ヒューマンインタフェース 45−17
情 報 メ デ ィ ア 8−17
（1992. 11. 13）

表示一体型液晶タブレットを用いた“未”ウィンドウシステムの設計と実現

河又恒久、宮島靖、早川栄一、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では、ペンだけで、文字、図形など多くの種類の要素をフリーハンドで入力できる。これらを計算機で扱うには、各要素に対して認識系が必要となり、そのシステムへの組み込み方が問題となる。

本報告では、対象をジェスチャに絞り、ジェスチャ認識を組み込んだ“未”ウィンドウシステムの設計、実現について述べる。その特徴は、ジェスチャ認識を変更可能にするため、“未”とジェスチャ認識を分離し、ジェスチャ認識を組み込む枠組みだけを用意したことである。これにより、ウィンドウの操作やシェルのアイコンの操作に、ジェスチャを使用でき、ジェスチャ認識を研究、評価するための基盤ができた。

The Design and Implementation of HITSUJI Window System on a Display Integrated LCD Tablet

Tsunehisa Kawamata, Yasushi Miyajima, Eiichi Hayakawa,
Mitarou Namiki, Nobumasa Takahashi

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

Display Integrated LCD Tablets can input various types of data, such as characters or figures, with the use of a pen. It is necessary to have a recognition system to process this data, so the problem is the implemention of this recognition system. This paper describes the design and implementation of the HITSUJI Window System(HITSUJI) which has made possible the realization of gesture recognition. To allow modifications to the gesture recognition system, it has been seperated from HITSUJI and a framework has been prepared to contain this sytem. HITSUJI allows the manipulation of the window and icons with gestures, and provides a platform to research and evaluate gesture recognition.

1．はじめに

表示一体型液晶タブレットは、液晶ディスプレイ上に透明タブレットを重ねたデバイスである。これを用いた計算機の最大の特徴は、入力をスタイラスペン（以下ペン）で、出力と同一面に行うことである。そのため、従来のマウスやキーボード入力に比べて、次の利点がある。

(1) 紙とペンに近い感覚で使用可能のため、入力が容易であり、自然である。
(2) ペンだけで、文字、図形、絵、数式など、すべてフリーハンドで入力できる。
(3) 文字、図形などに対する編集、操作をジェスチャで行うことができる。

このように、ペンでは、人間に直感的な入力、操作方法のため、ペンを使用した応用プログラム（以下AP）は、従来のものとは異なるユーザインタフェース（以下UI）を実現できる可能性がある。例えば、実際に、当研究室では、次の研究を行っている。

ペン用ビジュアルシェル [1]

アイコンの操作などをジェスチャで行うシェルの研究と実現を行っている。

原稿用紙ワープロ [2]

原稿用紙を模したウィンドウに、文字を手書きで入力するワープロの開発研究を行っている。

文房具メタファを用いた図形入力環境 [3]

画面上の定規やコンパスなどの文房具メタファを使用して、図形を入力する環境の構築を行っている。

我々は、これらのAPを例にし、ペン入力のユーザインタフェース（以下UI）を研究するための基盤となることを目的として、ウィンドウシステム“未”（以下“未”）を作成した。

本報告では、“未”の設計と実現について述べる。

2．“未”の概要

“未”は、ユーザインタフェースを研究するという目的から、ウィンドウカーネル（以下カーネル）とUIマネージャの二層から構成されている。

カーネルは、ウィンドウの重なりと入力デバイスの管理を行う層である。

また、UIマネージャは、ウィンドウのアクセサリ、操作方法、APIなどウィンドウシステムのUIを提供する層である。

本章では、“未”のAPの入出力モデルについて述べる。

2．1 入力モデル

(1) イベント

イベントとは、ウィンドウ上で起こった入力に関するデータで、筆点列などのデバイスの生のデータや、ウィンドウの操作結果、また、筆点列入力をシステムで認識した結果のデータである。

“未”のイベントデータの流れを図1に示す。カーネル・UIマネージャ間のイベントをデバイスイベント、UIマネージャ・AP間のイベントをAPイベントと呼ぶ。デバイスイベントは、デバイスの生のデータであり、これをUIマネージャで、ウィンドウの操作や認識処理をした結果のイベントがAPイベントである。

