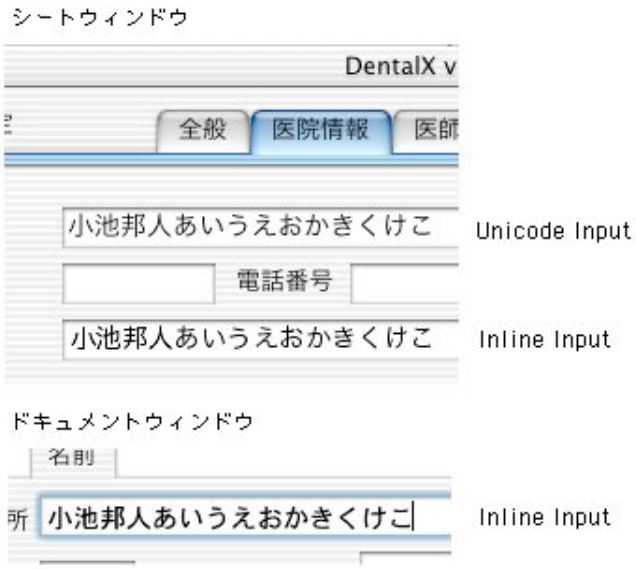
小池邦人のプログラミング日記》2002/2/6
＜TransitionWindow()でウィンドウを移動させる＞

今回から、Mac OS X 10.1 で利用できるようになったウィンドウに関係する新機能について解説します。最初は、Window Manager に属する TransitionWindow()という API を使ったサンプルアプリケーションを紹介します。

Mac OS X用のアプリケーションを開発していると、色々と理解に苦しむ現象に遭遇します（笑）。例えば、ドキュメントウィンドウとSheetウィンドウに、異なるタイプのTextEditコントロールを配置します。すると状況によって、テキスト入力で表示される文字の形状が変わります。

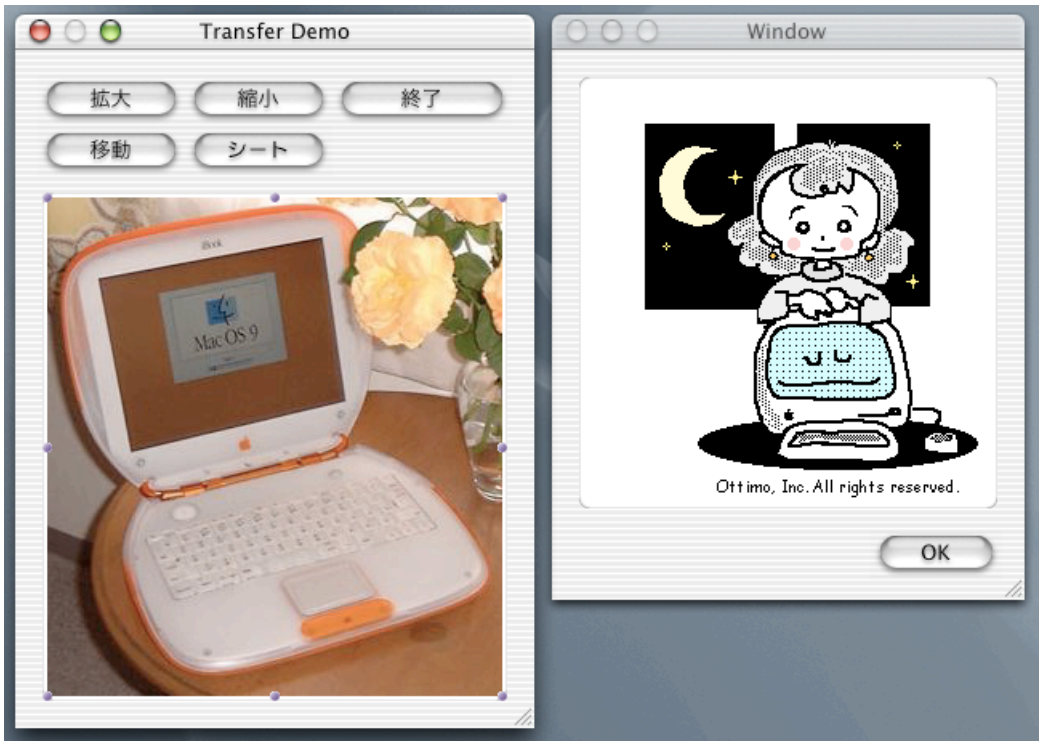


SheetウィンドウのUnicode InputタイプとInline Inputタイプの違いは、Quartz 2DとQuickDrawのアンチエイリアス表示の違いでしょうが、ドキュメントウィンドウのInline Inputタイプで文字が太くなるのは何故でしょうか？基本的には、Unicode Inputタイプが一番奇麗な文字表示なのですが、このタイプのTextEditでは、イベントクラスが「kEventClassTextInput」で、イベント種類が「kEventTextInputUnicodeForKeyEvent」のCarbonイベントが来ません。Inline Input

