ゲームプログラミング

▶　DirectInput－第５回　デバイスバッファ　◀

DirectInputでは、軸やボタンが「押された」「離された」瞬間に、これらのイベントに関する詳細な情報がデバイスバッファという領域に格納されます。デバイスバッファは、格闘ゲームのような複雑なコマンド解析などに利用することができます。

デバイスバッファ

デバイスオブジェクトはデバイスバッファという領域を持っています。デバイスバッファには、軸やボタンが押されたり離されたりした瞬間に、その詳細な情報が格納されます。この情報はDIDEVICEOBJECTDATA構造体に取得することができます。この構造体には、以下のような情報が格納されます。

```
struct DIDEVICEOBJECTDATA {
    DWORD    dwOfs;        // イベント発生源のデバイス定数
    DWORD    dwData;       // デバイスから取得したデータ(イベントの種類)
    DWORD    dwTimeStamp;  // イベントが発生したミリ秒単位のシステム時間
    DWORD    dwSequence;   // イベントのシーケンス番号。イベントは発生順に番号が割り当てられます
    UINT_PTR uAppData;     // アプリケーション定義のアクション値。通常は使用しない
};
```

デバイスバッファの情報を解析することにより、GetDeviceStateメソッドでは調べることができない「押された」「離された」というイベントを判定したり、「上上下下左右左右ＢＡ」といった複雑なコマンドの判定を行うことができます。

デバイスバッファの設定

デバイスバッファのサイズは、デフォルトで０に設定されており、イベント情報を格納することができません。デバイスオブジェクトのSetPropertyメソッドを呼び出し、適切なサイズに変更します。

SetPropertyメソッド
－説明－
SetPropertyメソッドは、デバイスの動作を定義するプロパティを設定します。プロパティには、デバイスバッファのサイズ、軸モード、軸の範囲などがあります。

－パラメータ－
１つ目の引数は、設定するプロパティを識別するGUIDです。おもに以下のものを使用。

DIPROP_BUFFERSIZE	デバイスバッファのサイズを設定
DIPROP_RANGE	軸が報告可能な値の範囲を設定
DIPROP_DEADZONE	軸のデッドゾーンを設定
DIPROP_AXISMODE	軸モードを設定

２つ目の引数は、プロパティの設定情報を格納する構造体に含まれるDIPROPHEADER構造体のアドレスです。

－戻り値－
成功した場合はDI_OKまたはDI_PROPNOEFFECT、それ以外はエラーの原因をエラーコードで返します。

デバイスバッファの設定を行う場合は、DIPROPDWORD構造体にバッファの設定情報を格納します。この構造体は、DIPROPHEADER構造体型のdiphメンバと、DWORD型のdwDataメンバで構成されています。dwDataはデバイスバッファの数を設定し、diphメンバは以下のように初期化します。

メンバ	設定する値
dwSize	構造体全体のサイズ。sizeof(DIPROPDWORD)
dwHeaderSize	DIPROPHEADER構造体のサイズ。sizeof(DIPROPHEADER)
dwObj	dwHowメンバがDIPH_DEVICEのときは０
dwHow	デバイスバッファの設定のときはDIPH_DEVICE

```
// バッファサイズ設定
LPDIRECTINPUTDEVICE8    lpDIDevice8; // デバイスオブジェクト(初期化済みとする)
```

```
DIPROPDWORD   dipd;
ZeroMemory(&dipd, sizeof(dipd));
dipd.diph.dwSize       = sizeof(dipd);
dipd.diph.dwHeaderSize = sizeof(dipd.diph);
dipd.diph.dwObj        = 0;
dipd.diph.dwHow        = DIPH_DEVICE;
dipd.dwData            = 16;   // バッファの数
lpDIDevice8->SetProperty(DIPROP_BUFFERSIZE, &dipd.diph);
```

