ヒューマンインタフェース 30－3
（1990．5．10）

ウィンドウシステムにおけるタッチスクリーンの使用について

来住伸子
日本アイ・ビー・エム株式会社　東京基礎研究所

タッチスクリーンのような非標準の入力デバイスをウィンドウシステム上のアプリケーションが使用するための問題点について考察し，タッチスクリーンの使用方法として，三層の枠組みからなる手法を提案する．ユーザーの様々な要求に答えるために，三層の枠組みがどのように利用できるかを，異なる方式のタッチスクリーンや異なる手法の使用例を通じて紹介する．

Supporting Touch Screens For Window Applications

Nobuko Kishi
IBM Research, Tokyo Research Laboratory, IBM Japan, Ltd.
5-19, Sanbancho, Chiyoda-ku, Tokyo 102, Japan

Although touch screens are widely used as input devices for simple applications, they are not often used for more complex applications such as window applications. This paper proposes a framework that supports various types of touch screen for window applications with graphical user interfaces. The main purpose of this framework is to offer a flexible design environment with three layers of services so that a variety of design alternatives can be explored for touch screens.

0．はじめに

計算機の普及にともない，計算機の使用される環境が多様になってきている．そこでタッチスクリーンのような今まで一般的でなかった入力デバイスの使用が増えつつある．この研究ではユーザーに使いやすくタッチスクリーンを提供すると同時に，ソフトウェア資源の再利用をはかる方法として，ウィンドウシステムのもとでタッチスクリーンをサポートする手法について考察し，実際に機能の一部を作成し使用してみた経験について紹介する．

1．マルチデバイスのサポートの必要性

新しい入力デバイスを提供するとき，ユーザーの立場からみると，自然に使用できると同時に、システムの機能がより多く提供されることが望ましい．現金引き出し機のように，タッチスクリーンを押しボタンの代わりに使用するというように要求が明確になっていて，機能が単純な場合，特定のタッチスクリーン専用のソフトウェアを使用することが多い．しかし，オフィスアプリケーションのような，機能がより多く，ユーザーの要求も多様な場合，タッチスクリーン専用のソフトウェアを作成することは，費用や時間の点で難しいことが多い．そのため，他の入力デバイス，例えばマウス用のソフトウェアを流用して使用することがある．デバイスドライバでタッチスクリーンにマウスと同じ振る舞いをさせるという方法がその一例である．

しかし，デバイスドライバ以外のソフトウェア部分にも，流用が不向きな点があることが多い．たとえば，ウィンドウシステムの多くはマウスでの使用を前提として設計されているので，小さすぎる画面要素やマウスのボタンを押したまま動かすという動作の使用といった点で，タッチスクリーンからの使用に向かない面がある．

最良の方法は，タッチスクリーン専用のウィンドウシステムの開発かもしれないが，時間と費用の点からの妥協として，既存のウィンドウシステムに手を加えるという形で，標準として使用されていない入力デバイスをサポートする手法を開発してみることにした．

2．三層の枠組みを使用したタッチスクリーンのサポートについて

入力手段へのユーザーの要求は種々ある．それら一つ一つを満たすときに，ソフトウェアの変更部分が少なくなるように，タッチスクリーンのサポートを三層に分けて行うことにした．

入力手段へのユーザーの要求を整理してみると，次のような分類の仕方が考えられる．

(a)ハードウエアに関するもの．

タッチスクリーンの場合でいえば，触感や，スクリーンの反射度や透明度を指す．

(b)ユーザーの動作に関するもの

タッチスクリーンの場合でいえば，たとえば，押す，離すといったユーザーの動作についての要求．

(c)機能に関するもの

アプリケーションの機能との調和などについての要求．例えば，あるコマンドを選択するのに，押しボタンとプルダウン・メニュウのどちらが，アプリケーションのユーザーに自然な操作かどうかというもの．

まず，(a)のタイプの要求を満たすには，使用するタッチスクリーンの方式を取り替えるということが考えられる．タッチスクリーンの方式には，抵抗膜，容量変化膜，など種々のものがあり，また，年々，解像度や見やすさなどの機能が改善されつつある．そこで，使用するタッチスクリーンの方式を変更したときに，ソフトウェアの変更部分を少なくてすむような仕組みが必要になる．

