製品

オブジェクト指向によるインタフェース設計を実現したNeXTStep

●米ネクスト社が昨年発表したThe NeXT Computer Systemの最大の魅力はユーザー・インタフェースに対する明解な哲学からトップダウンに設計された開発・操作環境，NeXTStepを装備していることである。
● NeXTStepはプログラマに対してオブジェクト指向型の開発環境を，エンドユーザーにはコマンドを使うことなくコンピュータの利用を可能にするイベント駆動ベースの操作環境を与える。
● NeXTStepの考え方が今後のユーザー・インタフェース構築手法に大きな影響を与えることは間違いない。現時点のテスト・バージョンはシステム・ソフトウエアまわりがやや不安定だが，今後，ライバルのMacintosh II相手にどこまで戦えるか楽しみである。

苫米地英人/カーネギー・メロン大学計算機科学部機械翻訳センター・同大学哲学科研究員

Macintosh IIキラー，The NeXT Computer Systemの登場はその仕様——CPUはクロック周波数25 MHz（メガヘルツ）のモトローラ68030マイクロプロセサ，浮動小数点演算用コプロセサは25 MHzの68882，56001 DSP（ディジタル・シグナル・プロセサ）チップ，8 MB（メガバイト）のRAM，17インチ・モニター，256 MBの光ディスク，OS（オペレーティング・システム）はMach，アプリケーション・ソフトとして数式処理プログラムMathematica，データベース管理システムSybaseなどを標準装備——がいずれも破格のものであったため，登場以前から早くもうわさが広がっていた。

実際，カーネギー・メロンの大学関係者向けコンピュータ・ストアの価格表には昨年夏からすでにこの組み合わせで，大学関係者価格6500ドルの表示が出ていた（同様のセットをMacintosh IIで組めば，大学価格でも1万ドルを越える）。

とは言っても，今のところNeXTは市場に出ていない。大学でもサイト・テスト用のリリース0.8とよばれるシステムをのせたNeXTマシンが各研究所で試験的に使われているだけである。

ただ，ウインドウ・ベースのUNIXマシンとして設計されただけのことはある。開発環境の面から見ると，Macintoshキラーと言うよりはSun-3キラーと言う感じであり，Sun-3，IBMRT/PC，HP 9000といったワークステーション市場に大きく食い込みそうなマシンである（写真1）。

NeXTの魅力は大雑把にいえば，UNIXハッカーにとっては，まずUNIX 4.3 BSDとバイナリ・レベルで互換性を持つ分散OS，Machが乗っていることである。そして，68030をCPUに使った大画面のマシンとして格安であることであり，GNU EMACSをはじめとするGNUのツールが乗っていることである。さらに，AIハッカーにとっての魅力は，Allegro CommonLispの標準装備であり，

大学などのユーザーであれば，“The CommonLisp”であるCMU-CommonLispや数少ない本物のパラレルLispであるMultilispがMach上で使えることも大きな魅力である。

また，DSPチップの搭載はアプリケーションとして特別なハードを必要とせず音声自然言語理解システムを載せられることを意味する。このことは，我々のように音声認識への利用を考えているものには，他のマシンとは大きく違う要素を持っている（写真2）。

NeXTのもう一つの特徴は，ハッカー以外の（普通の）プログラマをもデベロッパの対象としたMachマシンだということである。CMU(カーネギーメロン大学）では，ハッカー以外の人間が，Mach上に直接何らかのアプリケーションを書くことは有りえず，ユーザー・インタフェースの構築はMachのカーネルによほど詳しくなければむずかしい。

NeXTではこれをNeXTStepと呼ばれる開発環境を用いて自動化することを試みている。NeXTStepはNeXT以外のマシン上にも供給されることになっており（実際，すでにIBM　RT/PC上にも載っている），Machの環境として将来が楽しみである。

オブジェクトの考え方を取り入れたユーザー・インタフェース開発環境として他に，Sun上の「Xview」や，NECのX Window用の「鼎（かなえ）」，キヤノンの「Concord」などがある。NeXTStepの明解な哲学は，今後こういった方向に大きな影響を与えていくことと思われる。

NeXTコンピュータのユーザー・インタフェース設計哲学

ネクスト社はユーザー・インタフェースを初心者とエキスパートの双方のニーズを満たすものでなければならないと考えている。このために，ユーザー・インタフェースは，第一に，初めてのユーザーや使用頻度の低いユーザーにとって学びやすく，覚えやすいものでなければならない。特に，長い期間コンピュータを触る機会がなかったとしても，再学習の必要がないものでなければならない。

第二に，エキスパートにとっては，早くて，効率の良いものでなければならない。特に作業中の仕事から注意をそらせてしまうようなものがユーザー・インタフェースの中にあってはならないということを強調している。

このような課題に対しては，グラフィックなイベント駆動型のユーザー・インタフェースが特に望ましい。これはグラフィックなオブジェクトが現実の様々のアイテムの特徴をとらえることによって，ユーザーが日常生活で経験している物の形や動きを参考にしてインタフェース上のイベントを利用できるからである。

例えば，グラフィックスで表されたボタンには，本物のボタンの様な機能を，また，ウインドウには紙や便せんのような機能といったように，グラフィックなオブジェクトに，実世界の対応物と同様な機能を持たせておく。これによって，コンピュータ操作が特殊な世界の出来事ではなく日常生活の延長であるようなユーザー・インタフェースを構築できる。

NeXTユーザー・インタフェースの基本原則

NeXTのユーザー・インタフェースは，いくつかの基本原則にのっとって設計されており，特に以下の4つを主要な原則としている。

＊ユーザー・インタフェースは，すべてのアプリケーションを通じて一貫していなければならない。
＊ワークスペースと個々のウインドウは，常にユーザーの管理のもとにある。
＊インタフェースは，ユーザーに違和感のないものでなければならない。
＊キーボードではなくマウスが主となる入力ツールである。

以下，各々の原則について説明する。

一貫性の原則

すべてのアプリケーションが同一の基本的なユーザー・インタフェースを持っていれば，個々のアプリケーションを学ぶことはより容易になる。これは，自動車の運転に慣れると，似通った状況のもとでは，ほとんど無意識に種々の操作ができるようになることに似ている。異なるアプリケーションでも，一般にメニューからの選択，ウインドウの操作などは共通している。すべてのアプリケーションが基本的なユーザー・インタフェースを統一しておけば，同一の基本的操作に慣れることにより，無意識にコンピュータの操作ができるようになるはずである。

ユーザー・コントロール

ワークスペース（ユーザーが作業す

NeXTの魅力

OSにMachが走り，WindowをサポートしてCommonLispが乗っているという環境は，われわれカーネギー・メロン大学（CMU）の研究者にとっては，文字どおりHome Environmentといえる。実際，NeXT上の環境は，CMUの各研究所でMach上の環境に慣れた者にとっては極めて自然なものに仕上がっている。

逆にいえば，NeXT上の環境は拡張UNIXの環境としてエンドユーザー側から見た場合，特に目新しいものはなく，期待はずれの部分があるのも否めない。

ただこれは，OSからかなり離れたAIなど（例えばLisp上での使い勝手）での話である。Windowまわりなどのユーザー・インタフェース部分に関しては，NeXTStepがオブジェクト指向のユーザー・インタフェースと開発環境を提供する。このため，CMUでのMach上の開発がMachとXWindow Version 11を熟知したハッカーたちに任かされているのに対し，NeXTでは一般のプログラマはもちん，エンドユーザーにさえユーザー・インタフェースが開発できるようになっている。

