オブジェクト指向とゲームプログラミング

▶ DirectInput編 ― 第６回　ゲームパッドの初期化 ◀

ゲームパッドの初期化

ゲームパッドは、キーボードやマウスと同じ手順で初期化することができます。しかし、ゲームパッドはそれらに比べ多機能であり、さらに「GUID_SysKeyboard」のような定義済みGUIDが使用できないため、いくつかの作業が必要になります。ゲームパッドの初期化は、以下のような手順になります。

①デバイス(ゲームパッド)の列挙
②DirectInputDevice8オブジェクトの生成
③データフォーマットの設定
④協調レベルの設定
⑤バッファサイズの設定
⑥軸の列挙
⑦軸の値の範囲設定
⑧デッドゾーンの設定
⑨アクセス権の取得

①デバイスの列挙
最初にIDirectInput8::EnumDevicesメソッドを呼び出し、ゲームパッドを列挙します。キーボードやマウスを複数使用するときも、このメソッドで列挙します。

IDirectInput8::EnumDevicesメソッド
―説明―
EnumDevicesメソッドは、条件を満たすデバイスを列挙します。

―書式―
```
HRESULT EnumDevices(DWORD dwDevType, LPDIENUMDEVICESCALLBACK lpCallback,
                    LPVOID pvRef, DWORD dwFlags);
```

―パラメータ―
１つ目の引数(dwDevType)は、列挙するデバイスのタイプです。おもに以下の値を使用します。

DI8DEVCLASS_GAMECTRL	ゲームコントローラ
DI8DEVCLASS_KEYBOARD	キーボード
DI8DEVCLASS_POINTER	マウスやトラックボールのようなポインタデバイス
DI8DEVCLASS_DEVICE	以上のいずれにも該当しないデバイス
DI8DEVCLASS_ALL	すべてのデバイス

２つ目の引数(lpCallback)は、条件を満たすデバイスが見つかるたびに呼び出される関数(このような関数をコールバック関数と呼びます)のアドレスです。

３つ目の引数(pvRef)は、コールバック関数に渡すLPVOID型の値です。コールバック関数が呼び出されるたびに、ここで指定した値が渡されます。

４つ目の引数(dwFlags)は、列挙の範囲を指定するフラグです。おもに以下のフラグを指定します。

DIEDFL_ALLDEVICES	すべてのインストール済みデバイス
DIEDFL_ATTACHEDONLY	インストール済みで正しく接続されているデバイス
DIEDFL_FORCEFEEDBACK	フォースフィードバック対応デバイス

―戻り値―
成功した場合はDI_OK、それ以外はエラーコードを返します。

```
// ゲームパッドの列挙(pDinputは、初期化済みのDirectInput8オブジェクト)
pDInput->EnumDevices(DI8DEVCLASS_GAMECTRL, (LPDIENUMDEVICESCALLBACK)DIEnumGamePadProc,
                     NULL, DIEDFL_ATTACHEDONLY);
```

２つ目の引数で指定したコールバック関数は、１つ目と４つ目の引数で指定される条件を満たすデバイスの数と同じ回数呼び出されます。この関数は、DirectInputが必要に応じて呼び出す関数で、プログラマが呼び出す必要はないものの、プログラマが定義しなければならない関数です。
関数が呼び出されると、１つ目の引数にデバイスの情報を格納したDIDEVICEINSTANCE構造体のアドレス、２つ目の引数にEnumDevicesメソッドで指定した３つ目の引数と同じ値が渡されます。戻り値は、

列挙を継続する場合はDIENUM_CONTINUE、列挙を停止する場合はDIENUM_STOPを返します。

②DirectInputDevice8オブジェクトの生成
列挙されたゲームパッドのDirectInputDevice8オブジェクトを生成します。キーボードやマウスと同じように、IDirectInput8::CreateDeviceメソッドで生成します。このメソッドの１つ目の引数は、ゲームパッドの列挙時に取得されるデバイスのGUID(DIDEVICEINSTANCE構造体のguidInstanceメンバ)を指定します。

```
// DirectInputDevice8オブジェクトの生成
LPDIRECTINPUTDEVICE8   pDIDGamePad;     // ゲームパッドのデバイスオブジェクト

// ゲームパッド列挙コールバック関数
BOOL CALLBACK DIEnumGamePadProc(LPDIDEVICEINSTANCE pDIDInst, LPVOID pRef)
{
    // デバイスオブジェクト生成
    if(pDInput->CreateDevice(pDIDInst->guidInstance, &pDIDGamePad, NULL) != DI_OK)
        // 生成失敗

    return DIENUM_CONTINUE;     // 列挙継続
}
```