つぎに，(b)のタイプの要求を満たすには，押す力や時間の調節や視差の補正を，ユーザーの体格や計算機の置かれている環境に合わせるように，調節できる仕組みが必要になる．

(c)のタイプの要求を満たすには，アプリケーションの目的に合ったユーザーインターフェース要素の追加変更ができる仕組みが必要になる．

そこでユーザーの異なる要求を満たす仕組みを，次のような三層のサービス層として提供することにした(図１)．

(1)デバイス層(Device Layer)

(2)手法層(Technique Layer)

(3)要素層(Component Layer)

デバイス層はデバイスからの物理的な信号をイベントに変換する．従来のデバイスドライバとほとんど同じ役割，タッチスクリーンの方式ごとに異なる信号を一定の種類のイベントに変換するという役割を果たす．

さらに，各方式特有の情報もパラメータとしてイベントに付加できる機能も提供している．

手法層は，渡されたイベントを各ユーザーインターフェース要素へのメッセージに変換して，各ユーザーインターフェース要素に送る．

要素層では，ユーザーインターフェース要素の機能や状態に応じて，メッセージがアプリケーションに意味のあるデータに変換される．

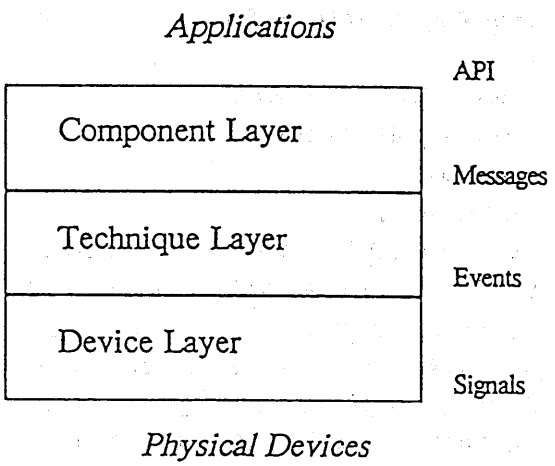

図１．三層のサービス層

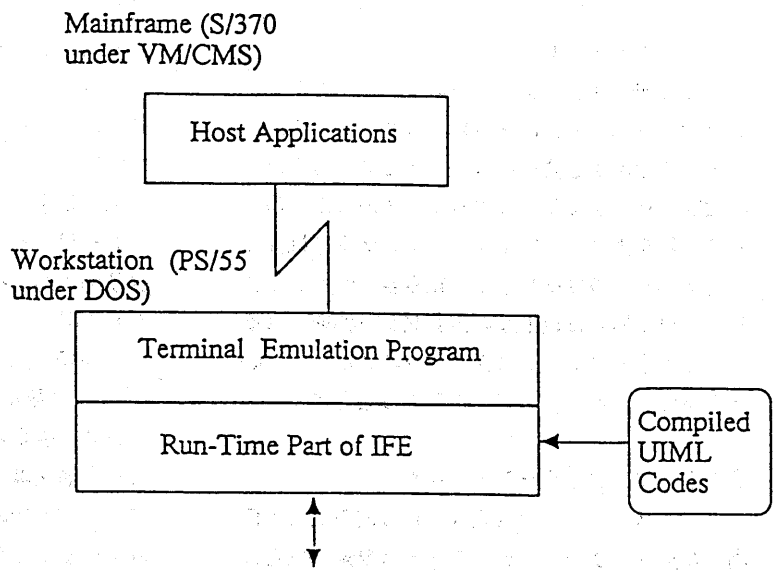

図２．三層の枠組みの実行環境

３．三層の枠組みの作成環境

ここで，この三層の枠組みを使用してみた環境について簡単に紹介する．この研究の背景となったのは，ＩＦＥ（インテリジェントフロントエンド）と呼ばれるシステムで，ホスト上のアプリケーションに対し，ＰＣ側の通信ソフトの機能を利用してユーザーインターフェースを改善しようというソフトウェアである．図２に示すようにホストとなるメインフレーム計算機に対し，ＰＣが端末として接続されている．このとき，ＰＣ側では端末機能をエミュレートする通信ソフトウェアが動いている．IFEはこの通信ソフトウェア上のアプリケーションとして動く．ホスト側から送られてきたデータの内容を解釈してからＰＣ画面上に表示し，ＰＣへのユーザー入力を解釈してホストへのデータ列を生成して返送する．

