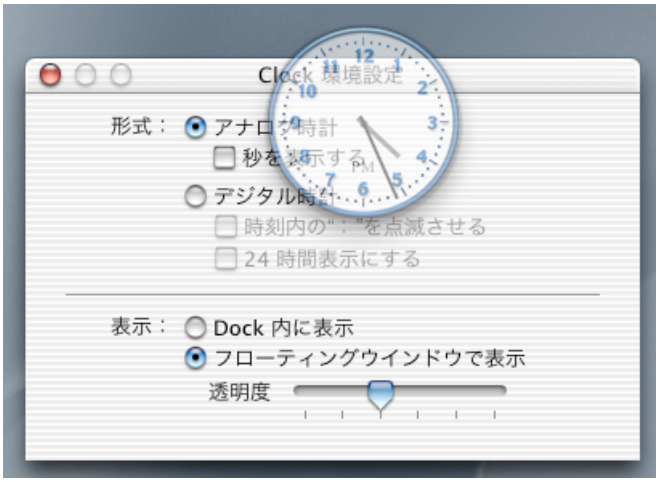
小池邦人のプログラミング日記》2002/3/11

～Overlay ウィンドウを利用する

今回は、Overlay ウィンドウ（下地が透明なウィンドウ）の利用方法を解説してみます。Overlay ウィンドウとは、Mac OS X から採用された新機能で、Quartz 2D による描画を使えば半透明のオブジェクトをモニタ上に表示することも可能です。

Group_Demo1 アプリケーションを紹介した時に、Sheet ウィンドウの親ウィンドウとして Overlay ウィンドウを使いました。Overlay ウィンドウとは、Mac OS X 10.1 から採用されたディフォルトで下地が透明なウィンドウのことです。下地が透明ということは、QuikDraw で図形を描画すると、それだけが画面上に浮き上がることになります。また、QuickDraw の代わりに Quartz 2D で半透明な図形を描画するとも可能です。Mac OS X 10.1 の Applications フォルダに「Clock」というアプリケーションがあります。これが時計の表示に使っているウィンドウが Overlay ウィンドウだと思われます。Quartz 2D の描画により半透明な表示を実現しています。

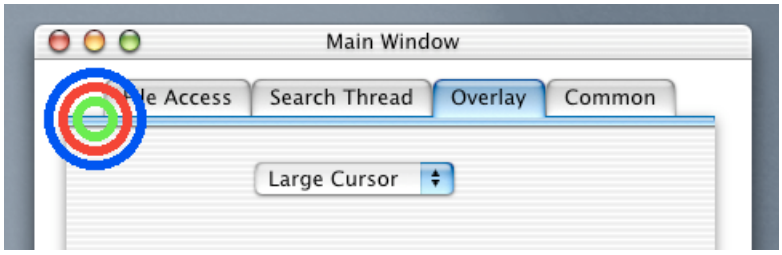


Overlay ウィンドウを作成するのには CreatWindow()を使います。この時のウィンドウクラス（WindowClass）には kOverlayWindowClass を選択します。ウィンドウクラスの

種類は、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。同時に、どのウィンドウクラスがどの CarbonLib のバージョンで利用可能なのかが併記されています。そちらも確認しておいた方が良いでしょう。

```
/*-------------------------------------------------------------------------*/
/* • Window Classes                                                        */
/*-------------------------------------------------------------------------*/
typedef UInt32 WindowClass;
enum {
  kAlertWindowClass             = 1L,   /* "I need your attention now."*/
  kMovableAlertWindowClass      = 2L,   /* "I need your attention now, but I'm kind enough to let you switch out of this app to do other things."*/
  kModalWindowClass             = 3L,   /* system modal, not draggable*/
  kMovableModalWindowClass      = 4L,   /* application modal, draggable*/
  kFloatingWindowClass          = 5L,   /* floats above all other application windows*/
  kDocumentWindowClass          = 6L,   /* document windows*/
  kUtilityWindowClass           = 8L,   /* system-wide floating windows (TSM, AppleGuide) (available in CarbonLib 1.1)*/
  kHelpWindowClass              = 10L,  /* help window (no frame; coachmarks, help tags ) (available in CarbonLib 1.1)*/
  kSheetWindowClass             = 11L,  /* sheet windows for dialogs (available in Mac OS X and CarbonLib 1.3)*/
  kToolbarWindowClass           = 12L,  /* toolbar windows (above documents, below floating windows) (available in CarbonLib 1.1)*/
  kPlainWindowClass             = 13L,  /* plain window (in document layer)*/
  kOverlayWindowClass           = 14L,  /* transparent window which allows 'screen' drawing via CoreGraphics (Mac OS X only)*/
  kSheetAlertWindowClass        = 15L,  /* sheet windows for alerts (available in Mac OS X after 10.0.x and CarbonLib 1.3)*/
  kAltPlainWindowClass          = 16L,  /* alternate plain window (in document layer) (available in Mac OS X after 10.0.x and CarbonLib 1.3)*/
  kAllWindowClasses             = (unsigned long)0xFFFFFFFF /* for use with GetFrontWindowOfClass, FindWindowOfClass, GetNextWindowOfClass*/
};
```

