          停　止                                                   */
/*****************************************************************************************************/
void DSStop()
{
    if(NULL == g_piAMMStream) {
        OutputDebugString("*** Error - マルチメディアストリーム未初期化(DSStop)\n");
        return;
    }

    g_piAMMStream->※ここは各自考えましょう;
}
```

(５)以下のプログラムは、マルチメディアファイルが停止された位置から再生を復帰するDSResume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。

DSResume関数
－説明－
DSResume関数は、マルチメディアファイルが停止された位置から再生を復帰します。

－パラメータ－
なし

－戻り値－
なし

－追加５－

```
/*****************************************************************************************************/
/*                                              復　帰                                               */
/*****************************************************************************************************/
void DSResume()
{
    if(NULL == g_piAMMStream) {
        OutputDebugString("*** Error - マルチメディアストリーム未初期化(DSResume)\n");
        return;
    }

    g_piAMMStream->SetState(※ここは各自考えましょう);
}
```

(６)以下のプログラムを入力し、ムービーが正しく再生、描画されることを確認しましょう。なお、このプログラムはGameFunc.cppのTest関数とTestProc関数に作成するものとします。

```
/*****************************************************************************************************/
/*                                            機能テスト                                             */
/*****************************************************************************************************/
void Test()
{
    DDLoadFromFile(DDS_OFFSCRN1, "BG.bmp");
    DDLoadFromFile(DDS_OFFSCRN2, "Chara.bmp");
    DDSetColorKey(DDS_OFFSCRN2, DDCKEY_SRCBLT, RGB(0, 0, 0));

    DSLoadFromFile("Opening.mpg");
    DSPlay(-1);

    GameFunc = TestProc;     // 初期化が終わったら、メインループ関数へ
}

/*****************************************************************************************************/
/*                                          機能テストメイン処理                                     */
/*****************************************************************************************************/
void TestProc()
{
    static POINT   ptSakura   = {0, 0}; // さくらの座標
    static int     nSakuraPtn = 0;      // さくらアニメパターン

    // さくら移動
    BYTE   byKeyState[256];
    DIGetKeyboardState(byKeyState);
    if(0 != (byKeyState[DIK_LEFT]  & 0x80))
        ptSakura.x--;
    if(0 != (byKeyState[DIK_RIGHT] & 0x80))
        ptSakura.x++;
    if(0 != (byKeyState[DIK_UP]    & 0x80))
        ptSakura.y--;
    if(0 != (byKeyState[DIK_DOWN]  & 0x80))
        ptSakura.y++;

    // バックバッファクリア
    DDColorFill(DDS_BACKBUF, NULL, RGB(0, 0, 0));

    // ムービー転送
    DSUpdateSurface();
    RECT   rcDest = {0, 48, 640, 432};
    DDBlt(DDS_BACKBUF, &rcDest, DDS_DIRECTSHOW, NULL, DDBLT_WAIT);

    // キャラクター転送
    RECT   rcChara;
    rcChara.left   = nSakuraPtn * 120;
    rcChara.top    = 0;
    rcChara.right  = rcChara.left + 120;
    rcChara.bottom = rcChara.top  + 120;
    DDBltFast(DDS_BACKBUF, ptSakura.x, ptSakura.y,
              DDS_OFFSCRN2, &rcChara, DDBLTFAST_SRCCOLORKEY | DDBLTFAST_WAIT);
    nSakuraPtn = (nSakuraPtn + 1) % 5;

    DDFlip();
}
```