第３回 Windows アプリ講座

１．日付表示と切り替える

前回作成した時計アプリにさまざまな機能を追加してみましょう。クリックでモードメニューを開き日付表示と切り替えられるようにしましょう。前回は time ライブラリによる時間取得を行いましたが、今回は WIN32API の時間取得関数を使います。

さらに、右クリックメニューを実装するためにリソースというものを使います。

```
////////////////////////////////////////////////////////////////////////03-01.cpp////////////////////////////////////////////////////////////////////////
#include <windows.h>
#include "resource.h"                    //リソースファイル

#define IDT_MYTIMER                      0x10000          //タイムアウト時のイベント ID

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam); //ウィンドウプロシージャの宣言

//グローバル変数
LPCTSTR szClassName = TEXT("MainWindow"); //ウィンドウクラス名
HFONT g_hTimeFont;
int g_Mode;

//ウィンドウクラスの登録
ATOM InitApp(HINSTANCE hInst)
{
        WNDCLASSEX wc;
                //ウィンドウクラス構造体
        wc.cbSize = sizeof(WNDCLASSEX);                         //ウィンドウクラス構造体のサイズを設定
        wc.style = CS_HREDRAW | CS_VREDRAW;                     //ウィンドウクラスのスタイルを設定
        wc.lpfnWndProc = WndProc;                               //ウィンドウプロシージャの設定
        wc.cbWndExtra = 0;                                      //ウィンドウに割り当てられる追加メモリ量
        wc.cbClsExtra = 0;                                      //ウィンドウクラスに割り当てられる追加メモリ量
        wc.hInstance = hInst;                                   //アプリケーションのインスタンスハンドルの設定
        wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);             //アイコンの設定
        wc.hIconSm = LoadIcon(NULL, IDI_APPLICATION);           //小アイコンの設定
        wc.hCursor = LoadCursor(NULL, IDC_ARROW);               //カーソルの設定
        wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);     //背景の設定
        wc.lpszMenuName = NULL;                                 //メニューバーの設定
        wc.lpszClassName = szClassName;                         //ウィンドウクラス名の設定
        return (RegisterClassEx(&wc));                          //ウィンドウクラスの登録
}

//ウィンドウの作成
BOOL InitInstance(HINSTANCE hInst, int nCmdShow)
{
        HWND hWnd;      //ウィンドウハンドル

        //ウィンドウの作成
        hWnd = CreateWindow(szClassName,    //使用するウィンドウクラス
                TEXT("真っ白ウィンドウ"),       //タイトルバーの名前
                WS_OVERLAPPEDWINDOW ^ WS_MAXIMIZEBOX ^ WS_SIZEBOX,//ウィンドウスタイル
                CW_USEDEFAULT,                  //ウィンドウの x 座標
                CW_USEDEFAULT,                  //ウィンドウの y 座標
                200,                            //ウィンドウの幅
```

```
                100,                    //ウィンドウの高さ
                NULL,                   //オーナーウィンドウハンドル
                NULL,                   //メニューハンドルか小ウィンドウの ID
                hInst,                  //アプリケーションのインスタンスハンドル
                NULL);                  //ウィンドウ作成データ

        if (!hWnd)return FALSE;         //ウィンドウの作成に失敗

        HMENU hSysMenu = GetSystemMenu(hWnd, FALSE);    //システムメニューのメニューハンドルを取得
        DeleteMenu(hSysMenu, SC_RESTORE, MF_BYCOMMAND);         //システムメニュー「元のサイズに戻す」を削除
        DeleteMenu(hSysMenu, SC_MAXIMIZE, MF_BYCOMMAND);        //システムメニュー「最大化」を削除
        DeleteMenu(hSysMenu, SC_SIZE, MF_BYCOMMAND);            //システムメニュー「サイズ変更」を削除

        ShowWindow(hWnd, nCmdShow);     //ウィンドウを可視化する
        UpdateWindow(hWnd);             //ウィンドウを更新する
        return TRUE;
}

//時刻文字列の取得関数
void GetTimeString(LPTSTR lpTimeString)
{
        SYSTEMTIME Time;

        GetSystemTime(&Time);
        wsprintf(lpTimeString, TEXT("%02d:%02d:%02d"), Time.wHour, Time.wMinute, Time.wSecond);
}

//日付文字列の取得関数
void GetDayString(LPTSTR lpTimeString)
{
        SYSTEMTIME Time;
        LPTSTR DayOfWeek[7] = {TEXT("日"), TEXT("月"), TEXT("火"), TEXT("水"), TEXT("木"), TEXT("金"), TEXT("土")};

        GetSystemTime(&Time);
        wsprintf(lpTimeString, TEXT("%02d/%02d/%02d(%s)"), Time.wYear, Time.wMonth, Time.wDay, DayOfWeek[Time.wDayOfWeek]);
}

