Java Watch on the X》
5 – ハードウエアアクセラレータをチェック(1)

Mac OS X の Java 1.3.1 Update 1 が、2002 年 2 月にリリースされた。いくつかのアップデート機能があるが、その中の 1 つに、ハードウエアアクセラレータによる Swing アプリケーションの高速化がある。アクセラレーションを有効にする方法の解説と、加えて、グラフィックスを利用するようなサンプルプログラムを作ってベンチマークテストをしてみた結果を説明しよう。

Java のアクセラレーション機能

Mac OS X は Java VM を搭載したことを大きな特徴としてプッシュしてはいるものの、残念ながら、日本のユーザに取っては、テキストフィールドがあるとアプリケーションの応答が極端に悪くなるといったことなどがあって、Java すなわち Swing を使ったアプリケーション作成を本格的に行うことはできなかった。しかしながら、Java 1.3.1 Update 1 によって完全ではないとしても、十分に使えるレベルにはなりつつある。日本のプログラマにとっては、やっと様子見をしなくても済む段階まで来た。その Java 1.3.1 Update 1 での新機能がハードウエアアクセラレーションだ。実際にはそれよりも前から機能自体は組み込まれていた。これまでは、「デベロッパ向けのプレビュー」という位置付けでもあり、そうした機能が今後利用できるという予告編のようなものであった。だが、Java 1.3.1 Update 1 ではそれが正式な機能としてリストアップされたのである。

ハードウエアアクセラレーションは、Swing のさまざまな処理を、グラフィックスカードを直接利用して処理をすることで、アプリケーション全体の処理能力を高めようというものだ。Java の場合には、やはり仮想マシンベースであることや、AWT の上に Swing があるといった多階層化されたフレームワークでもあり、速度的には確かに不利であることも言えるだろう。そこで、Swing の処理を、いわば、グラフィックス

のハードウエアを直接利用することで高速化を試みるのである。
ただ、それでは、実際にどういう処理がアクセラレートされるのかといった情報は今のところ得られていない。たとえば、処理のどの部分に高速化がかかるのかや、さらには AWT の処理まで高速化はあるのかどうかといったことが一切分からない。Java 1.3.1 Update 1 のドキュメントでは、Swing のグラフィックス呼び出しがハードウエアを直接利用するとしているという記述しか見られないのである。だから、逆に、この機能をうまく引き出すということや、この機能が適合する処理はどんなものがあるかといったことも判断しようがない。
さらに、ハードウエアアクセラレーションは、グラフィックスカードないしはグラフィックスチップごとに、利用する／しないを指定する形式になっている。単純なオン／オフではなく、たとえば ATI Rage では利用しないけど GeForce では利用するといった設定が可能になる。ただし、何でもいいからすべてオンにするということはできない。これについても、「効果のないカードでは指定をしないことを可能にする」という記述が見られるが、つまりは、アクセラレーションの効果がない場合が存在するということの裏返しではないかと想像される。

アクセラレーション機能の有効化

ハードウエアアクセラレーションを有効にするには、システムプロパティの

com.apple.hwaccellist

に対して、適用させたいグラフィックカードに対応したキーワードを設定する。カードごとに、以下のようなキーワードが定義されている。ここでも問題があって、機種ごとの対応カードについての情報がわずかしかないのである。

ビデオカード	キーワード
ATI Rage128（16MB）	ATIRage128_16777216
ATI Radeon（16MB）	ATIRadeon_16777216
ATI Rage128（32MB）	ATIRage128_33554432
ATI Radeon（32MB）	ATIRadeon_33554432
NVidia GeForce2（32MB）	NVidia11_33554432

