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１２回　アクティブ状態と非アクティブ状態　◁

アクティブ状態と非アクティブ状態

Windowsアプリケーションには、アクティブ状態と非アクティブ状態の２つの状態があります。
アクティブ状態とは、おもに前面に表示されているアプリケーションで、キーボードやマウスの入力権(フォーカス)が与えられ、実行権が優先的に与えられる状態です。通常はタイトルバーが青色になっています。
これに対し非アクティブ状態とは、フォーカスが与えられず、入力ができない状態です。通常はタイトルバーが灰色になっています。
ゲームプログラムでは、２つの状態を考慮したプログラムを作成する必要があります。

非アクティブ状態での制限

非アクティブ状態では、ウィンドウが最小化されたり、ほかのウィンドウに隠されたりしている場合が多いので、画面の一部または全体が隠れてしまいます。このような状態でゲームを進めてしまうわけにはいきません。
また、DirectXを使ったアプリケーションでは、非アクティブ状態になるとさまざまな制限が課せられます。たとえば、DirectX Graphicsで画面を占有している場合、非アクティブ状態になると強制的に最小化され、画面はほかのアプリケーションに渡されます。この状態では、画面に描画することができなくなります。また、３Ｄデバイスが消滅状態になり、ビデオカードへの操作が行えなくなる場合があります。DirectInputでは、入力デバイスを完全に占有することができますが、非アクティブ状態になったときには、キーボードやマウスの占有を解いて、ほかのアプリケーションが使用できるようにする必要があります。占有を解くと、キーボードやマウスから入力データを取得することはできないので、ゲームの操作を行うことはできません。
このように、非アクティブ状態では制限が多くなります。ほとんどの場合、画面も入力デバイスも使用できないので、コンピュータの思考ルーチン処理など一部の処理だけを行うようにするか、ゲームを一時休止するような設計にしなければなりません。

状態の判別

アプリケーションがアクティブ状態なのか非アクティブ状態なのかを判別するには、いくつか方法がありますが、アプリケーションの状態が変更になったときに発生するメッセージを使う方法がもっとも簡単です。

アプリケーションの状態が変更になるたび、WM_ACTIVATEAPPメッセージが発生します。このとき、ウィンドウプロシージャの引数WPARAMには、アプリケーションがどの状態に変更されたかが格納されています。WPARAMの値が０なら非アクティブ状態、非０ならアクティブ状態に変更されたことを表します。このWPARAMの値を変数に保存しておき、ゲームの処理を行うときに役立てます。

```
bool    m_Active;     // アプリケーションの状態を格納する変数。trueならアクティブ状態

// ウィンドウプロシージャ
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch(uMsg) {
      case WM_ACTIVATEAPP:
        if(wParam != 0)
            m_Active = true;          // wParamが非０ならアクティブ状態
        else
            m_Active = false;         // wParamが０なら非アクティブ状態
        return 0;
            ⋮(以下省略)
```

ゲーム処理では、次のように、アプリケーション状態によって処理を分岐させます。

```
// ゲーム処理
if(m_Active == true)
    アクティブ状態のゲーム処理          // アクティブ状態なら通常のゲーム処理
else
    非アクティブ状態のゲーム処理        // 非アクティブ状態なら専用の処理。たとえば、
                                        // コンピュータの思考など一部の処理だけ行う
```

これらの処理のほかに、ゲームを一時休止する処理と一時休止から復帰する処理も必要になります。ゲームを一時休止する処理では、DirectXが占有しているデバイスをいったん解放したり、音楽の再生を止めたりといった処理を行います。一時休止から復帰する処理では、デバイスの再占有や再構築、音楽の再生の復帰といった処理を行います。これらの処理はWM_ACTIVATEAPPメッセージが発生したときに行うか、アクティブ状態のゲーム処理で、デバイスが消失したことを検出したときに復元します。

```
// ウィンドウプロシージャ
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch(uMsg) {
      case WM_ACTIVATEAPP:
        if(wParam != 0) {
            m_Active = true;
            (ゲームの復帰処理を行う関数の呼び出し)
        } else {
            m_Active = false;
            (ゲームの休止処理を行う関数の呼び出し)
        }
        return 0;
            ⋮(以下省略)
```

課　　題

アプリケーションの状態を判別し、それに合わせた処理へ分岐させる機能を追加しましょう。

(１)アプリケーションの状態を示す変数を追加します。以下のメンバを適切な場所に追加しましょう。

```
bool   m_Active;  // アクティブ状態フラグ(true…アクティブ状態／false…非アクティブ状態)
```

(２)(１)を追加したクラスのコンストラクタ初期化子に、以下のプログラムを追加し、オブジェクト構築時にm_Activeが適切に初期化されるようにしましょう。

```
m_Active(false)
```

(３)CGameAppクラスにWM_ACTIVATEAPPメッセージを処理する機能を追加します。以下の追加１および追加２を適切な場所に追加しましょう。

－追加１－

```
case WM_ACTIVATEAPP:     return OnActivateApp(hWnd, wParam, lParam);
```

－追加２－

```
/*****************************************************************************************************/
/*                                    WM_ACTIVATEAPPメッセージ処理                                   */
/*****************************************************************************************************/
LRESULT CGameApp::OnActivateApp(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    if(wParam != 0) {
        ※ここは各自考えましょう(アクティブ状態フラグをアクティブ状態にする)
    } else {
        ※ここは各自考えましょう(アクティブ状態フラグを非アクティブ状態にする)
    }

    return 0;
}
```

※追加１，２のほかに、ヘッダファイルに追加しなければならないものがあります。

(４)ゲームの処理をアプリケーションの状態によって変更できるようにしましょう。

ゲームを実行する関数は、CGameProc::Run関数です。この関数内の処理をアプリケーションの状態によって分岐させるようにします。
CGameProcクラスのRun関数を以下のようにアクティブ状態かどうかを示す引数を取るようにし、関数内の処理もそれに合わせて分岐させるようにしましょう。また、この関数のプロトタイプも適切なものに変更しましょう。

```
bool CGameProc::Run(const bool inActive)
{
    if(inActive)
        ;   // アクティブ時の処理
    else
        ;   // 非アクティブ時の処理

    return true;
}
```

(５)CGameProc::Main関数の変更に伴い、CGameAppクラスのMain関数を以下のように変更しましょう。

```
int CGameApp::Main()
{
    while(true) {
        // メッセージ処理
        if(MessageLoop() == false)
            break;
        // ゲーム処理
        if(m_GameProc.???(m_Active) == ※ここは各自考えましょう)
            Release();
    }

    return 0;
}
```