大容量メディアを対象としたオブジェクト指向型の制御システム

‘ The object oriented control system for the large volume media era’

近 藤 克 彦
Katsuhiko KONDO

株式会社タクミ　　（〒104-0012 東京都中央区新川二丁目９番９号　SH ビル６Ｆ
E-mail: kkfunk@tacmi.co.jp ）

ABSTRACT. . The emerge of the large volume and high-speed accessible media accelerates the PC’ s integrating any functions from the creation of documents to multimedia files. It leads OS to be very unstable due to the confliction among applications running at the same time. In the Home-Computing environment, the mixture of the Set Top Box and PC at home will cause more complexity. This object oriented control system is designed for each application to run independently on BIOS with direct connection. The control system and applications are stored in the large volume media and become free from the hardware and application environment. The OBJECT, in other words any creations on the computing device, i.e. text document, movie and sound file will truly be the main object of the computing with this system.

.

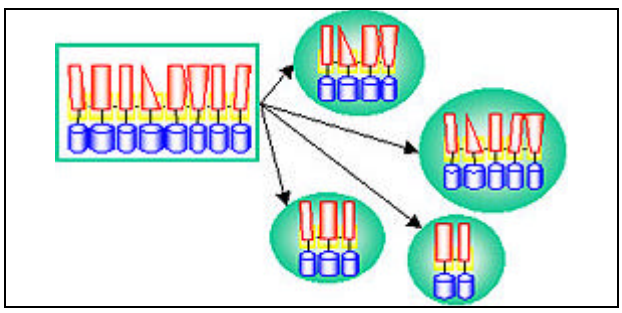
１．背景

－　大容量メディアの登場によるシステムの変化　－
大容量メディアの登場により、旧来データストレージを主流としていた目的からオペレーションシステムなどを含む形態へと現在のシステムは推移している。
また、ストレージメディアはさらに大容量化され高速化を行う方向に動いている。
この中で現在のシステム構成の中で不安定な要素になっているものがオペレーションシステムである。集中型のＰＣの性格によりワープロや表計算などのアプリケーションのみならずマルチメディアなどのコンテンツを再生するためのアプリケーションも含まれるようになった。
　これらのアプリケーションがひとつのオペレーションシステムの上で稼動するためにシステムとしては不安定な要素を生むことにつながるようになった。
今後はさらにホームコンピューティングやＳＴＢ（Set Top Box）などの登場によりシステムはさらに複雑になることが予想される。
　さらに今後登場する大容量メディアは持ち運びにも優れていることが予想される。また、今後登場するアプリケーションは今までのコンピュータシステムに求められている条件とは明らかに違うために新しい制御システムを構築する必要があると考える。

２．目的

(1) 1 のような背景の中で持ち運び可能な大容量メディアにデータ、アプリケーションおよび制御システムまで含めた環境を構築することを目的としている。ここでいう制御とは、動画を再生する場合統一された形態に対して大容量メディアに保存されている制御システムとハードウェアの構成を制御しているＢＩＯＳのみで行えることを指す。
(2) 制御システムは主軸として今後登場するであろうホームコンピューティングシステムやＥ－ＣＡＳＨなどのシステム用にそれぞれが構成され、他のシステムの影響をまったく受けることなく動作する。
(3) これら複数の制御システムは必要に応じてその機能を追加、削除することが容易に行える様にする。ここでいう追加削除はあくまで単体の機能を指す。
(4) これらの制御システムは、ＰＣやＳＴＢなどの実況環境に依存することなく同一のオブジェクトを用いることを可能としており、サーバーなどのプロセスと完全に同期が取れるように調整されている。これによりハードウェアの違いによる機能の不安定さや動作の違いを起こさないように制御することが出来る。