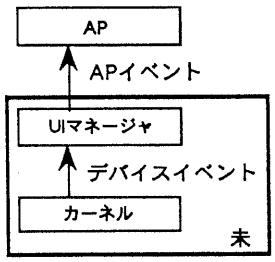

図1　イベントデータの流れ

(2) APの実行モデル

“未”では、APが、「このウィンドウのこの場所で、このイベントが発生したら、この関数を実行する」というイベント情報を登録して、それに該当するイベントが発生したら、指定した処理を実行するというモデルとなっている（図2）。これは、APが、詳細なイベントデータから内容を判定するという処理をなくすためである。このため、APは、必要なイベントだ

けを得られる。

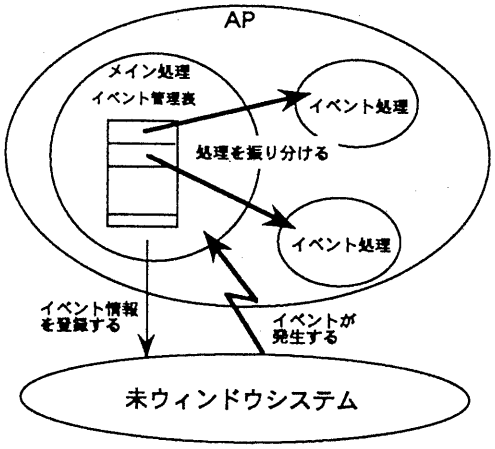

図2　APの実行モデル

2．2 出力モデル

(1) ウィンドウ

ウィンドウとは、“未”がAPに提供する一つの仮想画面である（図3）。この仮想画面に対して、次の四つの操作命令があればよい。つまり、生成とマップ、アンマップと消去の4種類である。生成でウィンドウを作成し、マップでウィンドウをディスプレイ上に表示する。アンマップ、消去は、その逆である。

“未”のウィンドウには、ウィンドウを操作するための要素をつけることができる。これらの要素をウィンドウのアクセサリと呼ぶ。アクセサリには、ウィンドウを操作するためのボタンやバーなどがある。“未”では、UIを研究する目的から、特定のアクセサリを用意するのではなく、自由にアクセサリを変更できる枠組みを提供した。

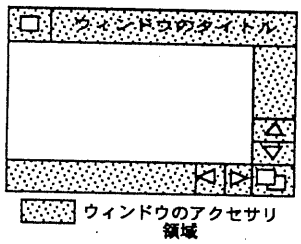

図3　ウィンドウ

(2) キャンバス

ウィンドウのアクセサリを除いた領域をキャンバス領域という。キャンバスは、APに与えられた文字どおり「キャンバス」で、APが何を描いても構わない。

“未”は、キャンバスの実体を持ち、その一部をウィンドウにマップするというモデルである。そのため、キャンバスの大きさは、ウィンドウの大きさに依存しない（図4）。

キャンバスへの描画は、専用の描画ライブラリで行う。

3．ペン入力システムの問題点

ペン入力の利点に、文字、図形などをフリーハンドで入力できることがある。例えば、‘あ’という文字を書いたとする。これが入力された時点では、計算機にとって、3本の筆点列でしかない。そのため、入力された筆点列をそのまま使用するなら問題はないが、これを計算機で、認識し、‘あ’と同定し、その後で、整形、編集するには、3本の筆点列を計算機に分かりやすい表現（内部表現）に変換しなければならない。‘あ’ならば、JISコードに変換することが考えられる。

このように、筆点列から計算機の内部表現に変換する処理を認識処理と言い、マウスのシステムにはないものである。

認識処理は、APで行うことと、システム側で行うことが考えられる。APで行う場合、各APに認識処理を組み込む必要があるが、これ

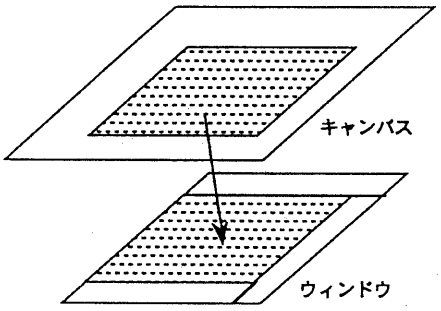

図4　キャンバスのマップ

は、APに負担がかかるだけでなく、各APによって認識の方法が異なるため、UIの一貫性が損なわれ、エンドユーザにも負担となる。

また、従来のウィンドウシステムを使用し、フロントエンドプロセッサ（以下FEP）で認識処理を行うシステムも考えられるが、この場合は、異なるウィンドウにまたがる筆点列を処理するのが困難である。なぜなら、このためには、FEPですべてのウィンドウの状態を把握していなければならないからである。これでは、ペン入力の一つの利点であるジェスチャを扱えない。

