OUKA
The University of Osaka
Institutional Knowledge Archive

Title	マルチウィンドウシステムの現状
Author(s)	松浦, 敏雄
Citation	大阪大学大型計算機センターニュース. 1988, 71, p. 45-55
Version Type	VoR
URL	https://hdl.handle.net/11094/65804
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

マルチウィンドウシステムの現状

基礎工学部情報工学科
松浦 敏雄

１．　まえがき

近年、ワークステーションの普及が著しいが、そのほとんど全てがビットマップディスプレイとポインティングデバイス（ほとんどの場合はマウス）を備えており、それらを利用したマルチウィンドウシステムが稼働している。マルチウィンドウシステムは同時に複数の作業が行え、さらに従来のキャラクタディスプレイと比べて、より良いユーザインタフェースを提供できる可能性をもっている。

本稿では、マルチウィンドウシステムについて、ユーザの立場およびプログラマの立場から眺めて、その現状ならびに問題点等について述べる。また、最近のワークステーション上での代表的なマルチウィンドウシステムについて概観する。

２．　マルチウィンドウシステム上でのユーザインタフェース

まず、マルチウィンドウシステムがどのように利用されているかについて、ユーザの立場から眺めてみる。

（１）複数端末の実現（端末エミュレータ）

現在のマルチウィンドウシステムで最も多く用いられているウィンドウは端末エミュレータである。端末エミュレータのウィンドウは通常のキャラクタディスプレイ端末とほぼ等価な機能を持っており、１つのウィンドウで１つの計算機端末の機能を果たす。マルチウィンドウシステムではビットマップディスプレイ上で、これらのウィンドウを複数個開いて利用できる。従って、１つの端末エミュレータウィンドウ上で、あるファイルを表示させながら、別のウィンドウで、他のファイルを編集したりできる。また、簡単なマウスの操作で任意のウィンドウ上の文字列をマルチウィンドウシステムが持っている内部のバッファにコピーすることができ、それを任意のウィンドウ上に入力として与えることができる（この機能をCut & Pasteという）。この機能は少量のデータ転送の際に手軽に用いることが出来る。

端末エミュレータはネットワーク環境でよりその威力を発揮する。最近のワークステーションはほとんどの場合ネットワークで接続されている。このような環境では通常ネットワークを経由したリモートログインの機能が提供されており、これを用いると１つのビットマップディスプレイ上で、複数のリモートの計算機を利用できる。

（２）より良いユーザインタフェースの実現

マルチウィンドウシステムの最大の利点は、ビットマップディスプレ

イとマウスをフルに活用した優れたユーザインタフェースを実現できる点にある。今のところ、マルチウィンドウシステム上のアプリケーションプログラムはあまり多くはないが、ビジュアルなインタフェースによって、優れたユーザインタフェースを提供している。一般にこのようなアプリケーションプログラムでは、ビデオやオーディオ装置などに見られるスイッチボタン、ボリューム、およびメータなどの部品を模倣してディスプレイ上に表示し、ユーザに対して視覚的に訴えるユーザインタフェースを実現している。これらの部品に対する操作もマウスを用いて直接各部品を操作できるようになっている。マルチウィンドウシステムでは、通常、アプリケーションプログラムからこのような部品が利用しやすいように適当なライブラリを備えている。

しかしながら、マルチウィンドウシステム上のすべてのアプリケーションプログラムが、優れたユーザインタフェースを実現しているのではないことに注意して頂きたい。マルチウィンドウシステムは、優れたユーザインタフェースを実現するための１つの手段に過ぎないのであって、ユーザインタフェース良し悪しは、あくまでアプリケーションプログラムの良し悪しにかかっている。

３．マルチウィンドウシステム上でのプログラミング

（１）プログラミング上の問題点

マルチウィンドウシステム上でのプログラミングは一般に難しい。（もちろん従来と同様のユーザインタフェースを実現すれば良いのなら、従来となんら変ることはない。）実際、例えば、X Window上で１つのウィンドウを開いて、"Hellow World"という文字列を表示するだけのプログラムを書くのにX Windowのライブラリ関数を16個も用いなければ成らないことが報告されている[9]。本節では、マルチウィンドウシステム上でのプログラミングが何故難しいかについて述べてみたい。

①画面の制御

マルチウィンドウシステム上のアプリケーションプログラムはマルチウィンドウであるがために以下の事柄を余分に考慮しなければならない。

- 開くべきウィンドウの位置と大きさの指定.
- 書くべき文字または図形のフォントの種類と大きさの指定.
- 書くべき文字を背景色と異なった色で書く.
- そのウィンドウが"expose"（他のウィンドウによって隠されていた部分が表に現れたことの通知）されたときの処理. 必要ならば再描画を行う.
- そのウィンドウが"resize"（大きさの変更があったことの通知）されたときの処理.
- そのウィンドウがアイコン化されたときの処理および逆にアイコンがウィンドウに戻されたときの処理（deiconify）.