本システムの目的としては(1)～(4)の機能を目的としこれらの機能を用いた制御システムを複数のＳＴＢおよびサーバー上で実現することを目的とする。



３．開発の内容、重要ポイント

(1) 開発内容

a) 既存のシステムではオペレーションシステムの上での動作のため複数のアプリケーションが同時に動作を行うと他のアプリケーションに影響を与えることが多い。本制御システムは大容量メディアの中でＢＩＯＳ上で動く独立したアプリケーションとしている。

b) 本制御システムはアプリケーション上の横の連携はパラメーターやファイルを用いることによりオペレーションシステムに依存することがない。

c) 複数のアプリケーションを起動する際にメモリー等の環境をＢＩＯＳ上からのチェックを行い、動作が出来ない場合メッセージを表示する。

d) 複数のアプリケーションを動作させる場合、他のアプリケーションに対して影響を与えることなく動作が行え、アプリケーションが終了する際にはメモリーなどの環境エリアから完全に消去される状態を作成する。

e) これらのアプリケーションは単体としてＳＴＢなどでも動作することを前提として同一のアプリケーション間ではアプリケーション単体の機能のみで制御システムは親和性を保つことを保証する。

f) 本制御御システムは基本的には、全ての制御システムおよびアプリケーションは大容量メディア内に格納され同様のハードウェア（ＳＴＢやサーバー）上では大容量メディアからの起動により機能や動作を保証する。

(2) 環境

環境として考えられる物は、機種およびオペレーションシステムに対する依存性が少なく、かつ、基本的なインターフェイスをデバイスドライバなどを介さずに操作できること、仮想システムを作動させても十分なパフォーマンスを出せることを前提として、以下の物が必要とされる。

・AT 互換機での動作
・TCP/IP プロトコルのネットワーク
・仮想システムを稼動させるために必要な高速な CPU、大規模ストレージおよび TCP/IP プロトコルを使用して通信が可能な機能を持つ実行環境

(3) 重要ポイント

本プロジェクトにおいての重要ポイントを以下に示す。

a) 起動システムエミュレータ上でアプリケーションが動作すること。
b) 基礎的な I/F が使用できること。
c) アプリケーションが独立したメモリマップ上で動作すること。

の３点が重要なポイントとなる。

a) においては、実機の IPL もしくはＢＩＯＳに相当する機能を実装する必要がある。
b) においては、仮想的な I/F をベースモジュールに実装する必要がある。
c) においては、ベースモジュールおよびアプリケーションモジュールが他の実行モジュールに干渉されずに動作する機構を実装する必要がある。

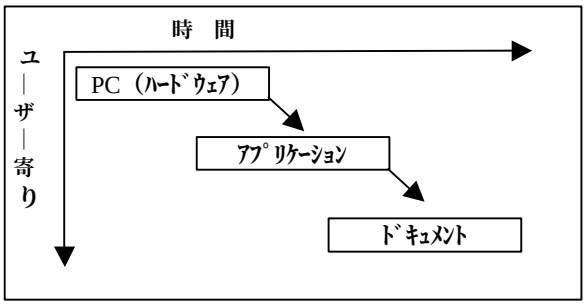
(4) 本研究の目標

従来の PC 用ソフトウェアは、アプリケーションソフトがまず『主役』として存在し、その主役であるところのアプリケーションソフトが生み出す副産物のようにドキュメントが扱われてきた。

しかし、本来、PC ユーザーの生産活動によって創造されるものはドキュメントであり、アプリケーションプログラムは道具である。

今　までのアプリケーションソフトは、道具ではなく、PC の世界では主役であった。

さらに遡れば、主役は PC たちハードウェアであり、アプリケーションソフトは PC をひきたてるための脇役だった。このように徐々にコンピュータ世界の中の主役はユーザー寄りにシフトしてきている。



こうした現象は具体的に現れてきている。

過去、オペレーティングシステム(OS)は、ユーザーが購入するのが当然だった。

しかし、Windows がプレインストールされるようになり、パソコンは OS とセットでとらえられるようになると、OS に独立した価値を見いだすユーザーは減ってきた。

つぎの段階では、さらにアプリケーションソフトに独立した価値を見いだすことがなくなってきた。つまり、OS やアプリケーションソフトの入手のためにお金を出すことに抵抗を感じるようになってきた。いまやパソコンを買えば OS は当然のようにインストール済みであり、ワープロソフトや、インターネット環境のためのソフトウェアも含まれている。以前は、「パソコン」とはハードウェアのことであったが、今となってはユーザーはそれら OS やソフトウェアをひとくくりにして「パソコン」と呼ぶようになってきている。

