ここまで来た HyperCard の開発環境

～ Review of Heizer's New Products ～

田中求之 Motoyuki Tanaka

PDF版について（１９９７年１０月２５日）

この文書は、技術評論社の月刊誌MacJapan（すでに廃刊）のための書いた記事の原稿です。雑誌には１９９２年１１月号～１９９３年１月号の３回にわけて掲載されました。このため、掲載されたものとは内容が異なっています（掲載のために原稿を分割する際に、加筆修正した部分があります）。今となっては、歴史資料程度の価値しかありませんが、HyperCardがプログラミング言語としてこれだけ熱かった時期があったのだということを知ってもらいたいと思い、公開することにします。

CompileIt! によって我々 HyperTalker に Toolbox の世界への扉を開いてくれた Heizer Software 社より，HyperCard の開発環境をさらに充実させる新しいツールがいくつか発売になりました．さっそく購入して使ってみましたので，使用レポートを兼ねて紹介することにします．

紹介する製品は次のものです．

１ CompileIt! Professional Extensions

２ MasterScript

３ WindowScript

１は CompileIt! に機能を追加するスタックです．２はスクリプト・エディタおよび各種のデバッグツールからなる開発環境．３は Dialoger Professional の後継者として発表された，ビジュアル・インターフェースを自由自在にデザインしコントロールするためのツールです．

§１　CompileIt! Professional Extensions

●　CompileIt! その後

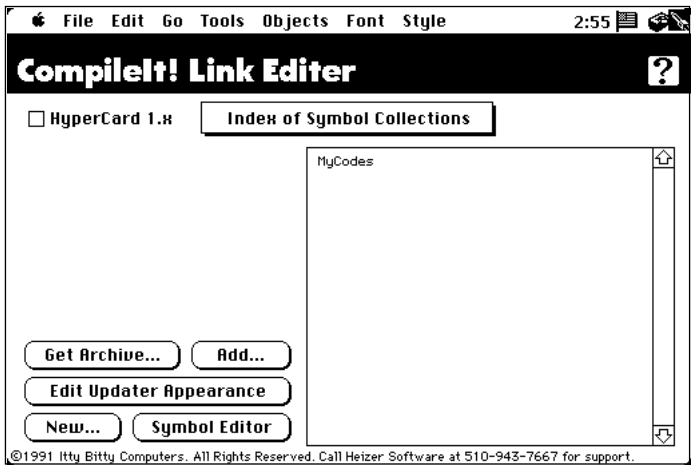
まず本誌５月号で紹介した HyperTalk コンパイラの CompileIt! が SuperCard 1.6 への対応や細かいバグフィックスによって，2.1a にバージョンアップされました．基本的な機能には変更はありません．登録ユーザーには Updater を含んだ CompileIt! News というディスクが無料で届いたはずです．このディスクには CompileIt! の作者である Tom Pittman 氏自らによる Technical Notes をはじめとして，さまざまな Tips の詰まったドキュメントやサンプルスタックが収められています．また，American Online には CompileIt! ユーザーの SIG が設けられているようで，そこでのやり取りの一部を収めたファイルも入っています．CompileIt! に対する同社の意気込みのようなものが感じられる，なかなか読みでのあるものでした．今回紹介する製品のうち，CompileIt! Professional Extensions はもちろんとして，MasterScript も CompileIt! との連係が計れるようになっており，どうやら Heizer Software 社は，CompileIt! を核にして HyperCard 上に新しい開発環境を築こうとしてい

るようです.

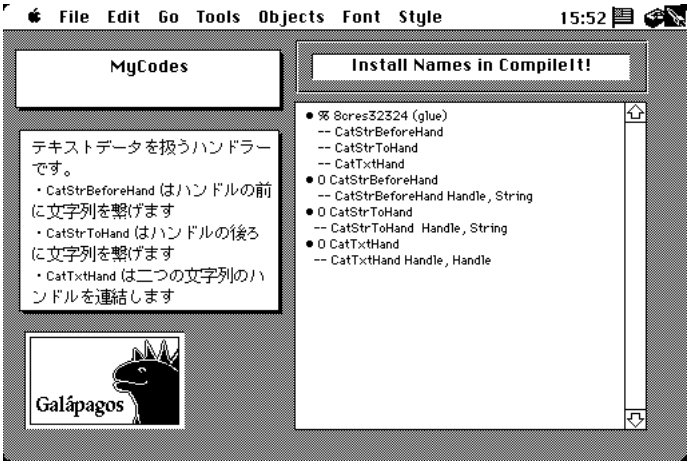
● CompileIt! Professional Extensions

CompileIt! Professional Extensions は, CompileIt! にさまざまな機能を追加するアド・オンスタックです. 現在, Vol.I CompileIt! Linker と Vol.II Developer Edition Interface の２つが発売になっています. 今後もシリーズで発売していくつもりのようです.

Vol.1 の CompileIt! Linker は, 他の言語の開発環境でいう Object Code Libraries を可能にしてくれるものです.



自分が良く使うハンドラーやファンクションを CompileIt! で CODE リソースとしてコンパイルし, この Linker を使うと, 最終的に図のようなハンドラーやファンクションの Symbol スタックを作成できます.

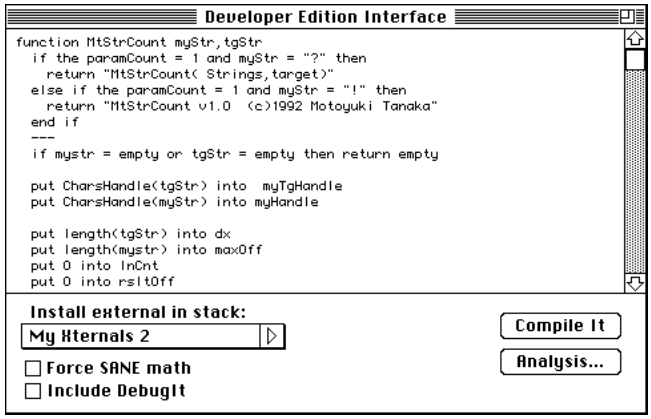


この Symbol を CompileIt! に登録すれば, 他の XCMD の作成の際に, 自作のハンドラーをHyperTalk のコマンドや Toolbox の procedure と同じように使えるようになります. 自分が良く使うハンドラーをこの Linker を使って Symbol として登録しておけば, 非常に効

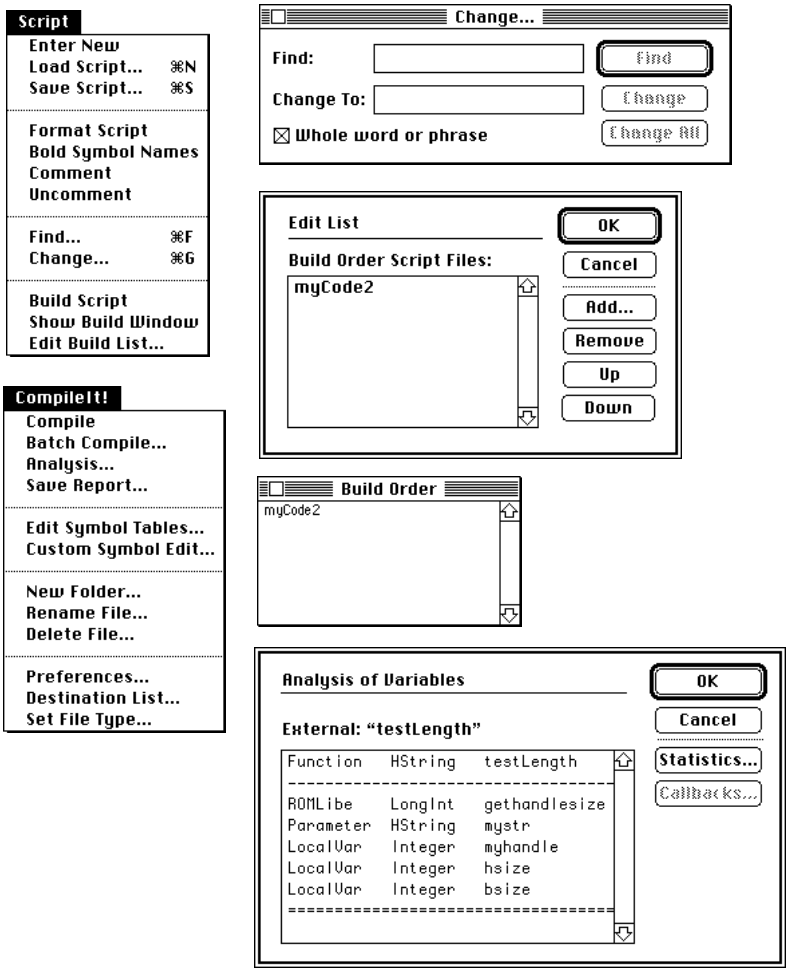
率的に XCMD を作れるようになります．また，私自身は試していないのですが，C や Pascal などの他の言語でコンパイルした CODE リソースを取り込むこともできるようです．ですから，大掛かりな XCMD をチームで開発するような場合などには重宝するでしょう．