これはNeXTの開発方針が特に教育用のグラフィック・インタフェースを重視したアプリケーションの開発が容易なマシンだったからで，ユーザー・インタフェースの仕様と開発の簡易化に，ネクスト社が特にNeXTStepの形で力を注いだためである。

あえていうならば，一部のハードまわりを除けば，ソフト面でオリジナリティを評価できるのはこのNeXTStep部分に限られる。それ以外はOSがMachそのままで，ファイル・ネットワークもNFSを流用している。また，NeXTStep以外の開発ツールはGNUベースである，というソフトウエア環境は，現存するソフトウエア・ツールで特に定評のあるものを，できるかぎり安く（大学製のフリー・ソフトウエアを利用して）寄せ集めたという感をぬぐえない。

とは言っても，米国防総省（DOD）が，標準OSにしようとしているCMU製のMachとGNUツールの組み合わせは，現時点で考えられるUNIXベースの開発環境として最高のものである。またアプリケーション・プログラムとして種々の高価なプログラムがバンドルされている。そしてオブジェクト指向によるインタフェース設計の自動化というテーマを具現化したNeXTStepの存在は，それだけでNeXTという寄せ集めマシンを1990年代のオズボーン1には絶対にしないだけの魅力を提供している。

ちなみに，Machはバークレイ版のUNIX 4.3 BSDとバイナリ・レベルで完全に上位コンパチである。またマルチプロセサによるパラレル・プログラミングのサポート，ネットワーク・トラスペアレンシなど，元のUNIXにない重要な機能も付加されている。

著者は，Mach上でMIT（マサチューセッツ工科大学）製の並列LispであるMultilispを利用している。このとき，プログラム開発をシングル・プロセサのIBM RT/PCやSun-3上で行い，出来上がったところでアンコール社のマルチプロセサ・マシン，Multimaxの上で走らせるといった技も可能である。

これは，シングル・プロセサの場合でもMachがOSレベルで，パラレリズムを提供しているからで，マルチプロセサの場合とまるで同じ環境でプログラミング可能だからである。

また，ネットワーク・トランスペアレンシのサポートにより，
「/‥/ノード名/usr/ユーザーID/ディレクトリ名/」
というパスで直接ネットワーク上の任意のマシンのファイルに自分のファイルと同じようにアクセスすることが可能である。

この他にもいろいろMachは4.3 BSDやSystem Vにはない機能を持っている。もともとUNIXが，ハッカー好みの比較的複雑なアプリケーション環境を提供しているうえに種々のMachの機能が付加されているため，アプリケーション開発の環境が特に複雑化している。

一方NeXTは，ユーザーとしてUNIXをこれまで使ったことがない層を特に対象としている。また，特に開発環境が複雑化しやすいグラフィックやサウンドを多用したアプリケーションのプログラミングを前提としてい

る。

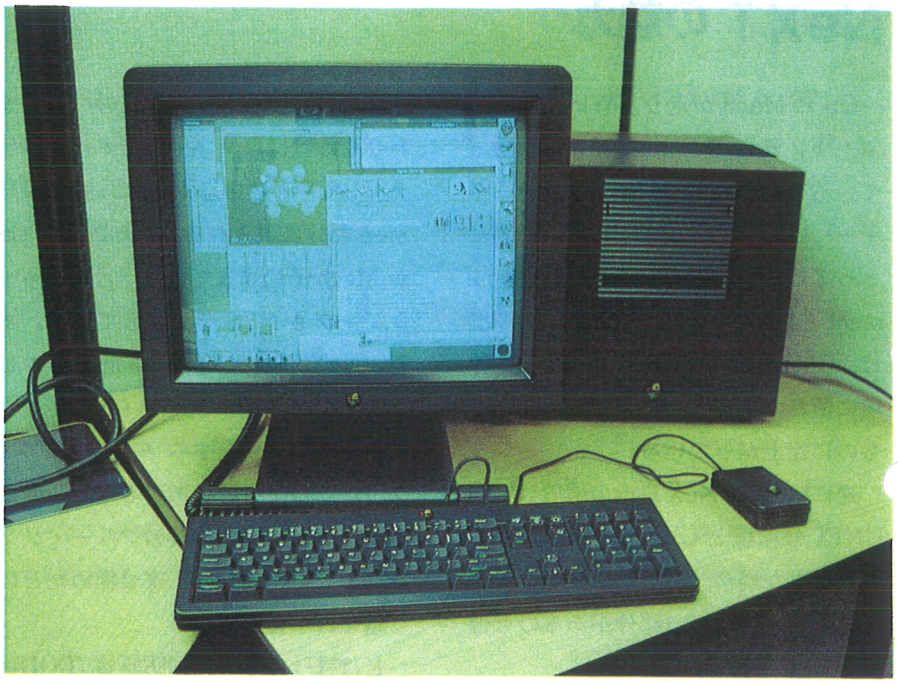
これは，高校や大学などの教育関係の市場を当初のターゲットにしているから，ハイパーメディアによる教育用ソフトの開発を容易にすることが1つの重要な課題だったのである。このためNeXTStepはユーザー・インタフェースとグラフィック・サポート環境を簡素化することに主眼をおいている。

ただ，これまでプロのグラフィック・インタフェース・ハッカー達がCMUにおいて行ってきたことが，誰にでも(そのうえオブジェクト・ベースのプログラミングで)，できるようになるシステムの構築は容易なものではない。実際，今年5月の段階でCMUの各研究所でサイト・テストされているバージョンでは特にウインドウまわりに多くのバグが見受けられた。

また，CMUに置かれているこれらのNeXT発売前モデルに関して言えば，システム全体としてもかなり不安定で，本格的なプログラミングのもとではかなりの頻度でマシンのクラッシュが見られた。

Machそのものは，すでに事実上すべてのVAXと，IBM RT/PC，Sun-3，Multimaxといったマシンのうえで安定して走っている。しかし数少ないNeXTのアプリケーション・ソフトウエアの安定性にはまだ不安が感じられる。89年夏の一般向け発売というスケジュールは，アップルなどへのけん制のための早期発表だろう。

写真1● The NeXT Computer System



る画面）とワークスペース上の操作のための道具（キーボードとマウス）は，ユーザーの管理下に属するものである。ユーザーが，常に好みのアプリケーションやウインドウを選べるものでなければならないし，また，ワークスペース上のウインドウを自由に思うようにアレンジできなければならない。また，2つのアプリケーションを利用しているときに，ユーザーの行える操作をむやみに限定することは望ましくない。

インタフェースの自然さ

グラフィカル・インタフェースの大きな利点は，ユーザーが違和感なく扱えることである。スクリーンが，現実世界の出来事を抽象化したひとつの仮想世界となり，それぞれのオブジェクトを，現実世界で慣れ親しんでいるものと同様な方法で操作することができる。このように，グラフィカル・インタフェースは，直感的な操作を可能にするものである。

マウスの利用

ユーザー・インタフェースにかかわるすべての情報は，スクリーン上にグラフィックなオブジェクトとして表され，すべてのスクリーン上のオブジェクトは主としてマウスでコントロールされる。

キーボードはあくまで文字入力のためのものであり，グラフィカル・インタフェースには，マウスがよりふさわしい。ただし，熟練者による入力の場合にはキーボード入力がより効率的なこともあるため，キーボード使用のオプションもサポートしている。ただし主入力はあくまでマウスであり，マウス入力があって，キーボード入力がない場合はありうるが，その逆はない。

