オペレーティング・システム　52－6
（１９９１．９．６）

ユーザインタフェース研究用ウィンドウシステム 未（HITSUJI）の設計と実現

河又恒久，小松徹，宮島靖，並木美太郎，高橋延匡
（東京農工大学　工学研究科　情報工学講座）

表示一体型タブレットを用いた手書きインタフェースは、ペンで直接操作ができるため、ジェスチャなどのマウスにはないユーザインタフェース（UI）を実現できる可能性がある。表示一体型タブレットでは、ペン入力に対して多くの処理を定義できるため、UIの設計に柔軟性が必要であると考える。

本報告では、ペン入力を指向し、UIの研究を行えるような、ウィンドウシステム「未」（HITSUJI）の設計と実現について述べる。このため、「未」は、UI管理の部分とデバイス仮想化の部分を分割して構築する。初版では、デバイス仮想化の部分を設計、実現し、UIを研究するための基盤を作成した。

Design and Implementation of the HITSUJI Window System for User Interface Experiments

Tsunehisa Kawamata,Tohru Komatsu,Yasushi Miyajima,
Mitaro Namiki,Nobumasa Takahashi

Department of Computer Science,
Tokyo Univ. of Agriculture and Technology
2-24-16 Naka-cho, Koganei-shi, Tokyo 184, Japan

Handwriting interface on a display integrated tablet may provide new interface through direct manipulation with a pen. Since pen action could be employed and interpreted in various ways, there are much wider variations in the interface design than with a mouse, so that design of user interface needs flexibility. This paper describes design and implementation of the HITSUJI window system which supports user interface design with pen and tablet. HITSUJI separates user interface management layer and device virtualize layer.

1．はじめに

グラフィカルユーザインタフェース（GUI）というと、ビットマップディスプレイとマウスを思い浮かべるが、最近では、液晶ディスプレイに透明タブレットを重ねたデバイス（表示一体型タブレット）が登場した。このデバイスは、入力と出力が同じ場所で行え、紙とペンを使用する感覚で計算機を操作することを目標としている。このような、手書きのインタフェースは、人間に自然であるため、GUIとして、有望な手段を提供してくれると思われる。

そこで、我々は、従来のビットマップディスプレイ＋マウスに加え、表示一体型タブレットを使用したGUIの研究を行うための環境として、ウィンドウシステム「未」（HITSUJI）を設計した。

本報告では、「未」の設計と実現について述べる。

2．UIの目標

ウィンドウシステムは、応用プログラム（AP）のユーザインタフェース（UI）構築環境を提供するものである。ここでは、「未」の目標とするAPのUIについて考える。

ビットマップディスプレイとマウスにより、急速にGUIが普及した。マウスは、ある程度人間に直感的で、操作しやすい環境を提供した。例えば、ファイルを消去する場合、ファイルを表したアイコンをドラッグし、ごみ箱アイコンの所で離すことで行える。しかし、ファイルの名前を変更する場合などは、メニューで“ファイル名変更”の項目を選び、キーボードで新たな名前を入力することとなる。このため、マウスは、基本的に、メニュー選択のデバイス（ポインティングデバイス）といえる。したがって、マウスを用いて、図や文字を入力するのは、困難な作業である。

一方、タブレットペン（ペン）を使用すると、マウスより、人間に直感的で、操作しやすい環境を構築することが可能となる。例えば、先のファイル名を変更する例の場合、アイコンの下に表示されているファイル名を二重線で消し、その下に手書きの文字を入力することで行える（図1－1）。また、ファイルの移動は、アイコンを丸で囲み、それを矢印で移動先のディレクトリウィンドウを指すことで行える（図1－2）。また、ペンを使用するので、文字や図形、絵の入力は簡単になる。

