高機能ワークステーションの概念と機能

―― クリエイティブワークステーション２０５０への適用 ――

こじま とみひこ　あきた ひでひこ　むらた ふみや
小島富彦[1]　秋田英彦[1]　村田文也[2]
（株）日立製作所　システム開発研究所[1]　神奈川工場[2]

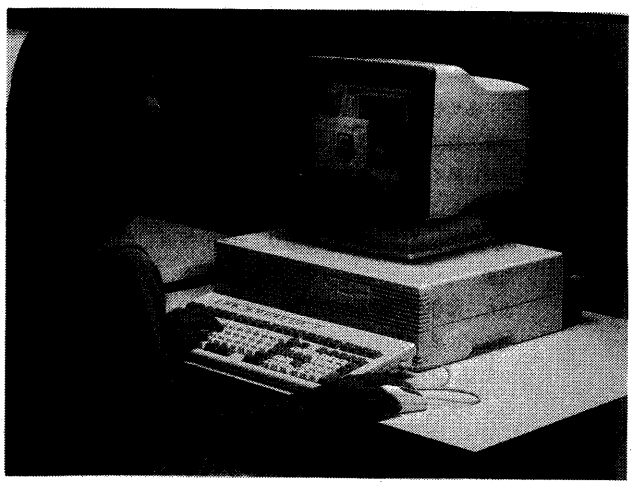
１．緒言

マルチウインドウ、マルチメディア文書処理、統合化操作環境、通信ネットワーク機能などに力点を置いた高機能ワークステーション(図１)の技術を開発した。その中核となる機能は、オペレーティングシステムの内部あるいは、その周りにビルディング・ブロック状に構成され、これらの基本機能を使用することによって高度なマンマシン・インタフェースや、アプリケーションの統合化を実現できる。本稿では、高機能ワークステーションの概念と機能について、ウインドウシステムと対話制御システムを中心に論じ、その上に構築される文書処理環境、仮想デスク環境、およびOAアプリケーションの統合化方式について報告する。

２．ソフトウェア階層

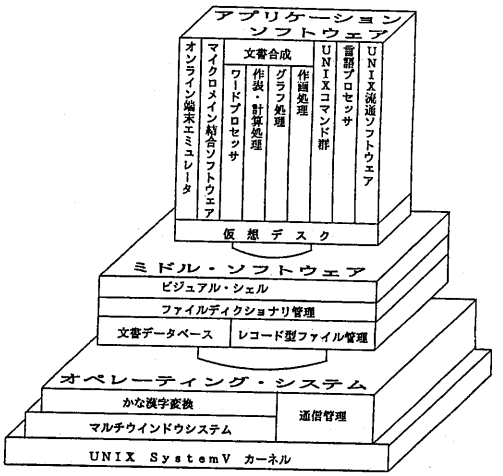
ソフトウェア全体の構成を図２に示す。オペレーティング・システム(OS)としてUNIX*を拡張し、新たにマルチウインドウ機能、かな漢字変換機能、通信機能を組み込んだオペレーティング・システム(HI－UX)を開発した。高性能化のために、独自の多重仮想空間制御、プロセス優先度制御、ダイナミックリンク制御などの新方式が採用されている。UNIX System VとのOSインタフェースの上位互換性を保持している。

オペレーティングシステムとアプリケーション(AP)層との中間に位置する基本ソフトウェアとして、文書データベース、ビジュアル・シェル、レコード型ファイル管理、ファイル・ディクショナリ管理を設けた。OSとAPの中間に位置することから、これらのプログラム群を総称して、ミドル・ソフトウェアと呼ぶことにする。ミドル・ソフトウェアは、単独のユーザ・プロセスとして、あるいはAPからダイナミック・リンクまたはスタティック・リンクで呼び出されてAPプロセスとして動く。



図－１　高機能ワークステーション２０５０の外観

アプリケーション層として、各種のOAソフトウェアや、オンライン端末エミュレータ、マイクロ・メインフレーム結合ソフトウエア、UNIXコマンド群などが同時にウインドウで動作できる。これらのAP群の起動制御をビジュアル・シェルが行なう。その際、あたかもオフィスで事務用品を扱うときのような感覚でオペレータがプログラムを起動できるように仮想デスク環境と呼ばれる方式（プログラム、アイコン、メソッド定義などから構成される環境）を開発した。



図－２　ソフトウェア・アーキテクチャの概念図

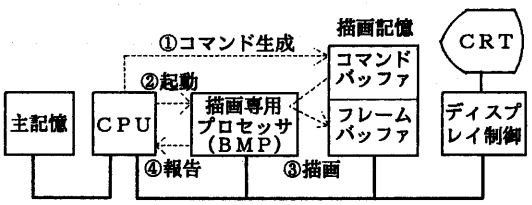
３．マルチウインドウ・システム

マルチウインドウ・システムとは、ワークステーションの画面中に複数のアクティブなワークステーション画面を同時に表示させるもので、次のような利点がある。(1)複数の文書を見ながら作業を進められる。(2)関連する複数作業をそれぞれのウインドウで同時に実行できる。(3)進行中の作業を中断せず、新しいウインドウを開くことにより、飛び込みの作業を処理できる。(4)ウインドウ画面の間でデータの交換が可能であるので、これを用いて表や文書の切り貼りを行なえる。(5)ホスト(または他のワークステーション)の複数アプリケーションと同時接続(マルチセッション機能)をし、複数ウインドウで異なるデータベース・システム(例えば本店と支店のデータベース)と対話しながら作業できる。このような効果をもたらすマルチウインドウ機能を開発した。その基本概念と構成方式を述べる。

