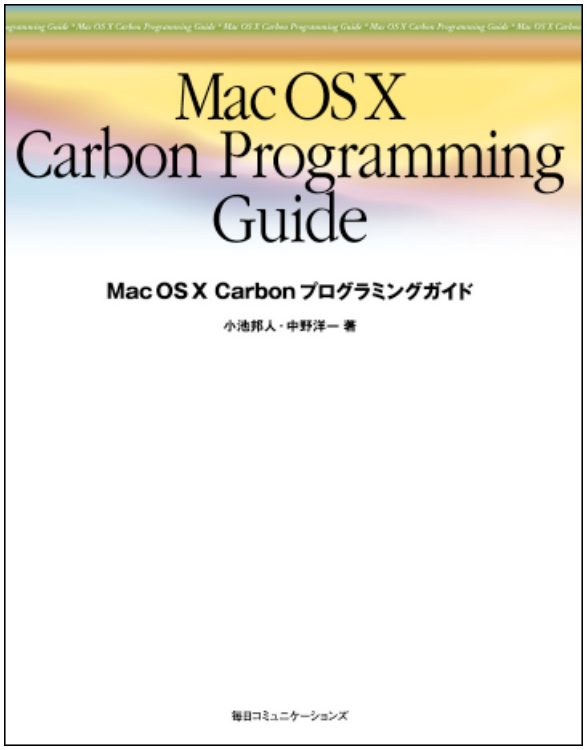
小池邦人のプログラミング日記》2002/3/25

～　Quartz 2D の秘密が明らかになった　その１　～

MACWORLD Expo/Tokyo の会期中に、アップル社が ADC メンバー対象に（Select メンバー以上？）Mac OS X の技術セッションを開催しました。今回は「Overlay ウィンドウを使ったオブジェクトのセレクション」の話をする予定でしたが、そのセッションで、Quartz 2D（Core Graphics）の機能についての「新事実」が判明しましたので、そちらを先行して紹介したいと思います。

その前にちょっと宣伝です（笑）。毎日コミュニケーションズから「Mac OS X Carbon Programming Guide」という技術解説書が出版されます。



この書籍は、私が今までウェブサイトで書き溜めてきた Mac OS X と Carbon についての解説を整理し、足りない部分を加筆した内容となっています。多くの読者から「書

籍としてまとめて欲しい」という要望が届いていたのですが、それを実現できてホッとしております。加えて、MLTE や DataBrowser コントロールなど、まだ言及していない技術については、「金魚すくい」で有名な中野洋一氏が詳しく解説してくれています。Mac OS X と Carbon について非常に「濃い」内容となっていますので、これからも Mac OS X や Carbon と付き合わなければいけない方は、ぜひ参考にしてみてください。

さて、アップル社が開催したセッションは、Mac OS X における「日本語環境」「グラフィックス」「ハードウェア」の各パートに別かれていました。セッションの解説者が Apple 本社から来日した技術担当でしたので、「プレ WWDC 2002」といった雰囲気で、なかなか充実した内容となりました。ただし、深い質問をすると「それは WWDC で詳しく紹介するから参加してね（ウィンク）」といった具合に、WWDC 2002 の宣伝も兼ねていたようです（笑）。出席者には、しっかり「WWDC ツアー」のパンフレット配られていましたし...。「グラフィックス」セッションの大部分は Quartz 2D に関することでした。すでに知っているトピックスが多かったのですが、Q&A コーナーでは以下の 3 つの質問をぶつけてみました。

（1）「Quartz Compositor」はデベロッパー側からアクセスできるのか？それに関する API のリファレンスやドキュメントはあるのか？

（2）ヘッダーファイルに定義されている Quartz 2D の「パタン機能」は、ちゃんと実装されていて利用することが可能なのか？

（3）Quartz API で描画された図形や画像を Carbon アプリケーションからプリントアウトするのにはどうしたら良いのか？

（1）は、「非常に複雑な処理をしているので、ごにょごにょ...」といった感じで、口を濁していましたが（笑）、用意してあったスライドで Compositor の仕組みを簡単に説明してくれました（用意してあるなら質問する前にやってね）。「Quartz Compositor」はウィンドウサーバで、加えて低レベルのイベントハンドリングも受け持っています。結局、Quartz 2D のドキュメントに解説されている内容以上の事実は分からずじまいで、このモジュールの API が開発者に開放されるかどうかについても、言及されませんで

した。

（2）は、「ちゃんと実装している」と話していましたが（ちょっと自信なさげ）、私が「動かないんだけど...」と突っ込むと、「非常に複雑な処理なので、ごにょごにょ...」といった感じで、うまくごまかされてしまいました。サンプルソースならメールで送ることが出来るとフォローしていましたが、「誰でも見られるようにオープンにしてよ！」と頼むと、「まだ、そのレベルじゃないので、ごにょごにょ...」という雰囲気でした（笑）。多分、あの感じでは、完全な実装にはほど遠い状態なのでしょう？　こちらについては、WWDC 2002 で再度追及してみたいと思います。

