オブジェクト指向と
ゲームプログラミング

▷　フレームワーク編 － 第３回　ウィンドウクラスの作成　◁

ウィンドウ

ウィンドウは、アプリケーションがWindowsからの指示(メッセージ)やユーザの操作情報を受け取るための窓口の役割を持っています。そのため、ウィンドウがないプログラムは正しく動作しない場合があります。
ウィンドウの生成は、以下の手順で行います。

(１)ウィンドウクラスの定義
(２)ウィンドウクラスの登録
(３)ウィンドウの生成
(４)ウィンドウの表示

(１)ウィンドウクラスの定義
ウィンドウクラスとは、ウィンドウの基本的な形状と機能のことです。ウィンドウクラスの定義は、WNDCLASSEX構造体の各メンバへの代入で行います。特に重要なメンバはcbSize, hInstance, lpfnWndProcです。使用しないメンバは0やNULLを代入します。
cbSizeメンバは、WNDCLASSEX構造体のサイズを格納します。APIに渡す構造体の多くは、自分のサイズを格納するメンバがあり、sizeof演算子を使ってサイズを取得し、代入します。
hInstanceメンバは、生成されるウィンドウが所属するインスタンスハンドル(WinMain関数の１つ目の引数)を代入します。
lpfnWndProcメンバは、もっとも重要なメンバで、ウィンドウプロシージャの関数名を代入します。メッセージループにより取り出されたメッセージは、ここで指定した関数に送出されます。

(２)ウィンドウクラスの登録
WNDCLASSEX構造体に定義したウィンドウクラス情報はRegisterClassEx関数でWindowsに登録します。この関数にはWNDCLASSEX構造体のアドレスを渡します。

(３)ウィンドウの生成
ウィンドウクラスを登録すると､CreateWindowEx関数でそのウィンドウが生成できるようになります。ウィンドウの生成に成功した場合は、ウィンドウを管理するためのハンドルが返されます。失敗した場合はNULLが返されます。

(４)ウィンドウの表示
ウィンドウの生成が成功しても、メモリに存在しているだけであり、画面には表示されません。ウィンドウは、ShowWindow関数で表示モードを設定することで初めて表示されます。ウィンドウ表示後に、SetFocus関数を呼び出せば、ウィンドウがキーボードの入力を受け取れる状態になります。

課　　題

ゲームアプリケーションのウィンドウの形状や位置を管理するCGameWndFrameクラスを作成しましょう。

一般的にウィンドウクラスは、さまざまなアプリケーションに共通する部分をクラスまたはインタフェース化し、それを継承または集約して拡張するような設計にします。
今回は、テーマをゲームに絞るので、そのような設計にはせず、CGameWndFrameクラスにすべてを定義することにします。

たいていのウィンドウクラスでは、ウィンドウの形状や位置だけでなく、送出されてくるメッセージの処理まで行うような設計にします。さらに、ウィンドウクラスがメッセージの影響を受けるほかのクラスに対し、受信したメッセージを配信するように設計しますが、プログラムが若干複雑になります。そこで、CGameWndFrameクラスではウィンドウの形状や位置のみを管理し、メッセージはアプリケーション全体に関わることが多いので、CGameAppクラスで処理を行う設計にします。

(１)CGameWndFrameクラスのヘッダファイル(GameWndFrame.hpp)を以下のように作成しましょう。

－GameWndFrame.hpp－

```
/*
===========================================================================================
                        オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
===========================================================================================

  【対象ＯＳ】
     Microsoft Windows2000/XP

  【コンパイラ】
     Microsoft Visual C++ 2005

  【プログラム】
     GameWndFrame.hpp
                ゲームアプリケーション用ウィンドウフレームクラスヘッダ

  【履歴】
     * Version    1.00       2005/03/dd  hh:mm:ss

===========================================================================================
*/

#pragma ????

/*******************************************************************************************/
/*                                      インクルードファイル                               */
/*******************************************************************************************/
#include <※ここは各自考えましょう>

/*******************************************************************************************/
/*                              ゲームアプリケーションウィンドウフレームクラス定義         */
/*******************************************************************************************/
class CGameWndFrame {
public:

private:
    CGameWndFrame(const CGameWndFrame&);
    CGameWndFrame& operator=(const CGameWndFrame&);
};
```

