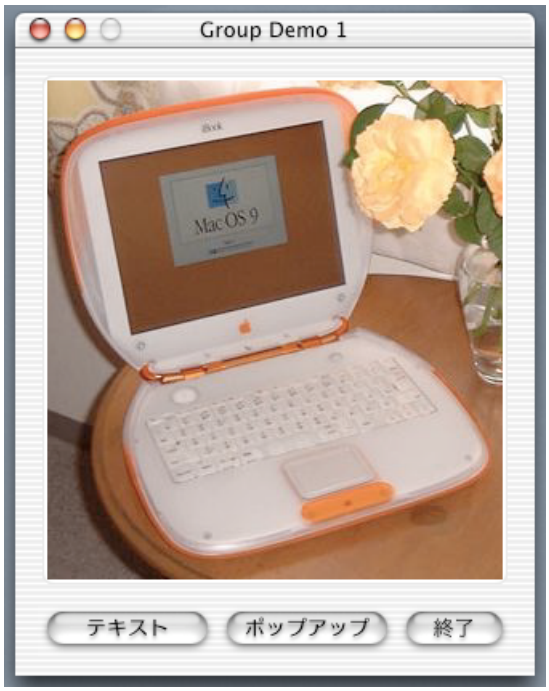
小池邦人のプログラミング日記》2002/2/15 – ウィンドウのグループ化　その１

今回は、Mac OS X 10.1 で利用できるようになった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。

今年の WWDC（Worldwide Developer Conference）は、5/6 から 5/10 までアメリカのサンホセ市で開催される予定です。Apple 社のサイトでは、まだ参加登録は始まっていないようですが、そろそろ開始されるのではないでしょうか？　今年のスケジュールでは、開始日がゴールデンウィークの終わりと重なっていますので、海外へ出るために国内移動をしなければならないデベロッパーにとっては、交通機関の予約などを早めにしておく必要があります。ところが、毎年アップル社がデベロッパーに斡旋する日本からのツアーは、スケジュールが決定するのが遅れがちでした。国内事情も考慮に入れて、今年は早めのスケジュール決定をお願いしたいと思います。

それでは、ウィンドウのグループ化についての話を始めます。Sheet ウィンドウが降りている場合でも、その親ウィンドウのタイトルバーをドラッグすることで、両ウィンドウを同時に移動させることが出来ます。これは、両方のウィンドウが「グループ化」されており、ユーザの操作に対して行動を伴にするようにシステム側で管理されているからです。ここでは、Window Manager の「グループ化」API を利用した「Group_Demo1」というサンプルアプリケーションを紹介いたします。アプリケーションを起動すると、「テキスト」「ポップアップ」「終了」という 3 つのボタンが配置されたウィンドウがオープンします。各ボタンコントロールに対応するコマンド ID は、先んじて Interface Builder で設定してあります。「テキスト」には‘text’ が、「ポップアップ」には‘popu’が、「終了」には‘quit’が割り付けられています。まずは、アプリケーションの機能を紹介することにしましょう。



「テキスト」ボタンをクリックすると、ピクチャ上にテキスト入力カラムがオープンし、そこでテキスト入力が可能となります。このテキスト入力カラムは最初からウィンドウに配置されているわけではありません。新たにオープンされた子ウィンドウの EditText コントロールが使われます。その証拠に、カラムの回りにはウィンドウと同じ「影」が表示されていることが分かります。このテキスト入力用ウィンドウは、親ウィンドウにと同じグループに属しています。よって、親ウィンドウに対す移動や最小化などの操作は、ちゃんと子ウィンドウ側にも反映されます。

Microsoft 社の Excel ではセル上に影付きの入力カラムが表示されますが、そうした機能もこの仕組みを利用すれば可能となります。ここでのテキスト入力カラムは、もう一度「テキスト」ボタンをクリックすることで消える仕組みになっています。

「ポップアップ」ボタンをクリックすると、ピクチャの「一点」から、Dock の「ジニーエフェクト」のように別のウィンドウが湧き出してきます。下になったウィンドウは、新しいウィンドウが表示し終わった瞬間に隠れます。新しいウィンドウには「OK」ボタンがあり、それをクリックすると、今度は先程の一点に吸い込まれるように消えます。ウィンドウが吸い込まれた瞬間に、前のウィンドウが再表示されます。この機能は Sheet ウィンドウが、親ウィンドウの上辺から降りてくる特徴をうまく利用しています。親ウィンドウのサイズを限りなく小さく（縦横 1 ピクセル）にすれば、Sheet ウィンドウは、あたかも 1 点から湧き出したような現れ方をするわけです。