このため、表示一体型タブレットでは、つぎのようなAPが作成可能となる。

・原稿用紙ワープロ

ウィンドウを原稿用紙とし、その中に文字を書く。もちろん、原稿中に校正記号を書くことも可能にする［1，2］。

・文房具メタファを使用した図形入力環境

定規などの文房具メタファを用意し、それを使用して図や表を作成する［3］。

我々は、上記の例で、表示一体型タブレットの可能性を追求することを目的とする。また、その支援系として、「未」を開発することにした。

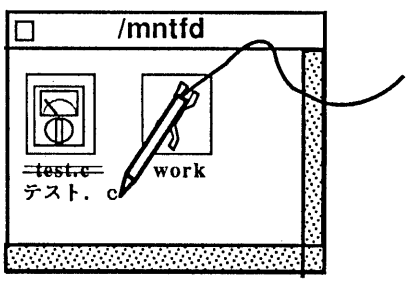

図1－1　ファイル名を変更する例

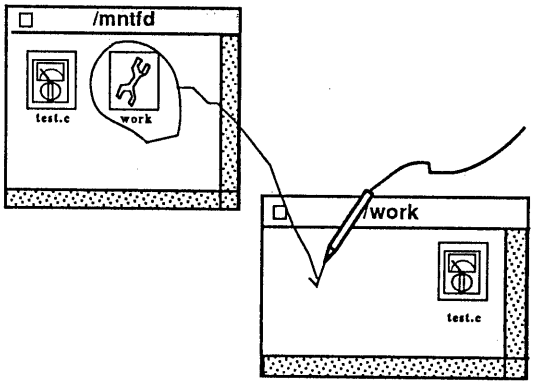

図1－2　ファイルを移動する例

図1　ペンを使った操作例

3.「未」の設計方針

上記の目標を達成するために、「未」の開発方針は次のとおりである。

(1) ペンを指向したウィンドウシステム

ペンを使用したGUI構築環境のためには、ウィンドウシステムはペン対応にする必要がある。

従来のマウス指向ウィンドウシステムで表示一体型タブレットを使うためには、マウスにはない処理、すなわち、筆点エコーバック、ペン入力認識などの機能を実現しなければならない。しかも、筆点のエコーバックは、リアルタイムの応答性能が本質的である。そのためには、ウィンドウシステムの大幅な改造が不可欠である。むしろ、タブレットペンの性能を活かすには、従来のウィンドウシステムを改良するというアプローチをとるよりは、ペンを指向したウィンドウシステムを開発すべきである。

(2) UI研究のためのウィンドウシステム

ペン入力だけでなく、マウスやキーボード入力のインタフェースは、現在のような体系で満足できるものとはいえない。例えば、キーボード入力先をどのウィンドウにするかという問題は、重なりの一番上のものか、マウスカーソルのあるものが一般的である。しかし、手書きインタフェースでは、状況に応じて、どちらも実現できる。メニューやダイアログに関しても同じことがいえる。

UIは、APの内容や対象とする人によって、異なるものである。そのため、UIを柔軟に構築できる機能がウィンドウシステムに必要である。

4.「未」の全体構成

ウィンドウシステムは、大きく分けて、入出力デバイスを仮想化する機能と、UIを管理する機能の二つのものから実現される。「未」は、これら二つの機能を分割して構築する。前者をウィンドウシステムカーネル（ウィンドウカーネル）、後者をUIマネージャと呼ぶ。全体構成図を図2に示す。

UI研究の目的から、APは、多種多様なUIを構築できなければならない。そのためには、ウィンドウシステムは、UIの管理に拡張性や柔軟性が必要である。例えば、タブレットペンのジェスチャが一つの方法とは限らない。文字の削除をするとき、横線二本で行いたいときもあれば、×印で行いたいときもある。そのため、UI管理の部分を分離する必要がある。

