CG プログラミング入門 (6)

Introduction to CG Programming (6)

長島 忍 Shinobu NAGASHIMA

Keywords

CG／プログラミング／C言語

1. はじめに

以前、マウスのクリックで多角形の頂点数を増やしたり減らしたりすることをした。今回は、マウスの移動で図形を描画することを試みる。これにより、マウスの軌跡で絵をかいたり、画面を見ながら図形を入力して絵を作ることができるようになる。またデザイン関係の人はよく知っているだろうが、タブレット（デジタイザ）という座標入力装置を使うと、マウスのプログラムはそのまま使用することができ、またマウスより使いやすいペンで、スケッチ図のトレースなど様々な用途に利用できるようになる。

2. イベント制御のしくみ

従来よく言われた「プログラム＝アルゴリズム＋データ構造」のプログラムは、メインプログラムから始まって、順次記述された処理をこなしていくやり方である。よく処理の流れをフローチャートで表現するとわかりやすくなると言われた。

Windows のアプリケーションプログラムは、局所的な処理の流れは従来と変わらないが、イベントが発生するとイベント情報を OS からアプリケーションプログラムに渡して処理をするやり方になっている。たとえばマウスを動かすと、マウスを動かした（動かしている）という情報が OS からアプリケーションプログラムに伝達される。このような方式をイベント駆動型といい、イベントは Windows ではウィンドウズメッセージと呼ばれている。

このイベント処理部分をもう一度復習してみる。Visual C++ で “Hello World” アプリケーションを作成し、ソースを見てもらいたい。この連載講座で毎回やっているように、Visual C++ を起動し、ファイルメニューから新規作成を選び、プロジェクトタブを開い

て Win 32 Application を選ぶ。プロジェクト名には cgex 6 など適当な名前を入力して OK ボタンを押す。次に「標準的な “Hello World” アプリケーション」を選んでいくととりあえず Hello という文字列を表示するプログラムが出来上がる。

ソースファイル cgex 6.cpp の中を見ると、自動的に 6 個の関数が作成される。その中の一つ、WinMain から実行が始まる。イベントを処理するのが、

```
LRESULT CALLBACK WndProc (HWND hWnd,
UINT message, WPARAM wParam, LPARAM lParam)
```

という関数である。関数名は WndProc であり、この名前は MyRegisterClass の中でイベント発生時の処理関数として設定するので、WndProc は自分の好きな名前に変更することができる。

次に WndProc の中を見てみると、

```
case WM_COMMAND:
case WM_PAINT:
case WM_DESTROY:
```

という case 文が目につく。これがイベントの種類である。これらの 3 つは、それぞれ、メニューを選択した場合、表示を行う場合、終了する場合に OS からプログラムに渡される情報に対応している。アプリケーションプログラムのウィンドウ上でマウスを動かすとマウスを動かしているという情報が OS からこのプログラムに伝えられるので、まずその座標を観察し、次に座標から簡単な図形を描くプログラムを作成してみる。

3. マウス移動表示プログラム

ウィンドウメッセージ（イベント）は上にあげた、WM_COMMAND、WM_PAINT のように先頭に WM_ がつく。マウスの移動によるイベントは、

```
WM_MOUSEMOVE
```

である。またマウスの左ボタンを押したり、左ボタンを離したりするときは、

WM_LBUTTONDOWN

WM_LBUTTONUP

というイベントが発生する。注意することは、普通マウスの操作ではクリック、ドラッグなどのやり方があるが、これには直接対応していない。すなわち、クリックした場合は、ダウンとアップのイベントが発生する。ドラッグ、ムーブに限らず、マウスを移動するとマウスムーブのイベントが発生する。マウスのようなイベントではその時点での座標が重要であるが、このようなデータは WndProc の引数で受け渡される。

座標値は lParam という LPARAM 型引数で受け渡される。これは32bit 符号なし整数と考えればよく、下位16bit に x 座標、上位16bit に y 座標が入っている。座標はディスプレイの座標ではなく、アプリケーションプログラムのウィンドウのタイトルやメニューなどを除いたクライアント画面と呼ばれる部分で測った数値である。左上が原点 (0, 0) で、y 軸が下向きになっている。

マウスの移動にしたがって座標を画面に表示するプログラムを図1に示す。いつものように WM_PAINT を修正し、新しいイベント処理を追加する。

case WM_MOUSEMOVE:

```
hdc=GetDC(hWnd);
wsprintf(szHello,"%d : (%d, %d) ",
wParam, LOWORD(lParam), HIWORD(lParam));
TextOut(hdc,1,1,szHello,strlen(szHello));
ReleaseDC(hWnd,hdc);
break;
```