③データフォーマットの設定
データフォーマットの設定は、IDirectInputDevice8::SetDataFormatメソッドで行います。ゲームパッドはあらかじめ定義されている変数を指定することができます。取得したい情報にあわせて、「c_dfDIJoystick」(一般的なゲームパッド)または「c_dfDIJoystick2」(フォースフィードバック対応)のどちらかを指定します。

```
// データフォーマットの設定
pDIDGamePad->SetDataFormat(&c_dfDIJoystick);
```

④協調レベルの設定
協調レベルの設定は、IDirectInputDevice8::SetCooperativeLevelメソッドで行います。ゲームパッドは、フォアグランド、排他占有モードに設定します。このほかのモードでも不具合があるわけではありませんが、フォースフィードバックの機能を使用する場合は、排他占有モードを指定しなければなりません。

```
// 協調レベル設定(フォアグランド・排他占有モードに設定。hWndはウィンドウのハンドル)
pDIDGamePad->SetCooperativeLevel(hWnd, DISCL_FOREGROUND | DISCL_EXCLUSIVE);
```

⑤バッファサイズの設定
バッファサイズの設定は、キーボードやマウスとまったく同じ方法で設定できます。

⑥軸の列挙
ゲームパッドに存在する軸に対して、報告させる値の範囲やデッドゾーン(軸が中心にあるとみなされる範囲)の設定をします。ゲームパッドによって軸の数や性質が異なるため、IDirectInputDevice8::EnumObjectsメソッドで列挙しなければなりません。

IDirectInputDevice8::EnumDevicesメソッド

－説明－
EnumObjectsメソッドは、デバイス上で使用可能なオブジェクトを列挙します。

－書式－
```
HRESULT EnumObjects(LPDIENUMDEVICEOBJECTSCALLBACK lpCallback,
                    LPVOID pvRef, DWORD dwFlags);
```

－パラメータ－
１つ目の引数(lpCallback)は、指定したオブジェクトが見つかるたびに呼び出されるコールバック関数のアドレスです。

２つ目の引数(pvRef)は、コールバック関数に渡すLPVOID型の値です。コールバック関数が呼び出されるたびに、ここで指定した値が渡されます。

３つ目の引数(dwFlags)は、列挙するオブジェクトのタイプを指定するフラグです。おもに次のフラグを指定します。

DIDFT_ABSAXIS	絶対軸
DIDFT_RELAXIS	相対軸
DIDFT_AXIS	すべての軸
DIDFT_PSHBUTTON	プッシュボタン
DIDFT_TGLBUTTON	トグルボタン
DIDFT_BUTTON	すべてのボタン
DIDFT_POV	視点コントローラ
DIDFT_ALL	すべてのオブジェクト

－戻り値－
成功した場合はDI_OK、それ以外はエラーコードを返します。

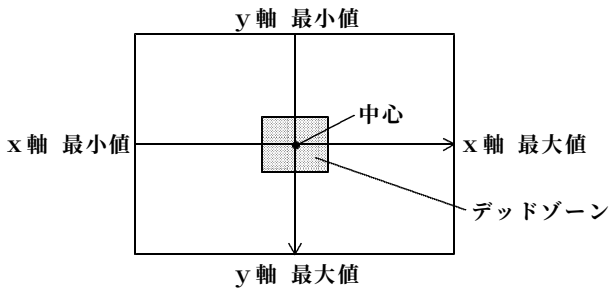
```
// すべての軸を列挙
pDIDGamePad->EnumObjects((LPDIENUMDEVICEOBJECTSCALLBACK)DIEnumGamePadAxesProc,
                         NULL, DIDFT_AXIS);
```