また、ウィンドウシステムによっては、描画の高速化などの目的でさらにいくつかの事項の指定を要求するものがある。上述の項目の全てがア

プリケーションプログラムの責任で行わなければならないわけではないが、これらの点を理解していないとウィンドウシステムのアプリケーションプログラムを書くのは難しい。

②イベントの取り扱い

マルチウィンドウシステムのプログラミングの難しさの第２の原因はマウスにある。今までのプログラミングにおいては、プログラムの動作と非同期な入力（装置）は通常キーボードのみであった。従って、プログラム中でキー入力を取り込む場合、必要なときにキーボードからデータを読み込む要求（例えばＣ言語ではgetchar()）を出せば良く、そのときにキーボードからのデータがまだ入力されていないときは、そこで待っていて、データが入力されてから、プログラムが続行できれば良かった。

ところが、マルチウィンドウシステムではマウスの入力も取り扱う必要があるため、ユーザからの非同期な入力はマウスとキーボードの２つになる。従って、プログラム中で”キーボードからの入力を待つ”と、その間のマウスからの入力が行えず、逆に、”マウスからの入力を待つ”と、その間のキーボード入力をとれない羽目になる。そこで、ほとんどのマルチウィンドウシステムでは、キーボードとマウスからの入力要求（イベントと呼ぶ）を１つの入力ストリームにして、それらのイベントの早いものから順に処理するような仕組みになっている。マルチウィンドウシステムのアプリケーションプログラムが取り扱わなければならないイベントは、マウス、キーボードからのイベントのほかに、前述の"expose"イベント、"resize"イベント、および、マウスがウィンドウに入って来たり、逆に出て行ったことを伝えるイベントなどがある。

（２）プログラミングのためのツールキット

マルチウィンドウシステムでは、上述したような画面制御（線分などの描画を含む）やイベントを取り扱うためのライブラリを用意している。このライブラリをウィンドウライブラリと呼ぶことにする。アプリケーションプログラマは、通常このライブラリを用いてプログラミングを行う。この場合、プログラミングの自由度は大きいが、プログラマはウィンドウシステムに対する詳しい知識が要求され、前述したように、そのプログラミングは容易ではない。

そこで、ほとんどの場合アプリケーションプログラムでよく用いられるような部品を簡単に利用できるようにライブラリを用意している。これをツールキットライブラリと呼ぶ。ツールキットライブラリにあらかじめ用意されている部品を利用してアプリケーションプログラムを作成するのは比較的容易である。従って充実したツールキットライブラリを備えることがウィンドウシステムにとって重要である。ウィンドウシステムによっては、自分で部品を定義してツールキットライブラリに登録出来る機能を持っているものもある。しかし、この部品の定義機能はそれらを単に利用するの比べると簡単ではない。

（３）より簡単に優れたユーザインタフェースを実現するには

既存のツールキットを用いると比較的簡単にマルチウィンドウのアプリケーションプログラムを作成できる。それでもツールキットを使うためのノウハウを修得する必要があり、そのためにはやはり使っているウィンドウシステムのポリシーを理解しなければならない。

もっと簡単に、良いユーザインタフェースを持ったアプリケーションプログラムができないのだろうか。そこで考えられるのは、アプリケーションプログラムからそのユーザインタフェース部を分離し、ユーザインタフェース部をなんらかの方法で記述し（もちろんこれが簡単でなければ意味がない）、アプリケーションプログラマはアプリケーションプログラムの本体のプログラミングに専念出来るようにする方法である。ここで記述されたユーザインタフェースの定義からビジュアルなユーザインタフェースが実現できれば、アプリケーションプログラマの労力が大幅に軽減される。このような方法は今のところあまり実現されていないが徐々に取り入れられるであろう。