//ウィンドウプロシージャ
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
        HDC hdc;                //デバイスコンテキストハンドル
        PAINTSTRUCT ps;         //ペイント構造体
        TCHAR TimeString[256];
        RECT rect;
        HFONT hOldFont;
        HMENU hMenu, hSubMenu;
        POINT MousePoint;

        switch(msg)
        {
        case WM_CREATE:
                g_hTimeFont = CreateFont(40,            //フォントの高さ
                        0,                              //フォントの幅
                        0,                              //文字送り方向の傾き
                        0,                              //ベースラインの傾き
                        FW_NORMAL,                      //フォントの太さ
                        FALSE,                          //斜体フラグ
```

```
                        FALSE,                          //下線フラグ
                        FALSE,                          //取り消し線フラグ
                        ANSI_CHARSET,                   //文字セット識別子
                        OUT_DEFAULT_PRECIS,             //出力精度
                        CLIP_DEFAULT_PRECIS,            //クリッピング精度
                        DEFAULT_QUALITY,                //出力品質
                        FF_DONTCARE | DEFAULT_PITCH,    //ピッチとファミリ
                        TEXT("MS ゴシック")             //フォント名
                        );
                break;

        case WM_CLOSE:
                PostQuitMessage(0);
                break;

        case WM_COMMAND:
                GetWindowRect(hWnd, &rect);             //ウィンドウの座標取得
                switch(LOWORD(wParam))
                {
                case ID_TIME:
                        g_Mode = 0;
                        MoveWindow(hWnd, rect.left, rect.top, 200, 100, TRUE);      //ウィンドウサイズの変更
                        break;

                case ID_DAY:
                        g_Mode = 1;
                        MoveWindow(hWnd, rect.left, rect.top, 300, 100, TRUE);
                        break;
                }
                break;

        case WM_PAINT:
                hdc = BeginPaint(hWnd, &ps);            //描画開始
                switch(g_Mode)
                {
                case 0:
                        GetTimeString(TimeString);
                        break;

                case 1:
                        GetDayString(TimeString);
                        break;
                }
                hOldFont = (HFONT)SelectObject(hdc, (HFONT)g_hTimeFont);
                TextOut(hdc, 10, 10, TimeString, lstrlen(TimeString));
                SelectObject(hdc, (HFONT)hOldFont);
                EndPaint(hWnd, &ps);                    //描画終了
                SetTimer(hWnd, IDT_MYTIMER, 1000, NULL);  //タイマーをセット
                break;

        case WM_TIMER:              //タイマーのタイムアウトメッセージ
                KillTimer(hWnd, IDT_MYTIMER);           //タイマーを破棄
                GetClientRect(hWnd, &rect);             //クライアント領域を取得
                RedrawWindow(hWnd, &rect, NULL, RDW_ERASE | RDW_INVALIDATE);//ウィンドウの再描画
                break;

        case WM_RBUTTONDOWN:            //右クリックが押された
                MousePoint.x = LOWORD(lParam);
                MousePoint.y = HIWORD(lParam);
                hMenu   =   LoadMenu((HINSTANCE)GetWindowLong(hWnd,   GWL_HINSTANCE),
```

```
(LPTSTR)IDR_RBUTTON);	//メニューのロード
			hSubMenu = GetSubMenu(hMenu, 0);
			ClientToScreen(hWnd, &MousePoint);	//クライアント座標からスクリーン座標へ
			TrackPopupMenu(hSubMenu, TPM_LEFTALIGN | TPM_TOPALIGN, MousePoint.x,
MousePoint.y, 0, hWnd, NULL);		//メニューの表示
			DestroyMenu(hMenu);		//メニューの削除
			break;


		default:
			return (DefWindowProc(hWnd, msg, wParam, lParam));
		}

		return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

リソースを使うにはリソースファイルとそのリソースファイルに対応したヘッダーファイルが必要です。

