ペンインタフェース研究・開発のためのウィンドウシステム“未”（HITSUJI）の設計と実現

早川栄一[†] 河又恒久[††] 宮島靖[†††]
加藤直樹[†] 並木美太郎[†] 高橋延匡[†]

本論文は，ペンインタフェース研究・開発を対象としたウィンドウシステム“未”（HITSUJI）の設計と実現について述べたものである．ペンを用いた計算機インタフェースは，新しいデバイスであることと自由度が大きいことから，システムソフトウェアにおいて，適切な仮想化が必要である．この問題に対して，我々は計算機システムにおけるユーザインタフェースの基盤であるウィンドウシステムに着目し，ペンインタフェースの研究・開発の支援を行うウィンドウシステムである“未”の設計と実現を行った．“未”では，ペンに特有な処理として，認識処理，特にジェスチャ処理に着目して，次の機能を提供した：（1）ジェスチャのシステムインタフェースとしての提供，（2）ジェスチャの誤認識による誤操作に対応するため，システムで統一したアンドゥ処理の提供，（3）ジェスチャによってウィンドウ間でのデータ通信が頻発することから，ウィンドウ間でデータ通信を行う機構の提供．また，ウィンドウシステムの構成要素を自由に変更可能にして，ペンインタフェース研究基盤として用いる．この結果，ペンインタフェースを対象としたウィンドウシステムにおける認識処理のインタフェースを明確にし，アンドゥ処理についても，統一的なインタフェースを提供することが可能になった．

Design and Implementation of “HITSUJI” Window System for Pen Interface Research and Development

EIICHI HAYAKAWA,[†] TSUNEHISA KAWAMATA,[†] YASUSHI MIYAJIMA,[†]
NAOKI KATO,[†] MITAROU NAMIKI [†] and NOBUMASA TAKAHASHI [†]

This paper describes the design and implementation of a window system named “HITSUJI” for supporting research and development into pen interfaces. An interface based on a pen requires appropriate virtualization in its system software as it is a new device which allows a greater degree of freedom than existing interfaces. In an attempt to meet this requirement, we focus on a window system which forms the foundation for the user interface of a computer system architecture. From this vantage point, the “HITSUJI” window system was designed and implemented. The recognition process (especially gesture recognition) is focused on in “HITSUJI” as a pen specific feature, and we present the following features: (1) gestures as a system interface, (2) a system unified undo interface in order to resolve misoperations resulting from recognition errors, (3) an inter-window data transfer mechanism operated by gestures. Many parts of this window system can be freely adapted to function as an environment for the research of pen interfaces. This work has defined an interface for recognition processes, and made possible the implementation of an undo feature, unified throughout the pen interface oriented window system it was developed in.

1. はじめに

表示一体型タブレットは，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．このデバイスの特徴は，スタイラスペン（以下ペン）で入力を行い，出力を入力と同一面に行えることである．これを使った計算機は，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操作ができる．

（2）文字，図形，絵，数式など多様なデータをペンだけで入力できる．

（3）文字，図形などに対する対象の指定と操作を直

† 東京農工大学工学部電子情報工学科
Department of Computer Science, Faculty of Engineering, Tokyo University of Agriculture and Technology
†† 日本電気株式会社
NEC Corporation
††† ソニー株式会社
Sony Corporation

接行うことができる．

これらは，従来のマウスやキーボードなどのインタフェースにはない特徴であり，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．このインタフェースをペンインタフェースと呼ぶ．

ペンインタフェースは，キーボードやマウスと比較して，新しいデバイスである．キーボードに関する仮想化は古くから行われてきた．また，マウスとビットマップディスプレイは，ワークステーション上で，指示と選択を中心として仮想化されてきた．これに対して，ペンと表示一体型タブレットでは，対象に対して直接操作が可能であり，また，多様なデータを入力できることから，設計の自由度が大きい．このことから，ソフトウェアアーキテクチャに対して，ペンインタフェースは大きな影響を与える．

特に，システム設計の観点からすれば，ペンインタフェースに対して，ウィンドウシステムのソフトウェアアーキテクチャをどのように設計するかが重要な問題となる．それは，ウィンドウシステムは，ハードウェアの入出力資源の管理を行い，さらに，上位層に対しては UIMS（User Interface Management System）などのユーザインタフェースの管理を行ってきたからである．

ウィンドウシステムアーキテクチャとペンインタフェースとの関わりで，従来の研究をとらえると，二つの方式が存在する．一つは，従来構成にペン用の機構を追加したもの，もう一つは，ペンインタフェースからシステム構成やデザインを設計しているものである．

前者には，Windows for Pen Computing[1]，X Window System 上での実現[2] が存在する．これらは，マウス用に作成された多くのアプリケーションを容易に移植することができるが，操作インタフェースやウィンドウデザインの点で，ペンインタフェースの特徴を生かすことができない．また，マウス機能に対する拡張という形で実装されていることから，利用できる場面が限定される．

後者の代表としては，PalmTop[3]，PenPoint[4] がある．これらは，ノートブックのメタファを提供する NBI（Note Book Interface）やジェスチャなど，ペンインタフェースを中心に操作やデザインが設計されている．しかし，メタファによってウィンドウシステムの機能が規定されていること，図形や数式といった他の認識系を組み込む枠組みを提供していないことの二つの理由から，ペンインタフェースの研究基盤として用いることは難しい．