デバイスバッファの取得

デバイスバッファに格納された情報の取得は、デバイスオブジェクトのGetDeviceDataメソッドで行います。

GetDeviceDataメソッド
－説明－
GetDeviceDataメソッドは、デバイスに割りあてられたバッファから情報を取得します。

－パラメータ－
１つ目の引数は、デバイスバッファの情報を格納するDIDEVICEOBJECTDATA構造体のサイズ(バイト数)です。sizeof演算子で求めます。

２つ目の引数は、取得したデバイスバッファの情報を格納する領域です。DIDEVICEOBJECTDATA構造体配列の先頭アドレスを指定します。

３つ目の引数は、DWORD型の変数のアドレスです。この変数には、２つ目の引数で指定した配列の要素数を代入しておきます。メソッド呼び出し後、実際に取得した要素数に書き換えられます。

４つ目の引数は、データを取得する方法を示すフラグです。通常は０を指定します。

－戻り値－
成功した場合はDI_OKまたはDI_BUFFEROVERFLOW、それ以外はエラーの原因をエラーコードで返します。

```
// バッファ情報の取得
LPDIRECTINPUTDEVICE8   lpDIDevice8;     // デバイスオブジェクト(初期化済みとする)
DIDEVICEOBJECTDATA     diDOData[16];    // バッファ情報を格納する領域
DWORD                  dwItems = 16;    // 配列の個数
lpDIDevice8->GetDeviceData(sizeof(DIDEVICEOBJECTDATA), diDOData, &dwItems, 0);
```

上記のようにすると、DIDEVICEOBJECTDATA構造体のdwOfsメンバにイベントが発生したデバイス定数が格納され、dwDataメンバにイベントの種類が格納されます。dwDataメンバはDWORD型で32ビットありますが、「押された」「離された」というイベントの場合は、下位から数えて８ビット目に格納されます。このビットが１なら「押された」、０なら「離された」ことを表します。たとえば、Enterキーで「押された」イベントが発生したかを判定するには、以下のようになります。

```
DWORD   dwItems = lpDIDevice8->GetDeviceData(sizeof(DIDEVICEOBJECTDATA), diDOData, &dwItems, 0);
for(DWORD i = 0; i < dwItems; i++) {
    if(DIK_RETURN == diDOData[i].dwOfs && 0 != (diDOData[i].dwData & 0x80))
        // 「Enterキーが押された」イベントが発生
}
```

「離された」イベントを判定する場合は、以下のようになります。

```
if(DIK_RETURN == diDOData[i].dwOfs && 0 == (diDOData[i].dwData & 0x80))
    // Enterキーが離されたイベントが発生
```

バッファ情報の消去

デバイスバッファに格納された情報は、GetDeviceDataメソッドが呼び出されるまで消去されることはありませんが、メソッドに渡す引数によっては、バッファ内に格納されている情報をすべて消去することができます。消去するには、１つ目の引数は取得のときと同じにし、２つ目の引数はNULL、３つ目の引数は"INFINITE"を代入したDWORD型の変数のアドレス、４つ目の引数は０を指定します。

```
// バッファ情報の消去
LPDIRECTINPUTDEVICE8   lpDIDevice8;          // デバイスオブジェクト(初期化済みとする)
DWORD                  dwItems = INFINITE;
lpDIDevice8->GetDeviceData(sizeof(DIDEVICEOBJECTDATA), NULL, &dwItems, 0);
```

○ 課題 ○

キーボードとマウスにデバイスバッファを設定し、バッファ情報を取得する関数を作成しましょう。

(１)DIUtils.hに、デバイスバッファのサイズを定数として定義します。

－追加１－

```
// バッファサイズ
enum {
    DID_BUFSIZE = 16
};
```

(２)キーボードとマウスの初期化時に、SetPropertyメソッドでバッファ数を設定する処理を追加します。なお、バッファサイズはDID_BUFSIZEとします。

(３)以下のプログラムは、デバイスバッファから情報を取得するDIGetDeviceBuffer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。

