小池邦人のプログラミング日記》2002/3/28
～　Quartz 2D の秘密が明らかになった　その２　～

今回も、Expo 会期中に開催された技術セッションで明らかになった、利用価値の高い Quartz 2D の隠れ機能（笑）を紹介してみます。Quartz 2D を利用して PICT ファイルを高精度で描画するという「技」です。

今年も、近畿日本ツーリストから「WWDC 2002パッケージツアー」が発表されました。すでに以下のウェブサイトから申し込みが可能となっています。

http://www.knt.co.jp/branch/shi/wwdc/

昨年と違い、宿泊ホテルも「FAIRMONT HOTEL」がちゃんと確保されているそうです。しかし、参加人員が多いと会場から離れたホテルに移される可能性があるので、参加することが決定している人は、早めに申し込んでおいた方が良いでしょう。費用はお一人様168,000円です。ただし、シングルルームを確保すると「シングルルーム使用追加料金」95,000円が追加されますので、同室に宿泊するペアを見つけることは必須でしょうね。(毎年終盤になると関係者同士で取り合いになる)今年は、Mac OS X 10.2の話がメインだと思われます。噂では、Mac OS X 10.2プレビュー版（β版？）が配付されるのではないかとも言われています。

さて、Quartz 2Dを利用してPICTファイルを描画する機能ですが、ヘッダーファイルの「Universal Interfaces 3.4.1」を詳しく調べてみても、そんなAPIはどこにも存在していません。この機能に関係するAPIは、Mac OS X 10.1の「QD.framework」に属するQDPictToCGContext.hに定義されています。どうしてUniversal Interfacesの方にないのかは謎なのですが、とにかく今までは表に出たことがない機能です（筆者が知っているかぎりでは...）。 CodeWarriorなどから利用する場合には、ソースファイルに以下の

定義が必要となりますので注意してください。

```
#include <QD/QDPictToCGContext.h>
```

QDPictToCGContext.hには、全部で7つのAPIが定義されています。

```
/*
    Note: QuickDraw picture data typically comes in two forms: a PICT resource
    that begins the picture header data at the beginning of the resource and PICT
    files that begin with 512 bytes of arbitrary data, followed by
    the picture header data. For this reason, the routines that create a QDPictRef
    attempt to find the picture header data beginning at either the first byte
    of the data provided or at byte 513 of the data provided.

    Additionally the Picture Bounds must not be an empty rect.
*/

/* Create a QDPict reference, using `provider' to obtain the QDPict's data.
 * It is assumed that either the first byte or the 513th byte of data
 * in the file referenced by the URL is the first byte of the
 * picture header. If the URL does not begin PICT data at one
 * of these places in the data fork then the QDPictRef returned will be NULL.
*/
extern QDPictRef QDPictCreateWithProvider(CGDataProviderRef provider);

/* Create a QDPict reference from `url'.
 * It is assumed that either the first byte or the 513th byte of data
 * in the file referenced by the URL is the first byte of the
 * picture header. If the URL does not begin PICT data at one
 * of these places in the data fork then the QDPictRef returned will be NULL.
*/
extern QDPictRef QDPictCreateWithURL(CFURLRef url);

/* Increment the retain count of `pictRef' and return it.  All
 * pictRefs are created with an initial retain count of 1. */
extern QDPictRef QDPictRetain(QDPictRef pictRef);

/* Decrement the retain count of `pictRef'.  If the retain count reaches 0,
 * then free it and any associated resources. */
extern void QDPictRelease(QDPictRef pictRef);

/* Return the Picture Bounds of the QuickDraw picture represented by `pictRef'. This
    rectangle is in the default user space with one unit = 1/72 inch.
*/
extern CGRect QDPictGetBounds(QDPictRef pictRef);

/* Return the resolution of the QuickDraw picture represented by `pictRef'.
    This data, together with the CGRect returned by QDPictGetBounds, can be
    used to compute the size of the picture in pixels, which is what QuickDraw
    really records into pictures.
*/
extern void QDPictGetResolution(QDPictRef pictRef, float *xRes, float *yRes);

/* Draw `pictRef' in the rectangular area specified by `rect'.
 * The PICT bounds of the page is scaled, if necessary, to fit into
 * `rect'. To get unscaled results, supply a rect the size of the rect
 * returned by QDPictGetBounds.
*/
extern OSStatus QDPictDrawToCGContext(CGContextRef ctx, CGRect rect, QDPictRef pictRef);
```