今、アプリケーションプログラムが主役である時代が終わりつつあり、次は開発者ではなく、ユーザーの操作によって生産されるドキュメントが主役となる。つまりプログラムに価値があるのではなく、ドキュメントに価値がある時代になってきている。簡単に言えば『パソコンがオモチャから道具になってきた』と言える。

今後、コンピュータが、より人々の生活に密接に関わるようになるには、ハードウェアや OS、プログラムは脇役である道具として昇華されていく必要がある。

ハサミや箸、ペンなどは、日常、何気なく使われている道具はその存在を特に意識されることはあまりない。

しかし、それらはなくてはならない必需品であり、人間の生活を支える上で欠かすことの出来ないものである。

本来道具とはそうした性質を持つことが望ましい。

家電製品や自動車を例にとっても、当初はメカマニアの高価なオモチャだったが、今では誰でも使う道具になっ

ている。PC もそうなっていく。PC 用アプリケーションソフトも、そのように、脇役に徹しながらも必需品となることがこれからの姿であると言えるし、そのようなソフトウェア開発が、今後のソフトウェア開発者にとっての課題である。

今まで一部のプログラマーたちが世に送り出してきた芸術品としての『ソフトウェア』は、道具としての側面を大きく我々に見せてくれるようになった。もちろん、今後も芸術性や独創性を求める姿勢は必要だが、コンピュータがさらに世の中の生活に浸透していくためには、人の道具としての未来も、また必要になる。

今回の実験の主旨

(a) オブジェクト指向の概念をつきつめ、プログラム主体ではなく、ドキュメント主導型ソフトウェアの試金石としての新しい考え方の画像編集を模索する。

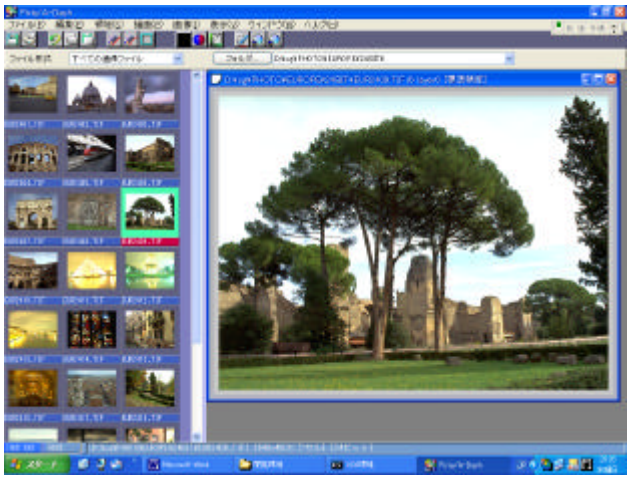
(b) プログラム（実行部）は画像ファイルの『属性』に過ぎず、それ単体として独立したものではない。

(c) あくまで主体は今までのようにデベロッパが産みだしたアプリケーションソフトではなく、ユーザーが産みだしたドキュメントである。

それらの観点を踏まえて、具体的に以下のようなテストプログラムを作成する。

４．プログラム

このプログラムは Microsoft Windows をベースとした「大容量メディアを対象としたオブジェクト指向型の制御システム」におけるサンプルプログラムである。