写真 2●カーネギー・メロン大学機械翻訳センターで使用している NeXT

ユーザー・インタフェースと構築ツールを統合した NeXTStep

ネクスト社の言葉によれば，NeXTStep の開発の動機は，「UNIX ベースのコンピュータは複雑さのため，科学者や技術者以外の人々の手の届かないものであった。また，UNIX アプリケーション開発に対するこれまでのアプローチの仕方ではグラフィックを使ったエンドユーザー向けアプリケーションの開発に法外な時間（inordinate amount of time）がかかってしまった」ことだという。

NeXT はこれに対し次のようなアプローチをとっている。つまり，ソフトウエア開発者にたいしては，オブジェクト指向型のソフトウエア開発環境を提供することによりアプリケーション・プログラムにおけるユーザー・インタフェース部の構築を容易にする。エンドユーザーにたいしては，イベント駆動ベースの使用環境を提供することにより，UNIX コマンドを使うことなくコンピュータの利用を可能にする。NeXTStep はこういったユーザー・インタフェースに対する明確な哲学に基づきトップダウンで設計されている。

またアプリケーション・ソフトウエア・デベロッパも現在のところ登録制で厳しい審査基準がある。登録デベロッパには NeXTStep の種々のユーザー・インタフェースにかかわるルールの遵守が強く求められており，インタフェースの均一化が図られている。

NeXTStep は，4 つの部分から構成されている。これらは，ウインドウ・サーバー，ワークスペース・マネジャ，インタフェース・ビルダー，アプリケーション・キットと呼ばれている（図 1）。

独自開発のウインドウ・サーバー

NeXTStep のウインドウ・サーバーはネクスト社が独自に開発したもので，個々のスクリーン上のウインドウの管理とマウスやキーボードにおけるイベントをアプリケーション・プログラムに送る役割を果たす。クライアント/サーバーのアーキテクチャがとられており，Mach 上の分散処理環境のもと，個々のアプリケーション・プログラムがクライアントとして並列的に，ウインドウ・サーバーのサービスを受ける形となっている。

図 1●NeXTStepのシステム構成（日経コンピュータ89年 5 月22日号より）

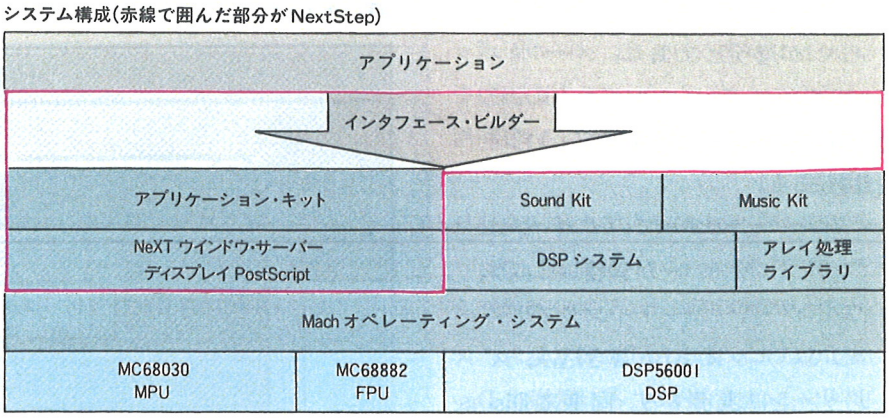


図2●スクリーン右端がドック（Dock）。ここに各アプリケーションに対応するドックド・アイコンを配置する（NeXT Technical Documentation Release 0.8より抜粋。以下の図も同じ）

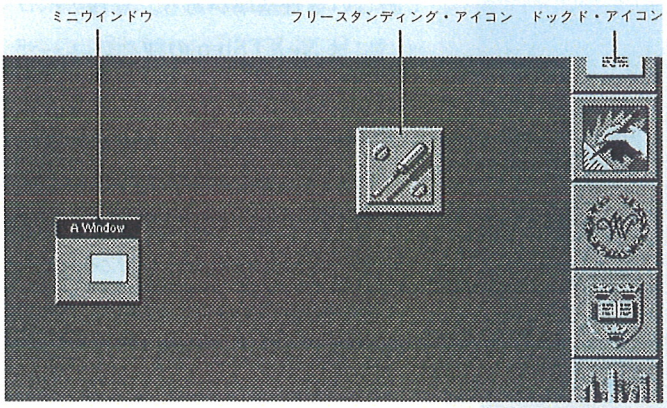
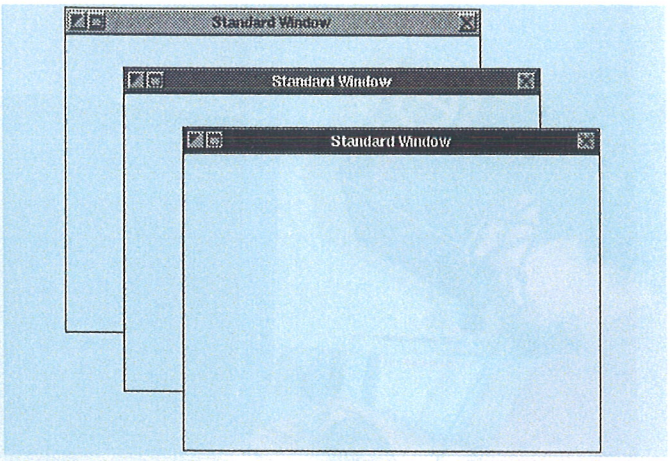


図3●スタンダード・ウインドウ



またMachのインタープロセス・コミュニケーション（IPC）とTCP/IP（Transmission Control Protocol / Internet Protocol）といった通信プロトコルもサポートしており，ネットワークを通じて他のMachマシンへのアクセスが可能である。例えばネットワークを介して他のNeXTのワークスペースにアクセスできる。

現在カーネギー・メロン大学ではMachマシンのネットワークでVAX上のプロセスの出力をRT/PCのXWindow上に表示する使い方をよくしている。これは複数のマシンを同時に利用するときに便利な使い方である。例えば，電子メールを受けるアカウントのあるマシンとプログラム開発をするマシンが違うときは，メール・ボックスのモニター・プログラムの出力を開発用マシンの画面に出すという利用法もある。

スクリーン上のすべてのイメージは，ウインドウ・サーバーが管理しており，ウインドウ・サーバー・インタプリタを介してビデオRAMに書き込み，スクリーン上に表示する。同様にDisplay Postscript対応のNeXT 400 dpi（ドット/インチ）レーザー・プリンタにもスクリーン上のイメージと同一のイメージを出力できる。

スクリーン表示したいアプリケーション・プログラムは，ウインドウ・サーバーに対して，PostScript言語によるメッセージを送ることによって表示する。

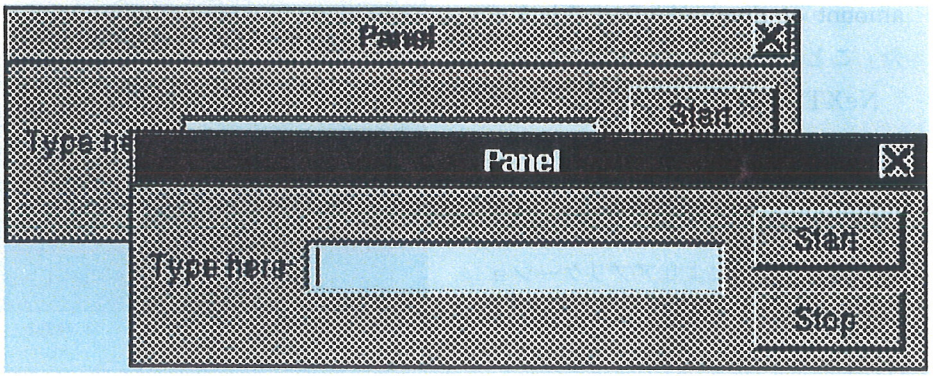
通常これらのPostScript言語によるマウスやカーソルにかかわるメッセージは，アプリケーション・キットが用意するため，プログラマが直接書く必要はない。また，アプリケーション・プログラマが直接PostScript言語を書くことも可能である。