そこで、基本的な認識処理をウィンドウシステム側で扱い、APにそのインタフェースを提供する。

認識処理をするには、文字なら文字認識、図形なら図形認識のように、対象要素に対して、各々認識系が必要となる。なぜなら、各要素ごとに認識処理方法に特徴があるからである。そのため、認識処理には、次の二つの段階がある。

・対象分類処理
・認識処理

前者は、入力された筆点列を各認識系に振り分け、後者は、その筆点列を各認識系で認識し、計算機の内部表現に変換する。

ペン用のウィンドウシステムに、これらの処理をどのように組み込むかということは、重要である。なぜなら、これらの処理は、APインタフェース（以下API）とエンドユーザのUIの両方に影響を与えるからである。

そこで、“未”では、この点を問題点と認識し、設計をする。

4．“未”の設計目標

“未”では、文字、図形や、それを操作するためのジェスチャなど、さまざまな種類の入力をペンで行なう。これらの入力されたデータを計算機で扱うには、それぞれの認識系が必要となる。例えば、文章に数式を入力できるワープロならば、文字、数式、ジェスチャの三つの認識系が必要となる。

“未”では、これらの認識系を組み込み、APにそのインタフェースを提供し、手書き入力用のAPを容易に作成できるようにしたい。

そこで、まず第一段階として、ジェスチャに対象を絞って、ジェスチャ認識を扱うことを目標とする。ジェスチャを採用した理由は、次のとおりである。

・文字、図形など、すべての要素に対して、それらを操作する操作系が必要である。ジェスチャは、ペン入力の操作系であるから、ジェスチャ認識は、すべての認識系の基本となる。
・ジェスチャは、文字などに比べて種類が少ないため、比較的容易に認識系を作成することができる。

5．“未”の設計方針

4章で示した設計目標をもとに、“未”の設計方針を次のように定めた。

(1) ジェスチャ認識を容易に変更できるようにする

ジェスチャは、図形、文字など、操作対象によって異なる。例えば、改行を示すジェスチャは、文字入力特有のものである。

また、ジェスチャには標準がないため、一つの操作に複数のジェスチャを実現できる。例えば、選択を示すジェスチャは、チェックマークでもよいし、丸で囲むのもよい。

このように、“未”は、複数のジェスチャ認識を作成して、それらを実験、評価するための基盤としたい。

そこで、“未”は複数のジェスチャ認識を容易に変更可能にする。

(2) 入力された筆点列をジェスチャ認識へ振り分けるために、ジェスチャモードを導入する

“未”で、ジェスチャ認識を扱うためには、カーネルからの筆点列をすべて認識系に渡すか、ジェスチャを示すものだけを認識系に渡すかの二つの方法が考えられる。

エンドユーザの意図どおりに認識するには、前者の方法は、本質的に無理がある。例えば、囲みのジェスチャが存在するとし、閉じた形状の筆点列は、すべて囲みのジェスチャとして認識する。このとき、例えば、お絵描きツールで円を描いても、数字の0を書いても、すべて囲

みのジェスチャとなってしまう。一度、APに筆点列を渡して、APがジェスチャかを判断する方法もあるが、この場合もジェスチャと正確に判断することは難しい。これは、入力した筆点列の意味を知っているのは、入力した人（エンドユーザ）だけだからである。

そこで、“未”では、後者の方法を採用する。この場合は、先の理由から、カーネルからの筆点列がジェスチャである、とエンドユーザに明確な指示をしてもらわなければならない。そのため、ジェスチャモードを採用する。エンドユーザは、ジェスチャを入力するときに、ジェスチャモードで入力する。