```
///////////////////////////////////////////////////////////////resource.h///////////////////////////////////////////////////////////////
#define IDR_RBUTTON				101
#define ID_TIME				40003
#define ID_DAY				40004
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////03-01.rc///////////////////////////////////////////////////////////////
IDR_RBUTTON MENU
BEGIN
    POPUP "右メニュー"
    BEGIN
        MENUITEM "時間表示",				ID_TIME
        MENUITEM "日付表示",				ID_DAY
    END
END
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

WM_RBUTTONDOWN はマウスの右ボタンが押された瞬間に送られるメッセージです。WM_RBUTTONDOWN というメッセージが送られたとき、LPARAM にはボタンが押されたときにマウスのクライアントエリアの座標が格納されています。LPARAM の下位 16 ビットに x 座標、上位 16 ビットに y 座標が格納されています。LOWORD マクロを使うと下位 16 ビットを HIWORD マクロを使うと上位 16 ビットの値を取り出すことが出来ます。

メニューの読み込みには LoadMenu 関数を使います。ここからさらに右メニューに必要な項目だけを取り出します。それが GetSubMenu 関数です。

メニューの作成が完了したら次にメニューを表示する座標を補正します。LPARAM に格納されているのはクライアントアリアの座標なのでそれをスクリーン座標に変換する必要があります。ClientToScreen 関数を使うと座標の補正が出来ます。

これでメニューを表示する準備が整いました。TrackPopupMenu 関数でメニューの表示を行います。最後に DestroyMenu 関数でメニューを削除します。

メニューから項目が選択されると WM_COMMAND メッセージが送られます。このとき WPARAM の下位 16 ビットには選択された項目の識別子が格納されています。あとは switch 文を使い識別子別に処理を記述すればよいのです。

2.チェックを入れてみる。

右メニューの項目に現在のモードに従ってチェックが入るようにしましょう。

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//チェックマークの処理
void ShowCheckMark(HMENU hMenu)
{
		MENUITEMINFO mii;				//メニュー項目の情報が記述された構造体

		mii.cbSize = sizeof(mii);
		mii.fMask = MIIM_STATE;		//項目の状態のみ所得

		if (g_Mode == 0)
		{
			GetMenuItemInfo(hMenu, ID_TIME, FALSE, &mii);		//項目の情報を取得
			if (mii.fState == MFS_UNCHECKED)
			{
				mii.fState = MFS_CHECKED;		//チェックをつける
				SetMenuItemInfo(hMenu, ID_TIME, FALSE, &mii);//項目の情報を設定する
				GetMenuItemInfo(hMenu, ID_DAY, FALSE, &mii);
				mii.fState = MFS_UNCHECKED;
				SetMenuItemInfo(hMenu, ID_DAY, FALSE, &mii);
			}
		}
		else if (g_Mode == 1)
		{
			GetMenuItemInfo(hMenu, ID_DAY, FALSE, &mii);
			if (mii.fState == MFS_UNCHECKED)
			{
				mii.fState = MFS_CHECKED;
				SetMenuItemInfo(hMenu, ID_DAY, FALSE, &mii);
				GetMenuItemInfo(hMenu, ID_TIME, FALSE, &mii);
				mii.fState = MFS_UNCHECKED;
				SetMenuItemInfo(hMenu, ID_TIME, FALSE, &mii);
			}
		}
}

//ウィンドウプロシージャ
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
	HDC hdc;			//デバイスコンテキストハンドル
	PAINTSTRUCT ps;		//ペイント構造体
	TCHAR TimeString[256];
	RECT rect;
	HFONT hOldFont;
	HMENU hMenu, hSubMenu;
	POINT MousePoint;

	switch(msg)
	{
	case WM_CREATE:
		g_hTimeFont = CreateFont(40,			//フォントの高さ
			0,					//フォントの幅
			0,					//文字送り方向の傾き
			0,					//ベースラインの傾き
			FW_NORMAL,			//フォントの太さ
			FALSE,				//斜体フラグ
```