この追加部分で、まず GetDC というのは、描画用のデバイスコンテキスト hdc を取得する Windows API (システム関数) である。次に wsprintf は C 言語の関数 sprintf と同じで整数値などを文字列に変換する関数である。sprintf を使用する場合は、〈stdio.h〉をインクルードする必要があるが、wsprintf の場合はインクルードの変更は必要ない。座標の x y 座標が引数 lParam の下位および上位16bit にあるので、マクロでそれぞれを取り出す。また wParam という引数には、マウスの移動時にボタンが押してなければ0、ボタンを押していれば1が入っている。それらを同時に表示してみる。TextOut は画面上に座標を指定して文字列を表示する API である。最後に GetDC で取得したデバイスコンテキストは、ReleaseDC で解放しなければならない。

実行結果を図2に示す。TextOut の部分を点の描画に置き換えると、マウスの軌跡で図を描くことができる。とりあえず図2のようなプログラムが出来れば、マ

図1 座標表示プログラム

図2 表示例

ウス操作で何かできるという確認ができた。いろいろなことに応用できると思う。

なお図1, WM_PAINTのBeginPaint, EndPaintも削除してよい(psを宣言したのに使用していないという警告メッセージは出る)。WM_PAINTで何もなくなったので、ウィンドウのサイズを変更したり、覆われていた他のウィンドウが取り払われても、何も表示はしない。

4. マウสดラッグによる円の入力

少し応用してマウスをドラッグして、任意の場所に任意の大きさの円を作成してみる。よくある描画ソフトでは、マウスでドラッグすると、ドラッグした線を対角線とする矩形に内接する楕円を入力するやり方が多い。この方式は、場所と大きさの指定がうまく出来ないこともあり、ここでは中心から円周上の一点までをドラッグして円を作成することにする。

ドラッグの開始・終了は上で述べたように、WM_MOUSEMOVEのwParamを見てもいいが、普通はボタンを押したり、離したりするときに発生するイベントWM_LBUTTONDOWN, WM_LBUTTONUPを利用する。

図3がプログラムである。WndProc内のcase WM_PAINT:~break;の間を取る。その次に、case WM_LBUTTONDOWN:~break;など3種類のイベント処理部分を追加する。さらにこの関数WndProcの先頭で変数宣言が必要になり、

```
static int ox,oy,updown = 0;
```

という1行を追加する。HDC hdc;などの変数宣言があるので、それらの中に挿入する。ox, oyは円の中心

図3 円作成プログラム

座標であり、updown はマウスを押したとき 1 に、離れたとき 0 に設定する変数である。図 3 のプログラムにはコメントが多いがこれらのコメントをはずす（つまり／を削除する）ことにより最終プログラムに変化する。

図 3 のプログラムで、WM_LBUTTONDOWN のときは、マウスの xy 座標をこの ox, oy に、updown に 1 を代入する。これらは static 宣言しておかないと内容が消えてしまう。次に WM_MOUSEMOVE の部分で、updown が 0 のときは何もしない。1 のときは x y 座標からまず円の半径を計算し、(ox, oy) を中心とする円を描画する。プログラム中の変数 x, y は ox, oy との差の絶対値である。したがって r はボタンを押した点と現在の点との距離になる。Arc は楕円、楕円弧を描画する Windows API である。ここでは円の描画に使用している。

これを実際に実行してみると図 4 のようになる。これは描画した円を消さないで、そのまま残っている。マウスをドラッグすると半径が変わり、いくつもの同心円が描かれることになる。この図も面白いので、場所を変えていくつか入力し表示することにした。

本来やりたかったように改良するには、図 3 のプログラムでコメント行をはずす。GetClientRect は Windows API で、ウィンドウの大きさを調べる。PatBlt というのは画面消去をする Windows API で、一般にはいろいろなパターンでウィンドウを塗りつぶす機能である。画面消去の場合は画面一杯に背景色で塗りつぶすようにする。WHITNESS という引数は白で塗りつぶす場合であるが、特定の色で塗りつぶす場合はブラシの作成などをしなければならない。

この他円以外に、中心座標がよくわかるように十字の直線をいれてみた。この直線を描画するのが、Arc の次にある 2 行である。図 3 を改良したプログラムの実行例を図 5 に示す。

図 4 実行例

図 5 実行例（改良版）

5. おわりに

Windows のプログラムを作るには、C++ 言語の文法の他に、何百とある Windows API やイベント（メッセージ）を知らなければならない。しかしこれらの API やイベントを処理できれば比較的簡単にプログラムを作ることができる。MS-DOS の時代にはマウスを使うには RS-232 C を初期化したり、つねに信号を監視して入力があればそれに対応する処理を記述しなければならなかったもので、それを思うと、現在容易にしかも統一的に、プログラムが作れるのは本当にありがたいことである。

●2003年8月11日受付

ながしま し のぶ

立教大学経済学部・教授

1979年東京大学工学部精密機械工学科卒、1985年工学博士。東京大学教養学部図学教室助手、講師、立教大学一般教育部助教授を経て1999年より現職。