これらのAPIからPICT画像へアクセスするのには、PicHandleの代わりにQDPictRefを用います。PQDPictCreateWithProvider()を使うと、PicHandleとしてメモリ上に展開されているPICTデータからQDPictRefを得ることができます。PICTファイルからダイレクトにQDPictRefを得るのには、QDPictCreateWithURL()にそのCFURLRef（使い方はCore

FoundationのCFURL.hを参照）を渡す方法を取ります。こうして得た、QDPictRefをQDPictDrawToCGContext()に渡せば、指定されたQuartz 2DのContext（CGContextRef）へPICT画像が描画されるわけです。まずは、メモリ上にPicHandleとして保存されているPICTデータを描画するサンプルを見てみます。

```
void drawQuartzPictData( WindowRef window )
{
    CGDataProviderRef   prov;
    OSType              type;
    QDPictRef           pref;
    CGContextRef        cont;
    PicHandle           pict;
    long                size;
    FSSpec              fsc;
    long                max;
    CGRect              crt;

    if( ! navMyGetFile( 'PICT',&fsc,&type,&max ) )
    {
        if( ! loadPictFile( &fsc,&pict ) )
        {
            HLock( (Handle)pict );
            size=GetHandleSize( (Handle)pict );
            prov=CGDataProviderCreateWithData( NULL,(void *)*pict,size,NULL );
            if( pref=QDPictCreateWithProvider( prov ) )
            {
                getWContext( window,&cont );
                if( cont )
                {
                    CGContextSaveGState( cont );
                    crt=QDPictGetBounds( pref );
                    QDPictDrawToCGContext( cont,crt,pref );
                    CGContextFlush( cont );
                    QDPictRelease( pref );
                    CGContextRestoreGState(cont );
                }
            }
            CGDataProviderRelease( prov );
            KillPicture( pict );
        }
    }
}

short loadPictFile(FSSpec *fsc, PicHandle *pict)
{
    long    len,flen,hlen=512L;
    short   fref,ret=-1;
    char    hbuf[512];
    Handle  hd;

    if( ! FSpOpenDF( fsc,fsRdPerm,&fref ) )
    {
        GetEOF( fref,&flen );
        len=flen-hlen;
        if( ! ( hd=NewHandle( len ) ) )
        {
            FSClose( fref );
            return( 1 );
        }
        HLock( hd );
        FSRead( fref,&hlen,hbuf );
        ret=FSRead( fref,&len,*hd );
        FSClose( fref );
        HUnlock( hd );
        *pict=(PicHandle)hd;
    }
    return( ret );
}
```