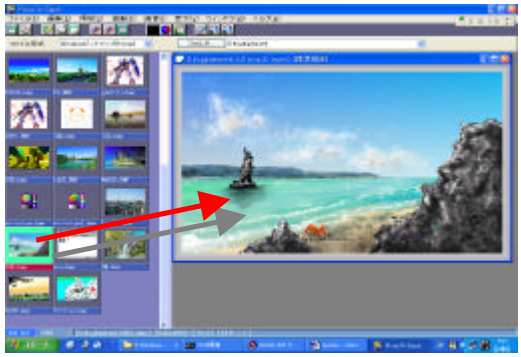
△Pixia/A'の画面

Pixia/A'は、今回の実験のために作成した特殊機構をもつグラフィックツールである。
今回の試みのために、いくつかの特殊機能を内蔵している。

(1) Pixia/A'の概要

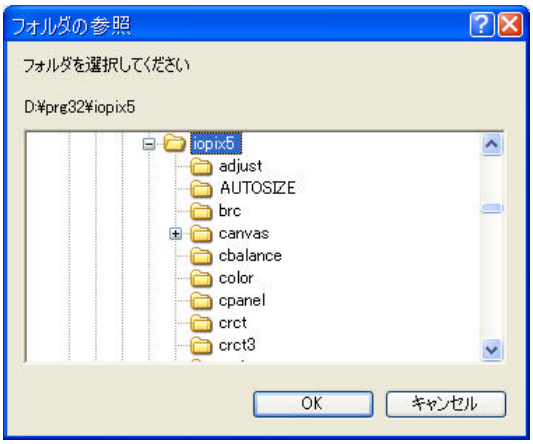
Pixia/A'は、Pixia をベースに今回の研究開発のためにカスタマイズした特殊なグラフィック加工ソフトウェアである。

アプリケーションウインドウ左側のイメージ一覧には、選択しているフォルダ（カレントフォルダ）に存在する画像ファイルがサムネイル一覧表示され、ウインドウ右側は編集・表示中の画像をウインドウ表示するために使われる。
左側のイメージ一覧にあるサムネイルをダブルクリックするか、または右側ワークエリアにドラッグ＆ドロップすることで、ウインドウ表示状態・編集状態となる。

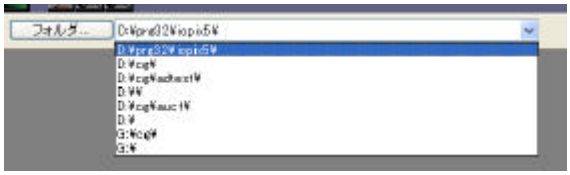


ツールバーには、画像一覧を表示するフォルダを迅速に選択するための機能が備えられている。

[フォルダ…]ボタンを押すと、以下のダイアログが表示される。
選択したフォルダの画像一覧がアプリケーションウインドウ左側に一覧表示される。



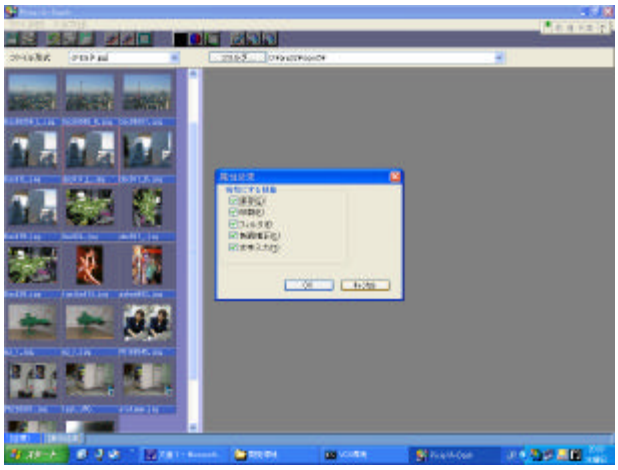
また、[フォルダ]ボタン右にあるコンボボックスには、過去に選択したフォルダの履歴が格納され、また使用頻度の高いフォルダ順にソートされている。
よく使うフォルダは、[フォルダ…]ボタンを押してフォルダツリーから選択することなく、このコンボボックスから選択することもできる。



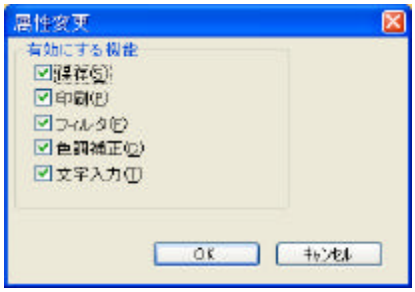
(2) 機能別フラグ埋め込み

実験の第一段階として、まず、現在インターネット環境で、もっとも一般的に使用されている JPEG 形式のフルカラー画像ファイルに Pixia/A'のもつ機能項目別のフラ