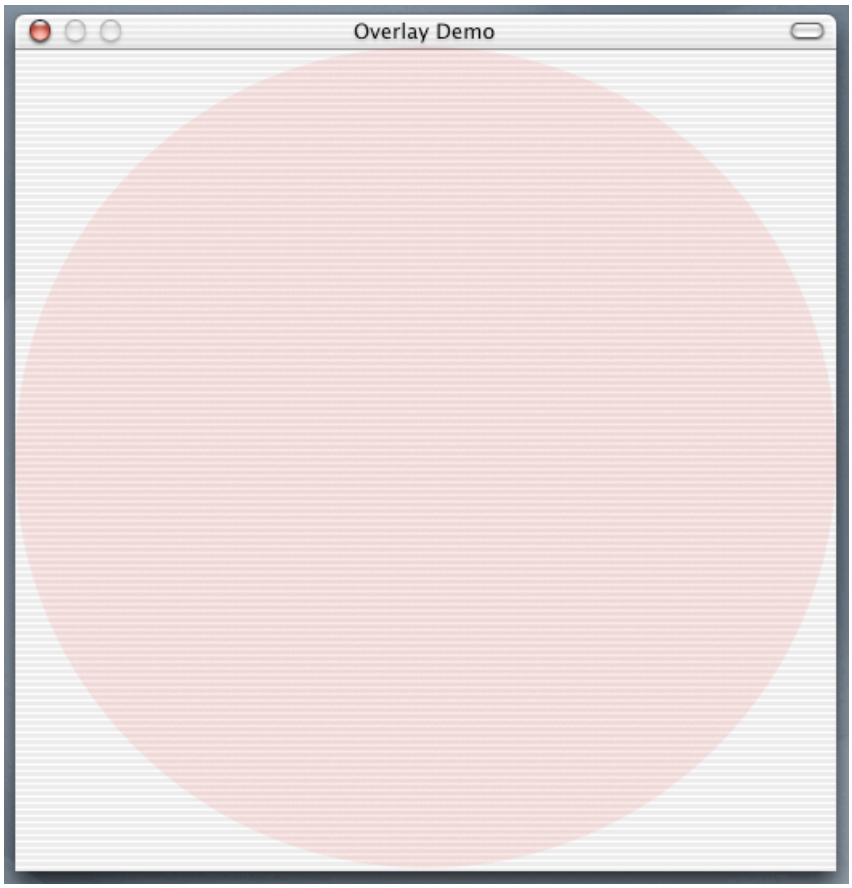
Overlay ウィンドウを使う簡単なサンプルアプリケーションは、CarbonLib 1.5 SDK の「Sample Code」フォルダにある「GrabBag」です。表示されたウィンドウの「Overlay」タブを選択して、メニューから「Large Cursor」を選択すると、マウスカーソルの位置にカラフルな三重丸が表示されるようになります。ソースコードを見ると、QuickDraw で Overlay ウィンドウに円を描き、それをマウスカーソルにあわせて移動させています。Overlay ウィンドウは下地が透明なだけであり、それ以外の性質は一般的なウィンドウと大差ないことが理解できます。