ただし，残念ながら，お世辞にも使いやすいとは言えないスタックです．機能の実現を優先して，インターフェースがいまひとつ練られていないという感じです．もっとも，スタックですから自分の使いやすいようにカスタマイズしてしまえばいいわけですが，機能的に複雑なだけに，そう簡単にはカスタマイズできません．

Vol. II の Developer Edition Interface は CompileIt! のフロントエンドとして使用するスタックです．



１枚のカードで CompileIt! のすべての機能を要領よく使えるようになるばかりでなく，CompileIt! には無い様々な機能が利用できます．CompileIt! の各種の設定はダイアログで行なうようになっています．メニューを見てもらえばどのような機能が使えるかはだいたい分かると思います．



実際に使用してみて便利だと思ったのは Build Script です．これは他の言語で include files と呼ばれる機能を実現するもので，コンパイルするスクリプトにおいて，他のファイルに収められたスクリプトを利用できるようになります．まず使用するハンドラーの入ったファイルを Build Over ウィンドウに登録しておきます．一つのファイルに複数のハンドラーが収められていてもよいので，自分がよく使うハンドラーを機能ごとに分類したライブラリ・ファイルを作っておいて，それを登録するのがよいでしょう．そしてスクリプトを書いてから，コンパイルする前にメニューから Build Script を選ぶと，書いたスクリプトを調べて，使用したハンドラーのスクリプトを Build Over ウィンドウに登録したファイルから抜き出して，スクリプトに追加してくれます．つまり，自動的に必要なスクリプトを探しだしてコピーしてくれるわけです．過去に書いたハンドラーを効率的に利用できるという点では先ほどの Linker と同じです．しかし，Linker の場合，すでにコンパイルしてしまったものを追加しますので，使用の際に修正することはできません．一方，こちらの Build Script ですと，単にスクリプトを探してきて追加してくれるだけですので，コンパイル前に必要に応じて修正することができます．また，Build Over ウィンドウに登録するライブラリ・ファイルにはテンプレートになるようなハンドラーを収めておいて，Build Script で追加した後に，テンプレートをもとにスクリプトを仕上げていくといった使い方もできます．ただし，スクリプトを追加するということは，当然ながら，それだけコンパイルに必要な時間も長くなります．ですから，使用に際して修正の必要がない，それなりに完成されたハンドラーの場合は，Linker で追加するほうが効率的です．

その他の便利な機能として，コンパイルのオプションを細かく設定できること，XCMD ごとに新しいスタックを作成してインストールすることが可能なこと，などがあげられます．マニュアルはスタックの形で付属しています．印刷しないと読みにくいものですが，内容は分かりやすいものになっています．

問題点としては，このスタックで HyperCard を終了しようとすると爆弾が出ること（マニュアルに注意書きがあります），コンパイルのオプションで Allow callbacks をチェックしないでおくと XCMD が作成されないこと（これはバグではないかと思います），などがあります．

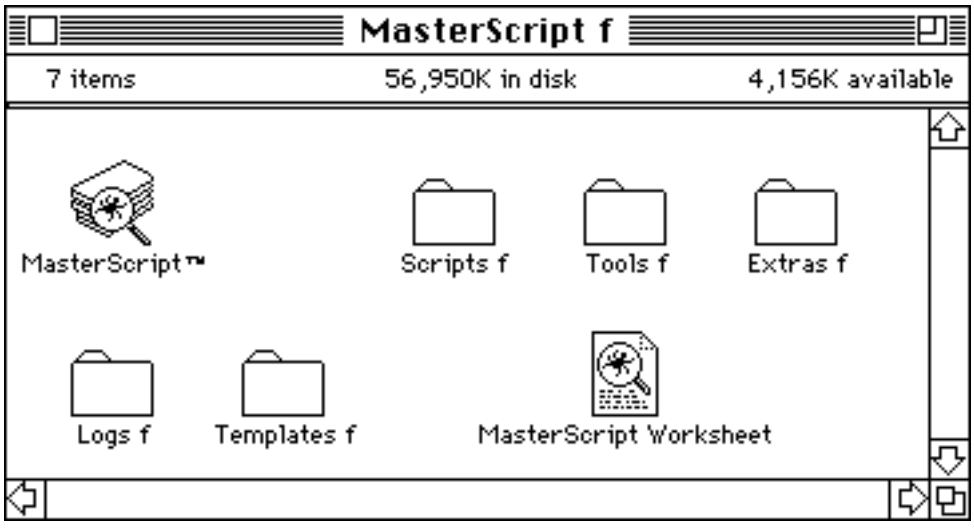
§ 2　MasterScript

MasterScript（以下 MS と略記）は，高機能な HyperTalk プログラム開発環境を提供する製品です．それぞれに便利な機能を備えた，Script(Text) Editor = Debugger, Messages Monitor, Variables Monitor, Externals Monitor の４つのツール（MS では module と呼びます）と，これらのツールを統合的にコントロールするためのコマンドを備えているほか，多くの機能を HyperTalk のスクリプトから使用するための Scripting Interface も備えています．さらに，WorkSheet というテキスト編集ウィンドウがコンソール・ウィンドウとして使えるようになり，このウィンドウから MS の様々なコマンドや HyperTalk のスクリプトを打ち込んで作業ができるのです．このように，MS はHyperTalk のプログラミング環境に，単にスクリプト・エディタのような部品を追加するだけでなく，それらを有機的に結び付けて開発が行なえる環境を提供してくれます．

製品には 140 ページほどのマニュアルが付属しています．たいていの機能はメニューやボタンで直感的に理解・使用できるようになっていますが，十分に使いこなすにはマニュアルを一通り読む必要があります．マニュアルは比較的分かりやすいものだと思います．

●インストール

System 6.0.5 以上で HyperCard 2.0v2 以上であれば使用できます．日本語版 HyperCard 2.0 でも問題ありません．MasterScript XCMD 他のリソースを HyperCard に直接インストール必要がありますので，MasterScript Installer スタックでインストールを行なうようになっています．さらに，"MasterScript f" というフォルダーを HyperCard と同じフォルダーの中にコピーしなければなりません．このフォルダーの中に MS が機能するために必要な様々なリソースやデータファイルが収められています．



高機能である分，メモリーを多く必要とします．英語システムで 2MB （ System 7 では 4MB ）が最低でも必要です．MultiFinder や System 7 の場合，HyperCard に割り当てるメモリーを 1500K 以上にする必要があります．XCMD のパラメーター・ブロックや callback をモニターするExternals Monitor を使用する場合には，MultiFinder 環境で使用し，

HyperCard 用に割り当てられたメモリーの中に MS の Private Heap を確保してそこへコードやリソースをロードするようしなければなりません．Private Heap は最低でも 400K は必要です．結局，MS をフル機能で快適に使うためには，MultiFinder か System 7 で，HyperCard に 1500K 以上のメモリーを割り当てて，そのうちの 500K を MS の Private Heap にしなければなりません．

インストールがすんだら HyperCard を立ち上げ直します．メッセージボックスから "MasterScript" というメッセージを送ると，MS が組み込まれます．MS が組み込まれると，メニューバーに "MasterScript" と "Windows" の二つのメニューが追加されます．9-inch モニターの場合はメニューバーが短いですから自動的に "Mscr" と "Wind" という短い名前のメニューになります．