グを埋め込み、画像ファイルによって使用できない機能と使用できる機能を指定することができるようにした。特殊な情報を埋め込む画像ファイルは、独自形式にしてしまうのが最も手っ取り早いが、JPEG 形式や BMP 形式など、一般に使われている形式を利用したほうが、実験価値が高いと判断した。
ここで特殊な属性を埋め込んだ JPEG 画像ファイルは、大きく JPEG のルールからは逸脱していないはずだが、インターネットエクスプローラでは読み込めなくなってしまった。しかし、PhotoShop など、一般のアプリケーションソフト複数では正常に読み込めることが確認できた。

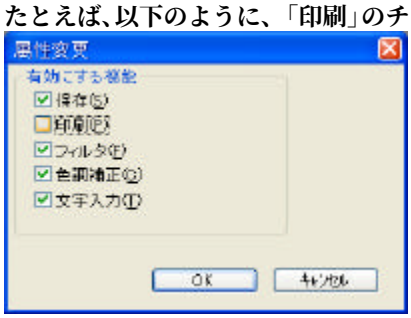


属性を埋め込むには、アプリケーションウインドウ左側のサムネイル一覧から、任意の画像ファイルを右クリックする。このテスト機構では、情報を埋め込むことができるのは JPEG 形式画像のみであるので、ツールバーの「ファイル形式」を JPEG に切り換えておく。

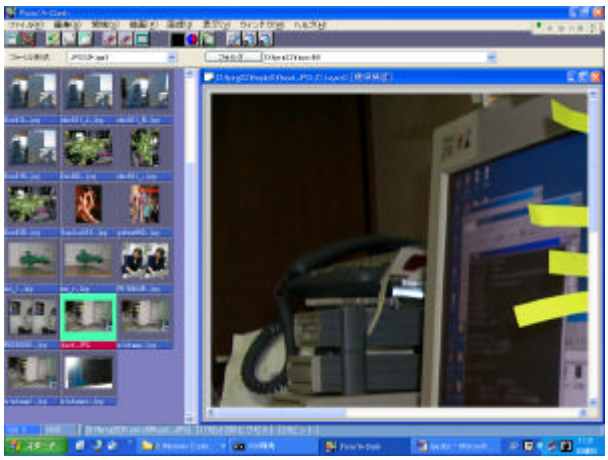


右クリックにより、上のようなダイアログが表示される。チェック（×印）が付加された機能は、その画像に対して有効な機能である。
チェックをはずした場合、その機能は、該当する画像に対して機能しなくなる。

たとえば、以下のように、「印刷」のチェックをはずす。

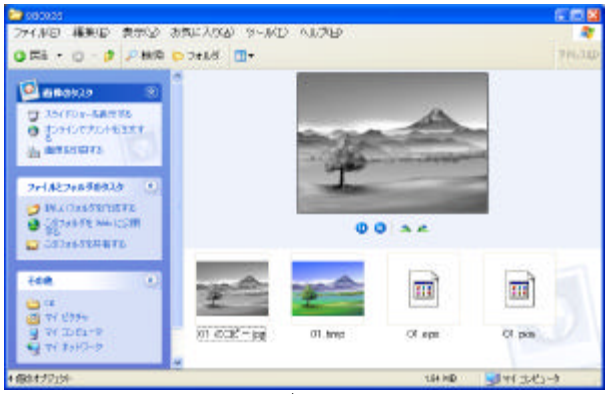


次に、画像をダブルクリック（もしくはドラッグアンドドロップにより）して編集状態にすると、以下のようにツールバーのプリンタアイコンに赤い斜線が引かれ、その機能が選択できなくなる。



