ゲームプログラミング

▶　基礎編－第８回　メッセージループ　◀

ゲームプログラムでは、Windowsから送られてくるメッセージを処理しつつ、リアルタイムにゲームの処理を行う必要があります。一般的なメッセージループはGetMessage関数を使い、メッセージがくるまでプログラムを休止します。この方法では、リアルタイムにプログラムを走らせることが困難なので、ゲームプログラムでは、PeekMessage関数を使ってメッセージループを確立します。

ゲームプログラムとメッセージ処理

Windowsアプリケーションは、Windowsから送られてくるメッセージを必ず処理する必要があります。メッセージが処理されないと、「応答なし」状態と見なされ動作が不安定になるだけでなく、ほかのアプリケーションにも悪影響を与える場合があります。

ワープロや表計算といったアプリケーションの場合、メッセージは非常に重要な役割を持っており、メッセージに応答しながら処理を進める形式がとられます。メッセージの取得にはGetMessage関数が使われ、メッセージがくるまでプログラムを休止し、ほかのアプリケーションにCPUの実行権を渡します。メッセージがきたときだけ(なにかイベントが起きたときだけ)プログラムが動作するわけです。

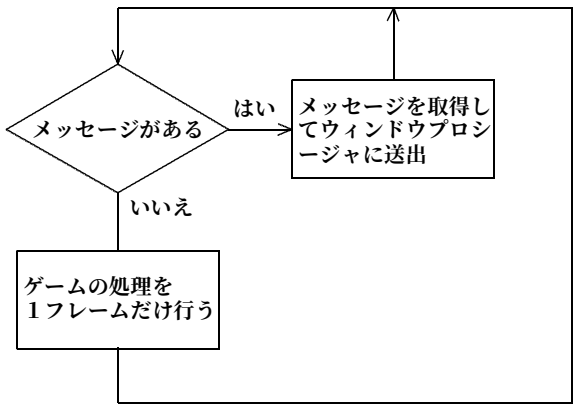
しかし、ゲームプログラムでは、ゲームの動作とメッセージはほとんど無関係であり、リアルタイムに動かし続けなければならないため、メッセージを待つ必要はありません。このようなプログラムでは、GetMessage関数は適さないので、代わりにPeekMessage関数でメッセージの取得を行います。この関数はメッセージを待つことがないので、メッセージがないときは直ちに次の処理を行えます。

ゲームプログラムでは、メッセージキューにあるメッセージをすべて処理し、なくなった隙にゲームの処理をします。こうすると、Windowsアプリケーションらしい振る舞いで、かつリアルタイムにゲームを動かすことができます。

```
while(true) {
    // メッセージキューに溜まったメッセージをすべて処理する
    while(0 != PeekMessage(&msg, 0, 0, 0, PM_REMOVE)) {
        if(WM_QUIT == msg.message)
            return msg.wParam;      // WM_QUITメッセージを受信した場合は、プログラム終了
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }   // while(PeekMessage)

    // メッセージがなくなったら、ゲームを処理する
    ※ここで、ゲームを処理する関数の呼び出しを行います
}   // while(true)
```

－メッセージループ・フローチャート－



○ 課題 ○

ゲームプログラムに適したメッセージループを実装しましょう。