我々の研究グループでは，これまで手書き原稿作成システム[5]，文房具メタファを備えた図形入力システム[6] などのペンインタフェースに関する研究を行ってきた．その結果，我々は，ペンインタフェースの研究，AP の開発の共通の基盤としてのウィンドウシステムの必要性を認識した．特に，ウィンドウシステムアーキテクチャからペンインタフェースを考えると，認識系の扱い，ペンによる資源操作機構の提供が重要な問題となる．

そこで，我々は，ペン入力の UI を研究する基盤としてのウィンドウシステムアーキテクチャの設計を目的として，ウィンドウシステム“未”（以下“未”）を作成した．

“未”の特徴は，ペンによる資源操作方式の一つであるジェスチャや認識処理のプログラムインタフェースを備えていること，アンドゥのプログラムインタフェースを実現したこと，ウィンドウ間の通信機構を提供したことにある．

本論文では，ペンインタフェースを指向したウィンドウシステムである“未”の設計と実現について述べる．

2. ウィンドウシステムアーキテクチャへの要求

本章では，ペンインタフェースを指向したウィンドウシステムアーキテクチャを設計するのに必要となる要求事項について分析する．

2.1 ジェスチャ

ペンインタフェースに特徴的な操作としてジェスチャがある．ジェスチャは，ペンデバイスのアクションによって，資源操作の指定を行うことである．ジェスチャは，操作を覚えなければならないが，ペンだけで計算機資源に対する操作が可能になる．

これをシステムソフトウェアから見た場合，ジェスチャをどのような機構で取り入れるか，さらに，どのようなプログラムインタフェースにするか，ジェスチャと通常の筆点列とをどのように区別するかが問題となる．

システムへの追加の容易さから考えると，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（FEP）による日本語入力の枠組み[7] を用いて認識処理を行うことが考えられる．しかし，この場合，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，ウィンドウ間の操作があるために，FEP の内部ですべてのウィンドウの情報を管理して

いなければならないからである．

ペン用のウィンドウシステムに，ジェスチャをどのように組み込むかということは，アプリケーションプログラムインタフェース（以下API）とエンドユーザのUIの両方に影響を与える重要な問題である．

2.2 認識処理

ジェスチャ以外の認識，例えば，文字認識，図形認識，数式認識なども，ペンにおいては重要な要素である．認識処理は筆点列とコードとの変換系であり，これによって，ペンによる入力をコード化することが可能となる．

認識処理を導入するには，ジェスチャと異なった筆点列の区別をする機構が必要になる．ペンから来るデータは筆点列でしかなく，これを振り分ける処理が必ず入るからである．そこで，認識処理とシステムとのプログラムインタフェースを明確に決め，複数の認識処理の追加や変更を容易にする必要がある．

2.3 ウィンドウ間でのデータ交換

ジェスチャでは，画面上のすべての資源を操作できることから，マウスやキーボードに比べて，ウィンドウ間でのデータ交換を自然に行うことができる．アプリケーション内でのウィンドウ間でのデータ交換と，異なったウィンドウ間でのデータ交換とを区別なく扱える．そのために，ウィンドウ間でのデータ交換のための通信機構を提供する必要がある．

これをウィンドウシステム側から見ると，複数のAPのプロセスとウィンドウとの関係を把握しつつ，データのやりとりを制御することになる．

2.4 アンドゥ処理

操作を取り消し，操作前の状態へ戻すアンドゥの機構は，UIにおいて一般的な機構となっている．これによって，操作の寛容性が生じ，より使いやすいシステムにすることができる．その一方で，システムプログラムは，アンドゥ機構を規定していない．それは，何を，どのようにアンドゥするかは，AP側にしか分からないからである．多くのシステムでは，操作の統一はAP側でのプログラミングによって保証してきた．

しかし，ジェスチャを含む認識処理が必要である．認識処理には誤認識の可能性があり，操作に対するやり直し（アンドゥ）の処理が重要になる．さらに，2.4節で述べたように，ウィンドウ間でのデータ交換を容易にすることは，アンドゥ処理を難しくする．つまり，ウィンドウ間の関係を見なければならず，AP単体でアンドゥ処理をすることはできないからである．

また，ジェスチャによって，ウィンドウの移動，生成，消去といった，システム資源の操作が可能になると，システム内部の資源もチェックしなければならず，単純にAP側で実現するだけでは，アンドゥインタフェースのプログラミングが面倒になり，容易にアンドゥを実現することはできない．

システムアーキテクチャとしてどのような機構を実現すべきか，また，処理とプログラムインタフェースをどのように分離するかが問題である．

2.5 ユーザインタフェース管理

現在，ペンインタフェースを用いたシステムは数多く出現しているが，マウスやキーボードと比較して，ペンインタフェースは固定化していない．これは，ペンインタフェースの一般的なモデルがまだ存在していないものと思われる．

このような状況では，システムソフトウェアで操作モデルや，描画モデルを規定するべきではない．現時点では，むしろこのような管理機構に対する実験を容易に実現できるような環境が必要であり，容易に変更，実験が可能な環境を構築すべきである．例えば，モジュール設計を進め，ユーザインタフェース管理の部分の変更を容易にすることや，ユーザインタフェースに関するデータを採る機構が必要となる．