6．“未”の設計

本章では、“未”の設計について述べる。

5章の方針にしたがって、“未”の構成を図5のようにした。この設計の特徴、各構成モジュールの機能について、次に述べる。

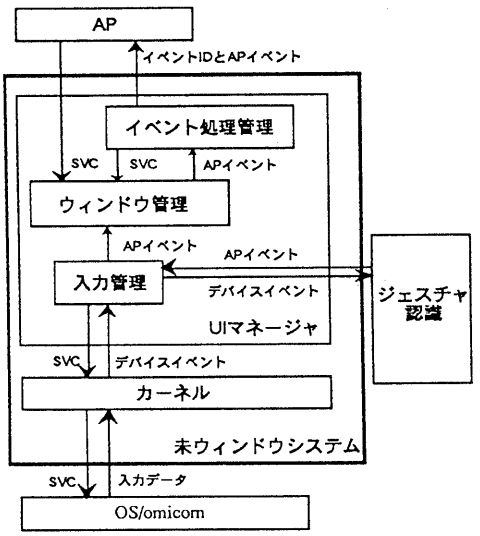

図5　“未”の構成

6．1 ジェスチャ認識の実行モデル

“未”では、複数の認識系を実現し、これを変更可能とするため、ジェスチャ認識を組み込まず、その枠組みだけを用意する。

“未”では、エンドユーザは、ジェスチャをジェスチャモードに切り換えて入力する。このため、“未”で、ジェスチャモードを管理する。“未”は、ジェスチャモードで入力された筆点列をジェスチャ認識に渡す。ジェスチャモードでない場合は、そのまま処理を続ける。ジェスチャ認識は、筆点列を認識し、その結果を“未”に返す。“未”は、それをイベントデータとして扱い、処理を続ける（図6）。

このように、“未”は、ジェスチャの種類や形状といったジェスチャ認識に依存している処理を行なわない。これにより、複数のジェスチャ認識を実現し、容易に変更可能な構造となった。

認識結果は、具体的には、APイベントであり、そのデータ構造は、あらかじめ“未”で決定する。しかし、図形、文字など操作対象により、その操作も異なるため、ジェスチャの種類に依存しないものとする。

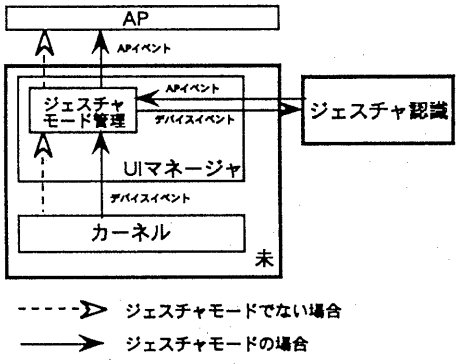

図6　ジェスチャの認識モデル

6．2 ジェスチャ認識とのインタフェース

ジェスチャ認識を変更可能にするため、“未”では、ジェスチャ認識を組み込むための枠組みだけを提供する。そこで、“未”とジェスチャ認識とのインタフェースを決定しなければならない。具体的には、これらの間でやりとりが行なわれるデータ構造を決定する。

“未”とジェスチャ認識は、次のデータをやり取りする。

“未”→ジェスチャ認識

筆点列のデバイスイベント

ジェスチャ認識→“未”
ジェスチャのAPイベント

この二つのデータが、“未”とジェスチャ認識の入出力データであり、それらのインタフェースとなっている。そして、デバイスイベントからAPイベントへの変換系が、ジェスチャ認識である。このため、“未”では、筆点列のデバイスイベントとジェスチャのAPイベントのデータを明らかにする必要がある。これらのデータ構造を、表1に示す。

“未”では、複数のジェスチャを実現できるため、ジェスチャ認識やそれを組み込んだ有効性、また、ジェスチャを使ったウィンドウの操作性などを評価をしなければならないと考える。

これらの評価をするには、ユーザの行動、すなわち、イベントのログデータを採ることが有効と考えた。そこで、“未”では、すべてのイベントデータに時間を入れ、イベントデータからログを採れるようにした。

表1-1　デバイスイベントのデータ構造

データ名	サイズ(BYTE)	内容
サブスイッチの状態	2	ペン横についているスイッチの状態
ディスプレイID	2	筆点列が入力されたディスプレイID
座標	4	筆点列の開始点のディスプレイ座標
ウィンドウID	2	筆点列の開始点があるウィンドウID
ウィンドウ座標	4	筆点列の開始点のウィンドウ座標
キャンバス座標	4	筆点列の開始点のキャンバス座標
筆点数	2	筆点列の筆点数
開始点	4	筆点列の開始点のタブレット座標*
中間点	4×最大筆点数	開始点以外の筆点のタブレット座標（開始点からの差分座標）
筆点最大領域	8	筆点の存在する最大矩形領域（開始点からの差分座標）

*タブレット座標　タブレットと画面の解像度が異なることから生じる入出力座標の違いをなくすために設けた座標系の座標である。ディスプレイ座標を8倍したものに相当する。

表1-2　APイベントのデータ構造

データ名	サイズ(BYTE)	内容
イベントの種類	2	ジェスチャなどのイベントの種類
イベント名	2	ジェスチャならば、移動などのイベントの名前
ウィンドウID	2	イベントの発生対象となるウィンドウID
イベントデータ	イベントの種類に依存する	イベントの種類に固有のデータ ジェスチャなら、範囲指定と元先のデータ