*UNIXは米国AT&T社ベル研究所の開発したオペレーティング・システムの名称です。

3．1　ディスプレイ描画ハードウェア機構

ビットマップ・ディスプレイの描画機構を、図3に示す。CPUが描画コマンド列をコマンド・バッファに生成して、描画専用プロセッサ(Bit Map Processor：BMPと略す)を起動すると、BMPがフレーム・バッファ上の画素データに展開する。それをディスプレイ制御が常時CRTにリフレッシュしている。コマンド列の実行が終了すると、BMPはCPUに割り込みで終了を報告する。CPUによるコマンド列の生成とBMPによる実行とは並列に進められる。コマンド列は、文字列・図形・アイコン描画、領域塗り潰し、描画属性(色・幅等)設定、クリッピング枠設定、等からなる。



図－3　ディスプレイ描画ハードウェア機構

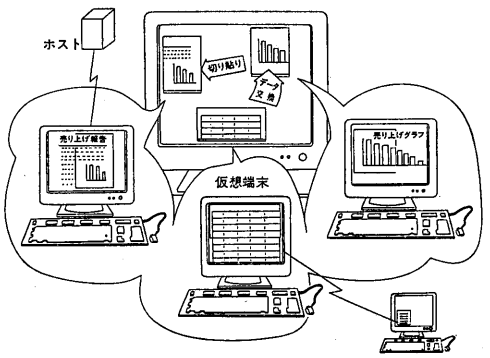
3．2　マルチウインドウの基本概念[1)]

ウインドウ・システムを、仮想的なワークステーション(仮想端末)群を作り出し管理するためのオペレーティング・システムの機能の1つとして位置付けた。今回開発したウインドウ・システムは、次のような機能的特徴を持つ。

(1)　大論理画面を持つ仮想端末

ビットマップ・プロセッサ、フレーム・メモリなど、ワークステーションのハードウェア機構を使って、ソフトウェア的に仮想端末群を作り出す(図4)。1台の仮想端末は、文字、図形、アイコン、画像などを自由に表示できるマルチメディア表示型のディスプレイと、キーボード、マウスを有する。仮想端末のディスプレイ画面(仮想画面)の寸法、色などをAPは自由に設定できる。実画面より大きく設定してもよい(大論理画面と呼んでいる)。APにとって理想的な端末に可能な限り近づけるためである。

マルチウインドウ・システムは、このような仮想端末を、1台のワークステーションの中に複数台収納する機構であるという考え方を取った。



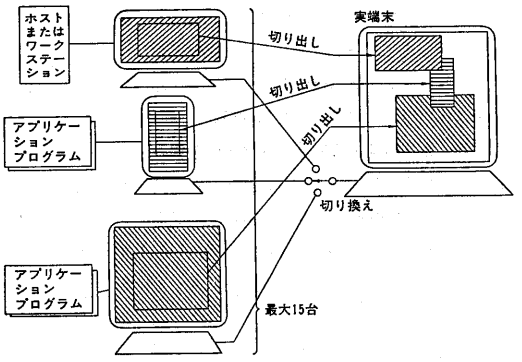
図－4　仮想端末の概念に基づくマルチウインドウ

(2)　複数プロセスのフォアグラウンド動作

複数台の仮想端末で、異なるアプリケーション・プログラム(AP)が同時に動く。また、1つのAPが同時に複数個の仮想端末を占有して動いてもよい。APはウインドウの重なり、寸法などを意識する必要はない。複数台の仮想端末のうち1台だけが対話権を持つ。ワークステーションのキーボード、マウスは、対話権を持つ仮想端末に接続される。原則として対話権の切り換えはオペレータが明示的にマウス操作で指示することにより行う。

(3)　オーバラッピング・マルチウインドウ

各仮想端末のディスプレイ上に、矩形の覗き窓を1個ずつ設けることができ、覗き窓を通して見える情報を、ウインドウ・システムが切り出してワークステーションのディスプレイ上に重ね合せて表示する(図5)。覗き窓が写像されたディスプレイ上の矩形をウインドウと呼ぶことにする。オペレータは覗き窓の位置、寸法、ウインドウ位置、上下の重なり順序などをマウス操作により自由に変更できる。



図－5　ウインドウのオーバラッピング表示

(4)　マルチメディア表示端末

仮想画面の代表的な利用方法は、紙に見せることである。文章しか書けない紙とかグラフしか書けない紙ではなく、文章(文字列)とグラフ(図形)と絵(画像)とが混在できる紙でなければならない。仮想端末をホストに接続された端末機に見せる場合でも、この機能は重要である。そこで、次のようなマルチメディア・データ表示方式を考案した。

表示の最小単位である1つの文字または1つの図形要素(折線・円・多角形・円盤・扇形・弧・アイコンなど)を「図素」と呼ぶ。図素の集団(受け皿)として「図群」を設ける。性格の異なるものを扱い易くするため、文字を受ける文字列図群と図素を受ける図形図群とに分けた。文字列図群は、文字を格子状に配置するための矩形の台紙(原稿用紙)であり、動的に生成して背景色を設定した後、行ピッチ、字ピッチ、行数、桁数を指定して形を決める。また、横書き／縦書きの区別、文字フォント・サイズ、罫線の有無などの属性を指定することができる。文字は常に上書きされる。つまり、何回書き換えても一定量の領域で状態を管理していられる。1文字毎に色を変えたり、アンダライン、アンダドット、リバースなどの属性を指定でき、行番号／桁番号を指定して任意の位置から書き続けたりすることができる。つまり、1つの文字列図群が1台のキャラクタ・ディスプレイに相当すると考えること

もできる。図形図群は共通の座標系に属する図素の集団である。つまり、図形図素は、局所的な座標を指定して描く。これらの図群を仮想平面上に、最大２５６層の深さの範囲で任意の位置に重ねて置くことができる。このような図群同士の位置関係、重ねかたの組合せで図や絵入りの複雑なマルチメディア文書の表示を実現する方式とした(図６)。