(3) 画像ファイルへの実行モジュール埋め込み

ここでは、特定のドキュメントファイル（今回は画像ファイル）をプログラムファイルと一体化させることを試みる。



IE から画像ファイルをダブルクリック

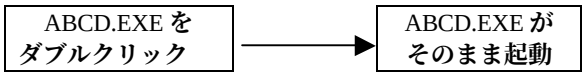
▼

その画像ファイルが閲覧できる状態になり、同時に実行部が画像に編集を加える。しかも、画像ファイルと実行部は１つのファイルになっている

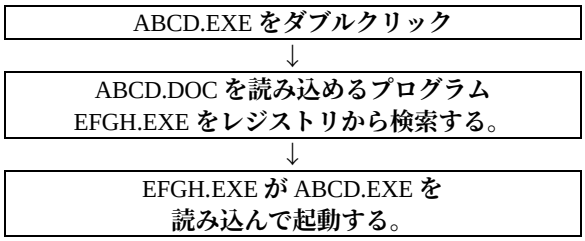
目標としたのは、IE などのファイル一覧表示プログラムから、特定のファイルをダブルクリックすると、その画像ファイルの中に埋め込まれた実行コードが働き、プログラムと画像ファイルを別々に配布することなくドキュメント部と実行部を同時に内包するファイルを作成することである。
今回は Windows を実験環境として選択しているので、Windows の制約をそのまま受けることになるが、それを前提にして、いくつか選択できる方法を考えてみる。

IE は画像ファイルがダブルクリックされた場合に

a) それが実行ファイル（exe、com、bat）であれば、そのまま実行する。

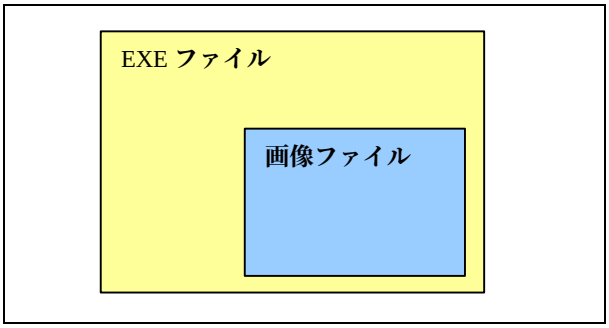


b) レジストリに関連づけが指定されているドキュメントファイルであれば、関連づけされている実行プログラムを起動させる。たとえば、xls ファイルをダブルクリックすればエクセルが起動する。



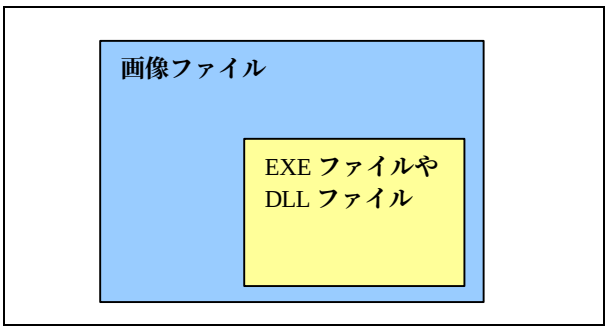
IE から画像ファイルをダブルクリックすると、その画像ファイルに埋め込まれた実行部が走る、と言った機構を実験するためには、以下の選択肢が考えられる。

c) 画像ファイルつき exe ファイル
　編集すべき画像データを埋め込まれた exe ファイルを作成する。exe ファイルなのでダブルクリックされれば、当然実行部が起動する。exe ファイルに連結された画像ファイルをロードし、編集をおこなえるようにすればいい。ユーザーからは exe ファイル１つにしか見えないわけだが、exe ファイルとして意識させるのではなく、『画像ファイルとして』意識されるようにする必要がある。



この場合の問題は、Windows では実行中のプログラムファイルを書き換えることができない点である。
つまり、画像ファイルであるとして編集した後、上書き保存ができない。Windows でなければ解決できる問題かもしれないが、現行の Windows で実践するのであれば、この方法は採用できない。

d) 実行プログラムつき画像ファイル
これは c) とは逆に、画像ファイルの中に EXE や DLL のような実行部を埋め込む。



このパターンが実現できるかどうか Pixia/A'を利用して実験してみた。
画像ファイルの中に DLL ファイルを埋め込み、画像ファイルのオープン後に、内包された DLL を実行させてみることとする。

Pixia/A'に以下のような機能を追加する。

■BMP 形式画像に特定の実行プログラム(DLL)を埋め込めるようにする。

■実行プログラムが埋め込まれた画像ファイルを Pixia/A'でオープンすると、画像ファイルが内蔵する実行部を動作させることができる。

ここでは、画像ファイル形式は BMP 形式とした。
フラグを埋め込む実験で述べた理由と同様に、特殊形式より一般的に利用されている形式のほうが実験価値が高いからである。
BMP 形式画像のファイルは、以下のような内部構成になっている。