6．3 ジェスチャモード

“未”では、ジェスチャはジェスチャモードで入力する。ジェスチャモードへの切り換えは、エンドユーザに行なってもらう。そのため、モードの変更方法が問題となる。なぜなら、モードの変更は、エンドユーザにとって煩わしいことだからである。したがって、なるべくユーザにモード変更を感じさせない方法にする必要がある。モード変更には、多くの方法が考えられる。その一例を次に示す。

・あらかじめ、ジェスチャアイコンをペンで触れる
・ペンが二つあるなら、ジェスチャ用のペンに持ち替える
・ウィンドウなどにジェスチャを入力する場所を用意し、そこに入力する

どの方法がエンドユーザにとってよいのかは、まだわからない。したがって、“未”では、ペンの横についている二つのスイッチのうち、一方を利用する。ペン横のスイッチを押しながら筆点列を入力したときは、ジェスチャモードでの入力とする。将来的には、ジェスチャモード切替えの方法を変更して、さまざまな方法を試せるようにする。

6．3 各モジュールの機能

ここでは、“未”の各モジュールの機能の説明をする。

(1) 入力管理部

入力管理部の機能は、次のとおりである。

・ジェスチャモードの管理
・カーネルからの筆点列の振り分け
・APイベントの作成

(2) ウィンドウ管理部

ウィンドウ管理部の機能は、次のとおりである。

・APからのウィンドウ操作の解釈
・ウィンドウのアクセサリ管理
・入力管理部からのAPイベントの解釈とウィンドウの操作

(3) イベント管理部

イベント管理部の機能は、次のとおりである。

・APの登録したイベント情報の管理
・APイベントの振り分け

7．実現

“未”は、現在、カーネル［4］が実現されている。今回は、6章の設計から、UIマネージャを作成し、“未”第一版（“未”初版）を実現した。“未”初版は、ウィンドウの操作やシェルのアイコン操作にジェスチャを使用するため、簡単なジェスチャ認識を作成し、“未”初版に組み込んだ。ここでは、“未”初版の実現について述べる。

7．1 ジェスチャ認識

7．1．1 扱うジェスチャの種類

“未”初版で扱うジェスチャの種類は、次の方針で決定した。

・対象は、ウィンドウやシェルのアイコンの操作とする
・APにも使えるように、ウィンドウやアイコンの操作に特化しない

これらの方針から、“未”初版では、文章編集などのジェスチャを採用せず、次の6種類のジェスチャを扱う。

・コピー
・削除
・選択
・スクロール
・範囲指定
・移動

7．1．2 ジェスチャ依存のデータ構造

認識結果のデータには、ジェスチャに依存したデータがある。これらのデータ構造を表2に示す。

7．1．3 ジェスチャ認識の実現

プロトタイプとして、すでに述べた6種類のジェスチャを認識する認識系を作成した。この認識系で入力できるジェスチャを表3に示す。

ジェスチャの形状を変更したい場合は、ジェスチャ認識を変更するだけでよい。

表2-1　ジェスチャのデータ構造（範囲指定）

データ名	サイズ(BYTE)	内容
ディスプレイID	2	ジェスチャの入力されたディスプレイID
ディスプレイ注目点	4	ジェスチャの注目点があるディスプレイ座標
左上方向範囲	4	ジェスチャが示す矩形範囲 （注目点からの右下方向への座標の差分）
右下方向範囲	4	ジェスチャが示す矩形範囲 （注目点からの左上方向への座標の差分）
キャンバス注目点	4	ジェスチャの注目点があるキャンバス座標
ウィンドウ注目点	4	ジェスチャの注目点があるウィンドウ座標

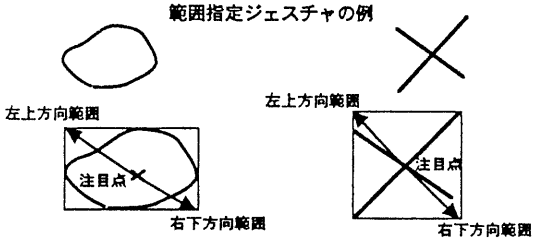

表2-2　ジェスチャのデータ構造（先元）

データ名	サイズ(BYTE)	内容
ディスプレイID	2	ジェスチャの入力されたディスプレイID
ディスプレイ元座標	4	操作元のディスプレイ座標
ディスプレイ先座標	4	操作先のディスプレイ座標
キャンバス元座標	4	操作元のキャンバス座標
キャンバス先座標	4	操作先のキャンバス座標
ウィンドウ元座標	4	操作元のウィンドウ座標
ウィンドウ先座標	4	操作先のウィンドウ座標
先ウィンドウID	2	操作先のウィンドウID