（４）ユーザインタフェースの自由度と標準化

ツールキットライブラリを充実させて、全てのアプリケーションプログラムがツールキットライブラリを用いるようにすることは、ユーザインタフェースの標準化につながる。このためユーザにとっては全てのウィンドウで操作方法が統一され使いやすくなる。こうやって優れたユーザインタフェースを実現しているウィンドウシステムにMacintosh[14]やAndrew[16]がある。

しかしながら、ツールキットの集合が不変である限りシステム全体の拡張性がなくなってしまう。ツールキットにない部品が必要になった場合（こういうことは、例えば、他のシステムで動いているアプリケーションプログラムを移植するときなどに起こり得る）、ウィンドウライブラリを直接用いて部品を作らなければならない。

X Window[6][7][8]では、ウィンドウライブラリは出来るだけ自由度を高めておいて、ツールキットライブラリで標準化を計っている。またこのツールキットも拡張できるようになっている。しかし、既に述べたようにこのことは標準化にとってはマイナスである。結局、ユーザインタフェースの自由度と標準化をどのようにバランスさせるかが、ウィンドウシステムにとって重要である。

５．マルチウィンドウシステムの実際

本節では、主なマルチウィンドウシステムについて個々に述べる。

（１）Ｓｕｎｖｉｅｗ

Sunview[3][4][5]はSun Microsystems社のワークステーションで稼働するウィンドウシステムである。画面への描画処理は全てライブラリ内で行われるので、高速に描画できる。また、Sunviewでは、ツールキッ

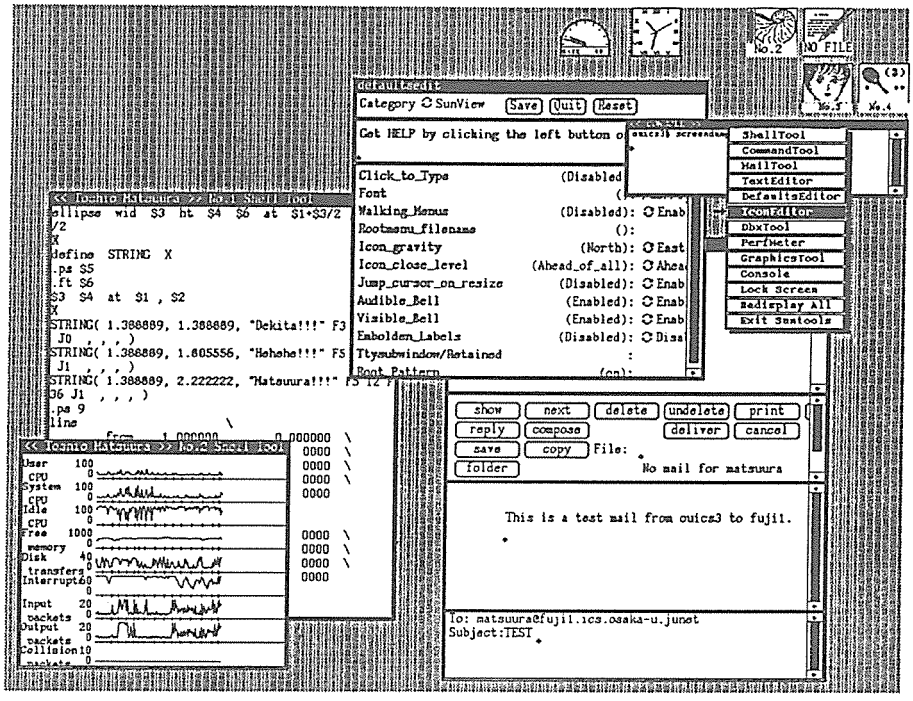

図1 Sunviewの画面表示例

トライブラリが充実しており、しかも、そのポリシーは比較的判りやすい。イベントの処理はすべてツールキット側で処理される構造になっている。アプリケーションプログラムから見た場合、ウィンドウ毎にイベント処理ルーチンが存在しており、ウィンドウの独立性が高く、見通しのよいプログラムが書きやすい。マルチウィンドウに対応した既存のアプリケーションプログラムとしては、dbxtool（デバッガ）,iconeditor（アイコンのイメージを作るためのビットマップエディタ）,defaulteditor（ウィンドウの種々の属性を設定するためのツール）,texteditor（いわゆるテキストエディタ）,mailtool（mailを読み書きするためのツール）などがあり、充実している。