１つ目の引数で指定したコールバック関数は、３つ目の引数で指定したオブジェクトの数と同じ回数呼び出されます。この関数は、DirectInputが必要に応じて呼び出す関数で、プログラマが呼び出す必要はないものの、プログラマが定義しなければならない関数です。
この関数が呼び出されると、１つ目の引数にオブジェクトの情報を格納したDIDEVICEOBJECTINSTANCE構造体のアドレス、２つ目の引数にEnumDevicesメソッドで指定した３つ目の引数と同じ値が渡されます。戻り値は、列挙を継続する場合はDIENUM_CONTINUE、列挙を停止する場合はDIENUM_STOPを返します。

⑦軸の値の範囲の設定
すべての軸に対し、報告可能な値の範囲を設定します。ここで設定した値は、軸をもっとも傾けたときの値になります。ｘ軸の場合は、もっとも左に傾けたときに最小値、もっとも右に傾けたときに最大値となります。ｙ軸の場合は、もっとも上(前)に傾けたときに最小値、もっとも下(手前)に傾けたときに最大値となります。どの軸も中心位置の値は、最大値と最小値を足して２で割った値です。
軸の値の範囲の設定は、SetPropertyメソッドで行います。１つ目の引数に「DIPROP_RANGE」、２つ目の引数に値の範囲を格納したDIPROPRANGE構造体に含まれるDIPROPHEADER構造体のアドレスを指定します。この構造体は、DIPROPHEADER構造体型のdiphメンバと、軸の最小値を格納するlMinメンバ、軸の最大値を格納するlMaxメンバで構成されています。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	軸を列挙した場合は、DIDEVICEOBJECTINSTANCE構造体のdwTypeメンバ
dwHow	軸を列挙した場合は、DIPH_BYID

⑧デッドゾーンの設定
軸がアナログの場合はデッドゾーンを設定します。デッドゾーンとは、軸を傾けても傾けたとはみなさない範囲のことです。デフォルト状態でのアナログの軸は、中心にあるとみなされる範囲がかなり狭くなっています。デッドゾーンを設定しないと軸が中心にあるつもりでも傾いているとみなされてしまうことがあります。
デッドゾーンの設定は、SetPropertyメソッドで行います。１つ目の引数に「DIPROP_DEADZONE」、２つ目の引数に値の範囲を格納したDIPROPDWORD構造体に含まれるDIPROPHEADER構造体のアドレスを指定します。この構造体は、DIPROPHEADER構造体型のdiphメンバと、デッドゾーンの範囲を格納するdwDataメンバで構成されています。diphメンバの設定は、軸の値の範囲を設定する場合とまったく同じです。dwDataメンバは、軸を中心としたデッドゾーンの範囲を０～10,000の範囲で指定します。たとえば０ならデッドゾーンなし、5,000なら軸を中心とした50%の範囲がデッドゾーンとなります。



▲軸の範囲とデッドゾーンの関係

```
// ゲームパッド軸列挙コールバック関数
BOOL CALLBACK DIEnumGamePadAxesProc(LPCDIDEVICEOBJECTINSTANCE pDIDObjInst, LPVOID pRef)
{
    // 軸範囲設定
    DIPROPRANGE   dipr;
    dipr.diph.dwSize       = sizeof(dipr);
    dipr.diph.dwHeaderSize = sizeof(dipr.diph);
    dipr.diph.dwHow        = DIPH_BYID;
    dipr.diph.dwObj        = pDIDObjInst->dwType;
    dipr.lMin              = -32767;    // 最小値
    dipr.lMax              =  32767;    // 最大値
    pDIDGamePad->SetProperty(DIPROP_RANGE, &dipr.diph);

    // デッドゾーン設定
    DIPROPDWORD   dipdw;
    dipdw.diph.dwSize       = sizeof(dipdw);
    dipdw.diph.dwHeaderSize = sizeof(dipdw.diph);
    dipdw.diph.dwHow        = DIPH_BYID;
    dipdw.diph.dwObj        = pDIDObjInst->dwType;
    dipdw.dwData            = 3000;     // デッドゾーンの範囲
    pDIDGamePad->SetProperty(DIPROP_DEADZONE, &dipdw.diph);

    return DIENUM_CONTINUE;
}
```

⑨アクセス権の取得

最後にIDirectInputDevice8::Acquireメソッドを呼び出しアクセス権を取得します。アクセス権を取得すると、ゲームパッドから情報を取得できるようになります。

```
// アクセス権取得
pDIDGamePad->Acquire();
```