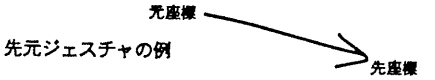

表3　ジェスチャ

種類	イベントデータ	ジェスチャ	備考
コピー	元先		どの方向へ描いても構わない 描きはじめが操作元である
削除	範囲指定		どちらから描きはじめても構わない
選択	範囲指定		1 2 左回の順番で描くこと
範囲指定	範囲指定		どこからでも、どの方向へ描いても構わない
スクロール	---	-----	用意していない
移動	元先		コピーと同じである 状況によってAPが判断する

7．2 ウィンドウの操作

実験として、ウィンドウの操作にジェスチャを採用した。“未”初版では、アクセサリに対してジェスチャを入力するとウィンドウの操作

を行う。ジェスチャとウィンドウの操作の対応を、表4に示す。

表4　ウィンドウの操作とジェスチャの対応

操作内容	操作開始場所	ジェスチャ
移動する	タイトルバー上	移動
リサイズする	リサイズボックス上	移動
アンマップする	アンマップボックス上	選択
重なりの一番上にする	入れ換えボックス上	選択
スクロールする	スクロールバー上	選択または移動
背景をスクロールする	上記の場所以外	スクロール

7．3 実現環境

“未”初版は、研究室独自開発のOS／omicronを使用して開発、実現した。

“未”初版は、OS／omicron第三版のデバイスドライバとして実現した。APから“未”初版への命令は、OS／omicron第三版へのSVC（Super Visor Call）となる。

ジェスチャ認識も、OS／omicron第三版のデバイスドライバとした。デバイスドライバは、コマンドで追加や削除が可能なので、ジェスチャ認識の変更が容易となっている．

また、開発は、OS／omicron上のC言語CATを使用し、カーネルは16000ステップ、UIマネージャは9000ステップとなっている。

“未”初版の実行画面を図7に示す。

8．おわりに

我々は、ペン入力のUIを研究するための基盤として、“未”を作成した。今回は、“未”初版として、ジェスチャ認識を組み込むための枠組みを設計、実現し、ジェスチャをウィンドウの操作やシェルのアイコン操作に使用することができただけでなく、ジェスチャ認識を研究、評価するための基盤を作成した。

今後の課題として、次のことを考えている。

（1）評価を行う

今回の設計で、“未”初版の評価を行うためのデータをイベントにいれた。これからは、評価項目を定め、イベントデータを収集し、評価を行う。

（2）APを作成するための環境整備

ペンを使用したUI構築ツールなどのペン入力用APを作成するための環境を整備する。

参考文献

[1]　宮島他：表示一体型液晶タブレットを用いたビジュアルシェル，情報処理学会第44回全国大会7K−2，1992

[2]　曽谷他：遅延認識方式による手書き原稿作成プロトタイプシステム，情報処理学会ヒューマンインタフェース研究会43−4，1992

[3]　風間他：文房具メタファを用いた手書き作図インタフェース，情報処理学会ヒューマンインタフェース研究会43−3，1992

[4]　河又他：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，情報処理学会オペレーティングシステム研究会52−6，1992

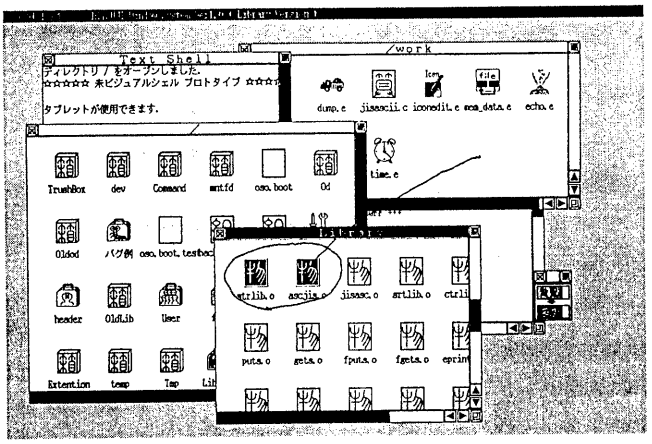

図7　“未”初版の実行画面