次に、各部分について述べる。

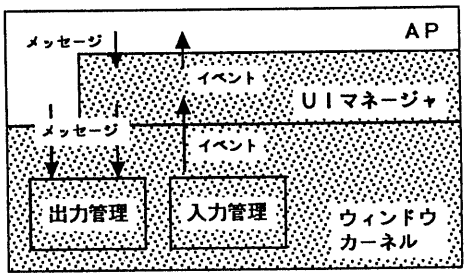

図2　未の全体構成

4.1 ウィンドウカーネル

ウィンドウカーネルは、入出力デバイスを仮想化する層であり、さらに次の二つに分けることができる。

(1) 出力管理

出力デバイスを出力資源に仮想化する部分である。オーバラップ方式のマルチウィンドウ機能を提供する。メニュー、アイコンなどUIに関する資源はUIマネージャで管理する。

(2) 入力管理

入力デバイスを仮想化する部分である。デバイスイベントとして、発生した順に入力データをAPやUIマネージャに伝える。

「未」が扱う入力デバイスは、キーボード、マウス、タブレットペンである。

4.2 UIマネージャ

UIマネージャは、UIを管理するための層である。ここでは、次のような機能を提供することを目標としている。

(1) UIに関する出力資源を管理する

- メニュー管理
- ダイアログ管理
- アイコン管理
- ウィンドウ管理

(2) 入力データを変換する
- イベント管理
- 仮名漢字変換
- 手書き文字認識
- 図形認識
- ジェスチャ認識

(3) 描画環境

5. APの入出力モデル

5.1 入力モデル

エンドユーザが行った入力は、正しくAPに伝えられるべきである。「未」は、このデータをイベントとして仮想化し、APへ知らせる。

ここでは、イベントについて述べる。

5.1.1 イベントの種類

イベントは、仮想化の度合いによって次の三種類のものが考えられる。

(1) 入力デバイスに密着したもの

例えば、ボタンが押された、放された、ペンが移動したというような、入力デバイスでのプリミティブ操作である。

(2) (1) を複数組み合わせたもの

(1) を複数組み合わせて仮想化したものであり、クリック（ボタンを押す、放す）、筆点列（ペン先のボタンを押しながら、ペンを移動）などが挙げられる。

(3) (2) の行われた場所が関係あるもの

(2) と、それが行われた場所を組み合わせて仮想化したものである。例えば、メニューの1番目の項目が選択された（メニューの1番目の項目でクリックされたとき）、アイコンが選択された（アイコンの場所を囲む筆点列）などがある。

状況によって、APが欲しいイベントの種類は異なる。例えば、グラフィック入力ツールなどは、(2) の筆点列が欲しい、また、メニューに対する操作結果は、(3) が欲しい。

「未」では、これらすべての場合に対応するようにしなければならない。(1) はウィンドウカーネル、(2) (3) はUIマネージャで管理する。それぞれ、デバイスイベント、ユーザイベントと呼ぶ。これらは、図3のように発生させる。イベント管理では、ユーザイベントとデバイスイベントの対応を管理する。

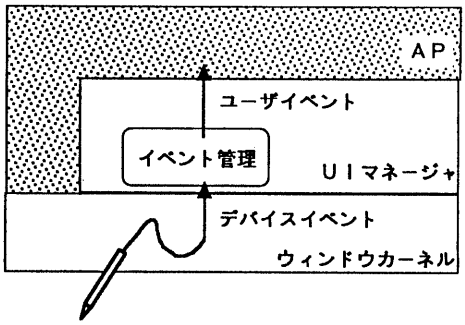

図3 デバイスイベントとユーザーイベント

5.1.2 イベントの得られる範囲

APは、発生したすべてのイベントが欲しいわけではなく、主に自分のウィンドウ内で発生したイベントが欲しい。

だが、それだけではなく、例えば、エディタが開いている複数ウィンドウ間で、文書の移動をジェスチャを使って行うと、ウィンドウ間のジェスチャを拾えない。

「未」では、背景もウィンドウとして扱う。そのため、ウィンドウ間ジェスチャを欲しいAPは、背景のウィンドウに対するアクセス権を持ち、そのウィンドウ内のイベントを得る。