４つの module のどれを組み込んで使うかを設定できます．必要のない module を外しておくことで，メモリーへの負担を軽くすることができます．

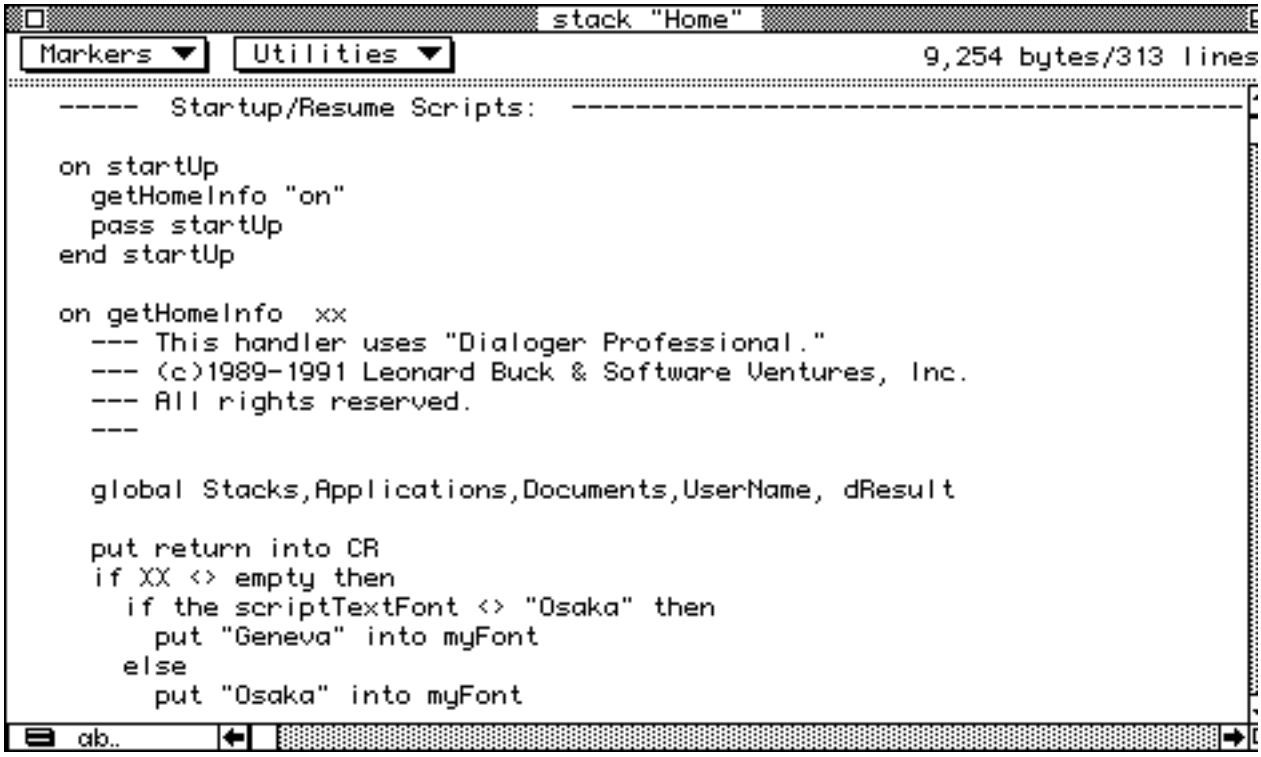
また，スクリプト・エディタを含めて，MS のウィンドウはすべてパレット・レイヤーに開かれるようになっています．つまり，MS のウィンドウは常にカードの上に表示されます．このため，複数のウィンドウを開くと，ウィンドウを隠すか閉じないとカードを操作できなくなります．もっとも，スクリプトを書いたりデバッグしているときに極力スタックにさわらないで作業ができるようにはなっています．

●機能・特徴

それでは，主な機能や特徴を紹介していきましょう．

◇スクリプト編集環境

スクリプトの編集は図６のようなウィンドウで行ないます．



MS はスクリプトだけでなくテキストファイルを開いて編集することもできるようになっているのですが，テキストを編集するウィンドウはすべてこの形のウィンドウになります．スクリプトをテキストファイルとしてセーブしたり，他のオブジェクトのスクリプトとしてセーブすることも可能です．また，どのオブジェクトにも属していない新しいスクリプト・ウィンドウを開き，スクリプトを書いてから，セーブする際に組み込むオブジェクトを選択できるようにもなっています．なお，テキストファイルを編集する場合，32K を越えるファイルでも扱えるようになっています．

このウィンドウにおいて次のようなことが可能です．

・ウィンドウの左上にある Markers メニューでスクリプトを mark しておくと，そこへ一気にジャンプできるようになります．mark は好きなだけ設定できます．すべてのハンドラーの先頭を mark するコマンドもあります．

・Font を自由に変えることができます．Null, Tab, Space, Return などの普段は表示されない Invisible Character を記号で表示させることができます．

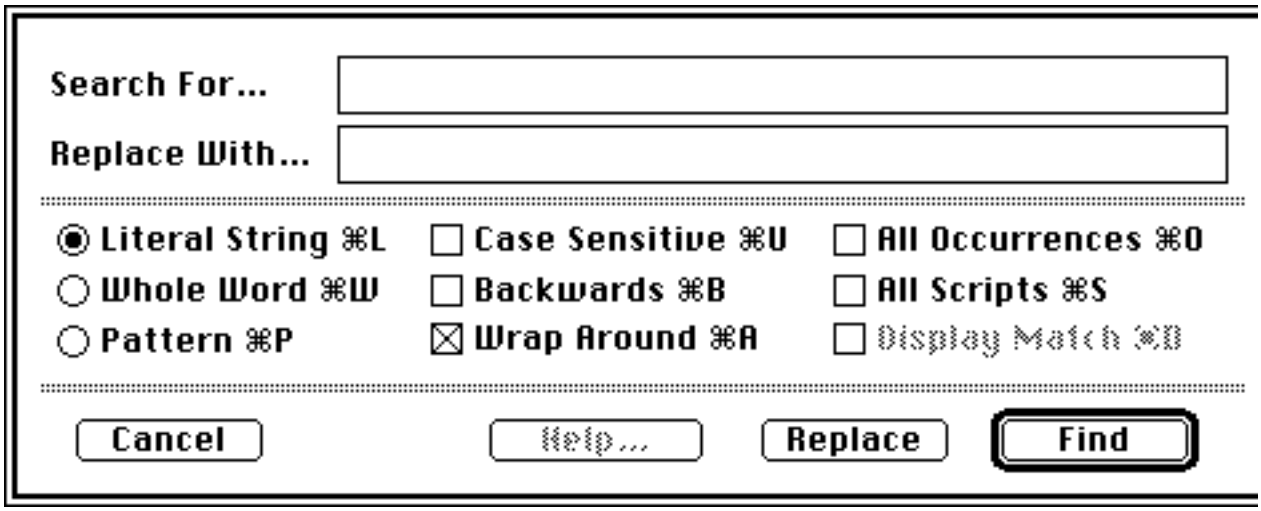
・ファイルの先頭に，そのスクリプトの持っているハンドラーの名前のリストを含む Script Header を付けることができます．Script Header の形式は自分でカスタマイズできます．

・印刷の際に Header と Footer を付けることができます．

・選択した複数行のスクリプトをその場で実行することができます．このため，少しずつテストしながらスクリプトを書き上げていくことができます．

・CompileIt! を持っているなら，選択したスクリプトをコンパイルすることも可能です．つまり，CompileIt! のフロント・エンドとして機能させられるわけです．コンパイル後のレポートを WorkSheet ウィンドウに出力させることができますし，コンパイル後の XCMD のインストール先にスタック以外のファイルを指定することさえ可能です．

・検索／置換もパワフルです．



まず，検索文字列の指定に正規表現 (Regular Expression) が使用できます．また，検索の際に All Occurrences をチェックしておくと，すべての該当する文字列を含む部分の行

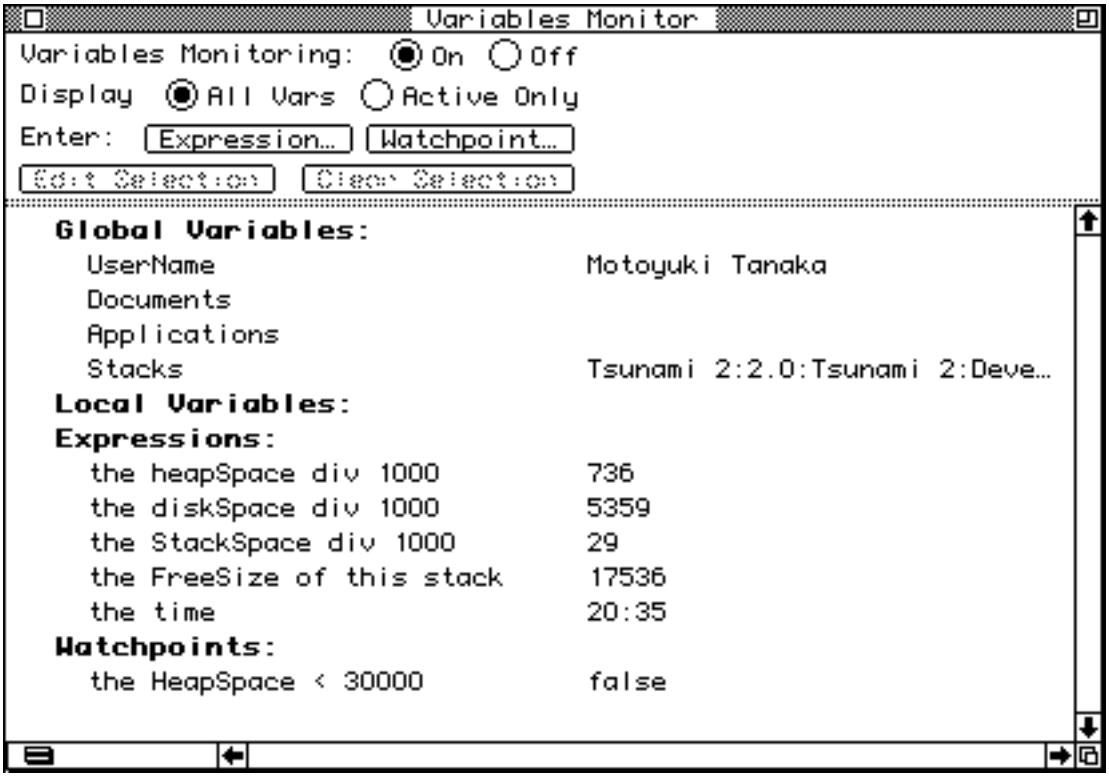
番号のリストをWorkSheet ウィンドウに出力します．All Scripts をチェックして検索を行なうと，現在のスタックのすべてのスクリプトを対象に検索を行ない，行番号とオブジェクトの名前からなるリストを WorkSheet に出力するようになっています．出力されたリストは MS のコマンドになっており，実際に見たい部分をリストから選んで，エンターキーを押すと，即座に，該当部分が選択された状態でスクリプトが開きます．