図群の種類に依らない操作として、移動、浮動化(ＥＯＲ描画)、消去、深さ変更などを設けた。また、図群グループという単位もある。ＡＰの処理の都合で複数の図群に分けたものを、オペレータには１つの集団に見せたい図素集合の移動などに用いる。

(５)　スクロール・ロックと連動スクロール

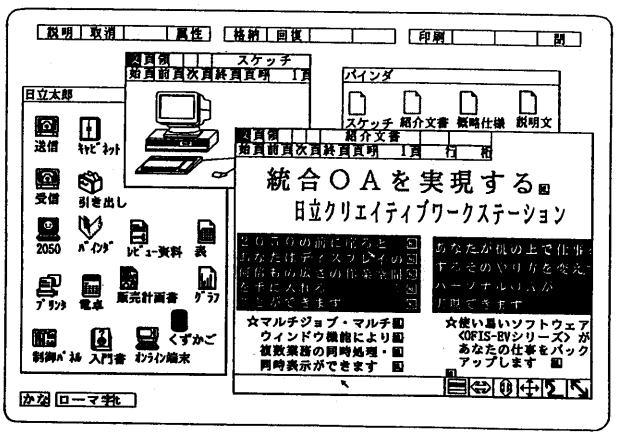
ウインドウ上に、スクロール・ロック領域、連動スクロール領域を設定できる。これにより、ＡＰは画面のタイトルや操作ガイダンスなどをウインドウに固定的に表示しておいたり、また、覗き窓の動きに連動して左右に動く補助ウインドウと上下に動く補助ウインドウを使って、巨大グラフの横軸、縦軸を覗き窓に連動して表示したりすることが可能となる。

(６)　かな漢字変換機能の内蔵化

全てのアプリケーションで共通に日本語を扱える環境を実現するため日本語入力機能をウインドウ・システムに内蔵した。連文節一括変換の可能な、かな漢字変換プログラムをキーボード・ドライバから呼出し、漢字コードに変換して、仮想端末毎の入力キューに入れる。ＡＰには仮想端末に漢字入力装置が接続されているように見える。かな漢字変換プログラムをＡＰから呼び出すことも、もちろんできる。

(７)　仮想画面読み出し

仮想端末に出力したデータを、ＡＰは再び読み出すことができる。読み出し機能を使って、ウインドウ間のデータ交換が可能となる。表作成ＡＰや、グラフ作成ＡＰが各々ウインドウに表示した表やグラフを、文書作成ＡＰが読み出して、報告書に挿入したり、編集したりすることを可能にする。読み出したデータの意味をＡＰが理解し再加工できるようにするためウインドウ・システム内では、文字はコードと属性で、図形はベクトル情報と属性でデータを保持する。



図－６　マルチメディア・データの表示

(８)　コマンド・メニュー

コマンド・メニューの表示・選択機能をＡＰに提供した(図６の画面最上部)。対話権を持つウインドウで走行しているＡＰのコマンド・メニューを画面に表示し、対話権が他のウインドウに写ったとき、そちらのウインドウで動いているＡＰのコマンド・メニューに、ウインドウ・システムで切り換える。

(９)　ＡＰ主導型のウインドウ・スクロール

ウインドウ・システムが行うスクロール機能のほかに、ＡＰ主導型のウインドウ・スクロール機能を提供する。仮想画面の最大サイズ(32768×32768ピクセル)を越える巨大論理画面を必要とするＡＰ(例えば、スプレッド・シート)は、この機能を使って、覗き窓が仮想画面の端まできたとき、仮想画面を入れ換えたりすることができる。

３．３　マルチウインドウ制御方式

マルチウインドウ・システムで用いた制御方式の中からいくつかを取り上げて述べる。

(１)　基本方式[1)]

ウインドウ・システムを、ＵＮＩＸに組み込むにあたり、次のような開発思想をとった。ＵＮＩＸは、端末等のデバイスをファイルとして扱えることを１つの特徴としている。この思想を活かすべく、仮想端末をファイルとして扱えるように、ウインドウ・システムをファイル・システムの中に埋め込む構造とする(従来のｔｔｙドライバーに位置付ける)。仮想端末にも他のデバイス同様に、スペシャル・ファイル(キャラクタ・スペシャル・ファイル)を作り、このファイルに読み書きすることにより、仮想端末にアクセスする方式とする。仮想端末用のスペシャル・ファイル(/dev/tty**)を予め仮想端末の最大台数分(１６台)だけ作成しておく。ＡＰから仮想端末の生成要求があったとき、空きスペシャル・ファイルの名称をウインドウ・システムが渡す。以後、ＡＰはそのスペシャル・ファイルをオープンして、従来の端末スペシャル・ファイルと同様に使用できる。これまで以上に、多様な表示制御データを送れるようになった点が従来と異なる。

仮想端末にアクセスするためのシステム・コールは、従来のＵＮＩＸと同じｏｐｅｎ，ｃｌｏｓｅ，ｒｅａｄ，ｗｒｉｔｅ，ｉｏｃｔｌを使用することとし、ＵＮＩＸコマンド群が、仮想端末上でもそのまま動くのを保証する。ｉｏｃｔｌ(入出力制御用システム・コール)については、仮想端末の機能を増強したことに対応して機能拡張する。ＡＰが仮想端末を生成したり破棄することは、従来ＵＮＩＸにない機能であるので、ｖｔシステム・コールを新設する。