欠点としては、ライブラリルーチンが大きく、メモリ、ディスクを圧迫することと、立ち上げに時間がかかることが挙げられる。しかし、Sunの新しいOS（SunOS 4.0）では、シェアードライブラリの機能が提供されているので、この問題もほとんど解決されている。

（2）X Window

X Window[6][7][8]は、MITとDECで共同開発されたウィンドウシステムである。現時点(1988年9月30日)での最新版はバージョン11リリース2(X.V11R2と略す)であるが、10月半ばにリリース3(X.V11R3)が供給される予定である。 X11R2のディストリビューションキットは約60MBものソ

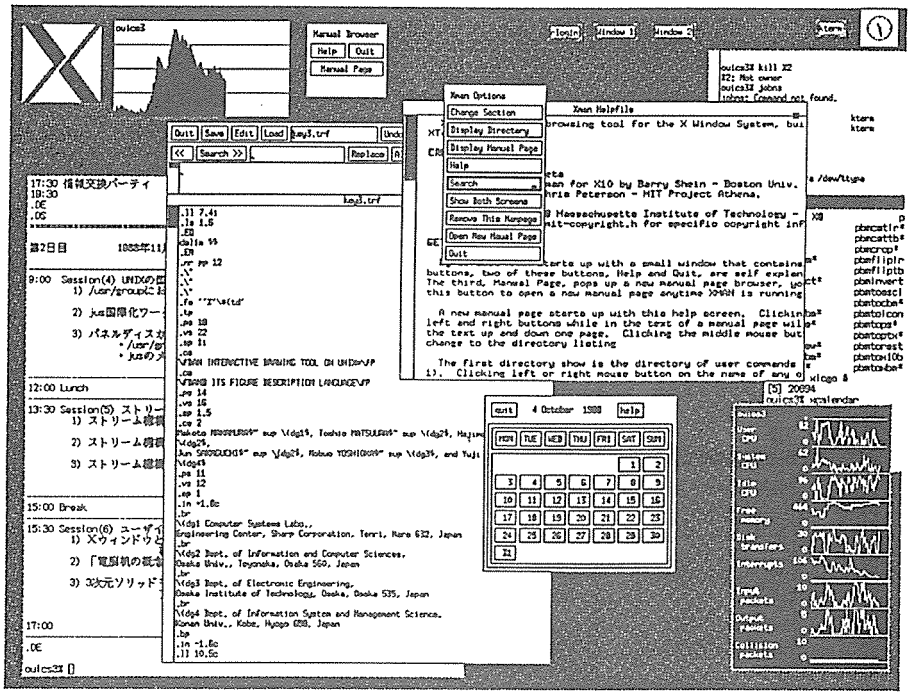

図２ X Windowの画面表示例

ースプログラムおよびドキュメントから構成されている。X Windowが稼働する計算機としては、DECのuVAX,Sun MicrosystemsのSun,IBMのRT-PC,HPのHP-9000,Apolloのdomain、SonyのNews、立石電機のLunaなどがある。ウィンドウシステムとしては新しい部類であるが、その移植性の良さならびに無料で入手できることから急速に広まりつつある。X Window自身も逐次改良されており、その開発スピードは極めて速い。（MITおよび世界中のX Windowに携わっている人々から、ネットワークを通じて、膨大なレポートが毎日届いている。）

X Windowは、サーバ・クライアント方式のウィンドウシステムである。この方式では、画面への描画およびイベントの取り扱いを１つの特別なプロセス（サーバと呼ぶ）が行い、サーバと個々のアプリケーションプログラム（クライアントと呼ぶ）との間はプロセス間通信によって、必要な情報をやり取りする。この方式では、描画およびイベントの処理をサーバが行うので、クライアントは小さくて済む。しかも、サーバとクライアント間の通信は、ネットワーク上で行えるので、サーバとクライアンとが同じ計算機上にある必要はない。

X Windowではいくつかのウィンドウマネージャが利用可能である。最も良く用いられるウィンドウマネージャはuwmと呼ばれるものであり、オーバラッピングウィンドウをサポートしている。uwmでは、.uwmrcと言うファイルを用意することで、任意のマウスボタンに対して自由に機