ユーザーインターフェースの改善すべき項目のうち，最初にでてきたものが，ウィンドウシステムを利用したグラフィカルユーザーインターフェース(GUI)の提供であった．そこで，IFE自身でDOS上で動くウインドウシステムを提供することにした．また，ＰＣ側の処理とホスト側の処理の対応付けは，簡単なプログラミング言語で記述して指定するようにした．この言語は，ＵＩＭＬ(User

Interface Mapping Language)という名称で，コンパイル型の言語である．

その後，タッチスクリーンのような，ホストアプリケーションでは従来使用されていない入力デバイスの使用が必要になった．そこで，ＵＩＭＬの機能を強化することによって，先述の三層の機能の一部を提供し，ウインドウシステムの機能と入力デバイスの特性をうまく組み合わせることにした．ＵＩＭＬは数種類の定義文から構成されるが，そのうちのタッチスクリーンのサポートに関連する部分のみについて，ここでは紹介する．

３．１．ワークステーション定義

ワークステーション定義は，ＰＣ側の画面で使用したいユーザーインターフェース要素の指定に使用される．たとえば，図３にあるような画面をＰＣ側で表示したい時は，付表１に示されているような記述を行う．図３の画面には，アクション・バー，プルダウン・メニュウ，選択リスト，プッシュ・ボタンなどのユーザーインターフェース要素が使用されている．ワークステーション定義の中には，これらについての記述がユーザーインターフェース要素の定義として複数含まれる．

一つのユーザーインターフェース要素の定義には次のような構文で表現されている．

:＜要素名＞　＜要素の表示属性＞
　{＜メッセージ名＞}　＜文の並び＞
　{＜メッセージ名＞}　＜文の並び＞
　　．．．．．．．．

ここで要素名は，使用するユーザーインターフェース要素の種類を指定する．要素の表示属性とは，位置，大きさ，色，表示する文字列のような要素の表示に必要な情報を指定する．メッセージ名とは，その要素が受け付けるメッセージの名前，文の並びとは，そのメッセージを処理する際に実行する文の並びに対応している．

例えば，付表１の４行目にある

:CHOICE "View"
　{SELECT} VIEW_MAIL(MAIL)

という記述はプルダウンメニュウのアイテムを定義している．':CHOICE' はプルダウンメニュウアイテムに対応する要素名，"View"はそのプルダウンメニュウアイテム上に表示する文字列，'{SELECT}'はメッセージの名前，'VIEW_MAIL(MAIL)'はSELECTメッセージを受け取ったときに実行する手続き名を表現している．

ユーザーインターフェース要素のいくつかは，親子関係を作ることができる．

たとえば，

:＜要素名１＞
　　:＜要素名２＞
:END＜要素名１＞

という構文は，要素名１をもつユーザーインターフェース要素の中に，要素名２をもつ要素が含まれていることを示している．

３．２．メッセージの定義

メッセージの定義は，デバイス層からのイベントを，各要素へのメッセージにどの様に変換するかを指定する．構文としては，

:MESSAGE　＜メッセージ名＞
　/＜デバイス名＞＝＜イベント名＞
　　＜パラメータ並び＞
　　．．．．．．．
:ENDMESSAGE

を使用する．たとえば，SELECTというメッセージが，

:MESSAGE　SELECT　/KEY=ENTER
　/MOUSE=MAINBUTTON DOWN
　/TP=TOUCH
:ENDMESSAGE

というように定義されているとすると，SELECTはキーボートからENTERキーが押されたイベント，タッチスクリーンが触られたイベント，マウスの主ボタンが押されたイベントの３つのどれかによって生成されるメッセージだという指定になる．デバイス名KEYはキーボード，MOUSEはマウス，TPはタッチスクリーンを示す．

３．３．ユーザーインターフェース要素とメッセ

ージの種類

ユーザーインタフェース要素の種類によって異なるメッセージを受け取る．IFEでは標準の状態では次のようになっている．

要素名	メッセージ名
アクションバー	POINT
バーアイテム	POINT SELECT
メニュウアイテム	POINT SELECT
選択リスト	POINT
プッシュボタン	POINT SELECT
テキスト入力域	SELECT
表示テキスト	なし

３．４．デバイスからのイベント

デバイスからのイベントは，各入力デバイスの発生する情報に基づいて生成されるが，それらの情報はデバイスの種類によって異なっている．キーボードからは，キーコードと押す・離すという情報，マウスからは，x-y座標とボタンの押す・離すという情報が発生する．