このような思想により、ＡＰとウインドウ・システムのインタフェースの大部分を、ｗｒｉｔｅシステム・コールで渡すデータ・ストリームによって取ることにする。他の方式案としてＵＮＩＸのシステム・コールを追加する考え方があるが、それだとＵＮＩＸのＯＳインタフェースの標準性を損うこと、またシステム・コールをバッファリングすることはできないので性能が低下するという点で得策でない。これに対して、デ

ータ・ストリームを使う方式は、ＵＮＩＸでよく使われるＡＳＣＩＩ端末の制御機能(制御文字)を拡張して、仮想端末用の制御機能の集合を定義する方式で、ＡＰから、ウインドウ・システムに指令を出したいときは、制御文字列を仮想端末に送ることにより行なう。この方式では、ＵＮＩＸの標準ＯＳインタフェースを保持でき、制御文字列をバッファに貯えておいて、まとめて送りつけることができるので、システム・コールのオーバヘッドが少ない。

制御文字列の拡張にあたっては、将来の発展性を考え、国際標準規格に準拠した。文字列の表示制御に関するものは、ＩＳＯ６４２９に一致させた。ウインドウの制御、図群の制御機能などについてもＩＳＯ６４２９で定められた制御文字拡張法(Private use)に基づき仕様を定義した。円や多角形などの図形図素についてはＩＳＯ６４２９では、負の値を表現するのに適さないので、ＪＩＳ６２２８規格で定められた制御文字拡張法に基づいて仕様を定義した。

多数の制御文字を覚えなくてもよいようにするため、またプログラムの可読性を高めるために、ウインドウ関数群を提供した。ウインドウ関数は関数名とパラメータから制御文字列を生成する。

(2) 仮想端末表示データの再展開[3]

1つのウインドウを取り除くと、それまで見えていなかった別のウインドウの最新状態が現れる。このように、ウインドウの位置・寸法・重なりをオペレータまたはＡＰからの指示で変更する機能を支える技術として、「再展開」と呼ぶ方式を開発した。ＡＰからの表示は全てウインドウ・システム経由とし、実画面に反映されたかどうかに依らず、仮想画面のどこに何が表示されているかをコード化形式で管理する仮想端末データベース(ＤＢと呼ぶ)を保持する(図７参照)。再展開とは、対象領域に見えるべきものをＤＢから捜して実画面に反映させる方式である。今回、それを短時間で行う方式を開発できたので採用した。本手法は、ウインドウ操作だけでなく、ウインドウ内の表示にも用いられる。例えば図群の消去処理では、ＤＢから当該図群に関するデータを取り除いた後、図群のあった領域を再展開する。

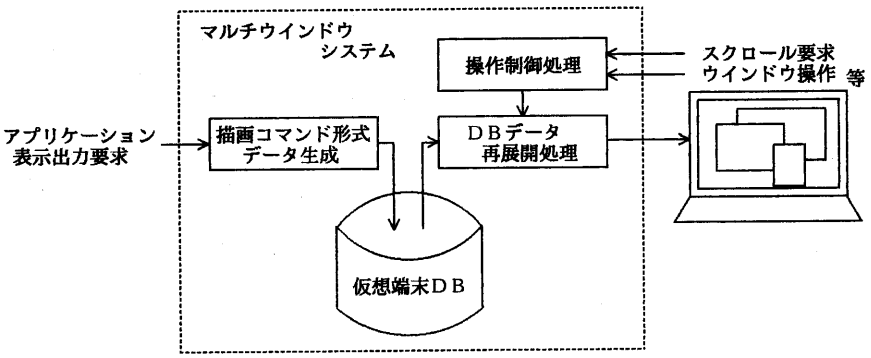
他の方式として、実画面には見えない部分に対しても、ＡＰが仮想画面に表示したものは裏のビットマップメモリに展開しておき、再展開の必要が発生した時点で、そこからイメージをコピーする方式が考えられるが、それだと大論理画面を含む複数の仮想端末をアクティブにするシステムでは、膨大な量のビットマップメモリが必要になり、データ交換後の文字の加工もできない。

(3) マルチプロセス環境での描画制御[3][4]

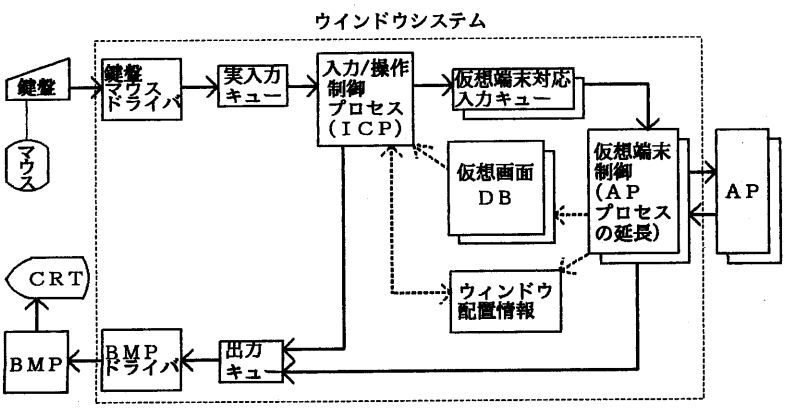
各ＡＰは、使用中の仮想端末が独立の端末であるかのごとく表示依頼を出すが、実際には１台の表示装置であり、干渉がある。また、ウインドウ内への表示の他に、随時オペレータまたはＡＰからウインドウ操作依頼が来る。このような状況で表示および管理情報に混乱をきたさないようにしてディスプレイへの描画に一貫性を持たせることが必要となる。これを実現するために、次のような描画の排他制御を行なう(図８)。