(２)CGameWndFrameクラスのソースファイル(GameWndFrame.cpp)を以下のように作成しましょう。

－GameWndFrame.cpp－

```
/*
===========================================================================================
                        オブジェクト指向ゲームプログラミング
    Programmed by Hibikino software. Copyright (c) 2005 Hibikino software. All rights reserved.
===========================================================================================

  【対象ＯＳ】
     Microsoft Windows2000/XP

  【コンパイラ】
     Microsoft Visual C++ 2005

  【プログラム】
     GameWndFrame.cpp
                ゲームアプリケーションウィンドフレームウクラス

  【履歴】
     * Version    1.00       2004/04/dd  hh:mm:ss

===========================================================================================
*/

/*******************************************************************************************/
/*                                      インクルードファイル                               */
/*******************************************************************************************/
#include "※ここは各自考えましょう"
```

(３)CGameWndFrameクラスのコンストラクタとデストラクタを作成します。ヘッダファイルに、コンストラクタとデストラクタのプロトタイプを記述しましょう。なお、デストラクタは、念のため仮想デストラクタとして宣言しましょう。

(４)CGameWndFrameクラスのソースファイルに、コンストラクタとデストラクタを実装します。以下のプログラムの足りない部分を補って完成させましょう。

```
/*****************************************************************************************************/
/*                                          コンストラクタ                                           */
/*****************************************************************************************************/
????????????::CGameWndFrame()
{
}

/*****************************************************************************************************/
/*                                          デストラクタ                                             */
/*****************************************************************************************************/
????????????::~CGameWndFrame()
{
}
```

(５)CGameAppクラスに、アプリケーション名を追加します。

アプリケーション名は、アプリケーションのいろいろな場面で必要になることが多いので、CGameAppクラスに宣言します。
以下の変数をCGameAppクラスのpublicなメンバとして追加しましょう。

```
// 定数
static const TCHAR*    APP_NAME;          // アプリケーション名
```

さらに、以下のプログラムを適切な場所に追加しましょう。

```
/*****************************************************************************************************/
/*                                          静的変数定義                                             */
/*****************************************************************************************************/
const TCHAR*    CGameApp::APP_NAME = "オブジェクト指向ゲームプログラミング";
```

ヒント：C++編 第５回「静的メンバ」

(６)CGameWndFrameクラスに必要なメンバ変数を追加します。

CGameWndFrameクラスは、ウィンドウを管理するクラスなので、メンバ変数にウィンドウのハンドルが必要になります。
以下のプログラムをヘッダファイルの適切な場所に追加しましょう。

```
HWND    m_hWnd;     // ウィンドウのハンドル
```

(７)CGameWndFrameオブジェクトの生成時に、メンバ変数m_hWndが確実に初期化されるようにします。
以下のコンストラクタ初期化子を適切な場所に追加しましょう。

```
m_hWnd(NULL)
```

(８)CGameWndFrameクラスに、アクセス関数を追加します。

メンバ変数へのアクセスは、そのクラスおよびサブクラスだけに限定し、外部からの直接アクセスはできないようにします。しかし、そのクラスが保持している情報が外部で必要になる場合があります。そこで、メンバ変数が保持している情報を返すだけの関数や、メンバ変数に情報を設定するだけ関数を必要に応じて作成します。これらの関数のことを、アクセス関数と呼びます。
今回は、(６)で追加したウィンドウハンドルが外部で必要になる場合が考えられるので、ウィンドウハンドルを返すだけの関数を作成します。
以下のプログラムをヘッダファイルの適切な場所に追加しましょう。

```
// アクセス関数
HWND GetHWnd() const { return m_hWnd; }
```

(９)CGameWndFrameクラスに必要な機能を追加します。最低限必要なのは、ウィンドウを生成する機能とウィンドウを破棄する機能です。それぞれ、Create関数、Destroy関数とします。これらの関数のプロトタイプは、以下のようになります。ヘッダファイルの適切な場所に追加しましょう。

```
bool Create(const HINSTANCE hInstance, const WNDPROC inWndProc);
void Destroy();
```

Create関数では､インスタンスのハンドルとウィンドウプロシージャのアドレスを受け取っています。インスタンスのハンドルはWinMain関数に、ウィンドウプロシージャのアドレスはCGameAppクラスにあるため、そのままではCGameWndFrameクラスでは取得できません。そこで、引数で受け取るようにしています。