さらに，タッチスクリーンの場合は，使用される方式によって発生する情報が異なっている．x-y座標はほとんどの方式で使用が可能だが解像度や座標系が異なることが多い．スクリーンを押した強さや，スクリーンに触っていた時間という情報を発生するのは，特定の方式に限られる．

従来のデバイス・ドライバでは，各入力デバイスに特有な情報は，デバイス・ドライバの層に隠すという方針がとられることが多い．しかし，IFEのデバイス層では，このようにデバイス特有の情報のうち，メッセージの生成に関わりを持ちそうなものを，イベントの付加情報，パラメータとして，手法層に渡すことにした．各デバイスのイベントには，次のようなパラメータが付けることが可能である．

デバイス	イベント	パラメータ
キーボード	キーコード	シフト状態
マウス	MAINBUTTON	up down at
	SUBBUTTON	up down at
	MOVE	main sub
タッチスクリーン		
	TOUCH	at for by within
	RELEASE	at within

タッチスクリーンのイベントのうち，TOUCHはスクリーンに触るというイベント，RELEASEはスクリーンから指を離すというイベントに対応している．パラメータ´at´は正規化されたx-y座標，´for´はスクリーンに触っていた時間，´within´は触った箇所のx-y座標に付け加えられる許容範囲の大きさ，´by´はスクリーンを押した深さの情報をイベントに付加するために使用する．

３．５．三層の枠組みとUIMLの関係

UIMLのワークステーション定義やメッセージ定義は，三層の枠組みの中で次のように利用される．

UIMLで記述された情報は，コンパイルされ，ワークステーション定義からは，ワークステーション画面のレイアウトが生成される．また，メッセージ定義から，ワークステーション画面上の各ユーザーインターフェース要素に使用すべきメッセージ定義の表が作られる．

実行時には，入力デバイスからの信号は，デバイス層でイベントに変換され，手法層に送られる．手法層では，どのユーザーインターフェース要素が，メッセージを受け取る可能性があるかを次のような情報を元に判断する．

※各ユーザーインターフェース要素の位置
　ワークスクリーン定義に含まれる各要素の定義によって配置された各要素の位置．

※各入力デバイスが使用しているカーソルの位置．
　この位置はイベントからのx-y座標についての情報などから計算する．

※各ユーザーインターフェース要素の状態．
　各ユーザーインターフェース要素は，active，prohibited，pointedの三状態になれる．要素は画面に表示されている時，通常activeの状態になっている．カーソルに指されている状

態の時，pointed，カーソルを移すことができない状態の時，prohibitedになる．

そして，手法層では，どの要素にメッセージを送るかを決定した後，その要素にメッセージの処理を任せる．送ろうとする要素にメッセージ定義がない場合，親子関係をたどり，親のメッセージ定義を使用する．

要素層では，各要素定義にもとづいてメッセージの処理を行う．要素の画面表示や，アプリケーションへのデータの受け渡しを行う．

４．三層の枠組みの使用例

４．１．異なるタッチスクリーンの取扱い

タッチスクリーンの方式には，抵抗膜，容量変化膜，赤外線，負荷ゲージなど種々の方式があるが，それぞれに一長一短がある．またユーザーの個人差によってもタッチスクリーンの使用方法を調整しなくてならないことがある．IFEのイベントにパラメータを追加したのはそれらを処理できるようにするためだが，ここでは，そのパラメーターの使用例を紹介する．

※押す圧力および時間の利用

負荷ゲージを使用するタッチスクリーンは，押す圧力を測定できる．また，タッチスクリーンの多くの方式では，スクリーンに触れていた時間を測定できる．従来，スクリーンに触れるというイベントしか使用しないことが多かったが，これらの情報は，誤選択率を下げたり，自然な応答の実現に使用することができる．

マウスを入力デバイスに使用する場合，マウスカーソルを要素の上に置くという操作と，マウスボタンを押すという操作の２段階の操作を通じて選択することが多い．画面に触るというイベントしか使用できないタッチスクリーンだと，この２段階操作のサポートが不自然になりがちだが，押す時間や力の情報をを利用するとより自然な動作が使用できるようになる．