navMyGetFile()は、「Navigation Service」を使いPICTファイルのFSSpec構造体（ファイ

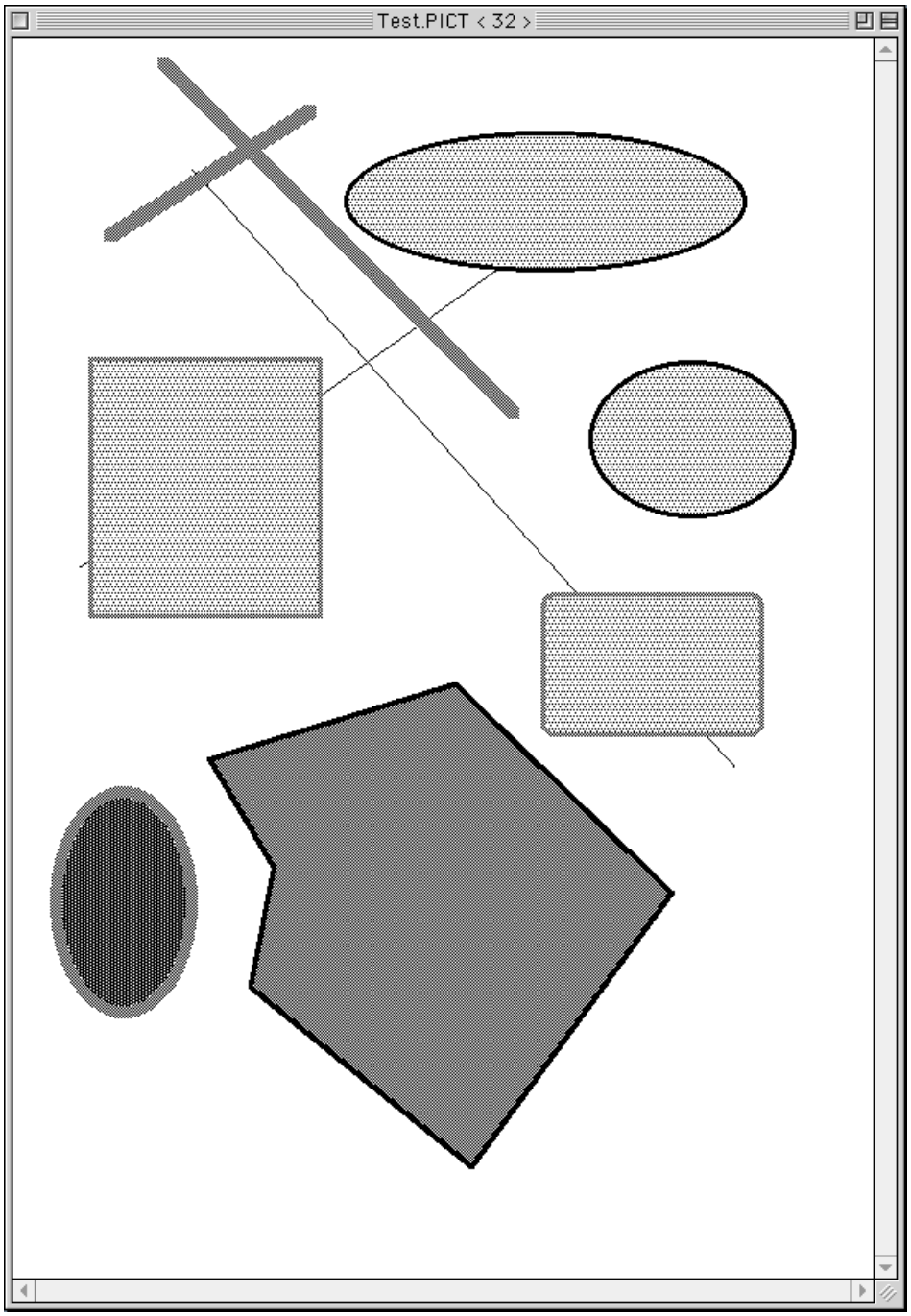
ル保存場所）を得るための自作ルーチンです。以前に何度か紹介しましたので、ここでの説明は割愛します。FSSpecを得たら、それをloadPictFile()ルーチンに渡してPicHandleを得ます。続いてCGDataProviderCreateWithData()にPICTデータのメモリ領域を指定し、CGDataProviderRefを確保します。これは、Quartz 2DでBitmapイメージを描画する時に利用した手法と同じです。最後に、得られたCGDataProviderRefをQDPictCreateWithProvider()に渡して、やっとQDPictRef（pref）を得ることができるわけです。

次は、Quartz 2Dによる描画の準備として、getWContext()によりウィンドウに確保しておいたCGContextRefを得ます。これも、以前に紹介した自作ルーチンです。描画に必要となる画像の矩形枠は、QDPictGetBounds()で得ておきます。この時の矩形枠表記は、QuickDrawのRect構造体ではなく、Quartz 2DのCGRect構造体ですので注意してください。最後に、確保しておいたCGContextRef、CGRect、QDPictRefの3つのパラメータをQDPictDrawToCGContext()に渡しCGContextFlush()を実行すると、Quartz 2Dにより描画されたPICT画像がウィンドウ上に表示されます。また、上記の処理を、QDPictCreateWithURL()を使う方法に変更してみると、以下ようなルーチンとなります。