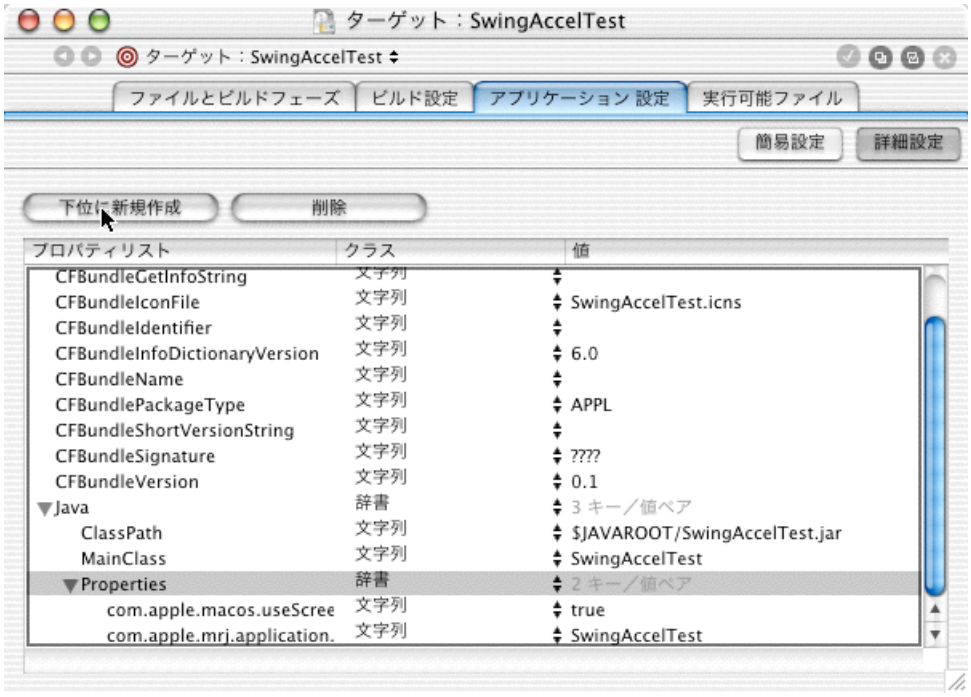
NVidia GeForce3（64MB）	NVidia20_67108864
NVidia GeForce4（64MB）	NVidia20_134217728
PowerBook G4 など？	ATIRage128_8388608

Java 1.3.1 Update 1 のドキュメントでは、一覧表が掲載されているが、たとえば、PowerBook G4 500MHz 機の場合のキーワードが掲載されていないなど、実際に使えるキーワードの全てが掲載されているとは限らないようだ。

Apple は、現在起動しているマシンでのビデオカードについての情報を得る、hwaccel_info_tool というコマンドライン形式の診断ソフトを提供している。ただし、これについては、入手できるのは現段階では ADC メンバーだけなので、こうしたツールの存在があることだけをお伝えしておこう。たまたま、筆者が PowerBook G4 を持っていたのでドキュメントのリストにあるもの以外のキーワードを見つけられたのかもしれないが、いずれにしても、どの機種ではどのキーワードを適用できるのかと言った一覧表は必要になるだろう。