2.6 “未”の設計方針

以上の考察に対して，通常のウィンドウシステムでは対応することができない．そこで，我々はペンインタフェースをターゲットとしたウィンドウシステムを作成することにした．このウィンドウシステムを“未”と名付けた．

“未”の設計方針を次のように定めた．

(1) ペンインタフェースに特徴的なジェスチャを扱える機構を用意すること．
(2) 認識処理に対する，システムインタフェースを提供すること．
(3) ジェスチャなどの認識処理に対応したアンドゥ機構を提供すること．
(4) ウィンドウ間の通信機構を用意すること．
(5) ペンインタフェースの研究・開発に必要となる，実験，評価が容易な環境とすること．

3. “未”の特徴

“未”の特徴は次のとおりである．

(1) ジェスチャ処理のプログラムインタフェースをウィンドウシステムのアーキテクチャの一部として実現する．

ジェスチャ処理機構は，ペンのウィンドウシステムにおいて必須の機能であると考え，ウィンドウシステムに組み込む．また，ペンインタフェースにおいて，

ジェスチャはすべての資源を操作するメタな操作であると考え，ペンによるデータ入力と容易に区別可能なインタフェースを設ける．

(2)認識処理のプログラムインタフェースを統一する．

ジェスチャの実装には必然的に認識系を伴うことから，認識系とのインタフェースが重要な役割を果たす．認識系とのインタフェースだけを決め，モジュールは独立させることで，ジェスチャ認識系のデバッグや機能追加を容易にする．

また，モジュールの独立性を高め，実験を容易にできるように，ジェスチャ認識，文字認識などすべての認識系で，システムとのプログラミングインタフェースを統一する．

(3)通信のイベント発生時に通信相手を特定できる通信機構を実現する．

ペンインタフェースでは，ジェスチャのように，ウィンドウをまたぐ操作が発生する．この場合，通常の通信とは異なり，対象となるウィンドウは，ユーザの操作によって初めて決定するものである．

このようにペンインタフェースを持つウィンドウシステムでは，ウィンドウアプリケーションの間の通信は非同期的に行われるので，このような通信機構を実現する必要がある．

そこで，“未”では，ウィンドウ間の非同期の通信機構を実現する．この機構をウィンドウ間通信と呼ぶ．

(4)ウィンドウシステムで，アンドゥ・リドゥに関するプリミティブを管理し，システムで統一的なインタフェースを提供する．

ウィンドウシステム内部にアンドゥ管理機構を実現して，システムで一貫したアンドゥ・リドゥ操作を可能にする．特に，システム内部では，アンドゥ・リドゥを行う枠組みだけを用意することにする．

筆点列をプリミティブとして，ウィンドウシステム内部で管理する．これによって，どのAPにどのアンドゥイベントを送るかは，ウィンドウシステム側で行い，実際のアンドゥ操作は，イベントを受け取ったアプリケーションが行うことになる．これによって，ウィンドウシステムで管理する資源と，APで管理する資源を分離することができる．

(5)実験環境として用いることから，ウィンドウ自身の管理，デザインの管理と他のサービスの部分をそれぞれ分離する．

物理的なデバイスの管理，ウィンドウの管理と，ウィンドウデザインは独立させることができる．また，認識処理部もデザイン管理とは独立した設計にする．

さらに，内部をモジュール化して実現することで，ペンインタフェースの研究環境として，各処理部分を変更しやすくする．

(6)ユーザ行動の評価のための枠組みを提供するために，イベントのログを採れるようにする．

“未”では，ジェスチャ認識やそれを組み込んだシステムの有効性，また，ジェスチャを使ったウィンドウの操作性などの評価をしたい．

それには，まず，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である．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する．

4. “未”の基本構造

本章では，UIマネージャが提供する“未”の全体構成と，AP入出力モデルについて述べる．

4.1 “未”の全体構成

“未”の全体構成を**図1**に示す．“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とUIマネージャの二層から構成した．

ウィンドウカーネルは，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，スクロールバーやラベルなどウィンドウのデザインに関係する部分や，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．このように二層に分離することで，ハードウェアの資源管理とユーザインタフェースに関する部分を分離して，機能の変更や追加などを容易にした．さらに，UIマネージャを構成するモジュールは，独立した複数の管理モジュールによって構成して，機能変更を容易にした．ジェスチャや文字などの認識系についても，ウィンドウシステムとは別のモジュールとして実現することで，追加，変更を容易にする．

4.2 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．

ペンのイベント発生頻度は，マウスに比べて多い．我々のシステムでは，1秒間に約120点発生する．これを各点についてイベントとして，APに渡していては，システムの実行オーバヘッドが大きくなる．そこで，筆点列イベントを定義する．これは，ウィンドウカーネル側で，ペンを画面につけてから離すまでの筆点の集合をイベントの発生単位とする．我々は，ペンで入