(10)CGameWndFrameクラスのソースファイルにCreate関数を実装します。以下のプログラムの足りない部分を補って完成させましょう。

```
/*****************************************************************************************************/
/*                                          ウィンドウ生成                                           */
/*****************************************************************************************************/
bool ?????????????::Create(const HINSTANCE hInstance, const WNDPROC inWndProc)
{
    Destroy();  // すでに作成済みかもしれないので、破棄しておく

    LPCTSTR   WindowClassName = "OOGAMEPROG_WINDOWCLASS";   // ウィンドウクラス名

    WNDCLASSEX   wcx;
    // ウィンドウクラスが登録済みか調べる
    if(::GetClassInfoEx(hInstance, WindowClassName, &wcx) == 0) {
        // ウィンドウクラス登録
        ::ZeroMemory(&wcx, sizeof(wcx));
        wcx.??????        = sizeof(wcx);
        wcx.style         = 0;
        wcx.???????????   = inWndProc;
        wcx.cbClsExtra    = 0;
        wcx.cbWndExtra    = 0;
        wcx.?????????     = hInstance;
        wcx.hIcon         = ::LoadIcon(NULL, IDI_APPLICATION);
        wcx.hCursor       = ::LoadCursor(NULL, IDC_ARROW);
        wcx.hbrBackground = (HBRUSH)::GetStockObject(BLACK_BRUSH);
        wcx.lpszClassName = WindowClassName;
        wcx.lpszMenuName  = NULL;
        wcx.hIconSm       = NULL;

        if(::???????????????(&wcx) == 0) {
            ::OutputDebugString("*** Error - ウィンドウ登録失敗(CGameWndFrame_Create)\n");
            return false;
        }
    }

    // ウィンドウスタイル設定
    const DWORD   WndStyle = WS_POPUPWINDOW | WS_CAPTION | WS_MINIMIZEBOX;

    // ウィンドウサイズ設定
    RECT   WndRect = {0, 0, 640, 480};
    ::AdjustWindowRectEx(&WndRect, WndStyle, FALSE, 0);
    const long   WndWidth  = WndRect.right  - WndRect.left;          // ウィンドウ幅
    const long   WndHeight = WndRect.bottom - WndRect.top;           // ウィンドウ高さ

    // ウィンドウ生成
    m_hWnd = ::?????????????(0, WindowClassName, CGameApp::APP_NAME, WndStyle,
                             0, 0, WndWidth, WndHeight, NULL, NULL, hInstance, NULL);
    if(m_hWnd == NULL) {
        ::OutputDebugString("*** Error - ウィンドウ生成失敗(CGameWndFrame_Create)\n");
        return false;
    }

    ::??????????(m_hWnd, SW_SHOW);       // ウィンドウ表示
    ::SetFocus(m_hWnd);                  // フォーカス設定

    return true;
}
```

※「CGameApp::APP_NAME」は、あるファイルをインクルードしないとコンパイルエラーになります。

(11)CGameWndFrameクラスのソースファイルにDestroy関数を実装します。以下のプログラムの足りない部分を補って完成させましょう。

```
/*****************************************************************************************************/
/*                                          ウィンドウ破棄                                           */
/*****************************************************************************************************/
void ?????????????::Destroy()
{
    // ウィンドウ破棄
    if(m_hWnd != NULL) {
        ::ShowWindow(m_hWnd, SW_HIDE);
        ::?????????????(m_hWnd);
        m_hWnd = NULL;
    }
}
```

(12)CGameWndFrameクラスのメンバは、以下のとおりです。

CGameWndFrame コンストラクタ

CGameWndFrameオブジェクトを構築します。

~CGameWndFrame デストラクタ

CGameWndFrameオブジェクトを解放します。

Create 生成

ウィンドウを生成します。

書式 `bool Create(const HINSTANCE hInstance, const WNDPROC inWndProc);`

Return	成功：true　それ以外：false
hInstance	アプリケーションインスタンスのハンドル
inWndProc	ウィンドウプロシージャ関数のアドレス

Destroy 破棄

ウィンドウを破棄します。

GetHWnd アクセス関数

現在保持しているウィンドウのハンドルを取得します。

書式 `HWND GetHWnd() const;`

Return	ウィンドウのハンドル

m_hWnd メンバ変数

CGameWndFrameオブジェクトが使用するウィンドウのハンドルです。

書式 `HWND m_hWnd;`