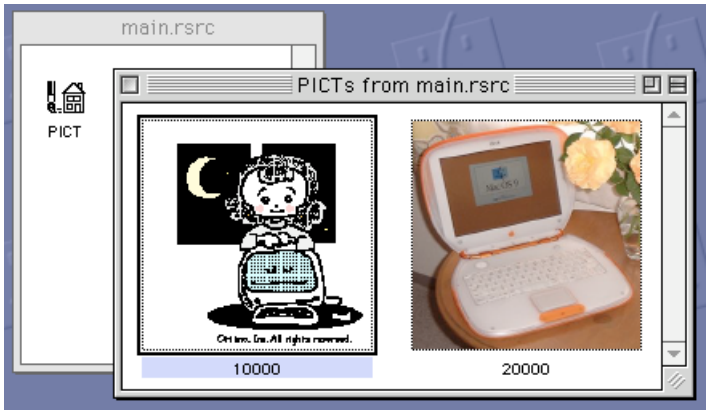
タイプの方はちゃんと来るのですが...。こうした不都合は、Mac OS X 10.1が、まだまだ開発途上だという証拠ですね。Mac OS X 10.2での改善に期待したいと思います。

Mac OS 8/9でウィンドウを移動した時、先んじてその外枠だけが移動し、マウスを放した瞬間にウィンドウ本体の描画がなされます。しかし、Mac OS Xでは、ドラッグ中でもウィンドウの内容がリアルタイムに描画されます。移動時の表示は非常にスムーズで、ユーザに対する視覚的効果も良好です。Dockに登録されているウィンドウが拡大しながら飛び出す（スケールエフェクト）時や、ウィンドウのLive Risizeingでも同じ効果を提供しています。残念なら、Dockの「ジニーエフェクト」をCarbonアプリケーションで使うためのAPIは用意されていません。しかし、ShowWindow()、HideWindow()、MoveWindow()、SizeWindow()といった旧Window Manager APIの代わりにTransitionWindow()を使うことで、Mac OS Xライクの良好な視覚効果を得ることが出来ます。Mac OS X用のCarbonアプリケーションでは、旧APIをTransitionWindow()に差し換えておくことをお奨めします。

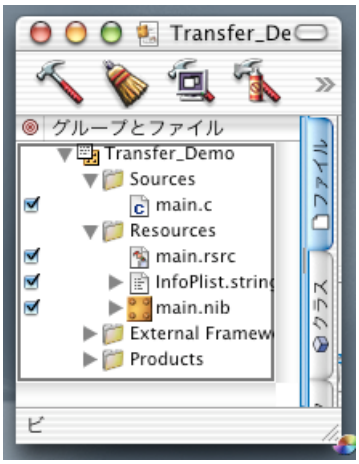
ここでは「Transfer_Demo」というアプリケーションを紹介します。このCarbonアプリケーションは、Develope ToolsのProject BuilderとInterface Builderで開発されています。Interface Builderで「MainWindow」「SheetWidnow」と名付けたふたつのウィンドウが作られ、main.nibというファイル名で保存されています。



両ウィンドウのImagWellコントロールに表示されているPICT画像は、リソースファイル（main.rsrc）として用意されています。



このリソースファイルもNibファイルと同様に、Project Builderのプロジェクトウィンドウのリストに加えておきます。



Project Builderに登録されているソースファイルはmain.cのみです。このプロジェクトをメイクしてアプリケーションを起動すると、Interface Builderで用意しておいたMainWindowの方がオープンします。そこまでの処理は、main.cソースファイルのmain()ルーチンに記述されています。

```
int main( int argc, char* argv[] )
{
    IBNibRef    nibRef;
    WindowRef   window;

    if( ! CreateNibReference( CFSTR("main"),&nibRef ) )
    {
        SetMenuBarFromNib( nibRef,CFSTR("MenuBar") );
        if( ! CreateWindowFromNib( nibRef,CFSTR("MainWindow"),&window ) )
        {
            setUpWindowEvent( window );
            TransitionWindow( window,kWindowZoomTransitionEffect,kWindowShowTransitionAction,NULL );
        }
        DisposeNibReference( nibRef );
        RunApplicationEventLoop();
    }
    return( noErr );
}
```