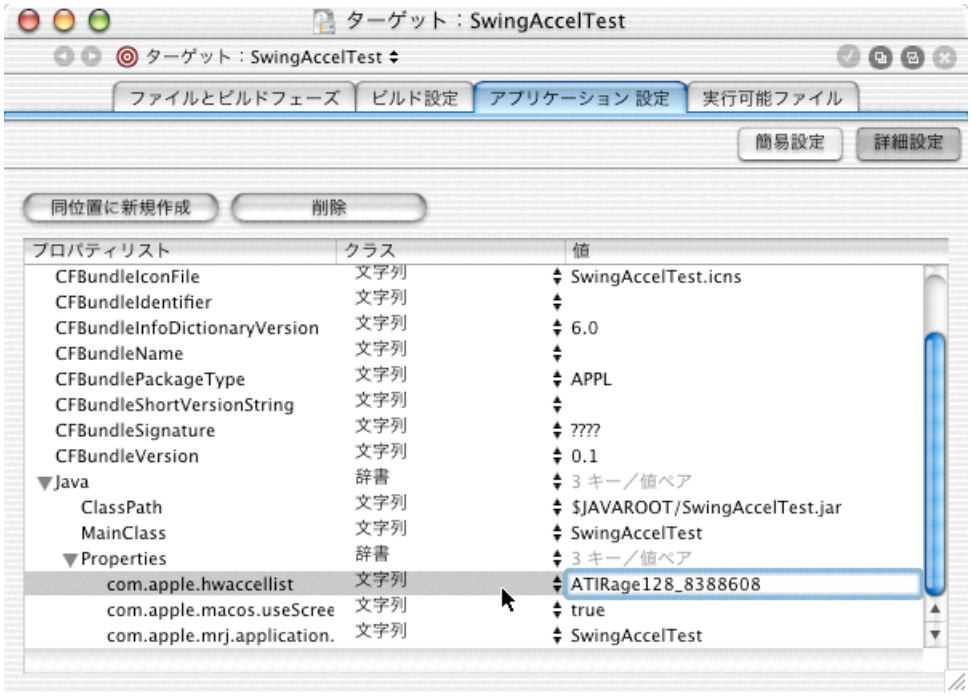
実際にプロパティを設定するには、Project Builder で、パッケージ形式のアプリケーションを作成しているのであれば、「プロジェクト」メニューから「アクティブターゲットの編集」（Command+option+E）を選択し、「アプリケーション設定」のタブを選択する。そして、「詳細設定」ボタンをクリックして、設定項目の階層表示を行う。Java、Properties と階層をたどり、Properties の項目を選択して「下位に新規項目」というボタンをクリックする。

プロパティ設定を追加する



すると、新しい項目が作成されるので、左側は「com.apple.hwaccellist」と指定し、右側はグラフィックスカードに対応したキーワードを指定する。右側のキーワードは、カンマで区切って複数指定が可能だ。だから、すべてのマシンでアクセラレートの機能を有効にするには、すべてのキーワードをカンマで並べれば良い。

