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５回　メッセージの処理　◁

課　　題

ゲームアプリケーションに必要なメッセージを処理しましょう。

(１)プログラムが正しく終了するようにしましょう。

ここまでのプログラムは、ウィンドウは消えるもののWM_QUITメッセージが発生しないため、メッセージループから抜けることができないので終了できませんでした。ほとんどのアプリケーションでは、メインウィンドウを破棄するとプログラムが終了するようになっています。

ウィンドウを破棄するとき、あるメッセージが発生します。プログラムを終了するには、ウィンドウプロシージャでこのメッセージを処理し、PostQuitMessage関数を呼び出す必要があります。この関数は、プログラムが終了しようとしていることをWindowsに知らせ、WM_QUITメッセージをウィンドウに送出します。引数は終了コードで、通常は０(正常終了)を指定します。
WM_QUITメッセージを受信すると、GetMessage関数が０を返すので、メッセージループを抜け、プログラムが終了するというわけです。

CGameAppクラスに、ウィンドウが破棄されるときに発生するメッセージを処理する関数を作成します。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。

```
LRESULT OnDestroy(const HWND hWnd, const WPARAM wParam, const LPARAM lParam);
```

(２)ソースファイルにOnDestroy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。

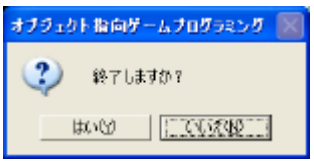
```
/*****************************************************************************************************/
/*                                        WM_DESTROYメッセージ処理                                   */
/*****************************************************************************************************/
LRESULT ????????::OnDestroy(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    ::??????????????(0);

    return 0;
}
```

(３)ウィンドウプロシージャCGameApp::WindowProc関数を以下のように変更しましょう。なお、足りない部分は各自補ってください。

```
LRESULT CGameApp::WindowProc(const HWND hWnd, const UINT uMsg, const WPARAM wParam, const LPARAM lParam)
{
    switch(uMsg) {
      case WM_???????:  return OnDestroy(hWnd, wParam, lParam);
    }
    return ::DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

(４)ウィンドウの閉じるボタンが押されたら、右のようなメッセージボックスを表示し、「はい」が選択されたらプログラムが終了するようにしましょう。



閉じるボタンが押されると、あるメッセージが発生します。まずは、このメッセージをリファレンスで調べてみましょう。メッセージボックス後の処理は、押されたボタンで分岐させます。「はい」ならアプリケーションの終了、「いいえ」なら何もしません。

CGameAppクラスに、閉じるボタンが押されたときに発生するメッセージを処理する関数を作成します。次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。

```
    LRESULT OnClose(const HWND hWnd, const WPARAM wParam, const LPARAM lParam);
```

(５)ソースファイルにOnClose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。

```
/*****************************************************************************************************/
/*                                        WM_CLOSEメッセージ処理                                       */
/*****************************************************************************************************/
LRESULT ????????::OnClose(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    if(::MessageBox(hWnd, "終了しますか？", CGameApp::APP_NAME, ※ここは各自考えましょう) == ?????)
        Release();

    return 0;
}
```

(６)ウィンドウプロシージャCGameApp::WindowProc関数を以下のように変更しましょう。なお、足りない部分は各自補ってください。

```
LRESULT CGameApp::WindowProc(const HWND hWnd, const UINT uMsg, const WPARAM wParam, const LPARAM lParam)
{
    switch(uMsg) {
      case WM_?????:    return OnClose  (hWnd, wParam, lParam);
      case WM_???????:  return OnDestroy(hWnd, wParam, lParam);
    }
    return ::DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

(７)Escキーを押したら、(４)で作成したメッセージボックスを表示し、「はい」が選択されたらプログラムが終了するようにしましょう。

キーボードのキーが押されると、あるメッセージが発生します。このとき、押されたキーを判定し、Escキーなら(４)～(６)で作成したプログラムを実行するようにします。

CGameAppクラスに、キーが押されたときに発生するメッセージを処理する関数を作成します。次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。

```
    LRESULT OnKeyDown(const HWND hWnd, const WPARAM wParam, const LPARAM lParam);
```