仮想画面データベース(ＤＢ)に対するアクセスの排他は、細かいフィールド単位ではなく、１回の表示依頼の処理の開始から終了までの間、更新対象ＤＢ全体を占有することによって行なう。実際の運用状態では、各ＡＰは専用に仮想端末を与えられるので、あるプロセスからの表示依頼を処理しようとした時、その仮想端末に別プロセスからの表示依頼が既に来ているために待たなければならない確率は低い。各々のＤＢの間には干渉要素を無くしてあるので、ＤＢが異なれば更新を並行して進められる。

ウインドウ配置情報が参照されている間は、新たな参照申請は即時許可される。配置情報を変更する時には、それを参照しているプロセスが全くないことを確認しなければならない。参照中のプロセスがあれば、変更したいプロセスはそれらの表示処理の終了を待つ。この状態では、新たな参照請求は



図－７　仮想端末ＤＢを介した表示出力方式



図－８　マルチプロセス環境での描画制御

変更終了まで待たされる。仮想端末への表示依頼の頻度に比べれば、ウインドウ配置の変更頻度は遥かに低い。
マウス操作をしている最中にウインドウ配置を変更されると、オペレータが戸惑うことになるので、マウスボタンが押されたならば、ＡＰによる配置変更を禁止する。参照は続けられる。つまり、この状態でもウインドウ内への通常の表示は続行される。ボタンが離された時点で、ウインドウ操作要求であると判断されたならば、配置情報の参照も禁止して、配置を変更する。

(4)　文字列データの表示制御[7]

文字列図群制御で利用するＢＭＰコマンドには、文字列描画、クリッピング枠の設定、ラスターオペレーション(イメージコピー、イメージクリア)等がある。高い描画性能を得るためにはＢＭＰコマンドのダイナミック・ステップ数を最小限に抑えなければならない。そこで、ＢＭＰコマンドの生成に際しては以下のような方式を取った。

(a)文字描画時のコマンド生成

ＢＭＰは、一個の文字列展開コマンドにより、行方向に並んだ一連の文字列を描画することができる。従って、図群の各行のうち、同一の可視領域に含まれる部分がそれぞれ一つの文字列展開コマンドとなるようにするのが描画効率上最も望ましい。図群上に文字を書き込んでゆく際には、一文字ごとにその文字の占める領域を求め、可視領域との交差を調べることにより、同じ可視領域に含まれる文字列が単一のコマンドとなるようにする。

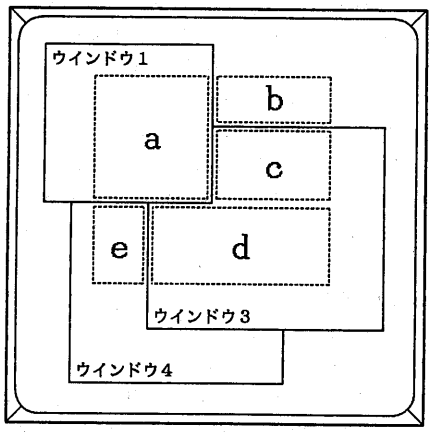
(b)編集操作時のコマンド生成

文字列図群では各文字のコードと属性のみを保持し、表示用のビットマップデータは持たない。したがって、編集結果を画面上に反映させるには、基本的には編集済の文字列データ全体を再びビットマップ展開する必要がある。しかし、この方法ではスクロールのように変更範囲が大きい場合には生成するコマンド量が膨大になり、表示性能の低下を招く。そこで、ビットマップ展開量を減らすため、フレームメモリ上に既に展開されているイメージデータを最大限に活用する方式とした。

(5)　逐次矩形分割表示[8]

逐次矩形分割表示方式とは、指定範囲(初期対象矩形)をウインドウの領域(参照矩形)との重畳位置関係に基づいで逐次矩形分割しながら、分割した矩形領域毎に該当するウインドウの内容を表示して行く方式である。

ウインドウの削除に対する適用例を図９に示す。図９は実画面上でウインドウ１、２、３、４がこの順序で重なり合った状態においてウインドウ２のみを削除する場合に、ウインドウ２によって直接被覆されていた範囲の表示内容を回復する状況を示している。初期対象矩形をウインドウ２の領域ａ＋ｂ＋ｃ＋ｄ＋ｅとし、その指定順位を２とする。まず最初に初期対象矩形を重畳順位が一番上の参照矩形(ウインドウ１)との位置関係で矩形分割する。分割した結果、３個の矩形ａ，ｂ＋ｃ，ｄ＋ｅが得られる。交差矩形ａは対象矩形(ウインドウ２)の指定順位より参照矩形(ウインドウ１)の重畳順位のほうが小さいので表示はしない。次に、非交差矩形ｂ＋ｃを新たな対象矩形として、次の重畳順位２の参照矩形(ウインドウ３)との位置関係で矩形分割する。分割によって得られた交差矩形ｃについてはウインドウ３の内容を表示する。非交差矩形ｂについては、さらに次の重畳順位３の参照矩形(ウインドウ４)との位置関係で矩形分割する。ただし、この例では分割は発生せず、また次の重畳順位の参照矩形がないので矩形ｂについては画面背景を表示する。矩形ｄ＋ｅについても同様の処理を行い、矩形ｄはウインドウ３、矩形ｅはウインドウ４の内容をそれぞれ表示する。



図－９　逐次矩形分割アルゴリズムによるウインドウの削除

４．ビジュアル・シェル[9)10)]

高機能ワークステーション上に使い勝手の良い利用者インタフェースを実現するため、マルチウインドウ・システムの機能を利用して、画面主体型の操作環境を提供する対話制御システム　―ビジュアル・シェル―　の方式を開発した。ビジュアル・シェルは、汎用の対話制御システムとして構築されているが、ここでは、わかりやすさのため、ＯＡ分野への適用を想定して、概念と機能を述べる。