◇デバッグ環境

スクリプト・デバッガ自体の機能は HyperCard 純正のものと違いはありませが，デバッガへの入り方に便利な方法が加わります．それは，スクリプト・エディタから直接にデバッグ・モードへ入れるというものです．スクリプトを編集している状態で，"MasterScript" メニューの Step か Trace を選ぶと，スクリプトをその場ですぐにデバッガで追いかけながら実行できるのです．文字通り，スクリプト・デバッガとスクリプト・エディタが統合されているわけです．スクリプトに複数のハンドラーが含まれている場合は，ダイアログでデバッグするハンドラーを選ぶようになっています．

◇ Variables Monitor

HyperCard の Variable Watcher にあたる，変数を表示する module です．



Expressions と Watchpoints を設定できるのが特徴です．

Expressions というのは，HyperTalk で値を取りうる式を設定しておくと，その式の値を常に表示してくれるというものです．例えば，the HeapSpace div 1000 という式を

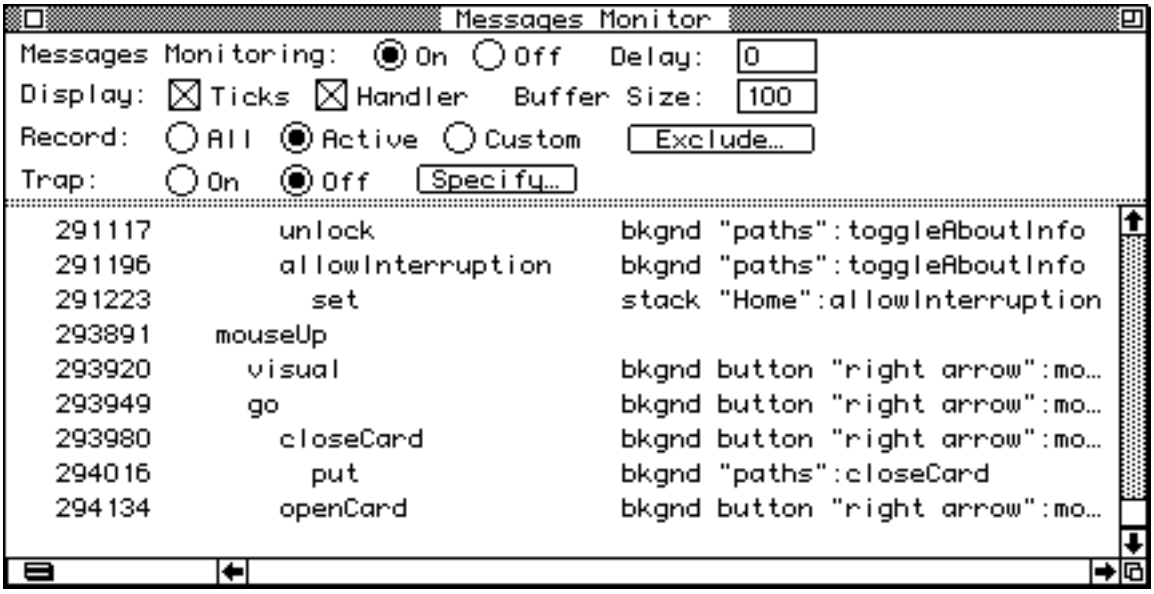
Expressions として設定すると，HyperCard の Heap の残りの大きさを K byte 単位で常に表示してくれるわけです．the time を設定すると時計としても使えます．

Watchpoint の方は，True か false の値をとる，いわゆる論理式を設定しておくと，その式の値が true になったところでアラートを表示してくれるものです．the FreeSize of this stack > 20000 という式を Watchpoints に設定すると，スタックのフリーサイズが 20K を越えた時点でアラートが出ます．

Expressions, Watchpoints とも複数の式を同時に設定できます．また，スクリプト・エディタとの連係も計られており，エディタにおいて選択した行のスクリプトを Expressions あるいは Watchpoints として設定できるようになっています．さらに，変数や Expressions, Watchpoints の設定や表示内容は，すべてこのウィンドウのプロパティとして HyperTalk で扱えます．

◇ Messages Monitor

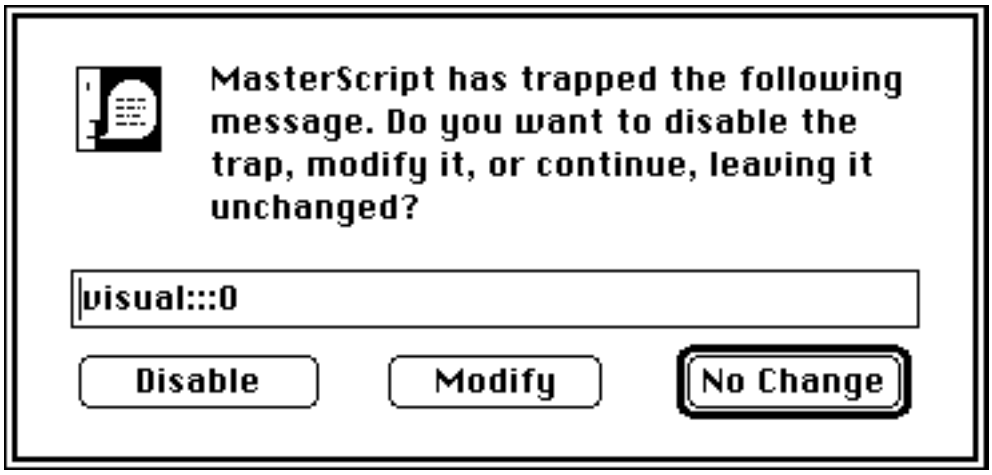
HyperCard の Message Watcher に較べて，格段に細かい設定を行なえるようになっています．



まず，表示する情報として，メッセージ名だけでなく，そのメッセージが記録された時間を ticks で表示させられますし，そのメッセージを発したハンドラーについても表示させられます．

表示するメッセージについても，Exclude で表示しないものを細かく設定できますし，逆に，Custom を選択すると，設定したメッセージだけを表示させることも可能です．

さらに，特定のメッセージをトラップすることが可能です．Trap を On にして，Specify... でトラップするメッセージを設定します．すると，そのメッセージが発せられた時点でスクリプトの実行が中断されてダイアログが表示されます．