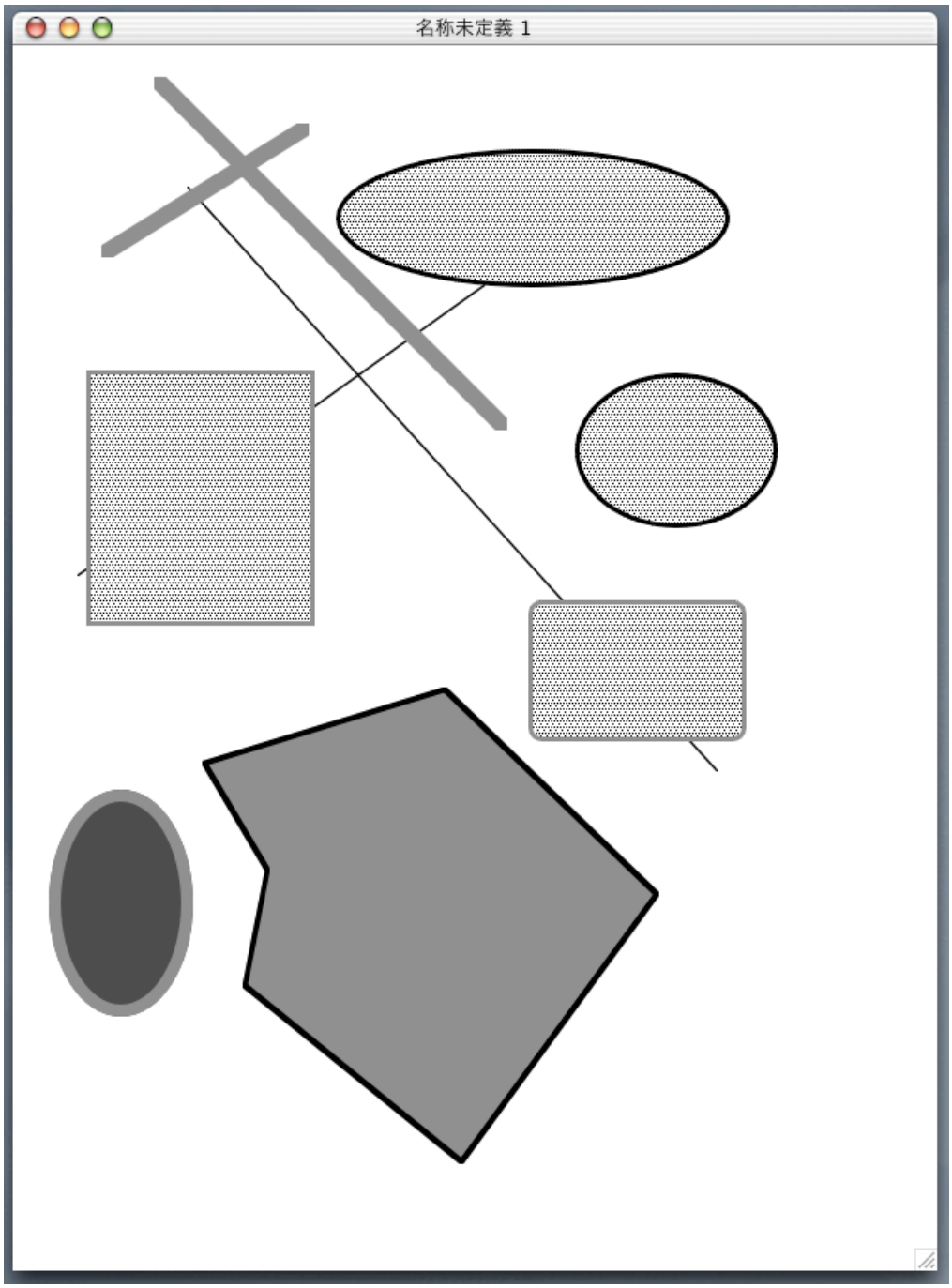
```
void drawQuartzPictFile( WindowRef window )
{
    OSType          type;
    QDPictRef       pref;
    CGContextRef    cont;
    FSRef           fref;
    CFURLRef        url;
    FSSpec          fsc;
    long            max;
    CGRect          crt;

    if( ! navMyGetFile( 'PICT',&fsc,&type,&max ) )
    {
        FSpMakeFSRef( &fsc,&fref );
        url=CFURLCreateFromFSRef( NULL,&fref );
        if( pref=QDPictCreateWithURL( url ) )
        {
            getWContext( window,&cont );
            if( cont )
            {
                CGContextSaveGState( cont );
                crt=QDPictGetBounds( pref );
                QDPictDrawToCGContext( cont,crt,pref );
                CGContextFlush( cont );
                QDPictRelease( pref );
                CGContextRestoreGState(cont );
            }
        }
    }
}
```

さっそく描画結果を見てみることにしましょう。最初は、昔懐かしいSuper Paint 3.0で作画したオリジナルのPICT画像です。QuickDrawの特徴である「ギザギザ」がよく出ていますね（笑）。



続いて、こちらが先程紹介したルーチンを使い（どちらを使っても結果は同じになる）、Quartz 2D経由で描画した結果です。アンチエイリアス処理や、グレースケールパタン、フレームラインの正確な幅などなど、非常に美しい描画となっています。今まで慣れ親しんできたPICTじゃないみたいですね。いや〜、大人になったPICTといった感じでしょうか！なんだか、子供の成長を喜ぶ親の心境ですね（笑）。



さて、次回こそはオブジェクトのセレクションに Overlay ウィンドウを利用する話をすることにします。残念ながら諸般の事情で、今回をもって MDOnline や MacWIRE Online での掲載は終了になってしまいますが、「プログラミング日記」自体は筆者のウェブサイトで継続掲載されますので、引き続きご愛読よろしくお願い致します。

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