４．１　ビジュアル・シェルの概念

ビジュアル・シェルは、次の設計目標を実現する対話制御ソフトウェアである。

(1)　統合化操作環境の実現

従来、コマンドの起動はオペランドが複雑だったり、形式がそれぞれ異なっていたりしてコマンド毎に起動法を記憶しなければならなかった。ビジュアル・シェルは、利用者に対し、応用ソフトウェアの起動操作手順に替るものを統一的に提供することにより、利用者インタフェースを簡略化し、操作性を向上させる。

(2)　現実の操作環境に合わせる。

現実の事務処理環境では、先ず、処理の対象となるキャビネットや文書を選びだし、次にその対象に対する操作を行う。操作には、中を開けて内容を見る，文書を削除する，等がある。又、許される操作の種類は対象によって決定される。ビジュアル・シェルでは、現実の環境に合わせた操作性を実現するため、処理対象(オブジェクト)を先ず選択し、その後に対象に許された操作群の中から、行いたい操作を次々と選択する操作環境を実現する。

(3) 視覚化された操作方法

従来のコマンド・システムでは、処理対象であるファイルやディレクトリ、操作群であるコマンド名等は、常には表示されておらず、操作者が頭の中の記憶を頼りに操作していた。ビジュアル・シェルは、オブジェクトであるファイルやディレクトリを表示しておくことにより、視覚化されたインタフェースを提供する。更に、ファイルやディレクトリをその内容を表現する図柄(アイコン)として表示することにより、視覚的な理解を容易にする。又、対象に許される操作群も対象が選択された時点で操作名をメニュー領域(コマンド・ビューポート)に表示する。

(4) マウスを用いたメニュー操作方式

計算機システムに不慣れな人は、キーボードを打つのが苦手という人が多い。従来、標準的なオペレーティングシステムでもメニュー方式によるコマンド入力方法は数多く見られたが、メニューを選択するのは、番号をキーボードから入力する方式が多かった。ビジュアル・シェルは、表示されたメニュー及び操作対象(ファイル)を直接画面上でマウスにより指示させることにより、キーボードからの入力量を減らす。

(5) 利用者毎の環境設定を可能にする。

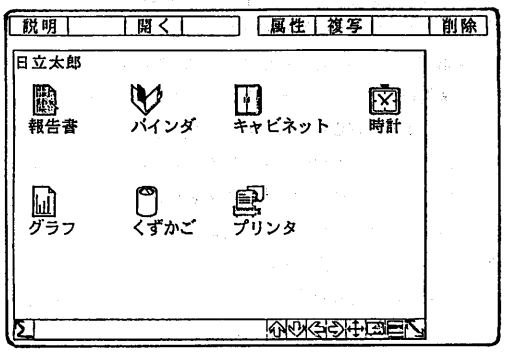
利用者の経験や好み、考え方によって、仕事の進め方、計算機システムに対する習熟度に違いがある。従来の対話システムでは、システム側が提供した使い勝手を個人の好みで簡単に変更することが出来なかった。ビジュアル・シェルでは、各種の変数や個人ライブラリを持つことにより、個人環境の自由な設定を可能とする。また、利用者個人の環境をセッション毎に退避、回復することによりマルチ・ユーザで使用しても個人毎の環境が確保されるようにしている。

4．2 ビジュアル・シェルの基本操作

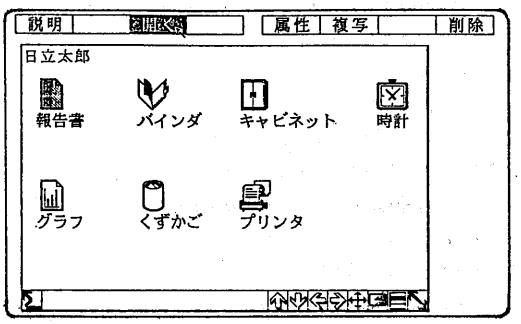
ビジュアル・シェルの基本的操作方法を述べる。ビジュアル・シェルは起動されると、指定されたファイルやディレクトリに対応するアイコンを画面上に表示し、利用者が処理対象を選択できるようにする。表示するアイコンは利用者の事前設定による。利用者は、マウスを操作して処理したいアイコン上にマウス・カーソルを移動し、マウスの左ボタンを押下する。ボタンが押下されると、ビジュアル・シェルは、そのアイコンに定義されている操作群(内容表示、編集、他)をメニューとして表示する(図１０)。利用者は表示されたメニューからマウスを用いて、起動したい操作を選択する(図１１)。ビジュアル・シェルは、自分自身や他のＡＰが使用しているウインドウと別に、新たにウインドウを生成し、並列的に実行可能な環境を設定し、クラス定義を解釈し、利用者が選択した操作を実行するコマンドを起動する(図１２)。

4．3 ビジュアル・シェルの機能

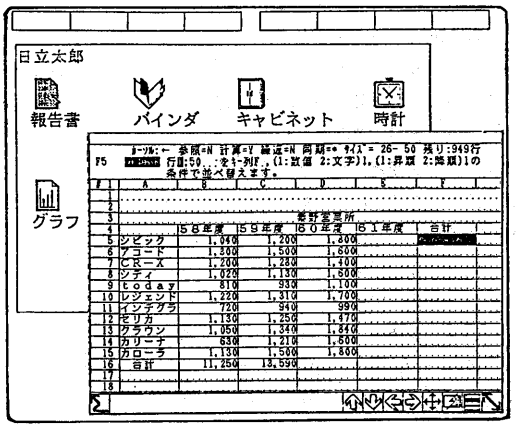
ビジュアル・シェルはＵＮＩＸにおけるファイル・システムの階層的構造を実際の作業環境におけるバインダや文書等の階層的構造に対応づけている。その上でディレクトリとファイルのみを対象とした限定されたオブジェクト指向の考え方を導入した。ビジュアル・シェルでは、ファイルやディレクトリを更に細かく分類して、各々の分類毎に自分自身に対する操作群(クラス定義)を定義できるようにした。ファイルがどのクラスに属するかを設定