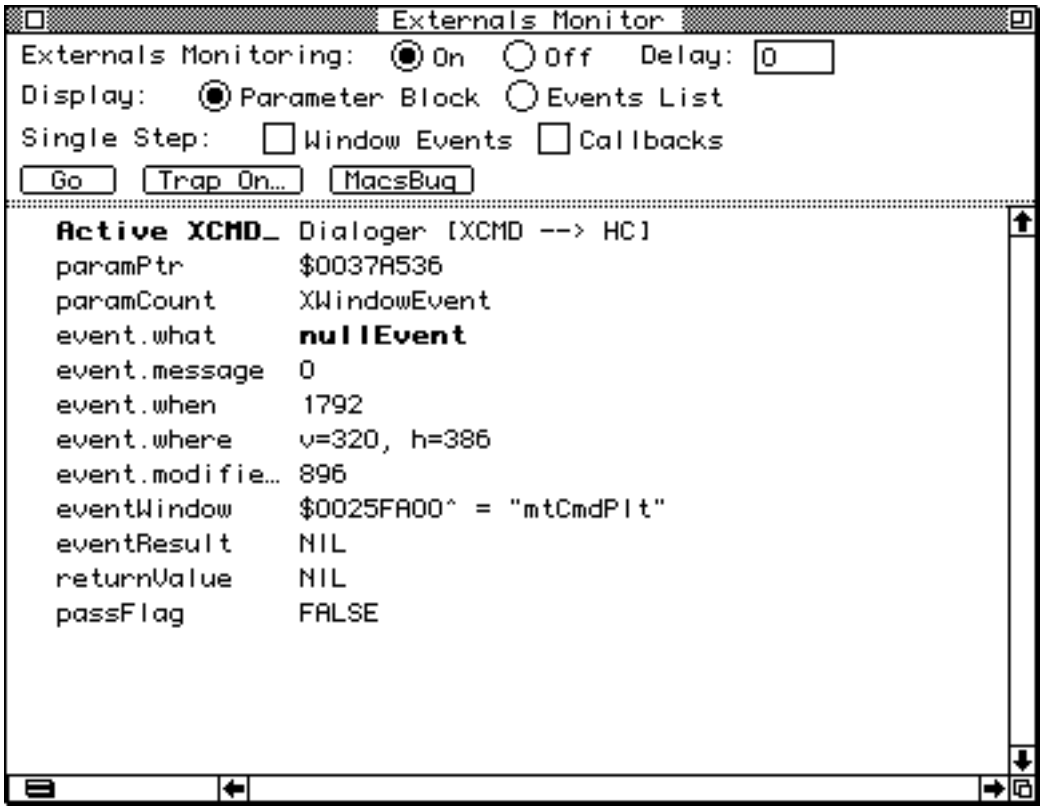


トラップの指定は，どのオブジェクトの，なんというハンドラーが，そのメッセージを何回発した時点でトラップするか，といった具合に非常に細かく設定することも可能です．また，検索のときと同じ形式の正規表現を使用できますので，バックグランドのメッセージならトラップするというような設定もできます．

Variables Monitor と同様に，各種の設定や表示内容については，この Messages Monitor ウィンドウのプロパティとして扱えるようになっています．また，HyperTalk でしか設定できないのですが，MsgsOutputFile プロパティにテキストファイルの名前をセットすると，その時点のモニターの表示内容を指定したテキストファイルに書き込むようになっています．さらに，ファイルへの出力の属性を redirect にセットすると，メッセージをモニターに表示せずに直接ファイルに記録していくようになります．

◇ Externals Monitor

HyperCard が XCMD や XFCN を呼び出すときは，パラメーター・ブロックというデータ構造を通じてやり取りを行ないます．Externals Monitor は HyperCard と XCMD の間でやり取りされるパラメーター・ブロックの内容をモニターするものです．



外部ウィンドウを作成・使用する XCMD の場合は，外部ウィンドウのイベント情報が収められた XWEventInfo レコードもモニターできます．

パラメーター・ブロックの内容をすべて表示するか，単に XCMD から送られた callback を記録するか，表示の形式はこの２通りから選べます．Single Step をチェックしておくと，callback や ウィンドウのイベント情報がやり取りされる度に，その内容を表示して，スクリプトの実行が中断されます．また，Trap を設定しておくと，指定した callback が送られた時に，それをトラップして実行を中断させることが可能です．

◇ Scripting Interface と MasterScript Command

MS の機能の多くは，HyperTalk によって設定・コントロールができるようになっています．"MasterScript" と "Windows" の２つのメニューに登録されているコマンドや，これまで紹介してきた各種 Monitor の設定や表示内容について，HyperTalk でアクセスすることができます．また，MS 独自のコマンドを持っており，これも HyperTalk で利用できます．このように HyperTalk によるインターフェースが設定されていることによって，使用したいコマンドをキーボードで直接入力して実行したり，MS の機能を使用した開発・デバッグ用スクリプトを書いて実行することができるようになっています．

MS 独自の内部コマンドしては，次のものがあります．

１ Menu Commands : MS の２つのメニューの各項目は，メニューを選ぶ代わりに，HyperTalk で MS のコマンドして実行することもできるようになっています．

２ Send Windows Commands : HyperTalk の send コマンドによって，MS のウィンドウに送ることの出来るコマンドです．ウィンドウの選択や表示などのコマンドを送れます．

３ Window Properties : MS のそれぞれのウィンドウはプロパティをいくつも持ってお

り，それを HyperTalk で読みだしたり設定したりできます．プロパティの中には HyperTalk によってしか設定できないものもあります．

４ External Tool : これはリソースファイルとして付属してくる MS の外部コマンドです．"MasterScript f" フォルダの中に，"XTRN" という独自形式のリソースを収めたファイルをいれておくと，そのリソースを外部コマンドして実行できます．製品には ResStrip, ExtractData, EditMenus の３つの外部コマンドファイルが付属しています．

５ External Script File : テキストファイルに HyperTalk で書いたスクリプトを収めて， "MasterScript f" フォルダの中に入れておくと，そのスクリプトを MS の外部コマンドして実行できるようになります．ファイル名がコマンド名になります．スクリプトは，複数行でも，if/then/else や repeat を使ったものでも，あるいはハンドラーを丸ごと収めたものでも，いずれも OK です．また，このスクリプトの中で MS のコマンドを使うこともできますので，MS のマクロを作ることも出来ます．

以上の５種類のコマンドが，MS のコマンドして HyperTalk によって利用できます．スクリプトの中でこれらのコマンドを利用するときは，send とウィンドウのプロパティを除いて，masterscript <コマンド> という書式で呼び出します（masterscript XCMD のパラメーターとしてコマンドを指定する）．send とプロパティの場合は，HyperCard の普通のウィンドウと同じです．

◇ WorkSheet

MS を立ち上げると，さまざまな module と一緒に，必ず，WorkSheet というテキスト編集ウィンドウが組み込まれます．このウィンドウは他のテキスト編集ウィンドウと異なる，特別な機能を持っています．WorkSheet は MS のコンソール・ウィンドウとして扱われるようになっているのです．

普通のテキストファイルのウィンドウと同様に，自由にメモやスクリプトを書き込めますし，それを印刷したり，あるいは，選択部分を HyperTalk として実行することもできます．こうした機能以外に，コンソール・ウィンドウとして次のような特徴があります．

１ MS のコマンドをエンターキーで直接実行できる

WorkSheet でエンターキーを押すと，カーソルのある行の文字列が MS のコマンドとして実行されます．複数の行を選択した状態でエンターキーを押すと，それらの行が順にコマンドして実行されます． MS のコマンドに該当するものが無かった場合は，文字列が HyperCard へ送られるようになっていますので，MS のコマンドではないスクリプトも実行することもできるようになっています．つまり，WorkSheet は，MS が組み込まれたメッセージボックスが，ウィンドウになったものと考えてもらえばよいでしょう．

ファイルの最後の行ではなく，カーソルがおかれているか選択されている部分がコマンドとして実行されるということは，ウィンドウを逆スクロールして過去に実行したコマンドをもう一度実行させられるわけですし，パラメーターだけ変更して実行するといったことも簡単にできるわけです．さらに，テキスト編集ウィンドウには mark 機能が備わっていますので，良く使うコマンドは mark しておけば，いつでもすぐに呼び出して実行できます．

２ コマンドのエラーメッセージが出力されるウィンドウである

MS のコマンドでエラーが起きると，この WorkSheet にエラーメッセージが表示されます．

３　検索のレポートが出力されるウィンドウである

検索の際に，"All Occurrences" や "All Scripts" をチェックすると，該当する行とオブジェクト（あるいはウィンドウ）の名前をリストにしたものが，この WorkSheet に出力されます．

４　CompileIt! のコンパイル結果が出力されるウィンドウである

選択したスクリプトを CompileIt! によってコンパイルした場合，コンパイル結果のレポートをこの WorkSheet に出力させることができます．どのテキスト編集ウィンドウでもコンパイルすることは可能ですが，コンパイルのレポートを得ることができるのはこの WorkSheet だけです．

●使用してみて

HyperCard は HyperTalk のプログラミングと実行が手軽にできることが魅力の一つです．ちょいとスクリプトを書いて，とりあえず動かしてみて，後から思いつくままにその場で改良を加えていく…　こういうお気楽プログラミングの喜びを与えてくれたからこそ，多くの人が HyperTalk でスタックを作っているのではないでしょうか．この MS はそうしたお気楽プログラミングに使うには重装備過ぎます．普通にスタックを作るのにここまでの機能は必要ないでしょう．もし純正のスクリプト・エディタに不満ということであれば，前に紹介した ScriptEdit 2.0 でたいていの人には十分でしょう．スクリプトを書くということだけに限れば，ScriptEdit 2.0 の方が圧倒的に使いやすいと私は思います．

しかし，一方で，HyperCard は十分に実用的なソフトウェアを作れるだけの力を秘めています．XCMD をうまく使って機能を補いながらながら，HyperCard / HyperTalk の特性を生かしたプログラミングを行なえば，他の言語で開発したソフトウェアに劣らないものを作ることができます．そうした本格的な開発を行なう方，プロの方などには，この MS は非常に重宝するでしょう．

