実時間処理を行う

DirectShowを使ったリアルタイム処理

本章では，DirectShowを使ったリアルタイムの処理について紹介します．第6章で紹介したように，DirectShowのキャプチャ・グラフとサンプル・グラバを組み合わせれば，ビデオ・デバイスからの画像の取り出しが可能になります．このことにより，ビデオからの画像を通常のA-D変換などによる実時間の計測と同じように利用することが可能になります．

8-1　複数のカメラからのキャプチャ

Windows XPなどの最近のバージョンではマルチメディア機能が強化されていて，ある程度のハードウェアのスペックをもつパソコンでは，複数台のカメラの接続が可能になりました．

図8-1は，CPUクロック1.5GHzのCeleron Mプロセッサ，USBとIEEE1394インターフェースをもつノート・パソコンに，カード型IEEE1394インターフェースを1台増設し，2台のDVビデオ・カメラ，2台のUSBカメラを接続した場合の「マイコンピュータ」のようすです．**図8-2**は，ビデオ関連のハードの状態を「デバイスマネージャ」で見たようすです．このパソコンの場合，この状態で「マイコンピュータ」から4台のビデオを同時にプレビューすることができます．ですから，プログラムから複数台のカメ

図8-1　4台のカメラを接続したパソコンの「マイコンピュータ」
4台のカメラが認識されていることがわかる．

図8-2　4台のカメラを接続したパソコンの「デバイスマネージャ」

ラからのキャプチャも可能なことが予想できます．

図 8-1，**図 8-2** で使用したパソコンの場合，DV の系統は一つのカメラについて一つのホスト・コントローラが使用され，USB の系統は二つのカメラを一つのホスト・コントローラがまかなっていることになります．パソコンに使用される USB ホスト・コントローラによっては，**図 8-3** に示すように二つ目の USB カメラが使用できない場合もあります．このような場合は，ホスト・コントローラを増設する必要があります．また，DV の系統は 2 台のカメラを 1 台のホスト・コントローラで同時に使用すると，一応 2 台のプレビューはできますが，モザイク・ノイズが現れたりして処理が追いつかないようです．

このあたりの能力はさまざまな要因がからむので一概にはいえませんが，数台のカメラ程度ならホスト・コントローラさえ確保しておけば，一般的なパソコンの性能でも大丈夫なようです．

● 複数キャプチャのプログラミング

前述したように「マイコンピュータ」からのプレビューで同時プレビューが可能ならば，DirectShow での複数のカメラからのキャプチャを行うことができます．考え方としては，**図 8-4** に示すように複数のカメラを使用する場合でも同時にコントロールすればよい場合は，フィルタ・グラフ・ビルダは一つだけ用意して，キャプチャ・グラフをカメラ分だけ用意します．つまり，フィルタ・グラフ・ビルダは，複数

図 8-3　2 台の USB カメラの同時プレビューができない場合
使用するチップやシステムによっては複数台のカメラの認識，個々のプレビューができても同時プレビューができない場合がある．

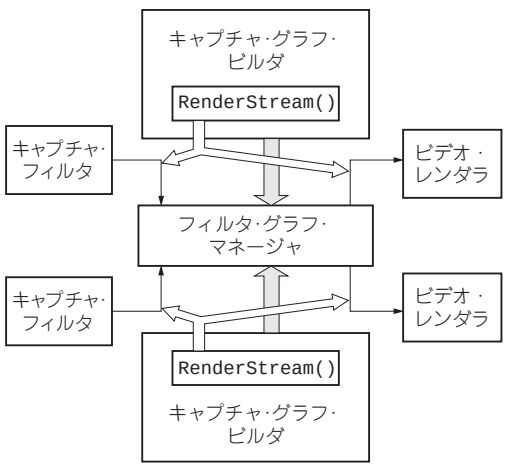

図 8-4　複数のキャプチャを行う場合のフィルタ・グラフ

図 8-5　複数キャプチャを実行したようす

リスト 8-1　複数キャプチャ

list502.h，list502b.cpp を一緒にビルドすること．

```
//
// list801.cpp　複数のカメラを使ったキャプチャ ver1.0
//  ▶ビデオ・デバイスからのキャプチャのみ
//  ▶［プロジェクト］［リンク］に strmiids.lib を追加
//  ▶COMの操作はすべてHRESULTを返すが，プログラムを簡単にするため
//    最低限の吟味しか行っていない．
//  ▶最大カメラ数は10を想定
//
#include <windows.h>
#include <dshow.h>
#include <stdio.h>
#include <conio.h>

void main ()
{
        int i;
        ULONG cFetched;

        CoInitialize (NULL) ;                          ← COMの初期化

        ---- キャプチャ・フィルタの準備 ----
        キャプチャ・デバイスを探す

        ICreateDevEnum * pDevEnum = NULL;
        IEnumMoniker * pClassEnum = NULL;
        IBaseFilter  * pbf [10] ;
        IMoniker  * pMoniker [10] ;                    ← 最大カメラ数10

        デバイス列挙子を作成
        CoCreateInstance ( CLSID_SystemDeviceEnum, NULL, CLSCTX_INPROC,
                           IID_ICreateDevEnum, (void ** ) &pDevEnum) ;

        ビデオ・キャプチャ・デバイス列挙子を作成
        pDevEnum -> CreateClassEnumerator ( CLSID_VideoInputDeviceCategory,
                                            &pClassEnum, 0) ;
        if  ( pClassEnum == NULL ) {
                printf ("ビデオキャプチャデバイスは存在しません¥n") ;
                pDevEnum -> Release () ;
                CoUninitialize () ;
                return ;
        }

        ビデオ・キャプチャ・デバイスのオブジェクトの
        インターフェースを最大10個得る
        pClassEnum -> Next (10, pMoniker, &cFetched) ;
        printf ("カメラ数 %d 個¥n", cFetched ) ;
        for ( i = 0; i < cFetched ; i++) {
                pMoniker [ i ]  -> BindToObject ( 0, 0, IID_IBaseFilter, (void**) &pbf [i]  ) ;
                pMoniker [ i ]  -> Release () ;
                printf ( "%d 個目のIBaseFilter取得 ¥n", i+1 ) ;
        }
        pDevEnum -> Release () ;
        pClassEnum -> Release () ;

        ---- フィルタ・グラフの準備 ----
        フィルタ・グラフを作り，インターフェースを得る．
        複数のカメラを一括制御ならフィルタ・グラフは一つでよい
        IGraphBuilder * pGraph = NULL;
        IMediaControl * pMC = NULL;
        CoCreateInstance ( CLSID_FilterGraph, NULL, CLSCTX_INPROC,
                           IID_IGraphBuilder, (void **) &pGraph) ;
        pGraph -> QueryInterface ( IID_IMediaControl, (LPVOID *) &pMC ) ;

        カメラ数分のキャプチャ・フィルタをフィルタ・グラフに追加
        for ( i = 0 ; i < cFetched ; i++) {
```