ゲームプログラミング

▶　第２回　ウィンドウ　◀

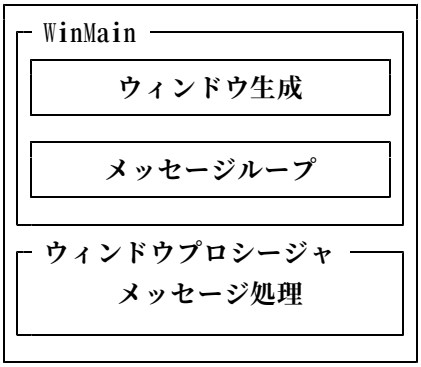
Windowsで動作するプログラムは、ウィンドウを生成する必要があります。ウィンドウは、複数のプログラムが協調して同時に動作するWindowsでは、非常に重要な役割を持っています。

Windowsプログラムの構成

Windowsの基本的なプログラムの構成は、WinMain関数とウィンドウメッセージ(※第３回参照：以下メッセージ)を処理するウィンドウプロシージャという２つの関数から成ります。

WinMain関数では、ウィンドウを生成し、Windowsから送られてくるメッセージを取り出してウィンドウプロシージャに送出するためのメッセージループを確立します。

ウィンドウプロシージャは、プログラマが定義しなければならない関数で、各メッセージをどのように処理するかを記述します。

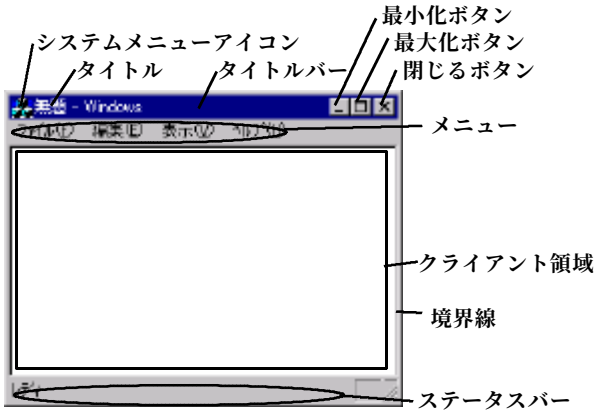


ウィンドウ

ウィンドウは、アプリケーションがユーザの操作情報やWindowsからの指示(メッセージ)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。前回作成したメッセージボックスもウィンドウから派生したもので、MessageBox関数が内部的にユーザから操作情報やWindowsからの指示を受け取り、処理しています。

ウィンドウの構成要素

一般的なウィンドウは、以下のように構成されています。



・システムメニュー…ここをクリックするとウィンドウの基本的な操作メニューが表示されます。
・タイトルバー……ウィンドウのタイトルを表示する領域です。ここをドラッグするとウィンドウが移動します。
・クライアント領域…テキストやビットマップを表示する領域です。

ウィンドウの生成

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（41～54行）
(２)ウィンドウクラスの登録（55行）
(３)ウィンドウの生成　　（59～64行）
(４)ウィンドウの表示　　（70行）

(４)は必ずしも必要ではありませんが､ウィンドウを表示しないとタスクバーに表示されなくなります。

(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。各メンバの意味は、以下のようになっています。

```
UINT     cbSize;         // WNDCLASSEX構造体のサイズ。sizeof演算子で取得
UINT     style;          // ウィンドウのスタイル
WNDPROC  lpfnWndProc;    // ウィンドウに送出されるメッセージを処理する関数
int      cbClsExtra;     // 拡張クラス情報
int      cbWndExtra;     // 拡張ウィンドウ情報
HANDLE   hInstance;      // ウィンドウが所属するアプリケーションのインスタンスハンドル
HICON    hIcon;          // ラージアイコンのハンドル
HCURSOR  hCursor;        // マウスカーソルのハンドル
HBRUSH   hbrBackground;  // ウィンドウ背景色
LPCTSTR  lpszMenuName;   // メインメニュー名
LPCTSTR  lpszClassName;  // 登録するウィンドウクラスの名前
HICON    hIconSm;        // スモールアイコンのハンドル
```