それでは、サンプルアプリケーションのソースコードを見てみます。まずは、メニューバーとメインウィンドウ（MainWindow）を Nib ファイルから読み込みます。続いて setUpWindowEvent()で Carbon Event Handler ルーチンをインストールしていますが、この仕組みについては前回の「Transfer_Demo」サンプルとまったく同じですので説明は省略します。

```
WindowGroupRef  sys_group;          /*  Group Window References */
WindowRef       sys_text;           /*  Popup TextEdit Window   */

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

    if( ! CreateNibReference( CFSTR("main"),&nibRef ) )
    {
        SetMenuBarFromNib( nibRef,CFSTR("MenuBar") );
        if( ! CreateWindowFromNib( nibRef,CFSTR("MainWindow"),&window ) )
        {
            setUpWindowEvent( window );
            flag=kWindowGroupAttrMoveTogether+kWindowGroupAttrSharedActivation+kWindowGroupAttrHideOnCollapse;
            CreateWindowGroup( flag,&sys_group );
            SetWindowGroup( window,sys_group );
            TransitionWindow( window,kWindowZoomTransitionEffect,kWindowShowTransitionAction,NULL );
        }
        DisposeNibReference( nibRef );
        RunApplicationEventLoop();
    }
    return( noErr );
}
```

ウィンドウをあるグループに属するようにするには、先んじて登録すべきグループを作成しておく必要があります。これには Window Manager の CreateWindowGroup()を用います。得られた WindowGroupRef を、登録したいウィンドウの WindowRef と一緒に SetWindowGroup()に渡せば、グループへの登録が完了します。WindowGroupRef は後から利用するので、外部変数の sys_group に保存しておきます。この処理で重要なことは、CreateWindowGroup()で作成するグループの性質をアトリビュートフラグ（WindowGroupAttributes）で設定することです。今回のグループには「ウィンドウの移動」「選択された場合のアクティベート」「Dock への引き込み」などの性質を共有させることにします。つまり、グループのどれかのウィンドウが Dock に引き込まれれば、同時にすべてのウィンドウが引き込まれるわけです。アトリビュートフラグの種類や内容については、Universal Interfaces の「MacWindows.h」に定義されています。

```
typedef UInt32 WindowGroupAttributes;

enum {
        kWindowGroupAttrSelectAsLayer = 1 << 0,
        kWindowGroupAttrMoveTogether  = 1 << 1,
        kWindowGroupAttrLayerTogether = 1 << 2,
        kWindowGroupAttrSharedActivation = 1 << 3,
        kWindowGroupAttrHideOnCollapse = 1 << 4
};
```

グループの設定が終わったら、TransitionWindow()でウィンドウ（MainWindow）をオープンします。ユーザによりウィンドウ上のボタンがクリックされると、先んじてインストールされている Carbon Event Handler ルーチンが呼ばれます。

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

    cls=GetEventClass( event );
    ekind=GetEventKind( event );
    if( cls==kEventClassCommand )
    {
        wptr=(WindowRef)userData;
        switch( ekind )
        {
            case kEventCommandProcess:

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

                        handlTextEdit( wptr );
                        ret=noErr;
                        break;

                    case 'popu':

                        handlPopUp( wptr );
                        ret=noErr;
                        break;

                    case 'ok  ':

                        HideSheetWindow( wptr );
                        DisposeWindow( wptr );
                        if( ! GetIndexedWindow( sys_group,1,kWindowGroupContentsReturnWindows,&wptr1 ) )
                            ShowWindow( wptr1 );
                        ret=noErr;
                        break;
                }
                break;
        }
    }
    else if( cls==kEventClassWindow )
    {
        if( ekind==kEventWindowClose )
        {
            QuitApplicationEventLoop();
            ret=noErr;
        }
    }
    return( ret );
}
```

「テキスト」ボタンがクリックされると handlTextEdit()ルーチンが実行され、「ポップアップ」ボタンがクリックされると handlPopUp()ルーチンが実行されます。「OK 」ボタンはポップアップされるウィンドウの方にあります。このボタンをクリックすると HideSheetWindow()が実行され、子ウィンドウは親ウィンドウへ引き込まれます。その後、ShowWindow()を使い隠していたウィンドウを再表示するのですが、この時に必要となる WindowRef は、GetIndexedWindow()で得ています。これは、ウィンドウ番号と kWindowGroupContentsReturnWindows を渡すことで、グループに属するウィンドウの WindowRef を得るための API です。

まずは「テキスト」ボタンをクリックでした時に呼ばれる handlTextEdit()ルーチンを見てみます。

```
void handlTextEdit( WindowRef window )
{
    IBNibRef    nibRef;
    Rect        drt;
    ControlRef  chd;