図－１０ 報告書アイコンを選択



図－１１ メニューを選択



図－１２ ウインドウが開かれ、ＡＰが起動された状態

するための機構としてファイルディクショナリ管理機能[2]を使用する。クラス定義は、シェル・プロシージャで記述された複数の操作定義からなる。各操作定義はＵＮＩＸの標準コマンド解析プログラムであるshellを用いて解釈させ、forやwhile等の構造を使えるようにして、作成の柔軟性と容易性を確保した。クラス定義ファイルは、ＡＳＣＩＩファイルとして、利用者が通常の編集コマンドを用いて変更可能としている。オブジェクトとしてのディレクトリやファイルは、１つのクラスにしか属せないが、後に述べるようにクラス定義用のファイルを格納するディレクトリを個人が自由に設定可能としているので、利用者は自分に適したクラス定義群を構築したり、ライブラリを作成したりできる。

利用者が設定可能な個人環境として、次の項目がある。

(i) クラス定義

クラス定義ファイルを格納するディレクトリを設定可能である。

(ii) アイコン定義

アイコン・ファイルを格納するディレクトリを設定可能である。

(iii) 表示名称

同一の文書を別の部署では、他の目的を持って見ることがよくあり、別々の名前を付けた方が理解しやすい場合がある。又、文書の先頭の1行を文書の名称としたい場合も多い。このとき、文書の内容が変更になるたびに名称を変更できる機能が必要である。このために、アイコンを表示する際の名称を複数個設定可能としている。

(iv) 画面設定

ビジュアル・シェルが表示している画面の大きさ、位置、アイコンの配置等の設定を行えるようにして利用者独自のレイアウトを可能にしている。

クラス定義を記述するための言語を規定するクラス定義解釈プログラムは、通常UNIXのBourne-Shellを用いているが、利用者の指定によって他のプログラム、例えばc-shellを用いてクラス定義の解釈を行なわせることを可能にしている。

ビジュアル・シェルは、APとのインタフェースを2つ持っている。1つめは、クラス定義実行時にAPが各種情報を得られるように、UNIXの環境変数を用いて、次の3つの情報を提供する。

・クラス名
・ファイル名
・メソッド名(操作名)

2つめに、ビジュアル・シェルは、利用者の操作を解析するだけでなく、実行されているAPからの指示も受け取る。ファイル削除や作成を行なうAPが、ファイルの作成、削除を行ったときに、ビジュアル・シェルにアイコン表示を依頼するためである。APとビジュアル・シェルとのやりとりはUNIXのプロセス間交信の機能を用いて行なう。

5. 統合OA環境への接近

高機能ワークステーション上に統合OA環境を構築するには、(1)AP機能の統合、(2)操作の統合、(3)ネットワークによる統合、の3つの課題がある。ここでは、(1),(2)への接近法を取り上げる。

5.1 統合文書処理環境[11)12)13)]

OSでデータを扱う単位としてのファイルより一段高い層に「文書」という概念を導入し、マルチメディア・データを扱う単位とする。文書処理に関する必要機能を基本ソフトウェアとして用意し、APに対して文書の作成、編集、ファイリング、転送などの各機能を標準的なインタフェースで実現することを可能とした。この基本ソフトウェアを「文書データベース」と呼び、ミドル・ソフトウェア層に置いた。文書DBは「文書を共通に扱う場」を実現するため、階層型領域構造を持つシートと、その編集、保管、表示、転送を行う関数ライブラリを提供する。これにより、(1)文字、表、グラフ、図形などの混在した高品質のマルチメディア文書を容易に作成でき、(2)シートを共有することによって、複数APの統合を図ることが可能となる。英文ワープロでの編集結果を機械翻訳プログラムに渡し、日本語に訳して日本語ワープロに渡す。加工した後、それを報告書作成プログラムに渡して、報告所の中に埋込むといったことをシートを介して行うことによりAP機能の統合を図る。これには、シート上の文書構造を統一することが必須で、文書DBがこの役割を果たす。文書DBは、ウインドウ・システムの提供するマルチメディア仮想端末の機能を使用してこれを行う。

5.2 ウインドウ間のデータ交換による統合

OA用のAP群(OFIS-EVシリーズ)の中に、文書処理環境を利用して動く1つのAPとして、文書合成プログラム(OFIS/REPORT-EV)がある。これは、文章、表、グラフ、図形などの混在した書類を作成するため、領域の貼り合わせ・重ね合せ機能を有しており、オペレータは、ワードプロセッサ、作表、作画、グラフ処理などの各APで作成したウインドウ上のデータを見ながら、マウス操作で画面の切り貼り(cut and paste)を行うことが出来る(図13)。これには、ウインドウシステムの持つ仮想画面読み出し機能が使われる。フレーム・メモリ上のビットマップ・データを切り貼りするのと違って、コード化・データとして保持している仮想端末DBを読み出す方式であるので、他のウインドウからデータを切り貼りした後、続けてそのデータを編集できるのが大きな利点である。さらに、切り貼り機能は、文書に限らずオンライン端末画面ほか全てのウインドウ画面に対して適用可能であることが大きな利点である。