5.2 出力モデル

APの出力モデルを図4に示す。ウィンドウカーネルでは、出力デバイスを出力資源に仮想化する。そこで、次のことを方針として、出力資源を決定した。

(1) ディスプレイの大きさに依存しないウィンドウ配置を実現する

(2) マルチディスプレイをサポートする

例えば、ペンとキーボードの入力をしているとき、ペンを使用しているときはタブレット画面、キーボードを使用しているときはディスプレイ画面を見たい。

(3) ウィンドウのデザイン、ディスプレイのデザイン、背景をユーザ定義可能にする（図5）。

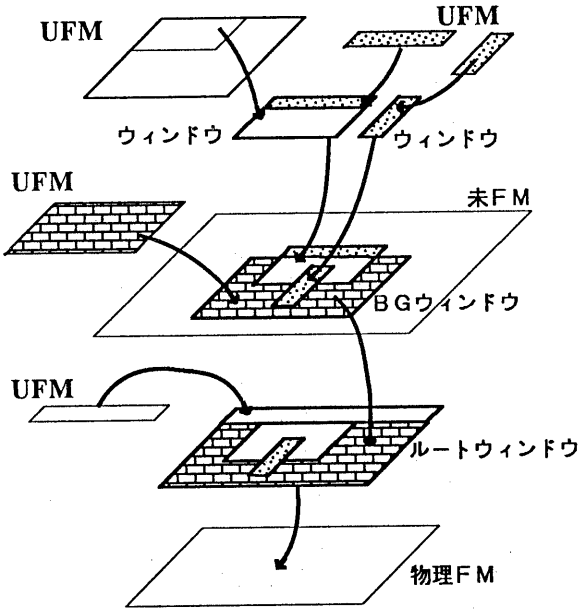

図4　出力モデル

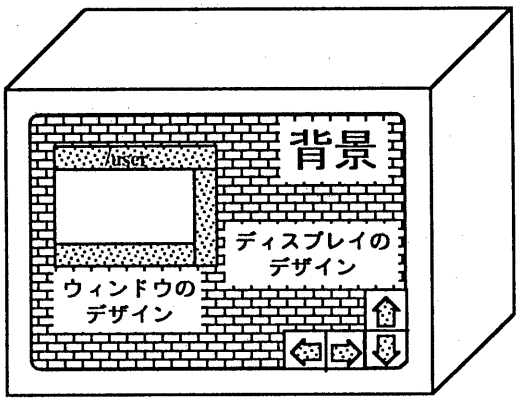

図5　ウィンドウのデザイン，
ディスプレイのデザイン，
背景

これらから、ウィンドウカーネルで扱う出力資源には、次の種類がある。

(1) 物理フレームメモリ（物理FM）
(2) 未FM
(3) ユーザFM（UFM）
(4) ウィンドウ

次にこれらの資源について述べる。

(1) 物理FM
ディスプレイ画面の内容を、実FMにビットマップで保持している資源である。

(2) 未FM
すべての出力資源に共通な、大きさ無限の座標系であり、仮想的なFMである。ディスプレイの大きさに依存しないウィンドウ配置と、マルチディスプレイを実現するために用意した（図6）。

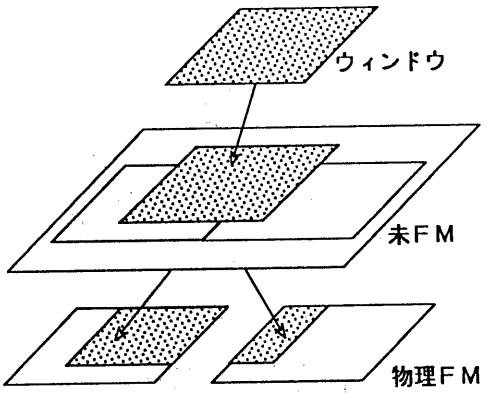

図6　未FM

(3) UFM
ウィンドウの内容を、実FMにビットマップで保持している資源である。APは、ここに描画する。UFMは、ウィンドウへマップすることができる。
資源の有効利用のため、一つのUFMを複数のウィンドウへマップできる。また、一つのウィンドウへ複数のUFMをマップすることもできる。これは、ウィンドウのデザインを行うのに有効である。