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つ目のパラメータ)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。ここで指定した関数でメッセージが処理されます。

(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関数でウィンドウを生成できるようになります。この関数の引数は、以下のようにたくさんあります。

```
DWORD     dwExStyle;     /* 拡張ウィンドウスタイル                          */
LPCTSTR   lpszClassName; /* 登録済みのウィンドウクラスの名前                */
LPCTSTR   lpszWindowName;/* ウィンドウのタイトル                            */
DWORD     dwStyle;       /* ウィンドウスタイル                              */
int       x;             /* ウィンドウの水平座標の位置                      */
int       y;             /* ウィンドウの垂直座標の位置                      */
int       nWidth;        /* ウィンドウの幅                                  */
int       nHeight;       /* ウィンドウの高さ                                */
HWND      hwndParent;    /* 親ウィンドウまたはオーナーウィンドウのハンドル  */
HMENU     hmenu;         /* メニューのハンドルまたは子ウィンドウのID        */
HINSTANCE hinst;         /* アプリケーションインスタンスのハンドル          */
LPVOID    lpvParam;      /* ウィンドウ作成データ                            */
```

１つ目の引数は、ウィンドウの拡張スタイルです。常に最前面に表示するようにしたりできます。
２つ目の引数は、ウィンドウ生成に用いるウィンドウクラスの指定です。(１)で定義したWNDCLASSEX構造体のlpszClassNameメンバと同じ文字列を指定します。
３つ目の引数は、ウィンドウのタイトル文字列です。ここで指定した文字列がタイトルバーに表示されます。
４つ目の引数は、ウィンドウの形状で複数の値を組み合わせることができます。
５つ目と６つ目の引数は、ウィンドウの初期座標です。
７つ目と８つ目は、ウィンドウの幅と高さで、ウィンドウ全体のサイズを指定します。

９つ目は、親ウィンドウのハンドルで、子ウィンドウを作成する場合に使用します。メインウィンドウの場合は「HWND_DESKTOP」やNULLを指定します。
１０個目の引数は、メニューがある場合にそのハンドルを渡します。
１１個目の引数は、生成するウィンドウが所属するアプリケーションのインスタンスハンドルを渡します。
１２個目の引数は、通常は使用しないのでNULLにします。
ウィンドウの位置、幅、高さにはCW_USEDEFAULTが指定できます。これを指定すると、Windowsが適当な値を設定します。

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、失敗すると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、メモリ上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関数を使って表示モードを設定することで初めて画面に表示されます。
ShowWindow関数の１つ目の引数は表示モードを設定するウィンドウのハンドル(CreateWindowEx関数の戻り値)、２つ目の引数は表示モードです。メインウィンドウの場合は、WinMain関数の４つ目のパラメータnShowCmdを渡します。

サンプルプログラム(WinMain.cpp)では、ShowWindow関数の前後にSetFocus関数とUpdateWindow関数を呼び出しています。SetFocus関数は、ウィンドウがキーボードの入力を受け取れる状態する関数です。UpdateWindow関数は、ウィンドウのクライアント領域(文字やグラフィックを描画する領域)を更新する関数です。

○ 課題 ○
ウィンドウを生成し表示してみましょう。

サンプルプログラム(WinMain.cpp)を入力して動作を確認します。一般的なウィンドウの機能はすべてそろっていて、ウィンドウの移動、サイズ変更、最大化、最小化などはすべて行えます。それらの機能を確認しましょう。

なお、このプログラムは不完全なため閉じるボタンを押しても終了しません。ウィンドウは消えますが、プログラムはメモリに存在しています。プログラムの完全な終了は、Ctrl+Alt+Delで強制終了する必要があります。この不具合は次回の課題で修正します。