```
                        FALSE,                          //下線フラグ
                        FALSE,                          //取り消し線フラグ
                        ANSI_CHARSET,                   //文字セット識別子
                        OUT_DEFAULT_PRECIS,             //出力精度
                        CLIP_DEFAULT_PRECIS,            //クリッピング精度
                        DEFAULT_QUALITY,                                    //出力品質
                        FF_DONTCARE | DEFAULT_PITCH,    //ピッチとファミリ
                        TEXT("MS ゴシック")                         //フォント名
                        );
                break;

        case WM_CLOSE:
                PostQuitMessage(0);
                break;

        case WM_COMMAND:
                GetWindowRect(hWnd, &rect);             //ウィンドウの座標取得
                switch(LOWORD(wParam))
                {
                case ID_TIME:
                        g_Mode = 0;
                        MoveWindow(hWnd, rect.left, rect.top, 200, 100, TRUE);      //ウィンドウサイズの変更
                        break;

                case ID_DAY:
                        g_Mode = 1;
                        MoveWindow(hWnd, rect.left, rect.top, 300, 100, TRUE);
                        break;
                }
                break;

        case WM_PAINT:
                hdc = BeginPaint(hWnd, &ps);            //描画開始
                switch(g_Mode)
                {
                case 0:
                        GetTimeString(TimeString);
                        break;

                case 1:
                        GetDayString(TimeString);
                        break;
                }
                hOldFont = (HFONT)SelectObject(hdc, (HFONT)g_hTimeFont);
                TextOut(hdc, 10, 10, TimeString, lstrlen(TimeString));
                SelectObject(hdc, (HFONT)hOldFont);
                EndPaint(hWnd, &ps);                    //描画終了
                SetTimer(hWnd, IDT_MYTIMER, 1000, NULL);  //タイマーをセット
                break;

        case WM_TIMER:              //タイマーのタイムアウトメッセージ
                KillTimer(hWnd, IDT_MYTIMER);           //タイマーを破棄
                GetClientRect(hWnd, &rect);             //クライアント領域を取得
                RedrawWindow(hWnd, &rect, NULL, RDW_ERASE | RDW_INVALIDATE);//ウィンドウの再描画
                break;

        case WM_RBUTTONDOWN:            //右クリックが押された
                MousePoint.x = LOWORD(lParam);
                MousePoint.y = HIWORD(lParam);
                hMenu   =   LoadMenu((HINSTANCE)GetWindowLong(hWnd,   GWL_HINSTANCE),
```

```
(LPTSTR)IDR_RBUTTON);   //メニューのロード
                        hSubMenu = GetSubMenu(hMenu, 0);
                        ClientToScreen(hWnd, &MousePoint);   //クライアント座標からスクリーン座標へ
                        ShowCheckMark(hSubMenu);
                        TrackPopupMenu(hSubMenu,   TPM_LEFTALIGN   |   TPM_TOPALIGN,   MousePoint.x,
MousePoint.y, 0, hWnd, NULL);           //メニューの表示
                        DestroyMenu(hMenu);          //メニューの削除
                        break;


        default:
                        return (DefWindowProc(hWnd, msg, wParam, lParam));
        }

        return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

項目がチェックされているかどうかを知るには項目の情報を取得する必要があります。その情報はMENUITEMINFO 構造体に記述されています。この構造体に取得した項目情報を格納するには fmask メンバに取得したい情報の識別子を入力し、GetMenuItemInfo 関数を使います。こうすることで項目情報が取得できました。項目にチェックが入ってるかどうかはfStateメンバにMFS_CHECKEDが格納されていればチェックされており、MFS_UNCHECKED が格納されていればチェックされていないと言うことになります。

以上のことを使い、片方がチェックされているときは一方をチェックされていない状態にすると言うことをやれば現在のモードにチェックが入るということが出来ます。