アクセラレートを有効にするプロパティを設定する

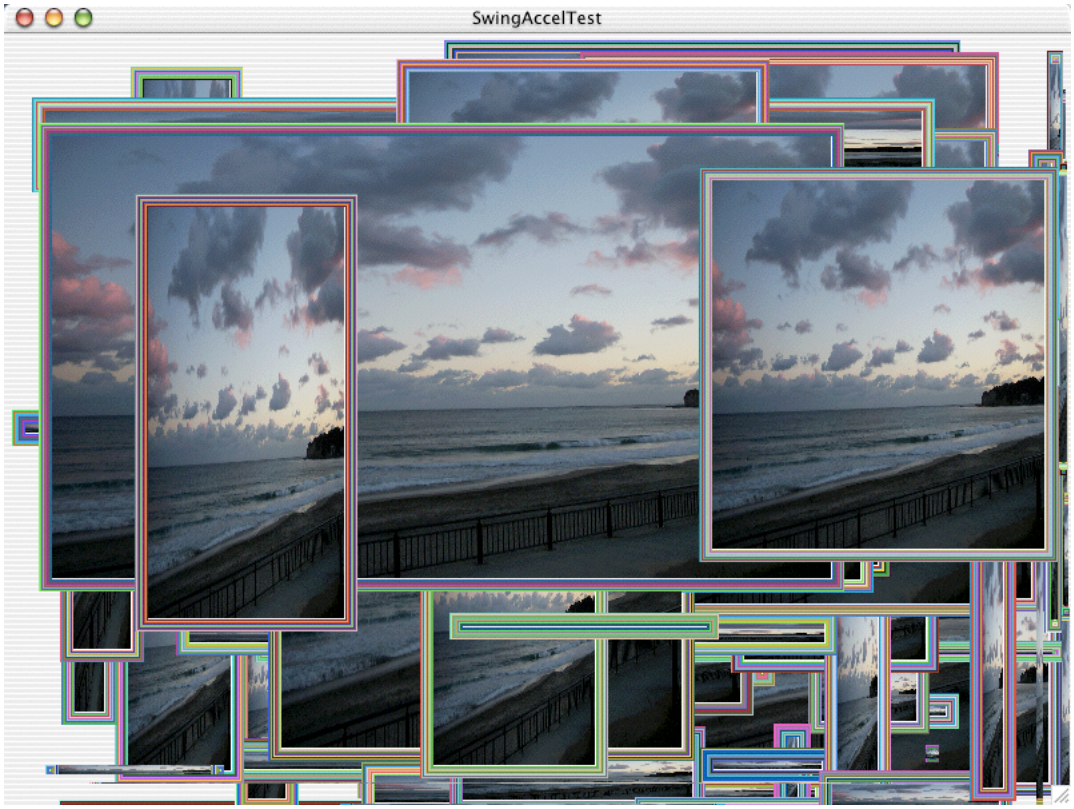


ベンチマークを行ってみた

さて、実際にベンチマークテストを行ってみた。「Swing が高速化される」というときの Swing の範囲が明確ではないため、はずしがあるかもしれないが、その辺りは要検討事項だろう。

テストプログラムは、JFrame のウインドウに、グラフィックスをたくさん表示するということをやってみた。グラフィックスとしては、パッケージの Resources フォルダにある 1 つの JPEG ファイル（1792×1184 ドット、606KB）を、画面上に表示した。

テストプログラムの実行例



JFrame は null レイアウトにして、座標をランダムに指定して、画像をたくさん表示するようにしている。1 つ 1 つの画像は、Canvas を拡張したクラスで表示するようにした。その拡張クラスで JPEG ファイルの表示を行うという具合であるが、画像の回りに、Graphics クラスの drawRect で 10 回ほどカラーをランダムに変更させて、ちょっとケバい枠を書いてみた。

ただ、ここで、「Canvas は AWT ではないか」という話も出てくるだろう。だが、JPEG 画像を手軽に表示するにはこの手法が使われるだろうから、まずはこれでチェックをしてみることにする。もし、AWT だとアクセラレーションがかからないのであれば、

ベンチマークテスト結果は、プロパティの設定に関わらず変化がないはずである。ベンチマークテストは、ウインドウを表示後、Canvas を拡張した JPEG 表示オブジェクトを 100 個ランダムに追加する時間を測定してみた。「追加」だから、厳密に言えば表示時間ではないのだが、体感と大きく離れた結果ではなかった。

◇テストプログラム

http://mdonline.jp/figs/02/028/SwingAccelTest.sit

気になるテスト結果だが、「ほとんど効果はない」というところのようだ。実は、測定結果が、毎回大きく変動する。統計処理をしなければならないような変動なのである。数回測定した結果はアクセラレーションの指定があれば約 8 秒、ない場合には 7 秒となっているが、どちらの場合も最大と最小が 2 倍近い開きがあるので、事実上変化はないと考えられる。テストマシンは 500MHz の PowerBook G4 である。なお、ビルドしたアプリケーションを Finder でダブルクリックし、Console アプリケーションに測定結果を出力して、実行時間を測定した。他のアプリケーションは起動しないようにした。他のアプリケーションが起動しているような場合でもチェックしてみたが、その状況によって実行時間の数字自体がかなり増えてくる。

もちろん、これだけの結果ではすべては分からないというのは当然のことだろう。AWT のコンポーネントでは有効ではないということも言えるかもしれないが、アクセラレータを有効にすると、時折 Console に「speed pen going down, clean up」と表示されるのが確認されるので、今回のベンチマークテストではアクセラレーションの機能はまったく使われなかったとも言えないと考えられる。これが、PowerBook だからということで、別のマシンなら早くなるのであろうか？　そのあたりはもちろん不明だが、もし、読者の方で、ベンチマークをしていただけるのであれば、測定結果をお知らせしていただければと思う。

なお、もう少し違うテストもやってみる予定である。

・・・・・・・・・・・・・この項、続く・・・・・・・・・・・・・[新居雅行]・・・・・・・・・・・・・