DIGetDeviceBuffer関数

－説明－

DIGetDeviceBuffer関数は、指定されたデバイスのバッファに格納された情報を取得し、指定された領域に格納します。

－パラメータ－

const DIDEV did…バッファ情報を取得するデバイス。DIDEV列挙型を指定
DIDEVICEOBJECTDATA diDOData[]…バッファ情報を格納するDIDEVICEOBJECTDATA構造体配列
const DWORD dwElements…２つ目の引数で指定した配列の個数

－戻り値－

バッファから取得したイベントの個数を返す。

－追加２－

```
/*****************************************************************************************************/
/*                                          バッファ取得                                             */
/*****************************************************************************************************/
DWORD DIGetDeviceBuffer(const DIDEV did, DIDEVICEOBJECTDATA diDOData[], const DWORD dwElements)
{
    ZeroMemory(diDOData, sizeof(DIDEVICEOBJECTDATA) * dwElements);  // 配列をゼロクリア

    if(NULL == g_lpDIDevice8[did]) {
        OutputDebugString("*** Error - デバイス未初期化(DIGetDeviceBuffer)\n");
        return 0;
    }

    DWORD   dwItems = dwElements;
    const HRESULT   hr = ※ここは各自考えましょう(ヒント：デバイスバッファから情報を取得します);
    if(FAILED(hr)) {
        if(DIERR_INPUTLOST == hr)
            g_lpDIDevice8[did]->Acquire();  // アクセス権の再取得
        OutputDebugString("*** Error - バッファ取得失敗(DIGetDeviceBuffer)\n");
        return 0;
    }

    return dwItems;
}
```

(４)以下のプログラムは、デバイスバッファに格納された情報を消去するDIFlushDeviceBuffer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。

DIFlushDeviceBuffer関数

－説明－

DIFlushDeviceBuffer関数は、指定されたデバイスのバッファに格納された情報をすべて消去します。

－パラメータ－
　const DIDEV did…バッファ情報を消去するデバイス。DIDEV列挙型を指定

－戻り値－
　なし

－追加３－
```
/*****************************************************************************************************/
/*                                          バッファ消去                                             */
/*****************************************************************************************************/
void DIFlushDeviceBuffer(const DIDEV did)
{
    if(NULL == g_lpDIDevice8[did]) {
        OutputDebugString("*** Error - デバイス未初期化(DIFlushDeviceBuffer)\n");
        return;
    }

    DWORD   dwItems = ????????;
    if(DIERR_INPUTLOST == ※ここは各自考えましょう(ヒント：デバイスバッファの情報を消去します))
        g_lpDIDevice8[did]->Acquire();
}
```

(４)以下のプログラムを入力し、デバイスバッファから正しく情報が取得でき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));

    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

    DIDEVICEOBJECTDATA   diDevBuf[DID_BUFSIZE];
    for(DWORD i = 0; i < DIGetDeviceBuffer(DID_KEYBOARD, diDevBuf, DID_BUFSIZE); i++) {
        if(0 != (diDevBuf[i].dwData & 0x80)) {
            if(DIK_LEFT  == diDevBuf[i].dwOfs)
                ptSakura.x -= 2;    // '←'キーが押された場合、左に移動
        } else {
            if(DIK_RIGHT == diDevBuf[i].dwOfs)
                ptSakura.x += 4;    // '→'キーが離された場合、右に移動
        }
    }

    // 背景転送
    DDBltFast(DDS_BACKBUF, 0, 0, DDS_OFFSCRN1, NULL, DDBLTFAST_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);

    DDFlip();

    nSakuraPtn = (nSakuraPtn + 1) % 5;
```

```
    Sleep(15);  // フレーム制御関数。適当なものに変更してください
}
```

(５)(４)のプログラムを変更し、マウスの左ボタンが押されたらキャラクターを左に移動、マウスの右ボタンが離されたら右に移動するようにしましょう。