能の定義が行えるなど、極めて自由度の高いウィンドウシステムになっている。このように、自分の好みにあった環境を設定できるのは便利ではあるが、逆に、各個人ごとに操作方法が異なるため、”他人のウィンドウシステムは使えない”という弊害も生まれる。

X Windowにおけるウィンドウライブラリは**Xlib**と呼ばれている。Xlibは自由度の高いライブラリであり、ほとんど何でもサポートされている。そのため、逆にいろいろと指定しなければならないことも多く、ライブラリを使いこなすのは容易ではない。

ツールキットライブラリとしては**X Toolkit**がある。ただし、これは、X11R2になって始めて、実用的なものにまとめあげられたものであり、これからその充実が望まれるところである。**X Toolkit**は完全なオブジェクト指向の概念を用いており、クラス階層を用いて新しい部品を定義できる。しかしながら、C言語を用いてオブジェクト指向なプログラミングを行うのであるから、所々に無理な面が見られる。

X11R2には、漢字の表示できる端末エミュレータ(**kterm**)が含まれている。これは、東工大の佐野氏によって作成されたものをMITが採用して付録のソフトウェアとしてディストリビュートされている。**kterm**とXサーバとの通信にはJISコードが用いられている。**kterm**を用いることで異機種の計算機間でも、漢字を含めた通信が可能になる。