最初にNibファイルからメニューバーとメインウィンドウを呼び込みます。ウィンドウに配置されているボタンコントロールには、Carbonイベントで識別するために、Interface Builderにより固有のコマンドID（HI Command）が割り付けられています。次に、CreateWindowFromNib()で得たWindowRefをsetUpWindowEvent()に渡し、メインウィンドウにCarbon Event Handlerをインストールします。

```
void setUpWindowEvent( WindowRef window )
{
    EventTypeSpec   list[]={ { kEventClassCommand, kEventCommandProcess },
                             { kEventClassWindow, kEventWindowClose } };
    EventHandlerRef ref;

    InstallWindowEventHandler( window,NewEventHandlerUPP( myWindowEventHandler ),2,list,(void *)window,&ref );
}
```

InstallWindowEventHandler()で登録しているEventTypeSpecは2種類です。ひとつは、ボタンコントロールに割り付けたコマンドIDを認識するための物です。イベントクラスが「 kEventClassCommand」イベント種類が「kEventCommandProcess」のタイプです。もう片方は、ウィンドウのCloseボックスのクリックを認識するための物です。イベントクラスが「 kEventClassWindow」イベント種類が「kEventWindowClose」のタイプです。またユーザデータ（UserData）としてイベント対象であるウィンドウのWidnwRefが渡されています。

前準備が終了したら、ShowWindow()の代わりにTransitionWindow()を呼んでメインウィンドウを表示します。TransitionWindow()の最初の引数は、対象となるウィンドウのWindowRefです。2番目の引数にはどんなエフェクトを利用するのかを、3番目の引数にはどんなアクションに用いるかを設定します。利用可能なエフェクトとアクショについては、Universal Interfacesの「MacWindows.c」に定義されていますので参照してみてください。コメント文として処理内容が詳しく解説されています。

```
typedef UInt32 WindowTransitionEffect;

enum {

        kWindowZoomTransitionEffect   = 1,
        kWindowSheetTransitionEffect  = 2,
        kWindowSlideTransitionEffect  = 3
    };

typedef UInt32 WindowTransitionAction;

enum {
        kWindowShowTransitionAction   = 1,
        kWindowHideTransitionAction   = 2,
        kWindowMoveTransitionAction   = 3,
        kWindowResizeTransitionAction = 4
    };
```

ウィンドウを表示する場合には、エフェクトにkWindowZoomTransitionEffectを、

アクションにkWindowShowTransitionActionを選択します。最後の引数は、どの位置からズームを開始するのかを決定するための矩形情報（Rect）です。ここにNULLを代入しておけばディフォルト位置が選択されます。これでTransitionWindow()を実行すると、Mac OS 8/9のFinderと同じ視覚効果でウィンドウがオープンされます。グレイの矩形枠が徐々にポップアップして大きくなっていく感じです。ウィンドウを閉じる時にもTransitionWindow()を使いたければ、アクションにkWindowHideTransitionActionを選択します。残念ながら現状のMac OS X 10.1では、Dockの「ジニーエフェクト」ライクな視覚効果を得ることはできません。Mac OS Xの次期バージョンでは、このAPIの機能が拡張され、ジニーエフェクトと同じ視覚効果を得られることを望みたいと思います。

続いて、setUpWindowEvent()でCarbon Event Handlerルーチンとしてインストールされてるmy WindowEventHandler()を見てみます。

```
static pascal OSStatus myWindowEventHandler(EventHandlerCallRef myHandler, EventRef event, void* userData)
{
    OSStatus        ret=eventNotHandledErr;
    WindowRef       wptr,wptr1;
    IBNibRef        nibRef;
    unsigned long   ekind;
    long            cls;
    Rect            drt;
    HICommand       cmd;

    cls=GetEventClass( event );
    ekind=GetEventKind( event );
    if( cls==kEventClassCommand )
    {
        wptr=(WindowRef)userData;
        GetWindowBounds( wptr,kWindowStructureRgn,&drt );
        switch( ekind )
        {
            case kEventCommandProcess:

                GetEventParameter( event,kEventParamDirectObject,typeHICommand,NULL,sizeof(HICommand),NULL,&cmd );
                switch( cmd.commandID )
                {
                    case 'zom1':

                        InsetRect( &drt,-20,-20 );
                        TransitionWindow( wptr,kWindowSlideTransitionEffect,kWindowResizeTransitionAction,&drt );
                        ret=noErr;
                        break;

                    case 'zom2':

                        InsetRect( &drt,20,20 );
                        TransitionWindow( wptr,kWindowSlideTransitionEffect,kWindowResizeTransitionAction,&drt );
                        ret=noErr;
                        break;

                    case 'move' :

                        OffsetRect( &drt,100,100 );
                        TransitionWindow( wptr,kWindowSlideTransitionEffect,kWindowMoveTransitionAction,&drt );
                        ret=noErr;
                        break;

                    case 'shet':

                        if( ! CreateNibReference( CFSTR("main"), &nibRef) )
                        {
                            if( ! CreateWindowFromNib( nibRef, CFSTR("SheetWindow"), &wptr1 ) )
                            {
                                setUpWindowEvent( wptr1 );
                                ShowSheetWindow( wptr1,wptr );
                                ret=noErr;
                            }
                            DisposeNibReference( nibRef );
                        }
                        break;

                    case 'ok  ':

                        HideSheetWindow( wptr );
                        DisposeWindow( wptr );
                        ret=noErr;
                        break;
                }
                break;
        }
    }
    else if( cls==kEventClassWindow )
    {
        if( ekind==kEventWindowClose )
        {
            QuitApplicationEventLoop();
            ret=noErr;
        }
    }
    return( ret );
}
```