画面上の出力とプリンタの出力が一致

Display PostScriptインタプリタそのものはNeXT社の開発ではなく，Adobe Systems社の開発による。もともとデバイス・インディペンデントなプリンタ用記述言語として開発されたもので，MacintoshのLaser Writer IIでも採用されている。NeXTはこれをディスプレイにも利用しており，ディスプレイ上の表示とレーザー・プリンタからの出力が完全に同一のものになる。

アプリケーション・プログラムからのスクリーンへの表示がすべてPostScript言語を通じて行われるため，ス

図4●パネル・ウインドウ



クリーンへの出力は，マシン/OSに対しインディペンデントになる。ただ問題は表示速度である。NeXTは「ASCII PostScriptコマンド」などPostScriptが備える高速化の機能やネクスト社独自の工夫を取り入れているという。それでも大学向けサイト・テスト版(リリース0.8)では，表示の遅さが気になった。

ワークスペースとファイル・アクセスを管理するワークスペース・マネジャ

Macintoshのデスクトップにあたるのが，NeXTのワークスペース(186ページの写真3)である。スクリーン全体をMacintoshでは机に見立てていた。NeXTではこれを「ワークスペース(作業場)」と呼んでいる。ワークスペース・マネジャの仕事は画面上のワークスペースの管理やファイル，ディレクトリへのユーザーのアクセスを管理することである。したがって，MacintoshでいえばMultiFinderにあたる。

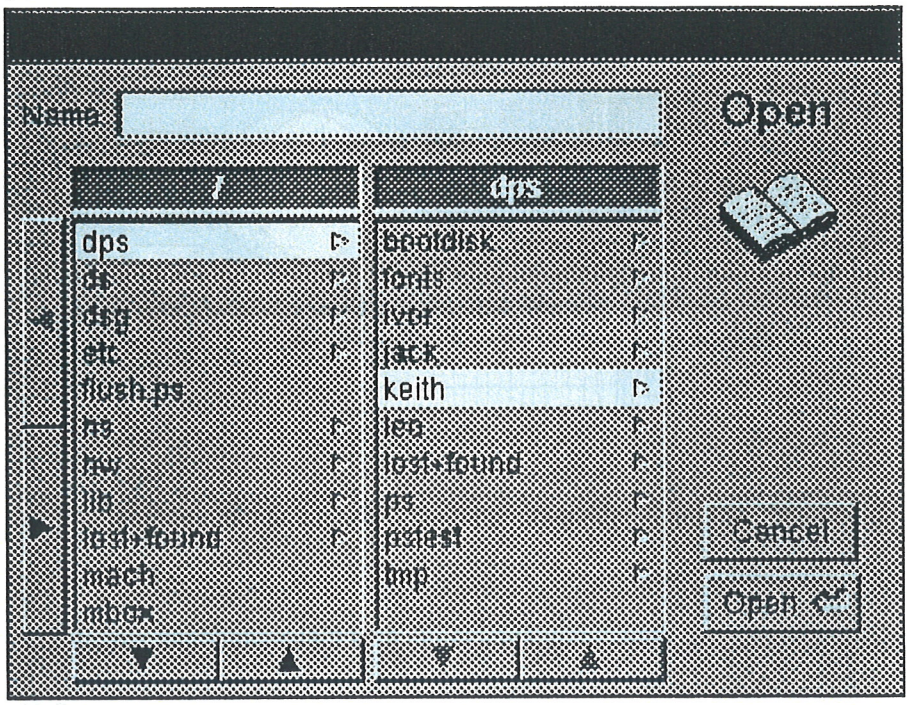
ファイルの検索や管理，ディレクトリ自身の表示，アプリケーションやユーティリティの実行など，UNIXファイル・システムとMachへのユーザーのアクセスすべてはワークスペース・マネジャを通じて行う。このため，ユーザーにUNIXの知識は必要ない。

また，必要であれば，UNIXシェルを開くこともできる。これは，VT-100エミュレートのウインドウを開きUNIXコマンド・インタプリタを走らせることによって可能である。ちなみにMachはC ShellとB Shellの両方をサポートしている。

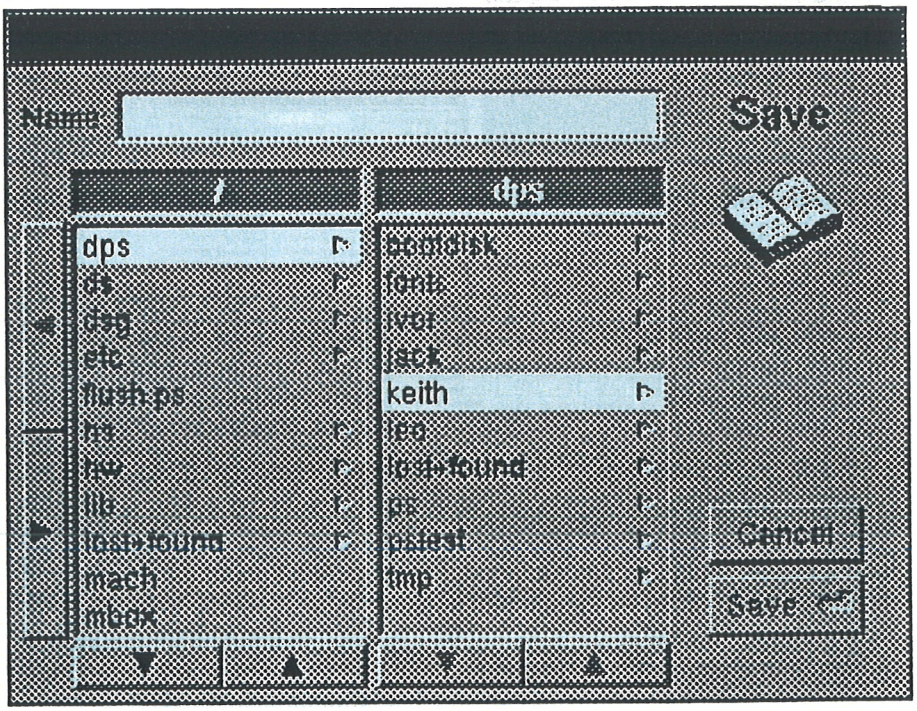
ファイルへのアクセスはMacintosh同様ディレクトリをマウスでクリックすることにより行える。ファイルの一覧を表示するブラウザ・ウインドウでディレクトリのヒエラルキの中身を1レベルずつ下げながらの

図5●パネル・ウインドウを用いた状態

(a)



(b)