（3）については、ついに「謎」が解決しました。まず重要な点は、Mac OS X ではプリントの最終出力には、必ず Quartz API が作成した PDF が利用されるという事実です。Cocoa アプリケーションでも Carbon アプリケーションでも、その仕組みは同じだそうです。ただし、Carbon アプリケーションからは、QuickDraw 環境で Quartz API を使わないとプリントアウトは実行できません。これは、いったいどういう事なのでしょうか？

以前に解説したと思いますが、Carbon アプリケーションのプリントアウトには、セッションプリントマネージャを利用します。関連ドキュメントによると、Quartz API で描画した図形をプリントアウトするのには、PMSessionGetGraphicsContext() に kPMGraphicsContextCoreGraphics を代入し、それにより得られた CGContextRef を用いて図形やテキストを描画するように指示されています。

```
short quartzPrint(WindowRef window,PMPrintSession pss,PMPrintSettings set,PMPageFormat form)
{
    short           ret=1;
    CGContextRef    cont;
    CFStringRef     jpb;
    Str255          str;

    GetWTitle( window,str );
    jpb=CFStringCreateWithPascalString( NULL,str,CFStringGetSystemEncoding() );
    PMSetJobNameCFString( set,jpb );
    if( ! PMSessionBeginDocument( pss,set,form ) )
    {
        if( ! ( ret=PMSessionBeginPage( pss,form,NULL ) ) )
        {
            PMSessionGetGraphicsContext( pss,kPMGraphicsContextCoreGraphics,(void**)&cont );
            drawQuartzPrint( cont );
            PMSessionEndPage( pss );
        }
        PMSessionEndDocument( pss );
    }
    return( ret );
}
```

サンプルルーチン内の、drawQuartzPrint()が Quartz API で実際に図形を描画している箇所です。

```
void drawQuartzPrint( CGContextRef cont )
{
    float               pi=3.14159265;
    CGPoint             cpt[4];
    MouseTrackingResult res=0;
    long                i;

    cpt[0].x=300.0;
    cpt[0].y=200.0;
    cpt[1].x=200.0;
    cpt[1].y=150.0;
    cpt[2].x=150.0;
    cpt[2].y=200.0;
    cpt[3].x=200.0;
    cpt[3].y=400.0;

    CGContextSetLineWidth( cont,10.0 );
    CGContextSetLineCap( cont,kCGLineCapButt );
    CGContextSetRGBFillColor( cont,0.0,1.0,1.0,0.5);
    CGContextSetRGBStrokeColor( cont,1.0,0.0,1.0,0.5 );
    for( i=0;i<4;i++ )
    {
        CGContextBeginPath( cont );
        CGContextAddArc( cont,cpt[i].x,cpt[i].y,30.0,pi*2.0,0.0,0 );
        CGContextFillPath( cont );
        CGContextBeginPath( cont );
        CGContextAddArc( cont,cpt[i].x,cpt[i].y,30.0,pi*2.0,0.0,0 );
        CGContextStrokePath( cont );
    }
    CGContextBeginPath( cont );
    CGContextAddLines( cont,cpt,4 );
    CGContextStrokePath( cont );
    CGContextSetLineWidth( cont,2.0 );
    CGContextSetRGBStrokeColor( cont,0.0,0.0,1.0,1.0);
    CGContextSetTextDrawingMode( cont,kCGTextStroke );
    CGContextSelectFont( cont,"Osaka",64.0,kCGEncodingMacRoman );
    CGContextShowTextAtPoint( cont,20,500,"Think Different",15 );
    CGContextFlush( cont );
}
```

ところが、このルーチンを実行してもプリントアウトはなされず、白紙が出力されるだけです。最初は「プリンタのドライバに問題があるのか？」と推測したのですが、ダイアログから「プレビュー」を実行しても同じ結果となるので、ドライバに依存した問題ではないという結論になりました。