(4) ウィンドウ
ウィンドウは、出力デバイスへの表示単位である。ウィンドウに対しては、次の操作ができる。
・内容の全部を未FMへマップする

・大きさを変更する
・未FM内を移動する

ウィンドウには、次に示す、二つの特別なウィンドウがある。これらは、UFMがマップされる点を除いては、普通のウィンドウとは、異なった機能を持っている。

(4－1) バックグラウンドウィンドウ（BGウィンドウ）

背景表示のためのウィンドウである。未FMにマップされ、必ずウィンドウの重なりの一番下にある。BGウィンドウの内容をルートウィンドウへマップできる。BGウィンドウの内容とは、その上にあるウィンドウの内容である(図7)。

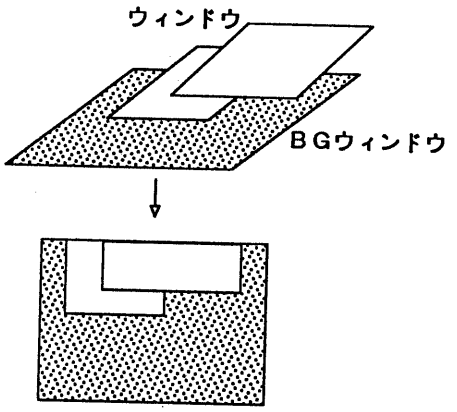

図7　BGウィンドウの内容

(4－2) ルートウィンドウ

ディスプレイ画面は未FMの一部である。ディスプレイ画面を未FM内で移動させるには、ウィンドウのようにディスプレイ画面のデザインが必要である。そのために、「未」では、ルートウィンドウを用意した。ルートウィンドウには、UFMとバックグラウンドウィンドウをマップでき、その内容の全部を物理FMへマップすることができる。

6．ウィンドウカーネル初版の設計

この章では、今回設計したウィンドウカーネルの初版について説明をする。

6.1　筆点のエコーバック

ペンを使用する場合、筆点のエコーバックが必要となる。エコーバックは、応答性能が重要であるため、OSレベルで行うことにした [4]。

6.2　UFMへの描画とカラー

APにとって、UFMへの描画は、描画コマンドで行いたい。現在、描画コマンドには、プリミティブな図形を描くものから、ページ記述言語まで多種にわたる描画形式が実現されている。そこで、初版では、これら多種にわたるコマンドを実現できるようにするため、ウィンドウカーネルでは、描画コマンドを扱わず、FMを用意して、ビットマップの描画を行う。そのため、描画コマンドは、UIマネージャで実現する（図8)。

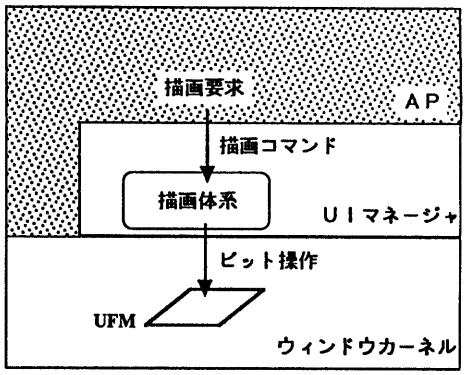

図8　描画モデル

初版では、表示一体型タブレットに重点を置いたため、カラーの問題は、次版以降での実現とした。しかし、我々は、カラーへの対応は、UIに必須であるという認識を持っている。将来は、描画コマンド、カラーともウィンドウカーネルで提供するつもりである。