ぞいていくこともできる（写真4）。

ユーザーによって異なった設定ができるマウス

NeXTのマウスは2ボタンで，通常は左右とも同じセットアップがされているが，Preferencesプログラムで異なったセットアップにすることもできる。

写真3● NeXTのワークスペース（米 *BYTE* 誌誌 88年11月号より）

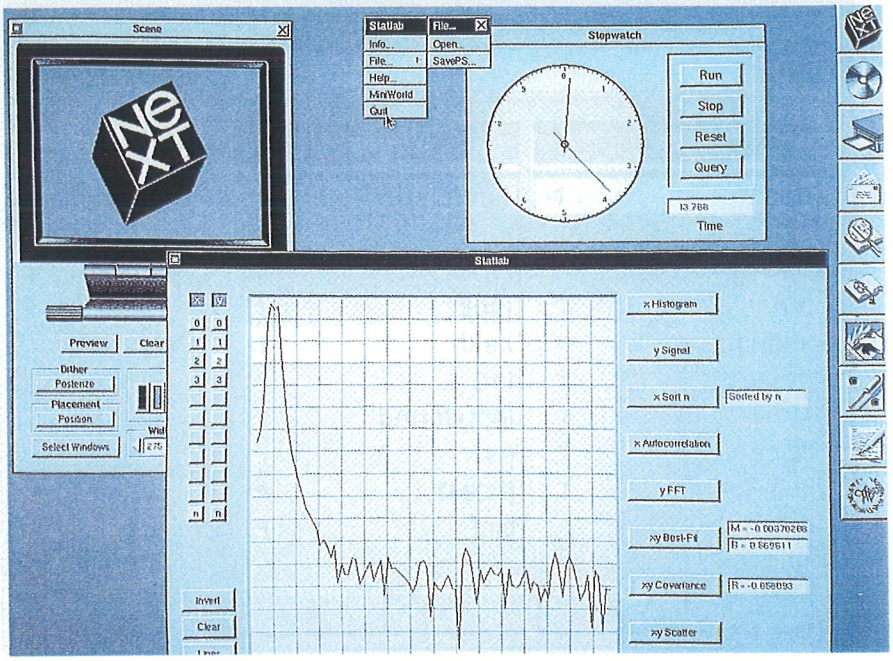
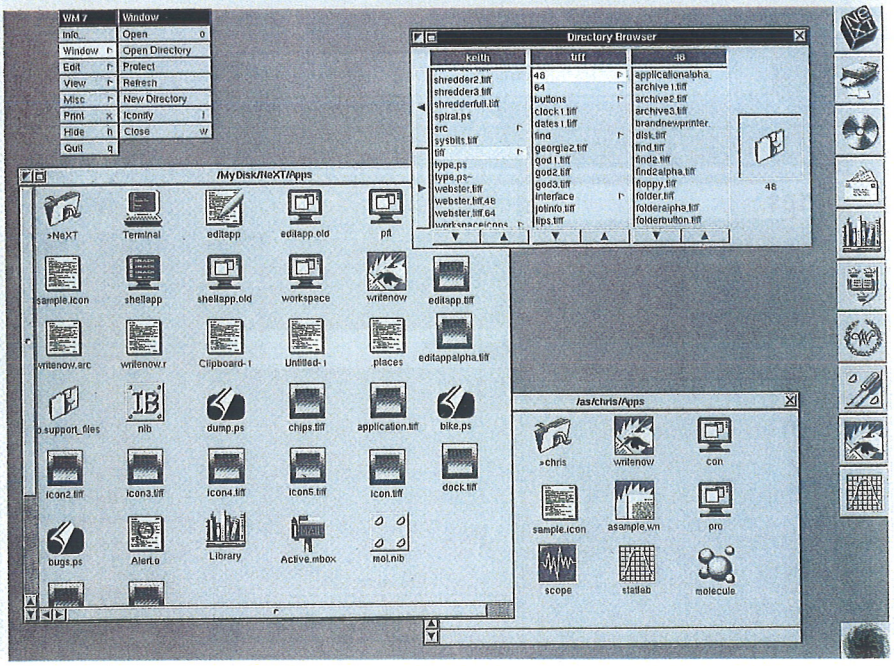


写真4●ワークスペース・マネジャの右上がディレクトリ・ブラウザで，ファイル一覧を表示している（『日経バイト』89年6月号より）



Preferencesプログラムは，Macintoshのコントロール・パネルにあたるもので，マウスとクリック間のインターバルや，キーボードのリピート・レートなどをユーザーが設定するためのもの。ただし，NeXTのリリース0.8の段階ではまだインストールされていない。

マウスのクリックは基本的にはMacintoshと同様であるが，3回クリックもある。ネクスト社は，論理的に，シングル・クリックの延長線上にあるイベントにのみダブル・クリックを使うようにとデベロッパ向けキットのなかで明記している。例えば，アイコンをシングル・クリックしてアプリケーションを選び，ダブル・クリックでオープンするなどである。

また，トリプル・クリックはダブル・クリックの延長線上にあるイベントにだけ使って構わないが，余り使わないことが望ましいとしている。4回クリック以上もサポートされているが使用はすすめられていない。

ドラッギングやスライディングなどもMacintoshや他のウインドウ・ベースのシステム同様，マウスによる主要な入力方法としてサポートされている。

アプリケーションのアイコンを配置するドック

スクリーンの右端はドック（Dock）といい，良く使うアプリケーションのアイコンを12まで置いておける（184ページの図2）。ディレクトリ・ウインドウから好みのアプリケーションをdockへドラッグするだけで，置いておくアプリケーションの種類をカスタマイズできる。ログアウトした後も，ド

ックの内容は記録しておける。

アプリケーション・プログラムの実行はMacintosh同様，ディレクトリ・ウインドウのアイコンをダブル・クリックすることで行う。実行中のアプリケーションは，すでにドック内にアイコンが登録されていなければ，ドック外にフリー・アイコンを表示する。このアイコンは，ドックにドラッグしなければ，アプリケーションが終了するときに消える。

ウインドウのタイプ

NeXTStepには，数種類のウインドウがある。それぞれ見た目が異なり，ユーザーの行為に対するレスポンスや，アプリケーション・プログラム中での機能も異なる。またウインドウのタイプには種々のサブタイプがある。

通常のアプリケーションが一般に利用するのがスタンダード・ウインドウである（184ページの図3）。これが普通のウインドウといわれているものに対応する。例えば，ファイルのエディットやゲームの表示画面などほとんどすべてのアプリケーションの標準ウインドウである。また，ほとんどのアプリケーションがスタンダード・ウインドウ以外に以下のウインドウを利用する。

パネル

他のウインドウのコントロールや，アプリケーションに命令を送るために使われる（185ページの図4，図5）。また，ユーザーへのヘルプ情報などにも利用される。Macintoshのダイアログ・ボックスに相当する。