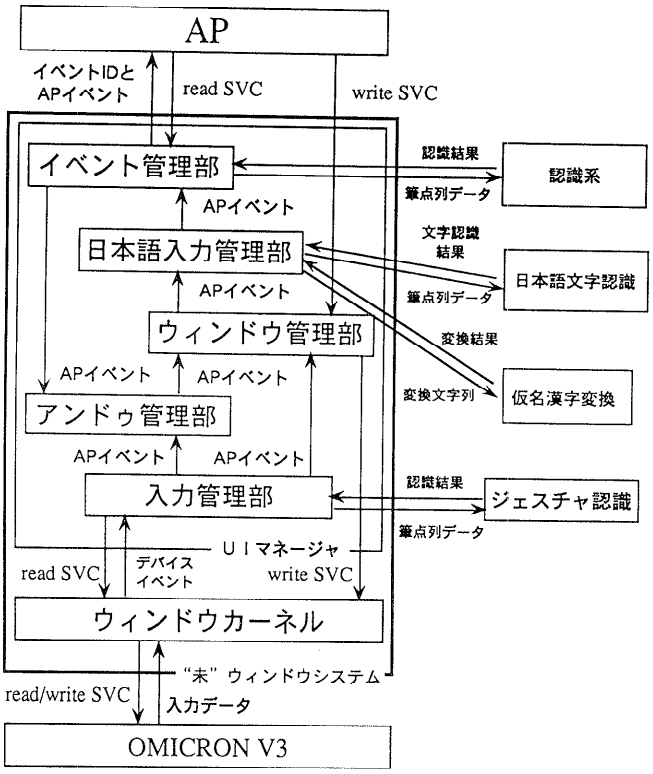


図1 "未"の全体構造
Fig. 1 Overall structure of HITSUJI.

力するデータとして，文字，図形，ジェスチャを想定している．これらは，筆点列という単位で表現することができ，また，そのほうがデータを扱いやすいからである．さらに，"未"では，デバイスイベント，APイベントという，二つのイベントの種類を用意する．前者は，デバイスの生のデータを表すデバイスイベントであり，後者はUIマネージャ内部で，より扱いやすい形に加工されたイベントである．

（2） APのイベントモデル

"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を提供する．これによって，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記述する必要がなくなり，また，APは，必要なイベントだけを得られる．

4.3 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．"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．ペンに関するアクセサリを研究を容易にするために，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．

（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呼ぶ．ペンインタフェースでは，ユーザはどのようなデータでも描くことが可能である．そのために，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．

5. 認識処理のインタフェース

本章では，"未"が提供する認識処理のインタフェースについて述べる．

5.1 認識の実行モデル

“未”では，複数の認識系を実現し，これを変更できるように，UIマネージャに組み込まず，その枠組みだけを用意する．

“未”は，認識対象となる筆点列（デバイスイベントで表現する）が入力された場合，それを認識処理部に渡す．通常の筆点列の場合は，そのまま処理を続ける．認識処理部は，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（APイベントで表現する）として扱い，処理を続ける．

このように，“未”は，認識系に依存した処理，例えば，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わない．これにより，複数の認識を容易に変更可能な構造となった．認識処理部とUIマネージャとの間のインタフェースは，あらかじめ“未”で決定する．ただし，図形，文字など操作対象により，その操作も異なることから，ジェスチャの種類に依存しないものとする．

5.2 認識処理とのインタフェース

“未”では，認識処理を組み込む枠組みだけを提供する．そこで，“未”と認識処理とのインタフェースを決定しなければならない．これは，これらの間でやりとりが行われるデータ構造を決めることによって実現する．

“未”は，認識処理に対して，デバイスイベントを渡し，その結果として，ジェスチャのAPイベントを受け取る．認識処理は，この二つのイベントの間の変換系として位置づける．これによって，この二つのイベントのデータ構造が，“未”と処理認識との間のインタフェースを規定する．このデータ構造を**表1**に示す．

ジェスチャの処理については，入力管理部から呼び出すことで，ウィンドウを含む資源操作を可能にする．

筆点列のジェスチャと入力との切替えについては，実現の容易さから，初版では，ペンに付いているスイッチを用いる．ジェスチャは，資源に対するメタな操作である．これによって，入力と操作とを容易に分離でき，ジェスチャによる資源操作が可能になる．

ジェスチャ以外の認識処理は，対象に適したモジュールから呼び出される．例えば，図形認識などをシステムのAPIとして実装するには，入力の領域を管理するイベント管理部で実現する．この場合，一つのウィンドウへの筆点列の入力が，認識系に渡されて，図形のコードに変換される．また，日本語文字認識であれば，仮名漢字変換と同様に，日本語入力管理部から呼び出される．

ジェスチャ以外の認識は操作ではなく，入力であるために，ウィンドウや領域の切替えが，データの区別の動作となる．

表1 ジェスチャ認識と“未”の間でやりとりされるデータ構造
Table 1 Data structure that is communicated between gesture recognition and HITSUJI systems.

（a） デバイスイベントのデータ構造
（a） Data structure of device event

データ名	サイズ（バイト）	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のID
座標	4	筆点列の開始点のディスプレイ上の座標
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上の座標
キャンバス座標	4	筆点列の開始点のキャンバス上の座標
筆点数	2	筆点列の筆点数
開始点	4	筆点の開始点のタブレット上の座標
中間点	4×n	開始点以外の筆点のタブレット上の座標（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）

（b） APイベントのデータ構造
（b） Data structure of AP event

データ名	サイズ（バイト）	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	n	イベントの種類に固有のデータ ジェスチャなら，範囲指定か，元先のデータ

5.3 ウィンドウ間通信

“未”でのウィンドウ間通信の方式を**図2**に示す．異なったアプリケーションの間で通信を行うためには，通信元は通信先のプロセスのIDを知る必要がある．“未”では，この通信を行うインタフェースを提供する．