課　題

CDInput8クラスに、ゲームパッドの初期化を行う機能を追加しましょう。

(１)ゲームパッドをカプセル化したクラスを作成します。以下のプログラムを適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                        ゲームパッドクラス定義                                       */
/*****************************************************************************************************/
class CGamePad : public ※ここは各自考えましょう {
public:
    CGamePad(IDirectInputDevice8* pDIDGamePad) : CInputDevice(pDIDGamePad) {}
    virtual ~CGamePad() {}
};
```

(２)ゲームパッドの定数を宣言します。以下のプログラムをInputDevice.hppの適切な場所に追加しましょう。

```
    DIDEV_AXIS_MIN      = -32767,  // 軸の最小値
    DIDEV_AXIS_MAX      =  32767,  // 軸の最大値
    DIDEV_AXIS_CENTER   = (DIDEV_AXIS_MIN + DIDEV_AXIS_MAX + 1) / 2,
    DIDEV_AXIS_DEADZONE =   2500   // デッドゾーン(25%)
```

(３)ゲームパッドは複数接続されることもあるので、配列で管理します。以下のメンバを適切な場所に追加しましょう。

```
    std::vector<IInputDevice*>   m_GamePad;      // ゲームパッド配列
```

(４)初期化されたゲームパッドをすべて解放する以下のReleaseGamePad関数を適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                        ゲームパッドデバイス解放                                     */
/*****************************************************************************************************/
void CDInput8::ReleaseGamePad()
{
    for(int i = 0; i < m_GamePad.size(); i++)
        ?????? m_GamePad[i];    // ゲームパッド解放
    m_GamePad.clear();
}
```

(５)コールバック関数のプロトタイプを宣言しましょう。

コールバック関数には、グローバル関数またはstaticな関数(クラス関数)でなければ指定することができません。staticでないメンバ関数を指定するとコンパイルエラーとなります。なぜかというと、staticでないメンバ関数は、引数にthisポインタが追加されるためです。グローバル関数やクラス関数は、プロトタイプで指定した引数のみ関数に渡されます。それに対し、staticでないメンバ関数は、暗黙的に引数にthisポインタが追加され、「thisコール」という形式で関数が呼び出されるため、引数の数と呼び出し形式が異なってしまいます。そのためコンパイルエラーになるのです。

しかし、コールバック関数にstaticでないメンバ関数を指定したい場合があります。上記の理由から直接指定することはできませんが、staticな関数とthisポインタを利用すれば、この問題を解決することができます。
まず、コールバック関数にstaticな関数を指定します。そして、DirectInputでは、コールバック関数に任意の値をひとつ渡すことができる、ということを利用して、この値にthisポインタを渡すのです。通常、staticな関数ではthisやstaticでないメンバにアクセスすることはできませんが、このようにすれば、staticな関数でもthisポインタを介してすべてのメンバにアクセスすることができます(staticな関数の中は、そのクラスに所属しているので、privateを含むすべてのメンバにアクセスできます)。thisポインタから任意の関数を呼び出せば、任意のstaticでない関数をコールバックできるというわけです。また、DirectInputではコールバック関数にプログラマ指定の値はひとつしか渡せませんが、構造体にまとめれば、いくつでも渡すことができます。

ゲームパッドの初期化に必要なコールバック関数は、ゲームパッドを列挙するときと、軸を列挙するときです。また、軸の設定を行う場合に複数の値が必要となるので、構造体にまとめてコールバック関数に渡します。以上をふまえると、以下のプロトタイプと構造体が必要になります。CDInput8クラスのprivateメンバに追加しましょう。

```
    // ゲームパッド列挙コールバック
    static BOOL CALLBACK EnumGamePadProc(LPDIDEVICEINSTANCE pDevInst, LPVOID pRef);
    // ゲームパッド初期化
    BOOL InitGamePad(const LPDIDEVICEINSTANCE pDevInst, LPVOID pRef);

    // ゲームパッド軸列挙コールバック
    static BOOL CALLBACK EnumGamePadAxesProc(LPDIDEVICEOBJECTINSTANCE pDevObjInst, LPVOID pRef);
    // ゲームパッド軸設定
    BOOL SetAxesProp(const LPDIDEVICEOBJECTINSTANCE pDevObjInst, LPVOID pRef);

    // ゲームパッド軸列挙引数構造体
    struct DIENUMAXESARGS {
        CDInput8*               pDInput;
        IDirectInputDevice8*    pDIDevice;
    };
```