図6●メニュー・ウインドウ

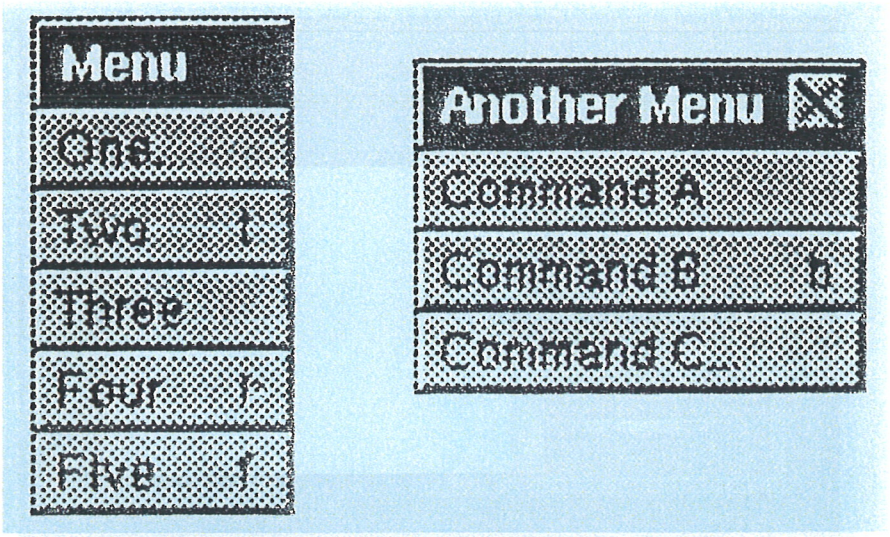
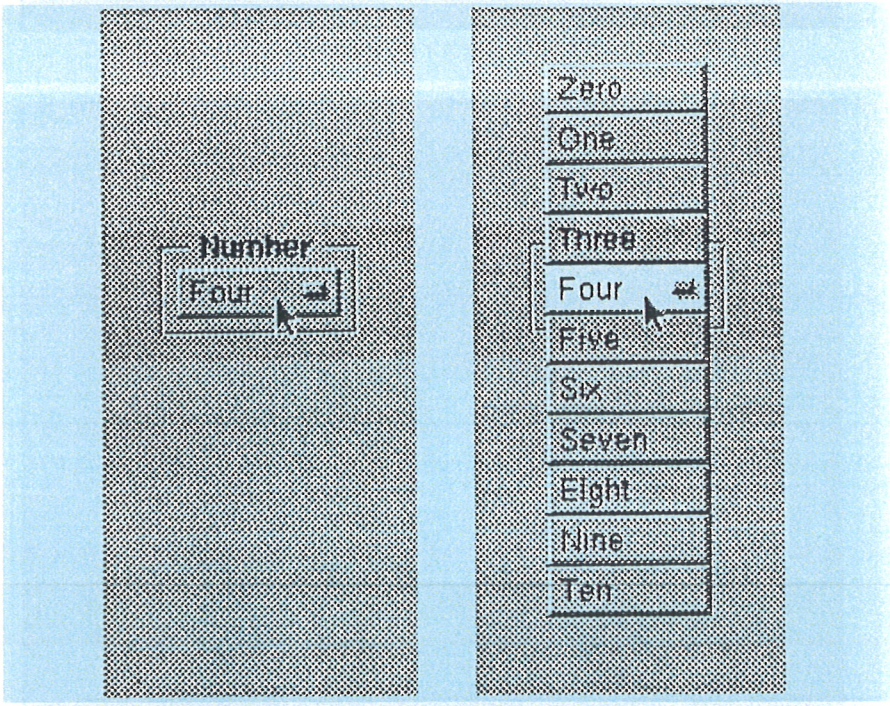


図7●ポップアップ・リスト



メニュー

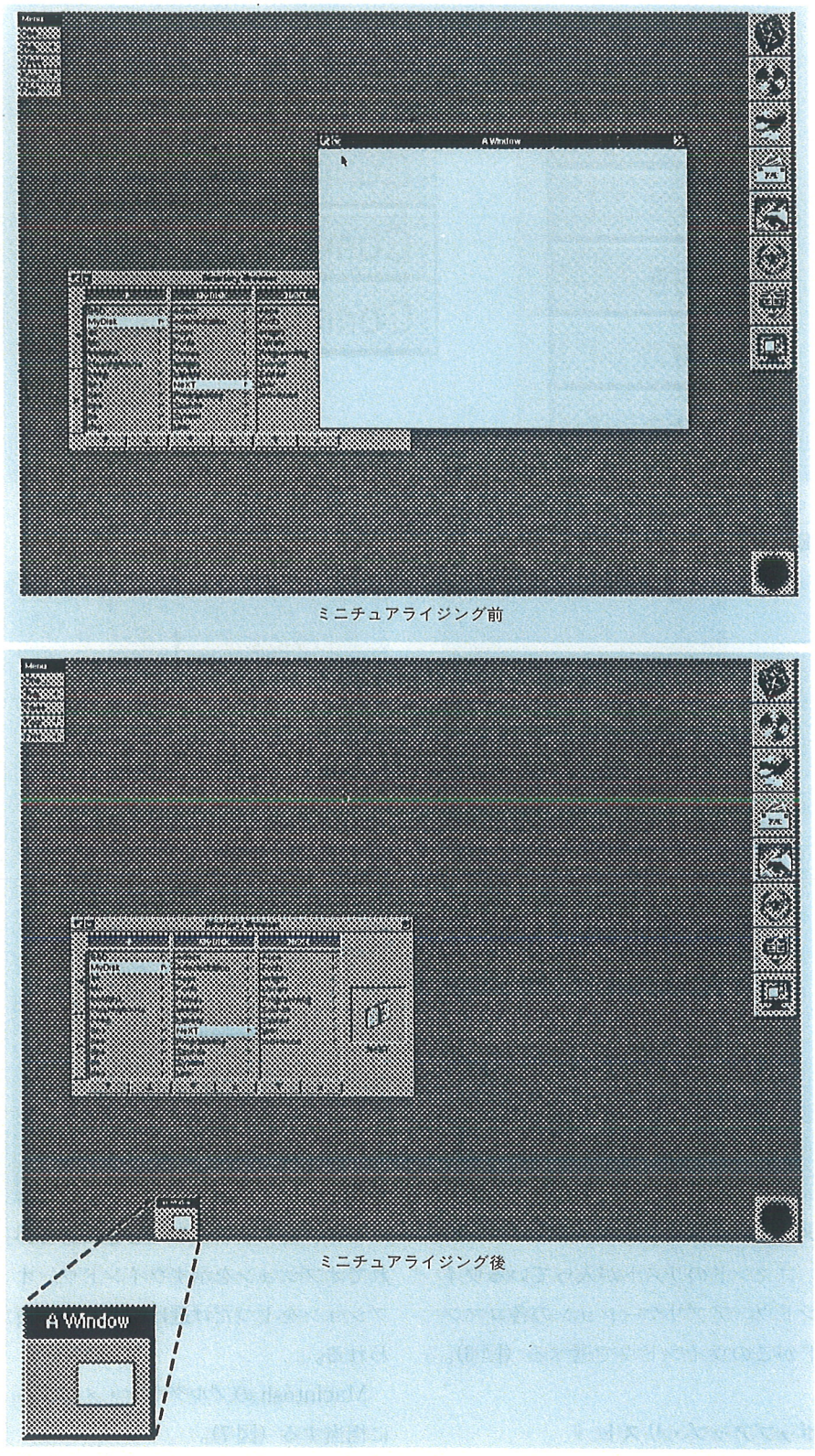
コマンドのリストが入っているウインドウ。アプリケーションの各コマンドがこのウインドウで選べる（図6）。

ポップアップ・リスト

ボタンをクリックしたときにあらわれてオプションを示すウインドウ。オプションを1つだけ選ばせるときに使われる。

Macintoshのプルダウン・メニューに相当する（図7）。

図 8 ●ミニチュアライジング。上の画面のウインドウを下の画面でミニウインドウに変えている



ミニチュアライジング前

ミニチュアライジング後

ミニウインドウ

ミニチュアライジング・ボタンをクリックすることによってウインドウをアイコン・サイズにミニチュア化することができる（図 8）。これによって，ワークスペース上の，スペースをセーブすることができる。X Window や Appolo Domain のディスプレイ・マネジャでシェル（ウインドウ）をアイコン化するのと同様である。