    GetWindowBounds( window,kWindowStructureRgn,&drt );
    if( sys_text )
    {
        DisposeWindow( sys_text );
        sys_text=NULL;
    }
    else
    {
        if( ! CreateNibReference( CFSTR("main"), &nibRef ) )
        {
            if( ! CreateWindowFromNib( nibRef, CFSTR("EditWindow"), &sys_text ) )
            {
                MacMoveWindow( sys_text,drt.left+62,drt.top+250,0 );
                SetWindowGroup( sys_text,sys_group );
                ChangeWindowGroupAttributes( sys_group,kWindowGroupAttrSelectAsLayer,kWindowGroupAttrLayerTogether );
                SelectWindow( sys_text );
                ChangeWindowGroupAttributes( sys_group,kWindowGroupAttrLayerTogether,kWindowGroupAttrSelectAsLayer );

                getMyControlRef( sys_text,20,&chd );
                setMyControlString( chd,"\pThis is the EditText." );
                ShowWindow( sys_text );
                AdvanceKeyboardFocus( sys_text );
            }
            DisposeNibReference( nibRef );
        }
    }
}

void getMyControlRef( WindowRef window,long id,ControlRef *chd )
{
    ControlID   cid;

    cid.signature='cntl';
    cid.id=id;
    GetControlByID( window,&cid,chd );
}

void setMyControlString( ControlRef chd,Str255 str )
{
    SetControlData( chd,kControlNoPart,kControlStaticTextTextTag,*str,str+1 );
}
```

最初に Nib ファイルから TextEdit コントロール用のウィンドウ（EditWindow）を作成し、適切な位置へと移動させています。オープンする前に、SetWindowGroup()で MainWindow と同グループに登録し、それの表側に表示させるために SelectWindow()を実行しています。ChangeWindowGroupAttributes()は、グループに新たに追加するアトリビュートと、グループから外してしまうアトリビュートを同時に設定できる API です。ここでは、一時的にウィンドウ選択を可能にするために「個別ウィンドウの選択が不可」というアトリビュート（kWindowGroupAttrSelectAsLayer）を外しています。これを外さないと、SelectWindow()を実行しても EditWindow が MainWindow の表側に現れません。最後は、getMyControlRef()で TextEdit コントロールの ControlRef 得て、サンプル文字列（This is the EditText.）を登録しています。ShowWindow()で EditWindow を表示したら、AdvanceKeyboardFocus()を実行して TextEdit コントロールをテキスト入力可能な状態にしておきます。

今度は「ポップアップ」ボタンをクリックした時に呼ばれる handlPopUp()ルーチンを見てみます。

```
void handlPopUp( WindowRef window )
{
    WindowRef   wptr1,wptr2;
    IBNibRef    nibRef;
    Rect        drt;

    if( sys_text )
    {
        DisposeWindow( sys_text );
        sys_text=NULL;
    }
    GetWindowBounds( window,kWindowContentRgn,&drt );
    drt.top+=92;
    drt.left+=128;
    drt.bottom=drt.top+1;
    drt.right=drt.left+1;
    if( !  CreateNewWindow( kOverlayWindowClass,kWindowNoAttributes,&drt,&wptr2 ) )
    {
        ShowWindow( wptr2 );
        if( ! CreateNibReference( CFSTR("main"), &nibRef) )
        {
            if( ! CreateWindowFromNib( nibRef, CFSTR("SheetWindow"), &wptr1 ) )
            {
                setUpWindowEvent( wptr1 );
                SetWindowGroup( wptr1,sys_group );
                ShowSheetWindow( wptr1,wptr2 );
                HideWindow( window );
            }
            DisposeNibReference( nibRef );
            DisposeWindow( wptr2 );
        }
    }
}
```

ポップアップされる Sheet ウィンドウ（SheetWindow）は、Nib ファイルから作成されています。しかし、このウィンドウは MainWindow を親にはせず、CreateNewWindow()で作成された縦横 1 ピクセルサイズの Overlay（下地が透明）ウィンドウを親として使っています。これが、SheetWindow が MainWindow の一点から湧き出したように見えるトリックの種明かしです。SheetWindow を ShowSheetWindow()でオープンしたら、MainWindow の方は HideWindow()で隠してしまいます。最後に、親ウィンドウの方も削除するのですが、SheetWindow は戻る場所を記憶していますので問題はありません。親ウィンドウに利用している Overlay ウィンドウの性質については、別の機会に詳しく説明します。

ここで解説した「Group_Demo1」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の話の続きとなります。ウィンドウタイトルに配置することができるようになった ToolBar ボタンの利用方法についても解説します。