6.3　プログラミングモデル

ウィンドウカーネルでは、資源をオブジェクトとみなし、APは、それに対してメッセージを発することで操作を行う。イベントは、ウィンドウカーネルからAPに対するメッセージとみることができる。ここでは、ウィンドウカーネルでのオブジェクトの種類（クラス)、オブジェクト操作の種類について述べる。

6.3.1 環境の導入

複数のウィンドウを同時に操作したいことはよくある。例えば、あるAPのウィンドウだけを移動したい場合など、複数のウィンドウをまとめて操作する仕組みが必要である。そこで、「未」では、環境という単位を設定する。環境は、複数のウィンドウの集合である。環境に対する操作は、環境に属するすべてのウィンドウに対する操作であり、環境に属するすべてのウィンドウに発生したイベントは、環境に対するイベントである。

環境には、APのウィンドウが使用するユーザ環境、ディスプレイ関連のウィンドウが使用するディスプレイ環境がある。

6.3.2 「未」が扱うクラス

「未」が扱うクラスは、次のとおりである。

ユーザ環境
ウィンドウ
UFM
未FM
BGウィンドウ
ルートウィンドウ
物理FM

6.3.3 オブジェクトに対する操作

APは、オブジェクトにメッセージを送ることで、オブジェクトの操作を行う。メッセージの種類と利用できるクラスの対応は表1のとおりである。

表1．1 メッセージの種類

メッセージ名	操作内容
Create／Delete	オブジェクトの生成／削除を行う
Open／Close	オブジェクトへのアクセス権を獲得／放棄する
Read／Write	オブジェクトの内容を読む／書く
Map／Unmap	オブジェクトをマップ／アンマップする
Move	オブジェクトを未FMの中で移動させる
Resize	オブジェクトの大きさを変更する
Top／Bottom	複数のオブジェクトの重なり順を変更する
Get_event	オブジェクトに発生したイベントを得る
Get_coordinate	座標を得る

表1．2 各オブジェクトに対する操作一覧表

クラス名	Create Delete	Open Close	Read Write	Map Unmap	Move	Resize	Top Bottom	Get_ event	Get_ coord
ユーザ環境	○	○	×	○*3	○*3	×	○*3	○*3	×
ウィンドウ	○	○	○*2	○	○	○	○	○	○
UFM	○	○	○	○	×	○	×	×	○
未FM	△*1	×	○*2	○	×	×	×	×	○
BGウィンドウ	○	○	○*2	○	○*4	×	×	○	○
ルートウィンドウ	○	○	○*2	○	×	×	×	○	○

物理FMに対しては何もできない。

○ 可能 ： △ 自動的に行われる ： × 不可能

注：*1 システム立ち上げ（立ち下げ）時に自動的に行われる。
*2 Read だけ可能である。
*3 それに属しているウィンドウに関するものを扱う。
*4 未FMの表示部分が移動する。

7．実現

「未」初版は、研究室独自開発のOS／omicronを使用して実現する。また、「未」は、OSの拡張として、第二版のユーザ拡張部に実装される。したがって、「未」へのメッセージは、SVC（Super Visor Call）を用いることになる。

現在、コーディング（入力関係は、3000ステップ、出力関係は、5500ステップ）が終了し、デバッグの段階である。開発は、OS／omicron上のC言語を使用した。

8．おわりに

表示一体型タブレットを有効に使用するため、UIの研究環境となるウィンドウシステムを設計した。「未」初版では、その基盤となるウィンドウカーネルを実現した。今後は、UIマネージャの設計と実現をする。

参考文献

[1]曽谷他：手書きユーザインタフェース第30回プログラミングシンポジウム，1990．

[2] 曽谷他：オンライン手書き文字認識系の使い勝手に関する考察，第32回プログラミングシンポジウム，1991．

[3] 風間他：手書きユーザインタフェースの研究～その図形への応用～，第42回情処全大，1991．

[4] 早川他：表示一体型タブレットの接続に対するOSの機能拡張の一方式，情処OS研究会資料，52－7，1991．