フリースタンディング・アイコンとドックド・アイコン

ワークスペース・マネジャが実行中のアプリケーションに割り当てたアイコンがフリースタンディング・アイコンで，ドック中にドラッグされたアプリケーションに対応するのがドックド・アイコンである（184 ページの図 2）。これらのアイコンはそれぞれ，ひとつのアプリケーションを表す。

ミニウインドウはひとつのウインドウを表すため，ひとつのアプリケーションに複数のミニウインドウがありうる。だが，フリースタンディング・アイコン，またはドックド・アイコンは 1 つしかありえない。

いってみれば，すべてのフリースタンディング・アイコンとドックド・アイコンはワークスペース・マネジャの直接管理のもとにある。

パネルとメニューのサブタイプ

また，パネルとメニューは，それぞれ以下の特殊なサブタイプを持つ。

アテンション・パネル

標準ウインドウで実行中のアプリケーションを続行させるために必要な入

力を促すためのウインドウ。例えばアプリケーションの終了前にファイルをセーブするかなどの質問はこのウインドウでおこなう。

メイン・メニュー

アプリケーションの実行中にコマンドを常に表示しているメニュー。タイトル・バーにアプリケーション名が表示される。サブ・メニューがここから選べる（図6）。

メイン・メニューとサブメニューにはメイン・メニューを頂点とするクラス・インヘリタンス（継承）関係がある。つまり，サブメニューは他のサブメニューのスーパーメニューになりうるのである。スクリーン上ではサブメニューをそのスーパーメニューからドラッギングにより，切り離すこともできる。

上記のウインドウのほかにスクリーン全体のバックグラウンドであるワークスペース・ウインドウがある。

NeXTStepにおける開発環境

インタフェース・ビルダーはアプリケーション・プログラムのユーザー・インタフェースをグラフィカルに構築するためのツールである。これによって開発中のアプリケーションのインタフェースを表示しているウインドウを開きながら，インタフェース・オブジェクトであるボタンやメニューを別のウインドウに表示，アクセスしながら開発できる。

インタフェースの設計はボタンやメニューといったインタフェース・オブジェクトを，アプリケーションのインタフェースを表示しているウインドウの望みの位置にドラッグするだけでできる。サウンドの追加やテストもその場でできる。

マウスやキーボードでのイベントは，すべてのインタフェース・オブジェクトが能動的に理解するため，ディスプレイ上でのアクションを自動的にとる。例えばボタンはクリックされたときみずから陰影を反転する。

インタフェース・ビルダーは個々のインタフェース・オブジェクトを連動させるためのツールも用意している。例えばボタンとパネルの連動やスライダとテキスト・フィールドの連動などである。またユーザーによるインタフェース・オブジェクトの定義も可能である。

基本的なインタフェース用のコードはすべてインタフェース・ビルダーが提供しているため通常はインタフェース部のプログラミングは不要である。ただし，Objective-Cによるプログラミングのサポート環境も提供している。支援環境には例えば，UNIXのmakefileスクリプトの自動生成や，基本的なソース・コードの生成などの機能がある。

インタフェース・ビルダーが利用するインタフェース・オブジェクトはアプリケーション・キットとして用意されている。アプリケーション・キットはウインドウ，スクローラ，ボタンといったインタフェース・ビルダーで使用するインタフェース・オブジェクトを含む約40のソフトウエア・クラスを定義している。

これらのオブジェクトはオブジェクト指向言語であるObjective-Cのメッセージ・パッシング（オブジェクトに対する動作の指示）によって種々のアプリケーション環境を実現する。アプリケーション・プログラムにおいては必要なソフトウエアのクラス・インヘリタンスの関係に好みのオブジェクトをおくことによって，インタフェース・ビルダーで利用できるオブジェクトをつくることができる。インヘリタンスがサポートさているため，上位クラスで定義された情報はすべて下位のオブジェクトに受け継がれる。また下位クラスのカスタマイズも当然可能である。

インターフェース・ビルダーの実際の使用法

ここで実際にインタフェース・ビルダーの使用法を見てみたい。インタフェース・ビルダーのアウトプットは以下の通りとなる。

＊main（）関数を含むファイル。
＊アプリケーションのインタフェース・オブジェクトのアーカイブ・ファイル。
＊カスタム・オブジェクトのテンプレート・ファイル。
＊コンパイルとリンク用のmakefile。

インタフェース・ビルダーを立ちあげると，メイン・メニューと，プロジェクト・ウインドウ，それに3つの基本ユーザー・インタフェース・オブジェクトのパネルが現われる。インタフェース・ビルダーの各コマンドはメイン・メニューから選ぶことができる。プロジェクト・ウインドウには，ユーザー・インタフェースを構築するため

のビルド・アンド・テスト・モードのコントロール用スイッチ，作業中のプロジェクトやオブジェクト名を表示するブラウザ，それに個々のオブジェクトのアイコンを表示できるボックスが入っている。

立ちあげ時のプロジェクト・ウインドウのタイトルはインタフェース・ビルダーであるが，プロジェクト名の指定により，タイトル名はプロジェクト名に変わる。3つのパネルは，それぞれビュー，ウインドウズ，メニューとタイトルが付けられており，作製中のプログラム用のアプリケーション・キット・オブジェクトのイメージが入る。

インタフェース・ビルダーはそれぞれのアプリケーションをプロジェクトと呼ぶ。各プロジェクトは，インタフェース・ファイル，いくつかのプログラム・ファイル，メイク・ファイルよりなる。インタフェース・ファイルは，インタフェース・ビルダー中でセレクトする各オブジェクトのスペックを記述する。プログラム・ファイルはインタフェース・ビルダーが生成するものとプログラマが記述するものがある。メイク・ファイルはコンパイラとリンカー用のUNIX makeユーティリティ向けファイルである。

Newコマンドによるプロジェクトの生成

プロジェクトの生成は，インタフェース・ビルダーのプロジェクト・メニューのコマンド「New」を選ぶことによって行う。メニューからNewを選ぶとメニューとスタンダード・ウインドウ，アプリケーション・アイコンが現われる。このアプリケーション・アイコンがアプリケーションのルート・オブジェクトを表す。ブラウザにより，インタフェース・オブジェクトをリストすると，ファースト・レスポンダと呼ばれるものが登場する。これが，キーボード，マウスといったところのイベントを最初に受けるオブジェクトである。

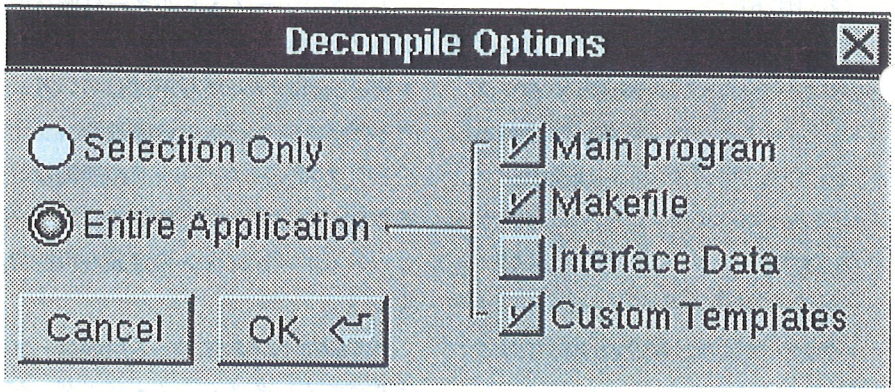
他に，メイン・メニュー，マイ・ウインドウ，インフォ・パネルといったオブジェクトもブラウザにリストされる。これらは，メイン・メニューがプロジェクトの最初のメニューのインスタンス，マイ・ウインドウがウインドウのインスタンス，インフォ・パネルがインフォ・パネルのインスタンスにそれぞれ相当する。