(５)ゲームパッドの初期化を行う以下のプログラムの足りない部分を補い、適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                        ゲームパッドデバイス生成                                     */
/*****************************************************************************************************/
DWORD CDInput8::CreateGamePad(const DWORD inCount)
{
    if(m_pDInput == NULL) {
        ::OutputDebugString("*** Error - DirectInput未初期化(CDInput8_CreateGamePad)\n");
        return 0;
    }

    ReleaseGamePad();
```

```
    if(inCount == 0)
        return 0;

    m_GamePad.resize(inCount, NULL);

    // ゲームパッドを列挙し、デバイスオブジェクトを生成する
    m_pDInput->???????????(????????????????????, (LPDIENUMDEVICESCALLBACK)??????????????, this,
                        DIEDFL_ATTACHEDONLY);

    // 生成できたゲームパッドを数える
    int   i, create = 0;
    for(i = 0; i < m_GamePad.size(); i++) {
        if(m_GamePad[i] == NULL)
            break;
        create++;
    }

    // 足りない部分にNull Objectを設定
    for(i = create; i < inCount; i++)
        m_GamePad[i] = new CNullInputDevice();

    return create;
}

/*****************************************************************************************************/
/*                                    ゲームパッド列挙コールバック                                     */
/*****************************************************************************************************/
BOOL CALLBACK CDInput8::EnumGamePadProc(LPDIDEVICEINSTANCE pDevInst, LPVOID pRef)
{
    return ((CDInput8*)pRef)->InitGamePad(pDevInst, pRef);
}

/*****************************************************************************************************/
/*                                      ゲームパッド初期化                                           */
/*****************************************************************************************************/
BOOL CDInput8::InitGamePad(const LPDIDEVICEINSTANCE pDevInst, LPVOID pRef)
{
#ifdef _DEBUG
    if(m_pDInput == NULL) {
        ::OutputDebugString("*** Error - DirectInput未初期化(CDInput8_InitGamePad)\n");
        return DIENUM_STOP;
    }
#endif

    // デバイスオブジェクト作成
    IDirectInputDevice8*   pDIDevice;
    if(m_pDInput->???????????(pDevInst->???????????, &pDIDevice, NULL) != DI_OK) {
        ::OutputDebugString("*** Error - ゲームパッドオブジェクト作成失敗(CDInput8_InitGamePad)\n");
        return DIENUM_CONTINUE;
    }

    // データフォーマット設定
    if(pDIDevice->???????????(&c_dfDIJoystick) != DI_OK) {
        ::OutputDebugString("*** Error - データフォーマット設定失敗(CDInput8_InitGamePad)\n");
        pDIDevice->Release();
        return DIENUM_CONTINUE;
    }

    // 協調レベル設定
    if(pDIDevice->??????????????????(m_hWnd, DISCL_FOREGROUND | DISCL_EXCLUSIVE) != DI_OK) {
        ::OutputDebugString("*** Error - 協調レベル設定失敗(CDInput8_InitGamePad)\n");
        pDIDevice->Release();
        return DIENUM_CONTINUE;
    }

    // バッファサイズ設定
    DIPROPDWORD   dipd;
    ::ZeroMemory(&dipd, sizeof(dipd));
    dipd.diph.dwSize       = sizeof(dipd);
    dipd.diph.dwHeaderSize = sizeof(dipd.diph);
    dipd.diph.dwObj        = ?;
    dipd.diph.dwHow        = ??????????;
    dipd.dwData            = DIDEV_BUFSIZE;
```

```
    if(FAILED(pDIDevice->??????????(????????????????, &dipd.diph))) {
        ::OutputDebugString("*** Error - バッファサイズ設定失敗(CDInput8_InitGamePad)\n");
        pDIDevice->Release();
        return DIENUM_CONTINUE;
    }

    // 軸を列挙し、プロパティを設定
    DIENUMAXESARGS   args = {this, pDIDevice};  // ２つの値を渡したいので、構造体にまとめる
    if(FAILED(pDIDevice->??????????((LPDIENUMDEVICEOBJECTSCALLBACK)????????????????, &args,
                                    DIDFT_AXIS))) {
        pDIDevice->Release();
        return DIENUM_CONTINUE;
    }