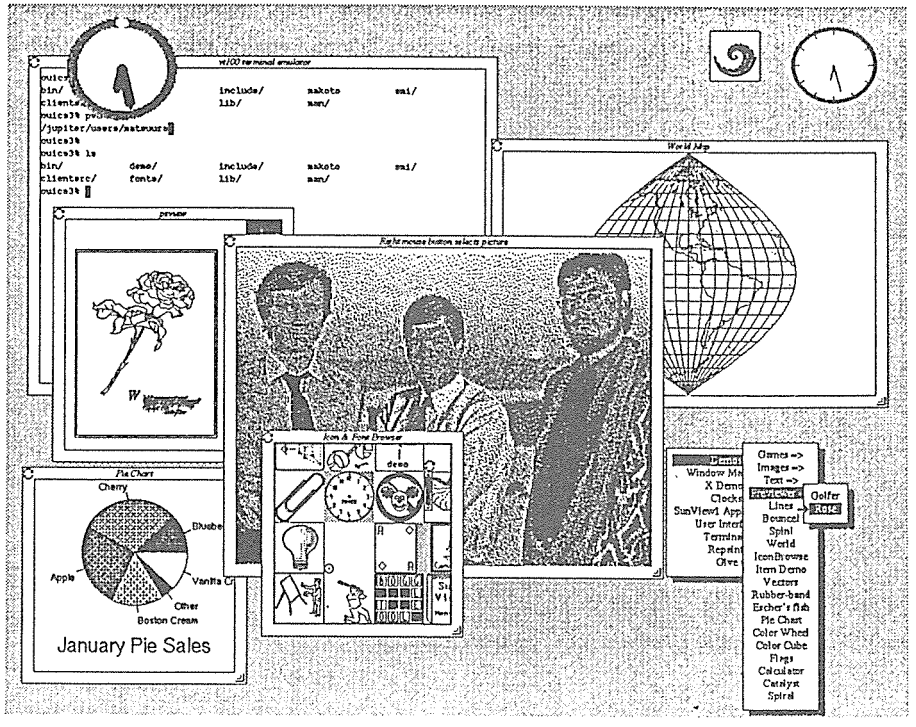

図3 ＮｅＷＳの画面表示例

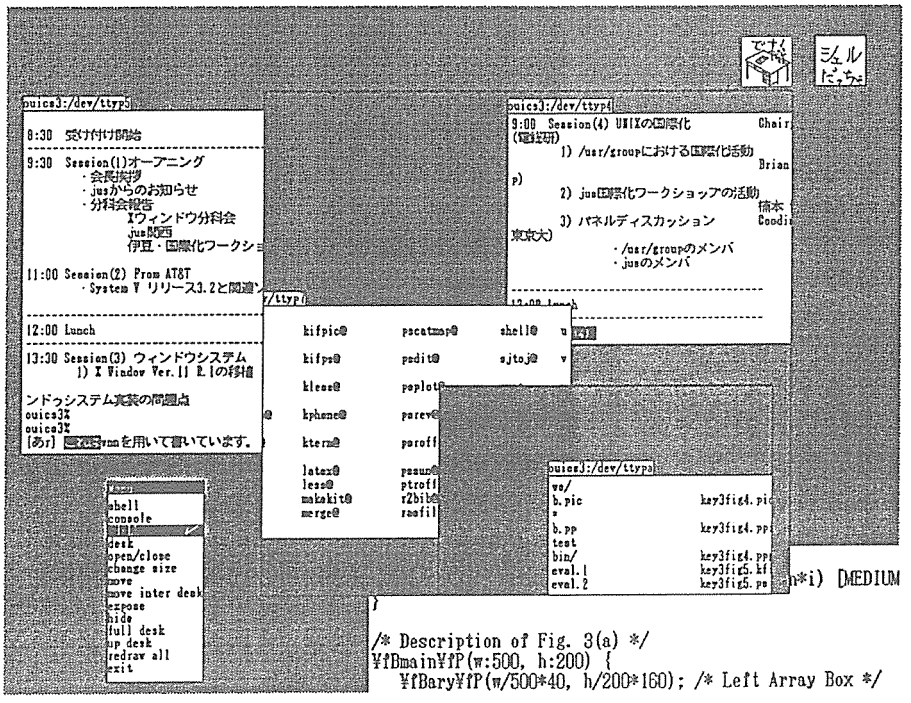

図４ ＧＭＷの画面表示例

（３）ＮｅＷＳ

NeWS[10][11][12]はSun Microsystems社がSun用に新しく作ったウィンドウシステムである。それぞれのウィンドウがPostScriptのインタプリタを含んでいるため、クライアントからPostScriptのプログラムをサーバに送るだけで、画面上に描画することが出来る。もちろん、画面のハードコピーを撮るのも簡単である。また、PostScriptのプログラムをサーバにダウンロードすることによってサーバの機能を拡張することができる。

NeWSでは、楕円形をしたウィンドウを作ることも容易である。NeWSのデモを見ているとその画面の美しさは他の追従を許さないものがある。

（４）ＧＭＷ

GMW[13]は京都大学、（株）アステックおよび（株）立石電機で共同開発された、おそらく、日本で開発された唯一つのまともなウィンドウシステムである。（言い替えれば、GMW以外の国産のウィンドウシステムで使い勝手の良いものはない。） GMWはユニークな描画モデルを持っており、例えば、１つのウィンドウが、あたかもマルチウィンドウを表示できるビットマップディスプレイの用に振舞うことが出来る。また、画面の大きさよりもはるかに大きなウィンドウを開くことも可能であり、その全体を眺める（詳細は見えない）こともできる。しかしながら、これらの機能を発揮できるアプリケーションプログラムはまだあまり見あ

たらない。

また、同じグループからGMWとともにWnnと呼ばれる日本語仮名漢字変換プログラムもディストリビュートされている。GMW自身がはじめから日本語の処理を意識して作られており、Wnnと併用することで、漢字の入出力も円滑に行える。

（５）その他のマルチウィンドウシステム

①Macintosh

Macintosh[14]は、Apple Computer Inc.のSteve Jobsらによって開発されたパーソナルコンピュータであって、そのウィンドウシステムは最も完成度の高いものの１つである。Macintoshのツールキットライブラリは充実しており、Macintosh上でアプリケーションプログラムを販売目的で開発する者は、このツールキットを正しく用いるように義務付けられている。このため、Macintoshではアプリケーションプログラム間の操作性が見事に統一されており、ほとんどマニュアルなしで操作できる。Macintoshが成功した理由は、アプリケーションプログラムの操作モデルを１つに限定して、そのモデルに必要かつ十分なツールキットの集合を提供し、さらに、ツールキットの使い方を徹底するために詳しい手引書[14]を用意したことにある。このアプローチは、ユーザ、および、プログラマの自由度を追及するX Windowとは対照的であり、興味深い。

②Smalltalk-80

Smalltalk-80[15]はXerox社のPalo Alto研究所で1970年代初頭から研究されてきたシステムで、オブジェクト指向の言語と環境を提供するものである。

Smalltalk-80のウィンドウシステムはMVCと呼ばれる描画モデルによって実現されている。MVCはModel、View、および、Controllerの略で、それぞれ、描画すべき内容、ウィンドウ自身の管理、および、キーボードおよびマウスからの入力を取り扱うクラスの名前である。MVCは完全なオブジェクト指向の概念に従って構成されているが、各アプリケーションプログラマは描画したいものそれぞれに対して、MVCのそれぞれのクラスを作成しなければならない。このとき、MVCの作業分担が明白でないため、わかり難いものとなっている。