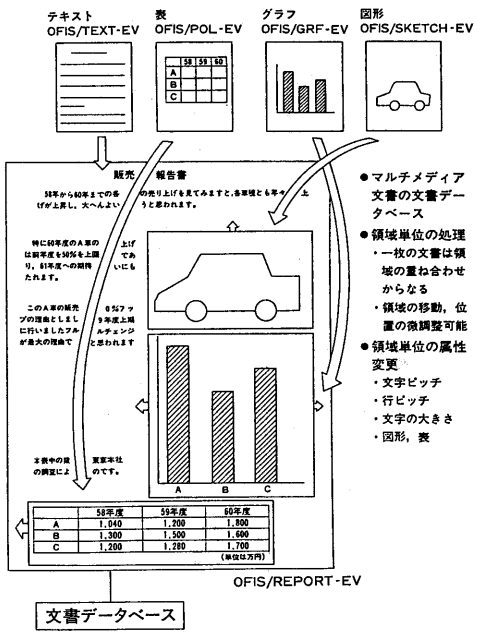


図-13 切り貼り機能を利用したマルチメディア文書の作成

5.3 仮想デスク環境[10)]

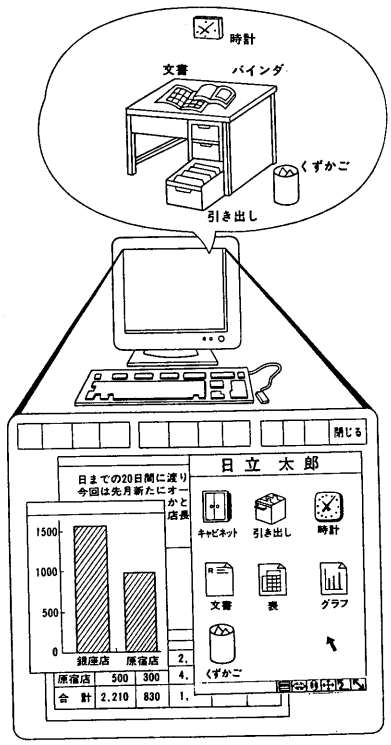
一般事務職の人が慣れ親しんだ机上で事務作業を行うのと同様

の感覚で、ワークステーションを操作できる環境の実現方式を開発した。この環境を仮想デスク環境と呼んでいる。机の引き出しをあけてバインダを取り出す。バインダを開いて書類を捜す。書類を取り出し、中のデータを写し取るといった一連の作業を、ワークステーション画面上でシミュレートする。いわば、オフィスの電子化である(図１４)。このような環境を提供するソフトウェア(ＯＦＩＳ／ＤＥＳＫ－ＥＶ)は、第４章で述べたビジュアル・シェルの機能を使用している。アイコン、クラス定義、表示名称などのオブジェクト情報をＯＡ分野向きに一式揃えることにより仮想デスク環境のベースができる。さらに、オブジェクトを操作するコマンド群を追加して仮想デスク環境を構築している。仮想デスク環境では、アイコンとマウスを操作しながら、ＯＳを意識せずにプログラムを走らせ、ファイルをオープンして目的とする結果を得ることができ、コンピュータのことを何も知らない人でも抵抗なく仕事できる。これが利点である。一方、コンピュータの端末に慣れた人にとっては、この操作インタフェースはまだるこしいと感じるかも知れない。そこで、アイコンの代りに、従来のように文字列による一覧形式の表示も可能とした。ＯＡ用の全てのＡＰ群をこのような仮想デスク環境の統一された操作で起動する方式により、統合化操作環境の実現を図った。

６．結言

仮想端末の思想を徹底して、マルチメディア表示機能にまで拡張したマルチウインドウ制御方式、および、視覚化された統一的な操作環境を実現する対話制御方式を中心に、高機能ワークステーションの概念と機能を述べた。また、ＯＡ分野への適用として、この仕組みの上に構築される文書処理環境、仮想デスク環境によりアプリケーション機能を統合化する方式について報告した。

これらの機能を組み込んで開発された高機能ワークステーションを実際に動かし、満足すべき性能・操作性が得られることを確認した。



図－１４　仮想デスク環境の概念

謝辞

マルチウインドウ・システムの研究では、当研究所の白濱伸雄研究員、岩見秀文研究員、和歌森文男主任研究員をはじめとする研究者が、同じくビジュアル・シェルの研究では橋本尚研究員、野瀬俊郎研究員をはじめとする研究者がアイディアを出し合った。これらの研究者ならびに文書データベースの研究、ワークステーションＯＳの研究に携わった研究者の皆様に深謝する。本稿では、これらの皆様を代表して報告した。また、ご指導をいただいた当研究所川崎淳所長、当社神奈川工場井上武洋工場長に深甚の謝意を表する。

参考文献

1)小島,白濱,岩見,秋田：マルチウインドウ･システムの開発思想
2)白濱,高梨,小島：高機能仮想端末群の仕様開発
3)白濱,中村,小島：マルチプロセス環境における描画制御方式
4)岩見,中村,小島：マルチウインドウ表示方式
5)和歌森,森下,白濱,小島：マルチウインドウ表示データ管理制御方式
6)高橋,山田,小島：ウインドウへの文字列データの表示方法
7)高梨,中村,小島,戸木：マルチウインドウ・システムにおける入力制御方式
8)橋本,野瀬,萬田,小島：ビジュアル・シェルの開発
9)野瀬,橋本,石川,萬田：オブジェクトの属性を管理するファイル・ディクショナリ管理の開発
10)萬田,住友,野瀬,橋本：アイコン操作コマンドの開発
11)渡部,柳,吉田：統合文書処理環境の開発思想
12)柳,渡部,吉田：領域概念を持つ文書処理標準関数の開発
13)吉田,渡部,柳,金房：高品質日本語文書編集方式の開発
14)中嶌,福岡：レコード型ファイル管理の開発
15)近藤,福岡,村田：システム共通サブルーチンのリンク方式
以上、情報処理学会第32回全国大会講演論文集(1986.3)

16)高機能ワークステーション特集、情報処理、Vol.25, No.2 (1984.2)