たとえば，画面に触る動作は，カーソルを要素の上に置くという操作に対応させ，一秒間以上触っていた場合は，その要素の選択と見なすという方法がある．これをUIMLで表現すると，次のようになる．

```
:MESSAGE SELECT
  /TP=TOUCH for 1 sec
:ENDMESSAGE
:MESSAGE POINT
  /TP=TOUCH
:ENDMESSAGE
```

また，0.1ｍｍ以上の深さで押した場合，要素の選択と見なすという方法は，次のようになる．

```
:MESSAGE SELECT
  /TP=TOUCH by 0.1 mm
:ENDMESSAGE
:MESSAGE POINT
  /TP=TOUCH
:ENDMESSAGE
```

※視差と解像度

視差（実際に指が触っている箇所と，ユーザーの目に触っているように見えている箇所との距離）の大きさは，タッチスクリーンの方式や，ユーザーの身体の位置などによって異なる．また，タッチスクリーンの測定解像度も方式によって異なる．従来のやり方だと，あらかじめ，視差を調べ，定められた解像度のタッチスクリーン専用に画面上の要素の位置を定めることが多かった．

UIMLでは，解像度や視差が変更になったときでも，画面上の要素の配置を再利用する事を考えて，ユーザーが触れる要素の位置に許容範囲を付けられるようにした．例えば，

```
:MESSAGE SELECT
   /TP=TOUCH within 5 mm
:ENDMESSAGE
```

となっていると，要素から 周囲5mm以内に指が置かれると，TOUCHイベントを生成することができる．視差が大きい時や，低解像度のときは，許容範囲を大きくしてある程度対応できる．要素同士が重なる程，許容範囲を大きくしすぎたかどうかは，

UIMLをコンパイルするときにチェックする．

４．２．異なる手法のサポート

ユーザーが入力デバイスを使用する動作の種類や順序を，手法(Interaction Technique, Strategy) と呼ぶことがある．タッチスクリーンの同じ方式でも，イベントの扱い方によって，異なる手法をサポートすることができる．もっともよく使われる手法は，前述のように，タッチスクリーンに触るというイベントだけを使うというものだが，離すとイベントを利用する手法もある．UIMLでは，このように異なる手法をメッセージの定義を書き換えるをだけで採用することができる．

※画面に触るイベントのみを利用する例

```
:MESSAGE POINT
  /KEY=CURSOR /MOUSE=MOVE
:ENDMESSAGE
:MESSAGE SELECT
  /KEY=ENTER /MOUSE=MAINBUTTON DOWN
  /TP=TOUCH
:ENDMESSAGE
```

※画面から指を離すイベントを利用する例

```
:MESSAGE SELECT
  /KEY=ENTER
  /MOUSE=MAINBUTTON DOWN
  /TP=REPLACE
:ENDMESSAGE
```

４．３．新しいユーザーインターフェース要素のサポート

マウスボタンを押したままマウスを動かすドラッグのような動作は，タッチスクリーンでは自然に出来ない．とくに長い曲線を描くというような操作は，タッチスクリーンからでは難しいことが多い．しかし，スクロールバーからの入力のように，アプリケーションに一次元の数量が渡すことができればいい場合は，ドラッグを使わないユーザーインターフェース要素の使用によって，タッチスクリーンからの入力をより自然なものにする事が可能である．たとえば，マウスからの入力に対しては，通常のスクロールバーとして振る舞い，タッチスクリーンからの入力に対しては，ボタン入力によって値を変化させるという新しいスクロールバーを用意するという方法がある(図 4)．
このような異なるメッセージを処理できるユーザーインターフェース要素の振る舞いは，UIMLでは次のようにして記述できる．

```
:MESSAGE DRAG
    /MOUSE=MOVE MAIN
:ENDMESSAGE
:MESSAGE SELECT
    /TP=TOUCH
:ENDMESSAGE
:SCROLLBAR /X=10 /Y=1 /LENGTH=10
   {DRAG} VALUE1=QUERYMESSAGE(OFFSET)
          SETPAGE1(VALUE1)
   {SELECT} VALUE2=UPDOWNBUTTON()
          SETPAGE2(VALUE2)
:ENDSCROLLBAR
```