ウィンドウAから，ウィンドウBに対してデータを送るジェスチャを発行した場合，まず，“未”は，ウィンドウAが所属するプロセスへイベントを発行する．このプロセスが送信先をチェックし，自分の持っているウィンドウであれば，自分の内部で処理をする．これが，他のプロセスであれば，このプロセスが転送すべきデータを用意して，転送先のプロセスへイベントを発行するように“未”へ要求を出す．

これによって，送信先に，イベントとそれに伴うデ

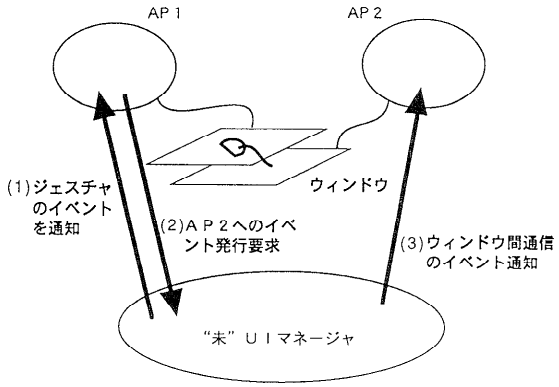


図2 ウィンドウ通信機構
Fig. 2 Inter-Window data transfer mechanism.

ータを送ることができる.

6. アンドゥ・リドゥ処理

本章では,"未"のアンドゥ・リドゥ機能について述べる.

6.1 アンドゥ・リドゥの実行モデル

"未"のアンドゥ・リドゥ処理を図3に示す.エンドユーザが入力を行うと,入力管理部でこのイベントの種類を判断する.この結果によって,後の処理が分かれる.

アンドゥやリドゥでなければ,イベントがそのままウィンドウ管理部,イベント管理部を経て,対象APに送られる.イベント管理部は,アンドゥ管理部へイベントの内容を送る.アンドゥ管理部は,このイベントをバッファへ格納して,アンドゥを複数回実行できるようにする.

イベントがアンドゥやリドゥならば,アンドゥ管理部で,バッファからイベントデータを取り出して,どのAPに対するイベントかを決定する.その後,アンドゥのAPイベントとなって,ウィンドウ管理部,イベント管理部を経て,APへ知らせる.このとき,ウィンドウ操作のアンドゥなら,ウィンドウ管理部でアンドゥ操作を行う.

6.2 "未"のアンドゥ機能

"未"のアンドゥ機能は次のとおりである.

(1) 複数回のアンドゥ・リドゥ

"未"では,複数のAPイベントを時系列でアンドゥ管理部のバッファに格納できるようにして,複数回のアンドゥ・リドゥを可能にする.バッファは,二つあり,これらをアンドゥバッファ,リドゥバッファと呼ぶ.これらのバッファはスタック構造になっている.

アンドゥの状態遷移図を図4に示す.アンドゥは,

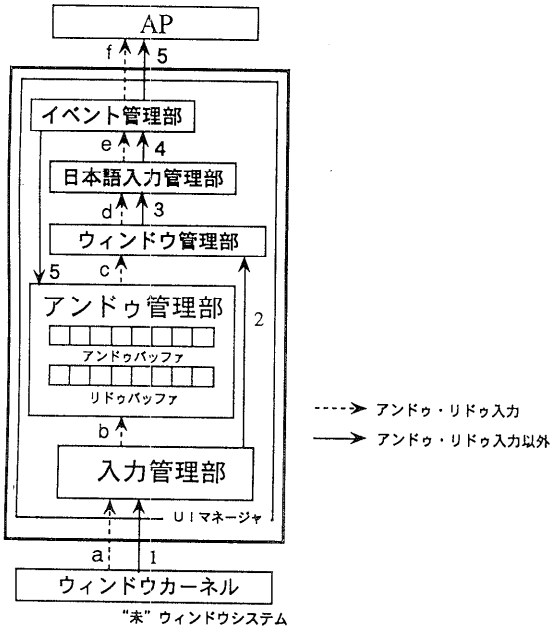


図3 アンドゥ・リドゥ処理モデル
Fig. 3 Undo/Redo process model.

リドゥで入力開始の状態に戻っている場合を除いて,いつでもできるが,リドゥはアンドゥをした後に,アンドゥをした回数だけが可能である.これは,リドゥはアンドゥのアンドゥであり,通常入力に対するリドゥは意味を持たないからである.

アンドゥが行われたときに,すでに対象となるAPの実行が終了している場合が考えられる.この状態でアンドゥを行うには,再びAPを実行して,終了する前の状態に戻す必要がある.

(2) アンドゥのユーザインタフェース

"未"ではアンドゥ,リドゥの入力の手段を提供して,アンドゥ・リドゥの操作を統一する.入力方式として,"未"ではジェスチャが使用できることから,アンドゥ・リドゥの入力もジェスチャで行うことが可能である.しかし,アンドゥ・リドゥは,誤操作の取消しなので確実に行いたい.そこで,画面上にコントロールボタンを設けて,これをペンで指示する方式を採用することで,ジェスチャ入力の誤認識によってアンドゥできないという状態を避けるようにした.

(3) APへの通知機構

ウィンドウの操作以外のアンドゥ・リドゥの処理を,ウィンドウシステム内部では行わずAP側で処理して,アンドゥ・リドゥ処理をAPから独立させた.