私自身は，MS の様々な機能を試しながら，HyperCard が 2.0 になったときの興奮を思いだしました．まともになったスクリプト・エディタ，デバッガや Watcher に出会い，わくわくしながらそれらを試した，あの時のような「幸福な」時間を MS と共に過ごしました．メニューから選ぶだけでそのまま Step できるエディタ&デバッガ，Expressions の設定で状態のモニターにも使える Variables Monitor，せっせとファイルに記録してくれる Messages Monitor，そして一度使いはじめると手放せなくなる WorkSheet…　私のように，HyperCard を何よりも HyperTalk のプログラムの開発・実行環境として使用している方，スクリプトを書いたり試しているときがスタックを作っている時間のうちでもっとも楽しいという方は，この MasterScript は良きパートナーになることでしょう．

●問題点

Mac SE 4MB，System 6.0.8 / 漢字Talk6.0.7，HyperCard 2.1 で動かしたところ，次のような問題が起きました．

まず，MS のバグではないかと思われるものとして，

1　Page Setup で，Header あるいは Footer のフォントを設定しようとすると，かならず爆弾が出ました．

2　MS に付属してくる外部リソースコマンドのうち，ResStrip を使おうとすると必ず爆弾が出ました．

3　MultiFinder で Private Heap に MS をロードして使用している際に Messages Monitor の表示オプションを変更しようとすると，よく爆弾が出ました．Finder で使っているなら問題は起きません．

4　HyperTalk で MS のダイアログを呼び出した場合，ダイアログが消えた後のmodule のウィンドウのアップデートがきちんと行なわれません．ダイアログの下になっていた部分が真白になってしまいます．メニューから呼び出したときには問題ありません．

次に，これは MS の仕様によるものとして，

5　スクリプト・エディタのようなテキスト編集のウィンドウでは日本語が全く使えません．フォントを日本語のフォントにしても，入力はもちろん，表示すらされません．これは，テキスト編集の機能を独自のルーティンで行なっているためだと思われます．

その他として，もっぱら私のマシンが SE 4MB であることから来ると思われるものとして，

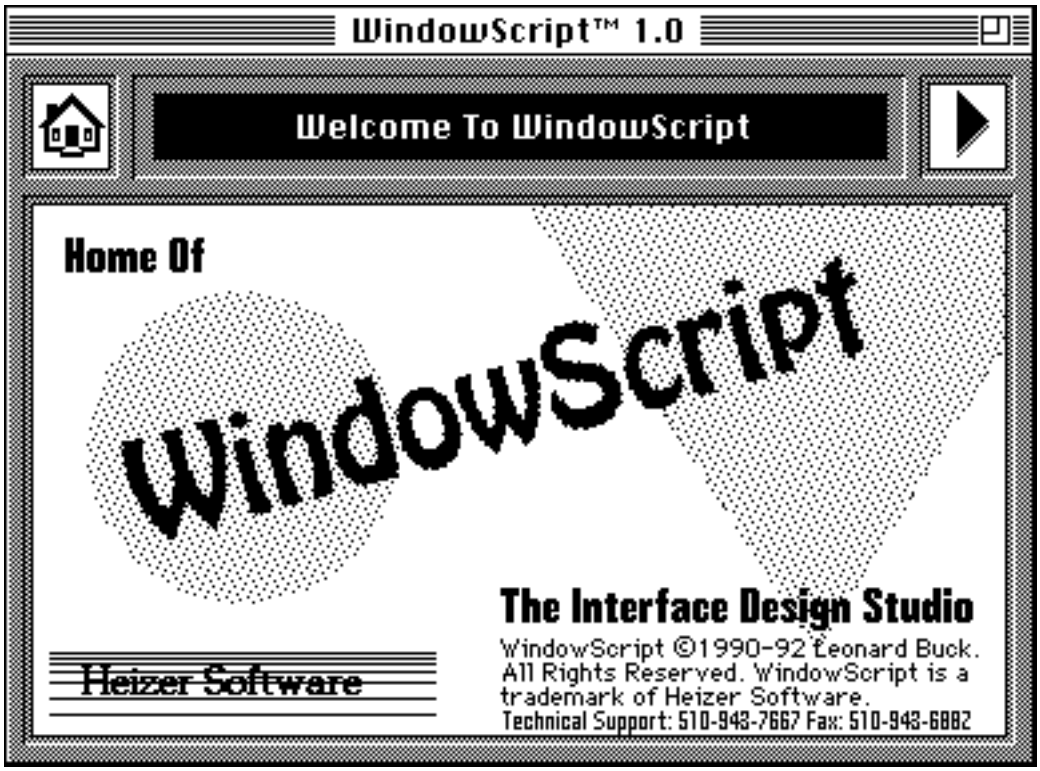
6　SE で使うとキー入力に対するレスポンスが遅いのが気になります．WorkSheet でコマンドを打ち込むぐらいであれば我慢できるのですが，スクリプトを書くには使えないと感じました．やはり高機能ゆえに SE には重すぎるのでしょう．（結局，スクリプトを書くときはエディタを ScriptEdit 2.0 に切り替えるようにしました）．

7　やはりメモリーが足りないと不安定になります．4MBでは，漢字Talk 6.0.7・MultiFinder で動かすには辛いですね．

8　それぞれの module のウィンドウが大きいですから，SE の 9 inch モニターでは，複数の module を同時に見るのは難しいです．デバッグの際にはこまめにウィンドウを切り替える必要があります．

§ 3　WindowScript

WindowScript（以下 WS と略記）は，一口で言えば，その名の通りウィンドウを HyperTalk スクリプトで扱えるようにするツールです．



ウィンドウ，ダイアログ，パレットが，HyperCard でボタン，フィールド，ペイントツールを使ってカードをデザインするのと同じ要領作成できますし，それを HyperTalk でコントロールすることが可能になります．アプリケーションと同等のインターフェースを手軽に構築することを可能にしてくれる，HyperTalk 用 Interface Builder です．

ウィンドウ（以下，ダイアログやパレットも含めた総称として「ウィンドウ」を使います）のアイテムとして１５種類のものが使用できます．

Button	Radio Button	Check Box
Label	Static Text	Edit Text
ICON, ICN#	Picture	QuickTime objects
List	Popup	Control
Box	RoundRect	Line

マックのビジュアル・インターフェースを構成する要素はすべて用いることができます．また，カラー対応ですのでカラーのウィンドウを作成することもできますし，QuickTime の Movie を使うことも可能です．System 7.0 であれば，すべてのアイテムにバルーンヘルプを組み込めるようにもなっています．

この WS は，前に紹介した Dialoger Professional の後を継ぐ製品として発表されました．作者も同じ Leonard Buck 氏です．しかし，Dialoger との互換性はなく，全く別の製品になっています．Dialoger に比べると，ウィンドウ（ダイアログ）の作成や，それをコントロールするスクリプトの書き方など，あらゆる点で使いやすさが増しています．また，WS には 100 ページほどのマニュアルが付属してきます（スタックにも同じ内容のヘルプが収められています）．スクリプトの書き方についての説明が不足しているように思いますが，必要な情報がコンパクトにまとめられたマニュアルになっています．

●インストール

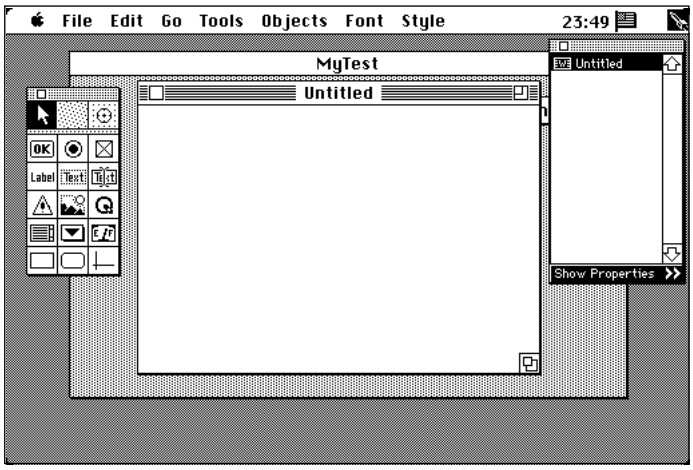
HyperCard 2.0v2 以上に対応しています．日本語版 HyperCard 2.0 でも使用できますが，後で述べるような問題を抱えていますので，英語版 HyperCard での使用が望ましいと思います．