    // アクセス権取得
    pDIDevice->???????();

    // ゲームパッド配列へ追加
    for(int i = 0; i < m_GamePad.size(); i++) {
        if(m_GamePad[i] == NULL) {
            m_GamePad[i] = new CGamePad(pDIDevice);
            break;
        }
    }
    pDIDevice->Release();

    // 列挙終了判定(配列の要素すべてが初期化された場合は、列挙を終了する)
    if(m_GamePad[m_GamePad.size() - 1] != NULL)
        return DIENUM_STOP;

    return DIENUM_CONTINUE;
}

/*****************************************************************************************************/
/*                                    ゲームパッド軸列挙コールバック                                   */
/*****************************************************************************************************/
BOOL CALLBACK CDInput8::EnumGamePadAxesProc(LPDIDEVICEOBJECTINSTANCE pDevObjInst, LPVOID pRef)
{
    return ((DIENUMAXESARGS*)pRef)->pDInput->SetAxesProp(pDevObjInst, pRef);
}

/*****************************************************************************************************/
/*                                        軸プロパティ設定                                           */
/*****************************************************************************************************/
BOOL CDInput8::SetAxesProp(const LPDIDEVICEOBJECTINSTANCE pDevObjInst, LPVOID pRef)
{
    IDirectInputDevice8*   pDIDevice = ((DIENUMAXESARGS*)pRef)->pDIDevice;

#ifdef _DEBUG
    if(pDIDevice == NULL) {
        ::OutputDebugString("*** Error - ゲームパッド未初期化(CDInput8_SetAxesProp)\n");
        return DIENUM_STOP;
    }
#endif

    // 軸範囲設定(絶対軸のみ)
    if((pDevObjInst->dwType & DIDFT_ABSAXIS) != 0) {
        DIPROPRANGE   dipr;
        dipr.diph.dwSize       = sizeof(dipr);
        dipr.diph.dwHeaderSize = sizeof(dipr.diph);
        dipr.diph.dwHow        = DIPH_BYID;
        dipr.diph.dwObj        = pDevObjInst->dwType;
        dipr.lMin              = ※ここは各自考えましょう;
        dipr.lMax              = ※ここは各自考えましょう;
        if(FAILED(pDIDevice->??????????(???????????, &dipr.diph)))
            ::OutputDebugString("*** Error - 軸範囲設定失敗(CDInput8_SetAxesProp)\n");
    }

    // デッドゾーン設定
    DIPROPDWORD   dipdw;
    dipdw.diph.dwSize       = sizeof(dipdw);
    dipdw.diph.dwHeaderSize = sizeof(dipdw.diph);
    dipdw.diph.dwHow        = DIPH_BYID;
    dipdw.diph.dwObj        = pDevObjInst->dwType;
```

```
    dipdw.dwData            = ※ここは各自考えましょう;
    if(FAILED(pDIDevice->???????????(??????????????, &dipdw.diph)))
        ::OutputDebugString("*** Error - デッドゾーン設定失敗(CDInput8_SetAxesProp)\n");

    return DIENUM_CONTINUE;
}
```

※このプログラムを完成させても、すべての純粋仮想関数を実装していないので、コンパイルできません。実装は次回の課題で行います。

(６)以下のプログラムをCDInput8::Release関数の適切な場所に追加し、DirectInput解放時にゲームパッドが解放されるようにしましょう。

```
    ReleaseGamePad();       // ゲームパッド解放
```

(７)CDInput8クラスの適切な場所に、ゲームパッドを制御するためのインタフェースを取得する以下の関数を追加しましょう。

```
    IInputDevice* GetGamePad(int i) const { return m_GamePad.at(i); }
```

(８)以下のマクロを適切な場所に追加し、短い名前でゲームパッドが使用できるようにしましょう。

```
    inline IInputDevice* DIGamePad(int i) { return DInput().GetGamePad(i); }
```

(９)ゲームパッドを使用する場合、以下のプログラムをアプリケーションを初期化する関数の適切な場所に追加しましょう(CreateGamePad関数の引数は初期化するゲームパッドの個数です)。

```
    // ゲームパッド生成
    if(DInput().CreateGamePad(1) == 0)
        return false;
```

キーボード、マウス、ゲームパッドのひとつ以上あれば良いという場合のプログラムも考えてみましょう。