AP側で処理するには,アンドゥ・リドゥが発生したことをAPが知る必要がある."未"では,アンドゥ・

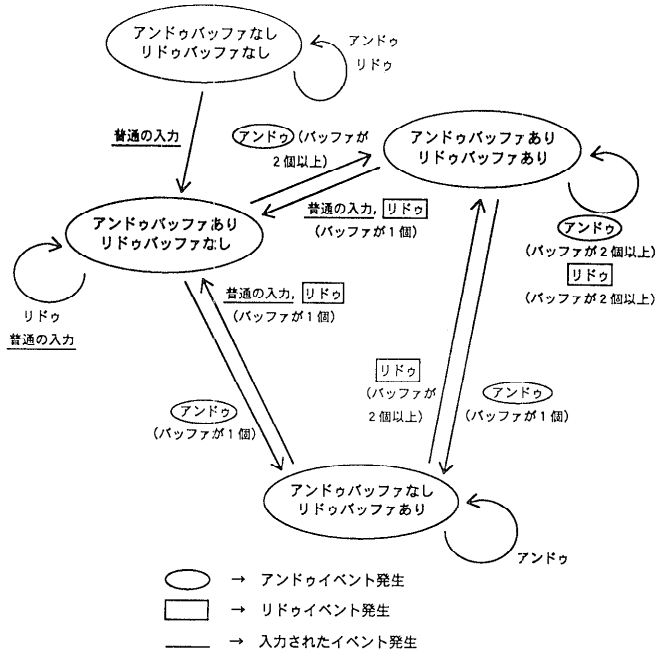


図4 アンドゥ・リドゥ時の状態遷移図
Fig. 4 State transition diagram of Undo event.

表2 “未”初版のジェスチャ
Table 2 Gestures of HITSUJI Version 1.

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 左図の順番で描くこと
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
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
スクロール			種類はあるが，ジェスチャは用意していない

リドゥは，通常のイベントと同じAPイベントとして実現し，通常のイベントプログラミングと同じインタフェースで処理可能にした．

このイベントデータは，アンドゥの対象となるウィンドウと対象イベントのIDを含んでいるので，APが対象となるアンドゥ操作を特定できる．

7. 実　　現

“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，実際に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

“未”初版で実現したジェスチャを表2に示す．これは，操作対象をウィンドウやシェルのアイコンの操作とし，さらに，APにも使えるようにジェスチャの形態をウィンドウやアイコンの操作に特化しないように決定した．

ジェスチャを範囲指定型，元先操作型の二つの型に分類した．範囲指定型は，ある範囲を指定するもので，削除のジェスチャはこの一種である．元先操作型は操作元と先があるもので，コピーや移動のジェスチャはこれにあてはまる．

また，ジェスチャは，イベントの形でAPが利用可

能である．実際に，“未”上で実現した手書き分散KJ法システム[8]では，個々のカード間のグルーピングにジェスチャを用いている．

7.2 ウィンドウの操作インタフェース

“未”初版では，ウィンドウの操作にジェスチャを採用した．具体的には，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．

“未”初版では，ウィンドウ操作に対して，アンドゥ・リドゥが可能である．ウィンドウ操作には，移動，リサイズ，重なり位置変更，スクロール，アンマップの5種類があるが，これらすべてに対して，アンドゥ・リドゥを提供する．具体的には，ウィンドウ管理部内部に，アンドゥ管理部と同様のバッファを用意して，ウィンドウ操作時のデータを格納する．

7.3 実現環境

“未”初版は，OS/omicron第2版を使用して開発，実現した．“未”初版は，OS/omicron第3版の上のサーバとして実現した．

ジェスチャ認識も，OS/omicron第3版のサーバとした．サーバ構成にしたことによって，認識処理部を含むシステムモジュールの変更が容易となっている．

“未”ウィンドウシステムは60人月かけて作成してきた．システム記述言語は，我々の研究室で開発した言語C処理系CATである．現時点での“未”初版の各モジュールのステップ数を**表3**に示す．

表3 “未”のステップ数
Table 3 Code steps of HITSUJI.

ウィンドウカーネル	
モジュール	ステップ数
ヘッダ	2530
SVC処理関係	3683
ディスプレイ関係	2265
資源管理表と操作関係	4201
デバイスドライバ	1660
その他	474
計	14339

UIマネージャ	
モジュール	ステップ数
SVC処理関係	965
ウィンドウ管理部と操作関係	5188
入力管理部	738
日本語入出力部	2201
アンドゥ管理部	330
イベント管理部	1831
計	11253

ウィンドウシステムでは，内部で管理する資源が多いことから，資源管理表ごとに管理部分と操作関数を定義し，それらを通してだけ表にアクセスするようにして，モジュール性を上げている．また，資源は相互関係を持つことから，表の間の一貫性の維持やチェックを行うようにコーディングしている．実際に，表3から“未”では，UIマネージャ，カーネルとも，ウィンドウやデザインなど資源の管理表とその操作関数の部分のステップ数が多いことが分かる．

他のサーバやアプリケーションとのインタフェース部分のコードが多いのは，OSのSVCによる“未”のSVC呼出しを“未”の内部表現へ変換する機構を持っているからである．また，仮名漢字変換部は，文字列編集の機能を含んでいるので，コード数が多くなっている．