これは，DRAG と SELECT の２種のメッセージを受け付けるスクロールバーを定義している．DRAG メッセージはマウス入力から送られていると仮定しているので，マウスの移動量(OFFSET)を利用して，新しいページ番号を設定している(SETPAGE1)．一方，SELECT メッセージはタッチスクリーン入力から送られていると仮定しているので，ボタンを表示し，ボタンによってユーザーが設定した値を利用して，新しいページ番号を設定している(SETPAGE2)．

５．まとめ

ウィンドウシステムでは標準的ではないタッチスクリーンの使用について考察し，タッチスクリーンのサポートのための枠組みを提案した．ユーザーの様々な要求に応ずると同時に，ソフトウェアの変更部分を少なくするために，三層のサービス層を使用した．提案した三層の枠組みのうちの

機能の一部を，ホスト端末のユーザーインターフェース変換システム（IFE：Inteligent Front End）で作成した．IFE自体は，メール文書管理システムと経営意志決定支援システムに使用されているが，タッチスクリーンのサポートとしては，デバイス層と手法層のサービスの一部を提供しているという段階である．マウスおよびキーボードから出来るウィンドウ操作が，タッチスクリーンからも同時に出来る．

IFEでの作成経験によって，入力デバイスの処理の調節に対する要求は多様であることと，そのための変更の範囲が三層の枠組みによって抑えられたことを確認した．しかし，枠組みを使用することによって，どれだけ変更量が少なくて済んだか，変更した設計がユーザーの使用感にどれだけ影響を与えたかといった点の具体的な評価は行っていない．IFEのウィンドウシステムの場合，DOSという制約のため，提供しているユーザーインターフェース要素の種類に限りがあるという点や，実際に使用できるタッチスクリーンの種類も限られているという点から，一般的な結論に結びつく評価は困難なためである．より機能の豊富な，標準のウィンドウシステムでの使用が可能になったときに，この枠組みの効果およびユーザーによる使用感の評価を行いたいと考える．

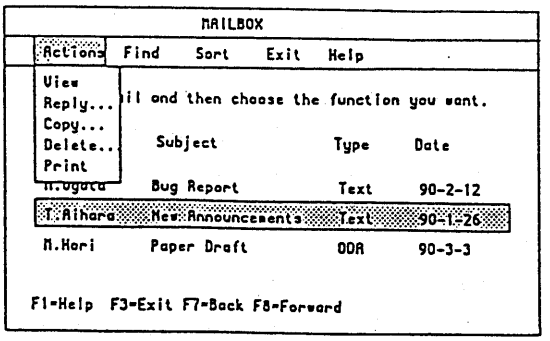

図３．IFEの生成するPC画面の例

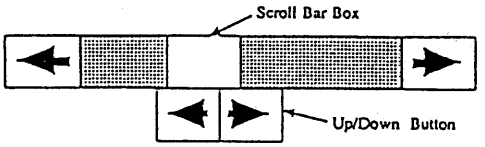

図４．タッチスクリーン向きの
スクロールバーの一例

付表１．図３のUIMLによる記述

```
:WSWINDOW
   :ACTIONBAR
      :ACTION "Action"
         :CHOICE "View"
            {SELECT} VIEW_MAIL(MAIL)
         :CHOICE "Reply"
            {SELECT} REPLY_MAIL(MAIL)
            ......
      :ACTION "FIND"
   ......
   :ENDACTIONBAR
   :BODY
      :INSTRUCT /X=9 /Y=3 "Select Mail ...."
      :SELECTIONFIELD /X=3 /Y=5 /TYPE=SINGLE_CHOICE
         :CHOICE MAIL1
            MAIL=MAIL1
         :CHOICE MAIL2
            MAIL=MAIL2
            ......
      :ENDSELCETIONFIELD
   :ENDBODY
   :FKEYAREA
      :FKITEM "F1=Help" /KEY=F1
         {SELECT} HELP()
      .....
   :ENDFKEYAREA
:ENDWSWINDOW
```


参考文献

1) 日本アイ・ビー・エム株式会社，ユーザー・インターフェース設計実行プログラムユーザーズガイド　N:SB18-1178, 1989

2) Masanobu Ogata, Toru Aihara, Nobuko Kishi and Yeong-Chang Lien, "Tools for Mapping User Interfaces," Proc. Of the Third International Conference on Human-Computer Interaction (HCI International '89), 1989.

3) Richard L. Potter, Linda J. Weldon, Ben Shneiderman, "Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies," CHI'88 Conference Proceedings, 1988.