Handlerルーチンは、ウィンドウのボタンがクリックされると呼ばれます。押されたボタンのコマンドIDを調べ適切な処理へ分岐していることが分かります。各ボタンに割り付けられているコマンドIDは、「拡大」が''zom1'、「縮小」が'zom2'、「移動」が'move'、「シート」が'shet'となっています。

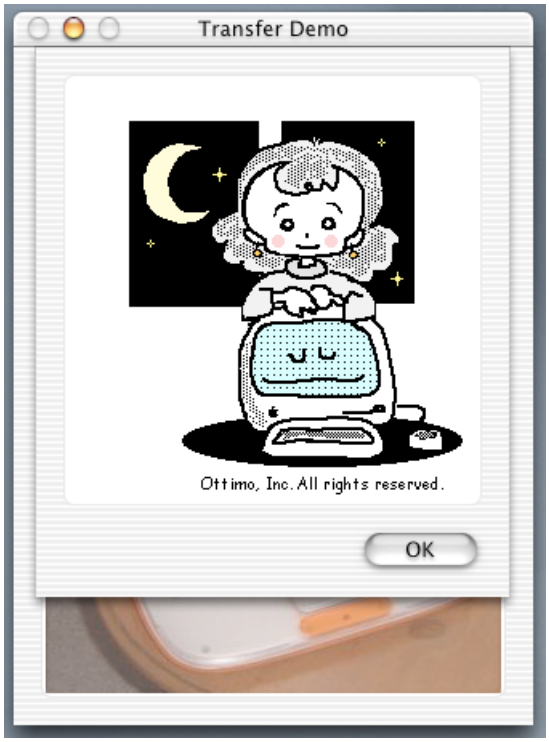
「拡大」ボタンのクリックでは、エフェクトにkWindowSlideTransitionEffectを、ア

クションにkWindowResizeTransitionActionを選択し、TransitionWindow()を実行します。矩形情報には、現在のウィンドウサイズより縦横20ピクセル大きなサイズが代入されています。この設定によりウィンドウが拡大されるわけです。「縮小」ボタンのクリックでは、それとはまったく逆の処理を行い、ウィンドウの縦横サイズを20ピクセルずつ小さくします。「縮小」ボタンを何度もクリックすると、ウィンドウサイズはどんどんと小さくなって行きます。

この機能は、Mac OS X 10.1の「初期設定」ダイアログで利用されています。機能アイコンを選択した時に、表示する内容によってダイアログウィンドウの縦サイズが延びたり縮んだりする時が、それに相当します。

「移動」ボタンのクリックでは、エフェクトにkWindowSlideTransitionEffectを、アクションにkWindowMoveTransitionActionを選択し、TransitionWindow()を実行します。矩形情報には、現在より縦横へ100ピクセルだけ移動した情報を設定します。これにより、ウィンドウは画面上をスライドするように移動します。「移動」ボタンをマウスクリックすると、ウィンドウがマウスから逃げるような感じになります。

最後の「シート」ボタンは、TransitionWindow()ではなくShowSheetWindow()を使い、Nibファイルから読み込んだSheetWindowを表示します。SheetWindowの「OK」ボタンにはコマンドIDとして'ok 'が割り付けられています。「OK」ボタンがクリックされると、Handlerルーチンは、HideSheetWindow()とDisposeWindow()を実行してウィンドウを閉じる処理を実行します。以下が、MainWindowで「シート」ボタンをクリックした時の様子です。



MainWindowのCloseボタンがクリックされた時には、QuitApplicationEventLoop()を実行してアプリケーションを終了させる処理がHandlerルーチンに記述されています。

「Transfer_Demo」サンプルアプリケーションは、以下のサイトに登録されていますので、試してみてください。Mac OS X 10.1と最新版のDeveloper Toolsが必要です。

http://www.ottimo.co.jp/library/

次回は、ウィンドウのグループ化について解説します。この機能を利用すると、マウスドラッグにより複数のウィンドウを同時に移動させるようなことが可能となります。

[小池邦人/オッティモ< http://www.ottimo.co.jp/>]