(８)ソースファイルにOnKeyDown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。

```
/*****************************************************************************************************/
/*                                       WM_KEYDOWNメッセージ処理                                      */
/*****************************************************************************************************/
LRESULT ????????::OnKeyDown(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    switch(??????) {
      case VK_??????:
        ::PostMessage(hWnd, WM_?????, 0, 0);    // WM_?????へ
        break;
    }

    return 0;
}
```

PostMessage関数は、指定されたメッセージをメッセージキューに投かんする関数です。Escキーが押されたとき、閉じるボタンを押したときと同じメッセージを発生させることにより、(４)～(６)で作成したプログラムを実行するようにしています。

(９)ウィンドウプロシージャCGameApp::WindowProc関数を次のように変更しましょう。なお、足りない部分は各自補ってください。

```
LRESULT CGameApp::WindowProc(const HWND hWnd, const UINT uMsg, const WPARAM wParam, const LPARAM lParam)
{
    switch(uMsg) {
      case WM_???????:  return OnKeyDown(hWnd, wParam, lParam);
      case WM_?????:    return OnClose  (hWnd, wParam, lParam);
      case WM_???????:  return OnDestroy(hWnd, wParam, lParam);
    }
    return ::DefWindowProc(hWnd, uMsg, wParam, lParam);
}
```

(10)アプリケーションの実行中に、スクリーンセーバが起動しないようにしましょう。

ゲームをプレイしているとき、ムービー鑑賞中など、一定時間キー入力がないために、スクリーンセーバが起動してしまうことがあります。スクリーンセーバは、スクリーンセーバが起動したときに発生するWM_SYSCOMMANDメッセージの処理で、１を返すことにより起動を阻止することができます。

以下のプログラムの足りない部分を補い、(１)から(９)を参考に、WM_SYSCOMMANDメッセージを処理するプログラムをCGameAppクラスに追加しましょう。

```
/*****************************************************************************************************/
/*                                    WM_SYSCOMMANDメッセージ処理                                    */
/*****************************************************************************************************/
LRESULT ????????::OnSysCommand(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    switch(wParam) {
      case SC_???????????:
        return ?;   // スクリーンセーバを起動しない
    }   // switch(wParam)

    return ::DefWindowProc(hWnd, WM_SYSCOMMAND, wParam, lParam);
}
```

(11)Altキーを押したときに、メニューを表示しないようにしましょう。

Altキーを押すと、プログラムが停止し、メニュー選択待ち状態になります。このような状態では、ゲームも停止してしまいます。
この状態を回避するには、WM_SYSCOMMANDメッセージが発生したときに、WPARAMを調べ、SC_KEYMENUのときに0を返します。

以下のプログラムを完成させ、適切な場所に追加しましょう。

```
    case SC_???????:
      return 0;     // Altでメニューを表示しない
```

(12)Altキー＋F4キーが押された場合に、アプリケーションを直ちに終了するようにしましょう。

Altキーを押しながらキーボードのキーを押すと、WM_SYSKEYDOWNメッセージが発生します。このメッセージが発生したら、Alt＋F4キーであるかを調べ、そうである場合は、アプリケーションを終了させる関数を呼び出します。

以下のプログラムの足りない部分を補い、(１)から(９)を参考に、WM_SYSKEYDOWNメッセージを処理するプログラムをCGameAppクラスに追加しましょう。

```
/*****************************************************************************************************/
/*                                    WM_SYSKEYDOWNメッセージ処理                                    */
/*****************************************************************************************************/
LRESULT ????????::OnSysKeyDown(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    switch(??????) {
      case VK_??:
        ???????();
        return 0;
    }

    return ::DefWindowProc(hWnd, WM_SYSKEYDOWN, wParam, lParam);
}
```

(13)IMEツールバーを消去しましょう。

ウィンドウを生成したとき、IMEツールバーが表示される場合があります。追加２のプログラムを実行すると、消すことができます。なお、Immで始まる関数を使用する場合は、ライブラリ"imm32.lib"が必要になります。

以下のプログラムを適切な場所に追加し、IMEツールバーを消去しましょう。

－追加１－

```
/*****************************************************************************************************/
/*                                          静的リンクライブラリ                                     */
/*****************************************************************************************************/
#pragma comment(lib, "imm32.lib")
```

－追加２－

```
    // IME消去
    HWND   hIMEWnd = ::ImmGetDefaultIMEWnd(m_hWnd);
    ::SendMessage(hIMEWnd, WM_IME_CONTROL, IMC_CLOSESTATUSWINDOW, 0);
```