ここで例えば，ブラウザの中のインフォ・パネルの名称をダブル・クリックすると，図9の一般的なインフォ・パネルが現われる。これらのパネルやウインドウは，通常ディスプレイ・マネジャ上で行うように，サイズや名前を自由に変更することができる。また，インスペクタと呼ばれるプログラムでオブジェクトの種々のインスタンス変数値を変更することもできる。

また，ビュー・パネルから，種々のコントロール・オブジェクト（アプリケーション・キット・オブジェクト）をドラッグすれば，新たなインタフェース・オブジェクトを加えられる。こうして作られたインタフェースは，シミュレーション・モードのもとでテストすることもできる。テスト後にディコンパイル・オプションを選べば（図

図9●インフォ・パネル

図10●ディコンパイル・オプションを指示するためのパネル



10)，メイク・ファイルなど上記の各ファイルが自動生成される。

各インタフェース・オブジェクトの定義後，アプリケーション・プログラムのクラスをエディットし(図11)，ツールズ・メニューのコネクト・コマンドを選びコネクト・ウインドウを開くことにより，グラフィカルに各オブジェクトをつなぎ合わせられる（図12)。

コネクト・ウインドウでは結合されるオブジェクトを置くセンダーとターゲットのそれぞれのスロットがあり，各オブジェクトをドラッグすることによって，メッセージ・パッシングの関係が定義できる。また，ターゲット・オブジェクトを別のオブジェクトに結合していくことにより，種々のメッセージ・パッシング経路を構築できる。

これらは，コネクト・ウインドウ内ですべてグラフィカルに行える。

◇　　◇　　◇

NeXT コンピュータは，ユーザー・インタフェース構築の簡易化のために，徹底したオブジェクト指向とマウス・ベースの環境を採用している。これにより，グラフィカルなインタフェースの設計が，特に容易になっている。現在の段階ではどの程度，一般のデベロッパによる生産性の向上が生み出されるか，不確定な部分が多い。しかし今後のユーザー・インタフェース構築手法の流れに，大きな影響を与えていくことは間違いない。

NeXT コンピュータそのものは，リリース 0.8 の段階では，バグも多く，また光磁気ディスク駆動装置の遅さなど，いまだ完成したマシンではない。判断は，一般向けバージョンが登場してから下したい。

ただ，ライバルの Macintosh の過去の蓄積はそう簡単に越えられるものではないことも明らかである。特に，日本語化に関しては，ようやく

図 11 ●クラス・エディタのパネル

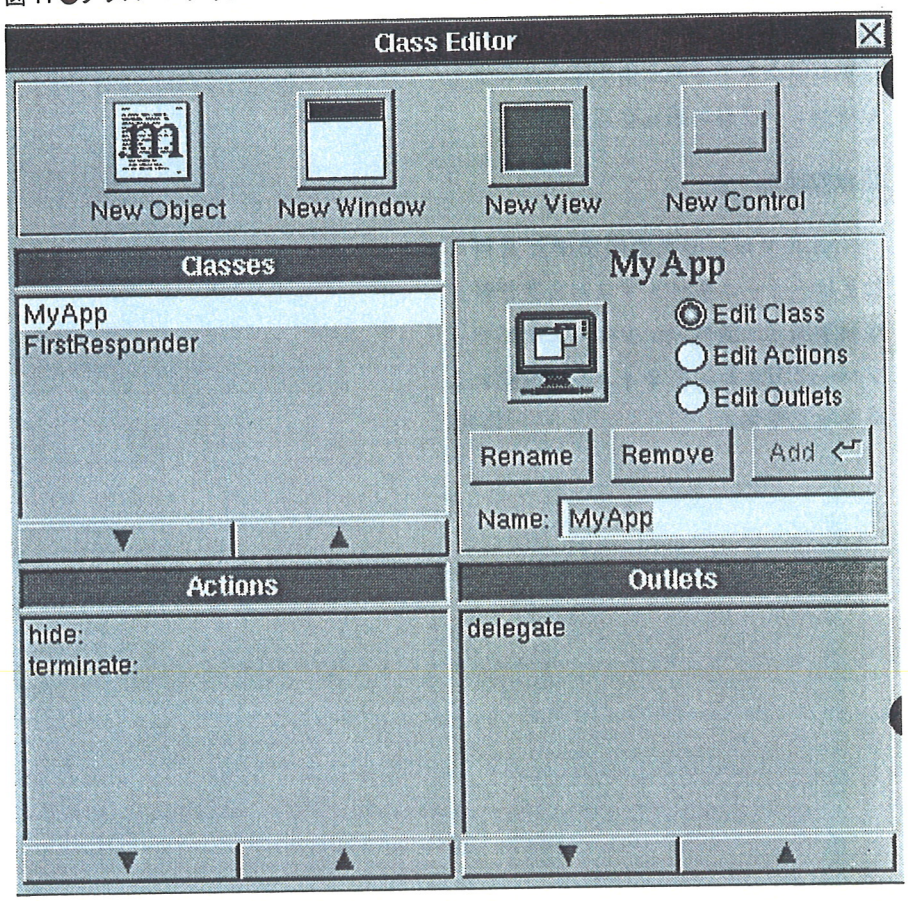
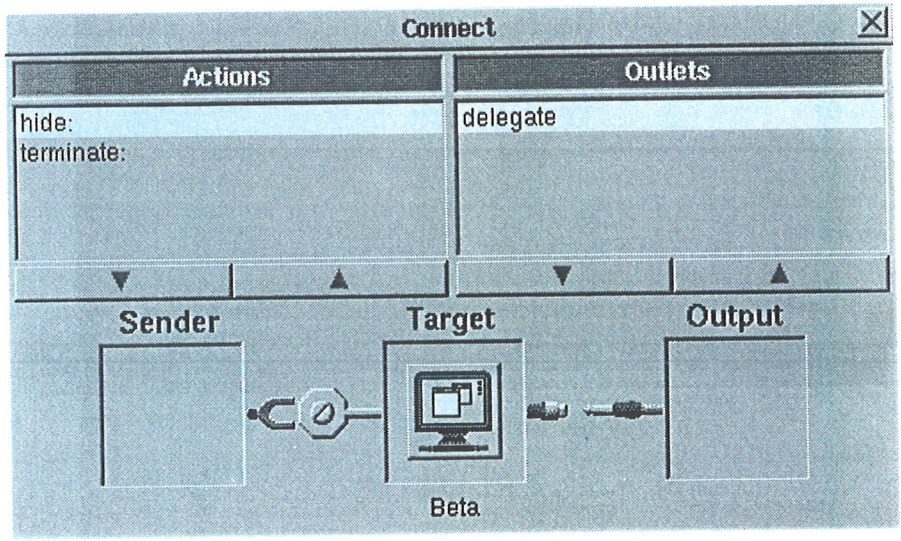


図 12 ●コネクト・ウインドウ



Macintoshが何とかなってきたところで，NeXTの日本語化には少し時間がかかりそうである。今後，Macintosh IIの最新機種，CXとまもなく発表予定の最新OS，システム7.0の組み合わせというライバルを相手に，どこまで戦っていけるか楽しみである。

感謝の辞

NeXTマシンの実際の使用と資料などは，カーネギー・メロン大学計算機科学部（米国ペンシルバニア州ピッツバーグ市）と，キヤノンの好意によるものである。 NC