それでは、Overlay ウィンドウを利用した「Overlay_Demo」サンプルアプリケーションを紹介します。Nib ファイルから、Overlay ウィンドウの下敷きとして使うウィンドウ（TitleWindow）を呼び出します。続いて CreateNewWindow()で Overlay ウィンドウを作成します。表示位置は、TitleWindow とピッタリ同じにします。両ウィンドウをグループ化し、マウスドラッグで同時に移動できるように設定します。Overlay ウィンドウを表側に持ってくるためのグループアトリビュートの調整も忘れないでください。この処理は、ChangeWindowGroupAttributes()ルーチンで行います。

```
WindowGroupRef  sys_group;      /*  Group Window References */

int main(int argc, char* argv[])
{
    float           pi=3.14159265;
    WindowRef       window,wptr;
    IBNibRef        nibRef;
    long            flag;
    CGContextRef    cont;
    Rect            drt;

    if( ! CreateNibReference( CFSTR("main"),&nibRef ) )
    {
        SetMenuBarFromNib( nibRef,CFSTR("MenuBar") );
        if( ! CreateWindowFromNib( nibRef,CFSTR("TitleWindow"),&window ) )
        {
            setUpTitleWindowEvent( window );

            flag=kWindowGroupAttrMoveTogether+kWindowGroupAttrSharedActivation;
            CreateWindowGroup( flag,&sys_group );
            SetWindowGroup( window,sys_group );

            SetRect( &drt,0,0,512,512 );
            OffsetRect (&drt,100,100 );
            if( !  CreateNewWindow( kOverlayWindowClass,kWindowStandardHandlerAttribute,&drt,&wptr ) )
            {
                setUpOverlayWindowEvent( wptr );
                SetWindowGroup( wptr,sys_group );

                SelectWindow( wptr );
                ShowWindow( wptr );;
                ShowWindow( window );
                ChangeWindowGroupAttributes( sys_group,kWindowGroupAttrLayerTogether,kWindowGroupAttrSelectAsLayer );

                if( ! CreateCGContextForPort( GetWindowPort(wptr),&cont ) )
                {
                    CGContextSetRGBFillColor( cont,1.0,0.0,0.0,0.1 );
                    CGContextAddArc( cont,256.0,256.0,256.0,pi*2.0,0.0,0 );
                    CGContextFillPath( cont );
                    CGContextFlush( cont );
                    CGContextRelease( cont );
                }
            }
        }
        DisposeNibReference( nibRef );
        RunApplicationEventLoop();
    }
    return( noErr );
}
```

ふたつのウィンドウを表示したら、Overlay ウィンドウにはウィンドウいっぱいに赤色の半透明な円を描画しておきます。この描画には、QuickDraw ではなく Quartz 2D を利用します。



ふたつのウィンドウの Carbon Event Handler ルーチンは、setUpTitleWindowEvent()と setUpOverlayWindowEvent()により別々にインストールされます。

```
void setUpOverlayWindowEvent( WindowRef window )
{
    EventTypeSpec   list1[]={ { kEventClassWindow, kEventWindowClickContentRgn } };
    EventHandlerRef ref;

    InstallWindowEventHandler( window,NewEventHandlerUPP( overlayWindowEventHandler ),1,list1,NULL,&ref );
}

void setUpTitleWindowEvent( WindowRef window )
{
    EventTypeSpec   list1[]={ { kEventClassWindow, kEventWindowToolbarSwitchMode },
                              { kEventClassWindow, kEventWindowClose }
                            };
    EventHandlerRef ref;

    InstallWindowEventHandler( window,NewEventHandlerUPP( titleWindowEventHandler ),2,list1,NULL,&ref );
}
```

TitleWindow には、ToolBar ボタンのクリックを認識するために、Carbon イベントの種類として kEventWindowToolbarSwitchMode をインストールしておきます。Overlay ウィンドウには、マウスドラッグによる落書き機能を実装します。そのために、マウスクリックを認識できる kEventWindowClickContentRgn をインストールしておきます。

Overlay ウィンドウ内でマウスクリックが起こると、そのマウス位置が Handler ルーチンの overlayWindowEventHandler()に渡ります。ちなみに、Overlay ウィンドウの透明部分はマウスクリックの対象にはなりません。イベント情報からマウス位置を抽出するのには ConvertEventRefToEventRecord()を使います。Quart 2D による図形の描画は paintOverlayWindow()で行われます。

```
static pascal OSStatus overlayWindowEventHandler(EventHandlerCallRef myHandler, EventRef event, void* userData)
{
    short       ret=eventNotHandledErr;
    WindowRef   wptr=NULL;
    EventRecord eve;
    CGrafPtr    cptr;

    if( GetEventClass( event )==kEventClassWindow )
    {
        GetEventParameter( event,kEventParamDirectObject,typeWindowRef,NULL,sizeof(WindowRef),NULL,&wptr );
        switch( GetEventKind(event) )
        {
            case kEventWindowClickContentRgn:

                GetPort( &cptr );
                SetPortWindowPort( wptr );
                ConvertEventRefToEventRecord( event,&eve );
                GlobalToLocal( &eve.where );
                paintOverlayWindow( wptr,eve.where );
                SetPort( cptr );
                ret=noErr;
                break;
        }
    }
    return( ret );
}
```