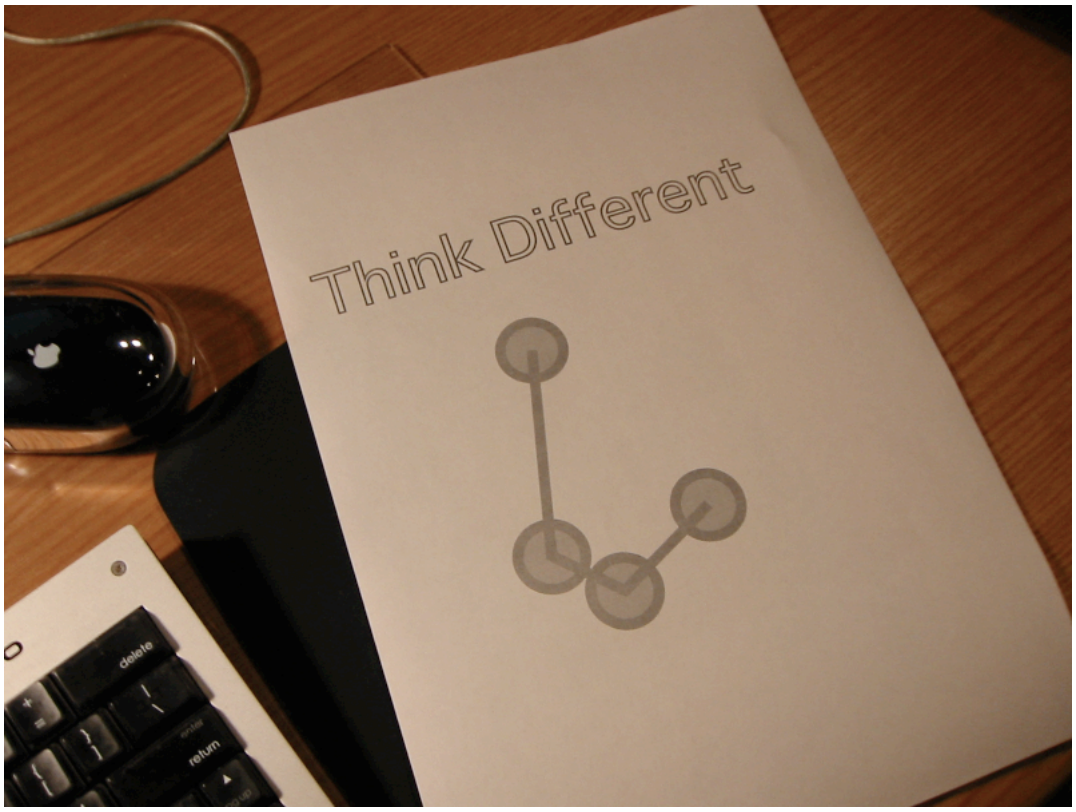
今回のセッションの解説によると、Quartz API による描画を Carbon アプリケーションからダイレクトにプリントアウトすることは不可能なようです。つまり、プリント環境に CGContextRef を選べないのです。具体的には、PMSessionGetGraphicsContext()に kPMGraphicsContextCoreGraphics を代入して CGContextRef を得る方法が使えないのです。その代わり、kPMGraphicsContextQuickdraw を代入し CGrafPtr を得、それを QDBeginCGContext()に渡すことで CGContextRef を得る方法を取ります。

```
short quartzPrint(WindowRef window,PMPrintSession pss,PMPrintSettings set,PMPageFormat form)
{
    CGrafPtr        cptr,port;
    short           ret=1;
    CGContextRef    cont;
    CFStringRef     jpb;
    Str255          str;

    GetPort( &cptr );
    GetWTitle( window,str );
    jpb=CFStringCreateWithPascalString( NULL,str,CFStringGetSystemEncoding() );
    PMSetJobNameCFString( set,jpb );
    if( ! PMSessionBeginDocument( pss,set,form ) )
    {
        if( ! ( ret=PMSessionBeginPage( pss,form,NULL ) ) )
        {
            PMSessionGetGraphicsContext( pss,kPMGraphicsContextQuickdraw,(void**)&port );
            SetPort( port );
            QDBeginCGContext( port,&cont );
            drawQuartzPrint( cont );
            QDEndCGContext( port,&cont );
            PMSessionEndPage( pss );
        }
        PMSessionEndDocument( pss );
    }
    SetPort( cptr );
    return( ret );
}
```

プリントアウトが終了したら、QDEndCGContext()を呼び出して CGContextRef を開放することを忘れないでください。QDBeginCGContext()と QDEndCGContext()は、「Universal Interfaces 3.4.1」の Core Graphics 関係のヘッダーファイルではなく、「Quickdraw.h」の方に定義されていますので注意してください。

この処理が一時的な対処方法なのかどうかは謎のままです。この手順を見てみると、プリント最終出力までの処理が、随分と遠回りをしているような気がします。こうしないとプリントアウトできないのは、Mac OS X の内部的な仕組みが、まだまだ未完成である証拠なのかもしれません？　しかし、出来ないなら出来ないでドキュメントに一言記述しておいてもらいたいものです。不可能な処理を試して悩むのは、まったくの時間の無駄なのですから（涙）。とにもかくにも、うちのプリンタでも Quartz API による図形描画をプリントアウトすることができました。記念すべき最初の出力ですね。



次回も、セッションで判明した利用価値が高い Quartz 2D の隠れた機能を紹介します。Quartz 2D を利用して PICT ファイルを高精度で描画するという「荒技」です（笑）。どうやら、Microsoft 社の Excel や Word では、すでにこの機能が使われているようです。

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