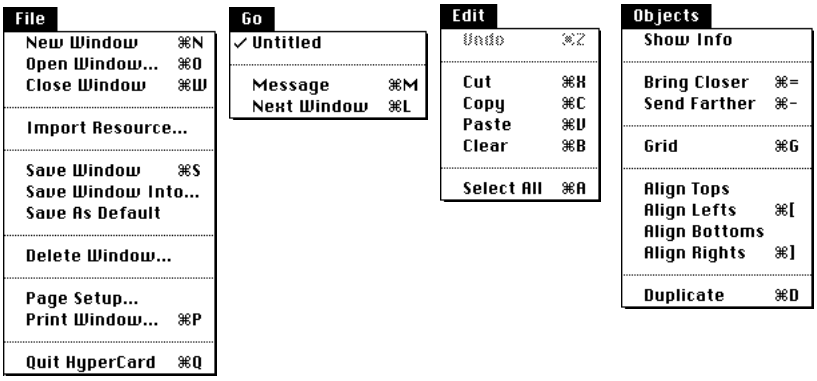
WS スタックの "Installation..." ボタンでインストールします．インストールすると，HyperCard の Edit メニューの一番下に，WS を呼び出すための "Window..." という項目が追加されます．ウィンドウを組み込みたいスタックで，この "Window..." を選ぶと，WS を使ってウィンドウが作れるようになっています．ちょうど HyperCard の Icon Editor と同じような感じで WS が組み込まれるわけです．

ウィンドウの作成・編集のためには WS をインストールする必要がありますが，できあがったウィンドウを使用するだけであれば WS スタックは不要です．ただし，ウィンドウを使うために必要な XFCN などのリソースをスタックにインストールする必要があります．リソースのインストールは WS スタックで行なえます．

●ウィンドウの作成

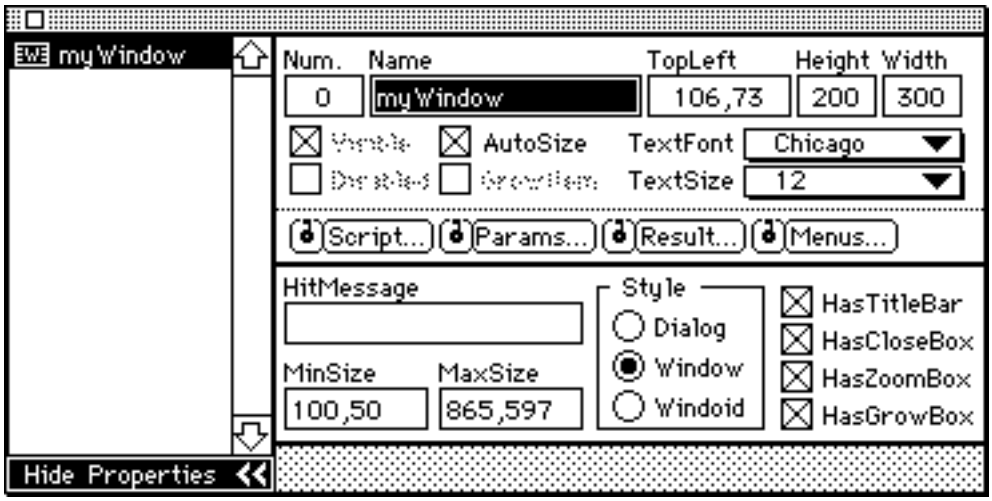
Edit メニューから "Window..." を選ぶと，ウィンドウを選ぶダイアログが現われます．ここで New を選ぶと，新しいウィンドウ，ツールパレットそして Property Picker が現われます．このとき，メニューも WS 独自のメニューに置き代わります．





ウィンドウのアイテムの作成は，ドローソフトでオブジェクトを書いていく要領で行ないます．つまり，ツールパレットから作成したいアイテムの種類を選び，そのアイテムを配置したい場所をウィンドウ上でドラッグすれば，そこにアイテムが作成されるようになっています．作成してからでも Selection ツールによってアイテムの位置は自由に移動できます．また，ボタンなどのいくつかのアイテムは，ドラッグした範囲にかかわらず，最初はマックの標準の大きさで作成されるようになっています．このため，適当にドラッグして配置していくだけで，マックらしさをきちんと保ったウィンドウが出来上がっていくようになっています．

Property Picker の Show Properties をクリックすると，Property Picker が広がってウィンドウやアイテムのプロパティ（属性）を細かく設定するようになります．



また，アイテムを Selection ツールでダブルクリックすると，そのアイテムのプロパティを設定する状態で Property Picker が広がるようにもなっています．アイテムにスクリプトを埋め込む場合やバルーンヘルプで表示する文章を設定する場合も，この Property Picker を使います．

ツールパレットでアイテムを選び，ウィンドウ上でドラッグしてそれを配置し，Property Picker でプロパティを設定していく．この繰り返しでウィンドウを作っていきます．ツールパレットから矢印ツールを選ぶと，作成途中のウィンドウでも，HyperCard から呼び出した場合と同じように動作するようになっています．ちょうど，HyperCard でスタックを作っている最中に，ブラウズツールにさえすれば，いつでもボタンやフィールドが機能するのと同じことが，ウィンドウにおいて可能になっているのです．ですから，動作を試しながら，少しずつ仕上げて

いけるようになっています.

●スクリプト

WS で作成したウィンドウは HyperTalk でコントロールできます．個々のアイテムのプロパティの変更はもちろん，新しいアイテムを作ってしまうことも可能です．ウィンドウ自体は WindowScript XFCN で呼び出すようになっていますが，呼び出されたウィンドウは，HyperCard のプロパティと同じように，set/get/put を使ったスクリプトでコントロールできます．いくつかのコマンドを send によってウィンドウに送ることも可能になってます．また，WS には wsGet, wsSet, wsSend というウィンドウのコントロール専用の XCMD や XFCN がありますので，こちらを使うこともできます．XCMD を使ったほうがパラメーターの設定が楽ですが，HyperTalk の set/get を使ったほうが処理の速度は早くなります．ただし，モーダル・ダイアログの場合は XCMD によってコントロールしなければなりません.

ウィンドウをコントロールするスクリプトは２通りの書き方が可能です．１つはアイテムに直接スクリプトを埋め込む方法．もう一つは，スタックにハンドラーとして書く方法です.

アイテムにスクリプトを埋め込む場合は，Property Picker でスクリプトを埋め込むアイテムを選んでから，Script... ボタンをクリックします．すると，HyperCard のスクリプト・エディタと同じような，スクリプトを設定するウィンドウが現われますので，ここでスクリプトを書きます．ウィンドウ自体を含めて，すべてのアイテムがスクリプトを持てるようになっています．アイテムに埋めこまれたスクリプトは，そのアイテムに相応しいイベント（たとえばボタンであればクリック）が起こった場合に実行されます．ウィンドウをコントロールするスクリプトでは，アイテムのプロパティを操作することが多くなるのですが，こうしたスクリプトを簡単に書くために Scripting Assistant が使えるようになっています．これはダイアログで，操作したいアイテムとそのプロパティを選ぶと，必要な HyperTalk の命令を組み立てて，スクリプト・エディタのカーソルの位置に挿入してくれるものです．これがあるおかげで，いちいちマニュアルでプロパティを確認する必要もなく，快適にスクリプトが書けるようになっています．ただし，この Scripting Assistant は Property Picker でアイテムにスクリプトを埋め込む場合にしか使えません.

ウィンドウをコントロールするスクリプトをスタックの方に書く場合は，イベントごとにウィンドウからスタックへメッセージが送られるように設定し，このメッセージを受けとめるハンドラーを書くことになります．スタックへ送るイベント・メッセージは，ウィンドウの HitMessage プロパティとして設定するようになっています．例えば，HitMessage プロパティを "myWindowHit" にセットすると，ウィンドウのイベントごとに myWindowHit メッセージがスタックに送られるようになるわけです．イベントの内容やイベントの対象になったアイテムの情報は，このメッセージのパラメーターとして渡されます．ですから，スタックの側には，このメッセージを受けとめる myWindowHit ハンドラーを書くことになります．この方法の場合には Scripting Assistant は使えませんが，そのかわり，デバッグには HyperCard のデバッグ機能が使えます．メッセージの送り手はウィンドウであっても，ハンドラーそのものは普通のメッセージ・ハンドラーですから，スクリプトを書いたりデバッグしたりする作業は，WS を使っていない場合となんら変わらないわけです．また，スクリプトの実行速度は，アイテムに埋め込むよりも，スタックにハンドラーを書いた場合の方が早くなります.