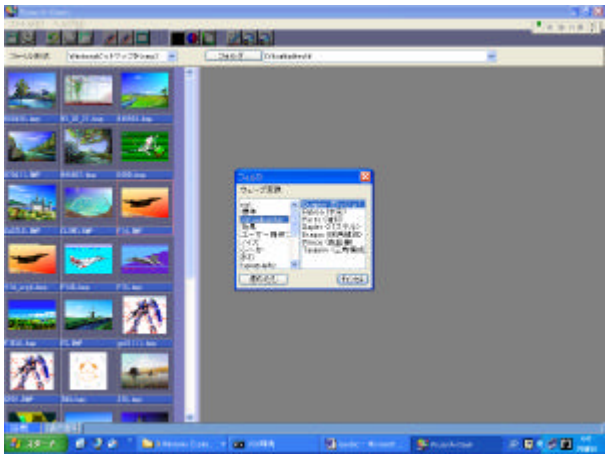
BITMAPFILEHEADER 部 bmfh	BITMAPINFOHEADER 部 bmih	画像データ列

構造体 BITMAPFILEHEADER には、bmfh.bfOffBits というメンバがあり、ファイルの先頭から画像データ列までの距離が格納されている。つまり、bmfh.bfOffBits の値を変えてやれば、画像データ列の直前には隙間があっても良いのである。

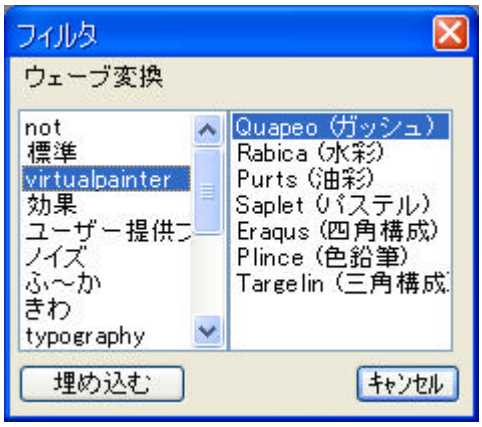
BITMAPFILEHEADER 部 bmfh	BITMAPINFOHEADER 部 bmih	隙間	画像データ列

この隙間に、任意の実行モジュールを埋め込むことで、『他のアプリケーションからも使用できる』実行モジュール内蔵画像ファイルが実現できる。
実行モジュールを内蔵しながらも、他のアプリケーションからまったく問題なく読み込める。
注：ほとんどのアプリケーションソフトは隙間をもつ BMP ファイルを正常に読み込めるが、たいていの BMP ファイルは隙間がゼロになっているため、隙間がゼロである前提でコーディングされたソフトウェアも少ないながらも一部存在する。そのようなソフトウェアでは今回実験した実行モジュールつき画像ファイルを読み込むことはできない。

Pixia/A'からの操作は以下のようになる。
ＢＭＰのみ対応であるので、ツールバーの「ファイル形式」は BMP に切り換える。
プログラムを埋め込みたいファイルを右クリックする。



右クリックにより、以下のようなダイアログが表示される。ここで使用できるモジュールを検索するので少し時間がかかる場合がある。（数十秒）



このダイアログで、画像ファイルに埋め込みたいモジュールを選択し、「埋め込む」ボタンを押す。
すると、該当する BMP ファイルに、選択した実行モジュールが埋め込まれる。

実行モジュールを含む画像ファイルをオープンすると、以下のようなダイアログが表示される。

実行モジュールが使用可能になれば、Pixia/A'のメニューの「画像ファイル内モジュールの実行」が選択できるようになる。この機能は画像ファイルが内蔵する実行プログラムを起動させるのである。

Pixia/A'は、画像ファイル内のモジュールを解釈し起動させるためのスタータとして機能しているだけで、画像編集機能は Pixia 内のものを機能させているわけではない。ここで機能しているのは Pixia 側の Virtua;lPainter ではなく、画像ファイルに埋め込んだ Virtua;Painter なのである。Pixia/A'は単なるモジュールスタータとしてのみ利用している。

今回は DLL ファイルを埋め込んだが、exe ファイルを埋め込むのも同じ方法で実現できる。