③Andrew

Andrew[16]はCarnegie-Mellon大学(CMU)とIBMの共同プロジェクトであり、キャンパス内に散在する数千台のワークステーションをネットワークで接続して統合化された環境を提供するためのものである。Andrewプロジェクトのウィンドウマネージャは、wmと呼ばれているが、タイリングのウィントウシステムである。アプリケーションプログラムはWYSWYGなエディタをはじめmailやnewsの読み書きのためのツール等、かなり充実している。ウィンドウライブラリは極めて単純でアプリケーションプログラムの作成はX Windowなどと比べるとはるかに容易である。これ

は、ウィンドウシステム自身が状態を持っており、ウィンドウライブラリを呼び出すときに煩わしいパラメータを沢山指定する必要がなく、適当に処理してくれるからである。Andrewシステムはこのように初心者でも容易に使えるような工夫が随所に見られる優れたシステムである。また、熟練した者に対しても煩わしさを感じさせないような工夫を行っている。

6．あとがき

マルチウィンドウシステムについて、ユーザ、および、プログラマの立場から眺め、問題点などを挙げてみた。また、実際の代表的なウィンドウシステムについて述べた。

ここで取り上げたのマルチウィンドウシステムはどれも実用レベルに達しているが、マルチウィンドウシステムの利点を最大限に生かしたアプリケーションソフトウェアはまだまだ少ない。今後は、AT&TとSun Microystemsの提唱するOpen Look[18]のように、ウィンドウシステムに独立なユーザインタフェースの標準化が進むものと思われる。しかし、本文でも述べたように、このような標準化はユーザの負担を軽減するかわりに、技術の進歩を止める働きをする可能性もあることを注意しなければならない。

参考文献

[1] Rosenthal, D.S.H.: Window System Implementations for Berkeley UNIX, Proc. Summer 1986 USENIX Conference, 1986.

[2] 萩谷 昌巳, 森島 晃年: ワークステーション上のウィンドウ・システムについて, コンピュータソフトウェア, Vol.5, No.2, April 1988.

[3] Sun Microsystems: SunView Programmer's Guide, 1986.

[4] Sun Microsystems: SunView System Programmer's Guide, 1986.

[5] 萩谷 昌巳: Sunウィンドウを解剖する, インターフェース, ＣＱ出版, Vol.12, No.5, pp.300-314.(1986.5).

[6] Scheifler, R. and Gettys, J.: The X Window System, ACM trans. on Grahpics, Vol.5, No.2, pp.79-109, 1986.

[7] 酒匂 寛：X-windowの概要, Bit, Vol.19, No.3, pp.20-30. (1987.3).

[8] 有座 道春, 石曽根 信：Ｘウィンドウシステムの最新技術動向ーバーション１１を中心にしてー, Jus UNIX Fair'87 SEMINAR 資料, pp.71-100, 日本ＵＮＩＸユーザ会(1987.12).

[9] Rosenthal, D.: A Simple X11 Client Program -or- How hard can it really be to wirte "Hello World"?, Proc. Winter 1986 USENIX Conference,1988.

[10] Sun Microsystems: NeWS Manual, 1987.

[11] Sun Microsystems: NeWS Technical Overview, 1987.

[12] 金崎 克巳：ＮｅＷＳの概要，Bit，Vol.19,No.3，pp.41-48. (1987.3).
[13] 萩谷 昌巳：ＧＭＷウィンドウシステムについて，Bit，Vol.19, No.3，pp.31-40. (1987.3).
[14] Apple Documentation Group:Inside Macintosh, Addison-Wesley, 1985.
[15] Goldberg, A. and Robson, D.: Smalltalk-80: The Language and its Implementation, Addison-Wesley, 1984.
[16] Morris, J.H., Satyanarayanan,M., Conner, H.M. and Howard, J.H.:ANDREW: A Distributed Personal Computing Environment, CACM, Vol.29, No.3(1986),pp.184-201.
[17] Adobe Systems Inc.: PostScript Language Reference Manual, Addison-Wesley, 1986.
[18] OPEN LOOK--Graphical User Interface Functional Specification, Prerelease Version, AT&T, July 1988.