さらに，スタックにスクリプトを書く方法と，アイテムにスクリプトを埋め込む方法とを併用することも可能になっています．ユーザーのアクションなどによってイベントが起こった際に，

アイテムがスクリプトを持っていればそれが実行され，なければ設定したメッセージがスタックへ送られるようになります．ですから，複数のウィンドウを持ったスタックの場合，ウィンドウに共通する処理はスタックにスクリプトを書いておき，ウィンドウごとに異なる部分についてはウィンドウのアイテムにスクリプトを持たせる，といったスクリプトの階層的な分散ができるわけです．

●ウィンドウの保存形式

WS のウィンドウは独自の LENS というリソースで保存されます．ダイアログであれウィンドウであれパレットであれ，すべてこの LENS リソースになります．通常の DITL や DLOG などのリソースとしてセーブすることはできませんし，逆に ResEdit などで作成した DLOG などのリソースを WS で直接使用することもできません．すでにある DLOG リソースを使用したい場合は，Import Resource を使って WS に取り込めば LENS に変換されるようになっています．

LENS リソースは，ウィンドウのタイプや各アイテムのプロパティ，スクリプトをテキストで記述したものとなっています．ウィンドウを開くダイアログで Open as TEXT をチェックしておくと，ウィンドウについて LENS リソースと同じ形式で記述したテキストを見ることができます．また，このテキストの内容を変更すれば表示されるウィンドウも修正されます．つまり，WS は， XFCN がウィンドウを表示する際に，LENS リソースに収められたウィンドウの記述テキストを読んで，それをもとにウィンドウを作成するのです．ですから，自分でウィンドウについて記述したテキストデータを用意しておいて，それを XFCN にパラメーターとして渡してウィンドウを作らせることも可能になっています．WS はリソース・インタープリターでもあると言ってよいでしょう．

マックの通常のウィンドウやダイアログとは大きく異なっていますが，この独自フォーマットの採用によって，ウィンドウのアイテムに直接スクリプトを埋め込んだりすることが可能になっているようです．

●使用してみて

図は試しに作ってみた簡単なエディタ・ウィンドウです．

ウィンドウの縦のサイズをマウスで自由に変えることができるようにし，Open ，Save ，フォントとサイズの変更といった基本的な機能を組み込み，ついでに文字とその背景とウィンドウのそれぞれの色をカラーテーブルで自由に設定できるようにしてみました．スクリプトを書くことも含めて，作りはじめて１時間足らずで完成．もちろん，ちゃんとエディタとして機能するものに仕上がりました．

マックのプログラミングに興味を持ってしまった人間にとって，ウィンドウを思いのままに作り上げて使いこなすということは，いつか辿り着きたい目標（夢?）の一つであると言ってよいと思います．今回，WS を手にして，その憧れのウィンドウが，こんなに簡単に作れてよいのだろうか？という驚きと興奮を感じました．確かに，HyperCard はカードという形でビジュアル・インターフェースを手軽にデザインできるようにしてくれました．しかし，HyperTalk のプログラムのインターフェースとしてカードを捉えたとき，カードでは表現したくてもできないことが多いのも事実です．WS の親ともいうべき Dialoger Professional はダイアログを思いのままにできるようにすることでこの制約のかなりの部分を取り払ってくれました．そして，この WS によって，私が HyperCard のカードで感じていた，言葉が使えないもどかしさのようなものが，完全に解消されたように思います．もちろん，単語を覚えただけでは英語が喋れないのと同じように，ウィンドウが思いのままになるからといってすぐに素晴らしいインターフェースを構築できるわけではありません．むしろ，選択の幅が広がる分だけ，困難さは増すでしょう．しかし，手軽に気軽にウィンドウを作って試せる WS であれば，その試行錯誤すらも楽しいものになると思います．

私のように，HyperCard をもっぱら HyperTalk のプログラミングと実行の環境として使っている方には，アプリケーションと同等のインターフェースをもたらすツールとして，お勧めします．

また，HyperCard のインターフェースとして使用する以外に，アプリケーションのウィンドウのデザインを考え，プロトタイプを作るものとしても，十分に使えるように思います．もっとも，アプリケーションで用いるようなリソースは作ってくれませんから，あくまでも，スケッチブックとして使うことになるでしょうが．

さらに，最近ではマックはカラーが標準といえるような状況になってきましたが，現時点では

HyperCard はカラーやQuickTime の Movie に対応していません．Picture XCMD や QuickTime 用の XCMD もありますが，決して使いやすいものではありません．WS であれば，カラーのウィンドウに QuickTime の Movie を表示してそれを HyperTalk でコントロールするといったことが簡単にできます．図はスタック作家のるじるしさんの絵を表示するカラーのウィンドウです．

このようなカラーのウィンドウがスタックの中で手軽に使えるようになるのです．また，白黒の環境で同じダイアログを呼び出すと，ちゃんと白黒で正しく表示してくれるようになっています．このようなわけですので，HyperCard のカラー環境対応を待ち切れない方，あるいはカラーでのプレゼンテーションに HyperCard を使いたいという方にも重宝するように思います．

●問題点

1　全体に重い．これは高機能の宿命かも知れませんが，ウィンドウを作成しているときのツールのレスポンスが鈍いです．Propertry Picker で編集するアイテムを切り替えたり，スクリプトを入力しようとすると，ちょっと待たされます．出来上がったウィンドウ自体の動作はそれほど遅くなりませんが，ウィンドウをリサイズしたときの画面のアップデートや， Edit Text フィールドへ文字を入力する場合は，レスポンスが鈍くなります．先ほどのサンプルのエディタも，文字入力には使えないという感じです．もっとも，LC で使ってみる機会があったのですが，これくらいのマシンだとそれほどレスポンスの鈍さも気になりませんでした．ついでに Quadra 900 でも試してみたところ，それはもう気持ち良くバリバリとウィンドウを操ることができました．やはり，快適に使用するためにはそれなりの CPU のパワーが必要なようです．色のこととといい，CPU のパワーのことといい，SE では不満の募る時代になってしまったことを実感させられました．

2　これはバグのようですが，Property Picker を使うと，本来関係のないはずのシステムフォントが勝手に Chicago フォントに切り替えられてしまいます．一度切り替わってしまうと，HyperCard を終了するまで Chicago のままになってしまいます．ですから，日本語版 HyperCard を使っていると，WS を使った後ではメニューやダイアログがすべて文字化けしてしまいますので，とてもじゃないですがそのまま HyperCard を使っていられなくなります．

このシステムフォントの切り替えを起こす犯人は，WS に付属してくる wsSet XCMD のようです．この XCMD はウィンドウのプロパティを変更するためのものなのですが，この XCMD でフォント関係のプロパティに変更を加えると，その際に勝手にシステムフォントを Chicago に変更してしまいます．Property Picker もこの XCMD を利用していますので，それでシステムフォントの切り替えを起こしてしまうようです．また，出来上がったウィンドウで wsSet XCMD を使ってフォントの変更などを行なった場合も，この問題が発生します．

とりあえず，フォント関係のプロパティの変更を行なうときは，wsSet XCMD を使わずに，set コマンドで行なうようにすれば問題は回避できますが，Property Picker については我慢するしかないようです．

3：WS を組み込もうとするさいに，ID = 28 のシステムエラーが出ることがありました．メモリーの StackSpace が足りなくなることがあるようです．もっとも，これは私の方の環境の問題なのかもしれません．しかし，メモリーに余裕があるほうが良いのは確かでしょう．

§４　おわりに

Toolbox を扱える HyperTalk コンパイラにはじまり，それに Linker が加わり，MPW のようなシェルが使えるようになり，自由にビジュアル・インターフェースを構築することも可能になりました．こうしたツールがそろった今，HyperCard も，他の言語のものに劣らない，立派なソフトウェア開発環境になったと言えるのではないでしょうか．もちろん，アプリケーションや INIT を作ったりすることはできません．あくまでもスタックを作れるだけです．しかし，たかがスタックとはいえ，マックの持つ膨大な資産をフルに生かしたプログラムに仕上げることが可能になったわけです．そのような作品を作れる道具がそろってきたのです．そして，何より

も重要なことは，「スタック作り」という一本の道でここまで辿り着けるということではないでしょうか．初めてボタンのスクリプトの visual effect を書き換えてみてプログラミングの面白さに触れたその日から，Inside Mac の森の中を右往左往する今日まで，HyperCard だけで，HyperTalk だけでやってこれたということが，私には素敵なことのように思えます．たかがスタック作りとはいえ，これだけの深さを秘めている HyperCard の素晴らしさに改めて感心するとともに，その深みへといたる道を開いてくれる Heizer Software の製品に出会えたことを喜んでいます．しかし，まぁ，ダニー・グッドマンの『ザ・ハイパーカード』をわくわくしながら読んでいた頃は，まさかスクリプトを書くために Inside Mac のページをめくることになろうとは思いもしませんでした．

最後になりましたが，サンプル・ウィンドウ作成にあたって，素敵な絵を使用させていただくことを快く認めてくださった、るじるしさんに感謝します．

※この記事で取り上げた製品（いずれも Heizer Software 社）

・CompileIt! Linker　　$49.00
・CompileIt! DEveloper Edition Interface　$49.00
（CompileIt! 2.1　$149.00 ）

・MasterScript　　$129.00
・WindowScript　　$149.00

Heizer Software
1941 Oak Park Blvd., Suite 30
P.O.Box 232019
Pleasant Hill, CA 94523
U.S.A.

なお，MasterScript と WindowScript は，Working Model （デモ版）が $10.00 で手に入ります．（ NIFTY-Serve にアップロードされています）