“未”初版のAPIを**表4**に示す．“未”初版の実行画面を**図5**に示す．

7.4 評　価

現在，“未”ウィンドウシステムはプログラム開発環境として用い，その上でAPを開発，実現している．主要なアプリケーションを次に示す．

- ビジュアルシェル“双（たぐい）”[9]
- 分散手書きKJ法システム[8]
- 手書きビジュアルプログラミング言語とそのプログラミング環境[10]

“未”の評価について次に述べる．一般に，資源管理方式やプログラミングモデルの異なるウィンドウシステムを評価するのは難しい．そこで，本論文では，“未”上のアプリケーションを例にして，“未”のモデルの有効性を示す．

表5は，“未”上で動作するコマンドインタプリタ“双”の各モジュールごとのステップ数である．

表4 “未”初版のAPI
Table 4 APIs of HITSUJI Version 1.

ウィンドウシステム関連	キャンバス関連
＿＿ウィンドウシステムをオープンする	＿＿キャンバスの大きさを変更する
＿＿ウィンドウシステムをクローズする	＿＿キャンバスを消去する
＿＿イベントキューの大きさを設定する	＿＿キャンバスをフラッシュする
ウィンドウ関連	**イベント関連**
＿＿ウィンドウを生成する	＿＿イベントを登録する
＿＿ウィンドウを消去する	＿＿イベントを消去する
＿＿ウィンドウをマップする	＿＿イベントを得る
＿＿ウィンドウをアンマップする	＿＿マウスに合わせて枠を移動する
＿＿ウィンドウタイトルを変更する	**その他**
＿＿ウィンドウを一番上にする	＿＿ディスプレイの内容をファイルに書き込む
＿＿ウィンドウを一番下にする	＿＿筆点列を消去する

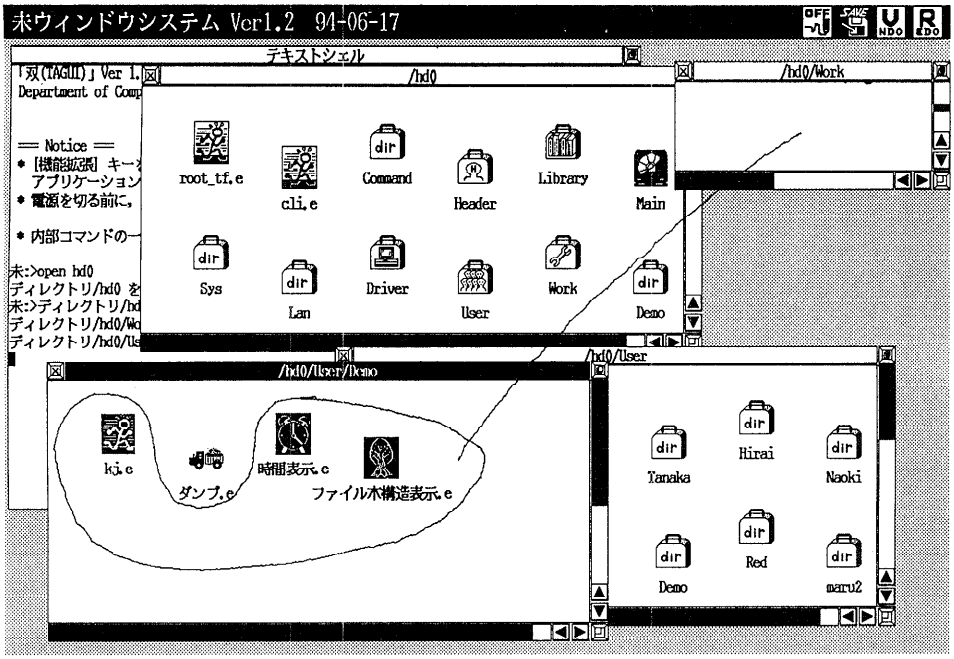


図5 “未”初版の実行画面
Fig. 5 Screen dump of HITUJI Version 1.

表5 “未”上のシェルのステップ数
Table 5 Code steps of a shell on HITSUJI.

ファイル	ステップ数
セットアップ	752
テキスト入力	1343
テキスト出力	1198
ペン入力	1451
ペン出力	2060
ファイル操作	2622
ビルトインコマンド	104
プロセス管理	1159
AP管理	502

“双”は，キーボード操作によるテキストシェルと，ペン操作によるビジュアルシェルという二つのインタフェースを備えている．ビジュアルシェルでは，ジェスチャによるファイルの操作が可能である．

表5を見ると，入力部分については，テキストによる部分と，ペンインタフェースによる部分とで，同程度のステップ数で実現されていることが分かる．ジェスチャをAPIとして提供したことから，ジェスチャに対応する実体を呼び出せばよく，入力の部分の処理は，テキストと同程度で記述できることが分かる．これによって，“未”が提供するAPIの妥当性が主張できると考えられる．

8. おわりに

本論文では，ペン入力のUIを研究するための基盤として作成した“未”の設計と実現について述べた．“未”では，ペンインタフェースに特徴的な操作である認識処理のインタフェースに着目し，ウィンドウシステムアーキテクチャとして，認識を組み込むための枠組みを設計，実現した．認識処理の一つとして，ジェスチャ認識を実現した結果，ウィンドウの操作やAPに対して，ジェスチャを使用することが可能になったとともに，ジェスチャ認識を研究，評価するための基盤として，用いることが可能になった．また，誤認識に対する操作の取消しと再実行を行うアンドゥ・リドゥ処理を実現した．特に，一貫したアンドゥ・リドゥ操作を行うためのAPIを提供した．