paintOverlayWindow()ルーチンでは、マウスドラッグにより半透明の青色の円を描画し

ます。半透明の図形描画と、描画を赤い円だけに制限するクリッピング処理には、Quartz 2D の能力を使っています。

```
void paintOverlayWindow(WindowRef window,Point pt )
{
    float               pi=3.14159265;
    MouseTrackingResult res=0;
    CGContextRef        cont;
    Point               pt1;
    Rect                drt;

    GetWindowBounds( window,kWindowContentRgn,&drt );
    if( ! CreateCGContextForPort( GetWindowPort(window),&cont ) )
    {
        CGContextBeginPath( cont );
        CGContextAddArc( cont,256.0,256.0,256.0,pi*2.0,0.0,0 );
        CGContextClosePath( cont );
        CGContextClip( cont );

        CGContextSetRGBFillColor( cont,0.0,0.0,0.0,1.0,0.1 );
        while( res!=kMouseTrackingMouseReleased )
        {
            TrackMouseLocation( GetWindowPort( window ),&pt1,&res );
            pt1.v=(drt.bottom-drt.top)-pt1.v;

            CGContextBeginPath( cont );
            CGContextAddArc( cont,(float)pt1.h,(float)pt1.v,20.0,pi*2.0,0.0,0 );
            CGContextFillPath( cont );
            CGContextFlush( cont );
            pt=pt1;
        }
        CGContextRelease( cont );
    }
}
```

続いて TitleWindow の Handler ルーチンである titleWindowEventHandler()を見てみます。ToolBar ボタンがクリックされると、TransitionWindow()によりウィンドウサイズの縦方向をゼロにします。つまり、TitleWindow はタイトルバーだけなってしまうわけです。再度クリックすればウィンドウサイズは元に戻ります。ToolBar ボタンをクリックした時にどちらに状態に切り換えるのかは、GetWindowBounds()で得た矩形サイズから判断しています。

```
static pascal OSStatus titleWindowEventHandler(EventHandlerCallRef myHandler, EventRef event, void* userData)
{
    short       ret=eventNotHandledErr;
    WindowRef   wptr=NULL;
    Rect        drt;

    if( GetEventClass( event )==kEventClassWindow )
    {
        GetEventParameter( event,kEventParamDirectObject,typeWindowRef,NULL,sizeof(WindowRef),NULL,&wptr );
        switch( GetEventKind(event) )
        {
            case kEventWindowToolbarSwitchMode:

                GetWindowBounds( wptr,kWindowStructureRgn,&drt );
                if( drt.bottom-drt.top==22 )
                    drt.bottom=drt.top+534;
                else
                    drt.bottom=drt.top+22;
                TransitionWindow( wptr,kWindowSlideTransitionEffect,kWindowMoveTransitionAction,&drt );
                ret=noErr;
                break;

            case kEventWindowClose:

                QuitApplicationEventLoop();
                ret=noErr;
                break;
        }
    }
    return( ret );
}
```

ToolBar ボタンのクリックにより下敷き部分が上に登り、Overlay ウィンドウだけが画面に表示されることになります。ただし、両ウィンドウはグループ化されていますので、タイトルバーをドラッグすれば Overlay ウィンドウも一緒に移動します。また、下敷き部分が消えてしまっても、Overlay ウィンドウへの描画はそのまま継続することが可能です。

ここで解説した「Overlay_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、オブジェクトのセレクションに Overlay ウィンドウを利用してみます。Finder のアイコン選択のような「雰囲気」を目指します。昔からあったペイントソフトの「ウニウニ波線」を Mac OS X ライクに書き換えるわけですね（笑）。

ᔓᔓᔓᔓᔓᔓᔓこの項、以上ᔓᔓᔓᔓᔓᔓᔓ[小池邦人/オッティモ]ᔓᔓᔓᔓᔓᔓᔓ