今後の課題として，次のことが挙げられる．

（1） ペンインタフェースに関する評価

今回の設計で，“未”初版の評価を行うのに必要となるデータをイベントデータに導入した．今後，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を作成する環境を整備する必要がある．

謝辞 高解像度の表示一体型液晶タブレットの使用にあたり，御援助頂いた日立製作所日立研究所の福永

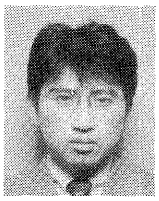
泰氏，正嶋博氏，荒井俊史氏に感謝いたします．

なお，本研究は文部省科学研究費試験研究B(1) 0558027により行われた．

参考文献

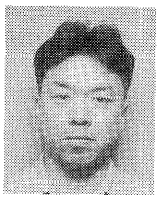
1) Microsoft Press: *Microsoft Windows for Pen Computing Programmer's Reference,* Redmond, Washington (1992).
2) 荒井俊史，正嶋 博，福永 泰：手書きヒューマンインタフェースの構築支援環境，情報処理学会ヒューマンインタフェース研究会，Vol. 35, No. 7 (1991).
3) SONY: PalmTop PTC-300/310使い方手引書，SONY (1992).
4) Carr, R. and Shafer, D.: *The Power of Pen-Point,* Addison-Wesley, Reading, Massachusetts (1991).
5) 曽谷俊男，福島英洋，高橋延匡，中川正樹：遅延認識を用いた手書きユーザインタフェースの基本設計，情報処理学会論文誌，Vol. 34, No. 1, pp. 158-166 (1992).
6) 風間信也，福島英洋，中川正樹：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HI研究会，Vol. 43, No. 3 (1992).
7) Kataoka, Y., Mirisaki, M., Kuribayashi, H. and Ohara, H.: A Model for Input and Output of Multilingual Text in a Windowing Environment, *Proc. UIST'91,* pp. 175-183 (1991).
8) 中島一彰，早川栄一，並木美太郎，高橋延匡：『紙』メタファによる手書き作図コミュニケーションと分散手書きKJ法システム，情報処理学会システムソフトウェアとオペレーティング・システム研究会，60-22, pp. 163-170 (1993).
9) 宮島 靖，河又恒久，早川栄一，並木美太郎，高橋延匡：ジェスチャ操作によるユーザインタフェースを持つビジュアルシェル，第46回情報処理学会全国大会論文集，8 H-03 (1993).
10) 小島大吾，並木美太郎，高橋延匡：ペンによるアニメーション作成ビジュアル言語の研究，第48回情報処理学会全国大会論文集，2 G-01 (1994).

(平成6年1月14日受付)
(平成7年2月10日採録)



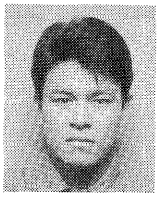
早川 栄一（正会員）

1989年東京農工大学工学部数理情報工学科卒業．1992年同大大学院電子情報工学専攻博士後期課程単位取得退学．同年同大学電子情報工学科助手．博士(工学)．オペレーティングシステムおよび日本語情報処理の研究に従事．ACM，IEEE，電子情報通信学会各会員．



河又 恒久（正会員）

1968年生．1991年東京農工大学数理情報工学科卒業．1993年同大学大学院博士前期課程（電子情報工学専攻）修了．同年日本電気（株）入社，現在に至る．ウィンドウシステムのユーザインタフェースに興味を持つ．



宮島 靖（正会員）

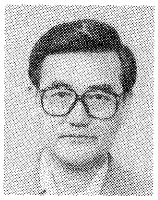
1968年生．1991年東京農工大学数理情報工学科卒業．1993年同大学大学院博士前期課程（電子情報工学専攻）修了．同年ソニー株式会社入社，現在に至る．ユーザインタフェース，特にペン入力を用いたコンピュータシステムに興味を持つ．

加藤 直樹（学生会員）

1969年生．1993年東京農工大学工学部電子情報工学科卒業．同年同大学大学院博士前期課程（電子情報工学専攻）進学，現在に至る．ヒューマンインタフェース，ペンコンピュータシステムに興味を持つ．

並木美太郎（正会員）

1984年東京農工大学工学部数理情報卒業．1986年同大学院修士課程修了．同4月(株)日立製作所基礎研究所入社．1988年東京農工大学工学部数理情報工学科助手．1989年4月電子情報工学科助手．1993年11月電子情報助教授．博士(工学)．オペレーティングシステム，プログラミング言語，ウィンドウシステム，並列処理，コンピュータネットワークおよび日本語情報処理の研究・開発に従事する．ACM，IEEE，電子情報通信学会各会員．



高橋 延匡（正会員）

1957年早稲田大学第一理工学部数学科卒業．同年4月(株)日立製作所中央研究所入社，HITAC 5020モニタ，TSSの開発などに従事．1977年東京農工大学工学部数理情報工学科教授，1989年電子情報工学科教授．オペレーティングシステム，日本語情報処理などの研究，教育に従事．理学博士．現 本学会監事．