筑波大学大学院博士課程

システム情報工学研究科修士論文

空中での手の動きを用いた
ペン入力インタフェースに関する研究

鈴木 優

(コンピュータサイエンス専攻)

指導教員　田中 二郎

2008年3月

概要

入力デバイスとしてスタイラスを用いるシステムは，手書きでの絵や文字を描くといったクリエイティブな作業を行うのに適したシステムである．しかしながら，そのシステムのインタフェースは必ずしもスタイラスでの操作に適したものであるとはいえない．この理由として，マウスでの操作を前提とした現在の WIMP インタフェースをそのままスタイラスで操作するシステムにも流用していることと，スタイラスの特徴を活かした入力操作が少ないことが考えられる．例えば，スタイラスでのポインティング操作はペン先をディスプレイに接触させる際の正確なポインティングが非常に難しいため，従来のメニューバーや小さいアイコンを基本としたメニューインタフェースは使いやすいインタフェースであるとはいえない．

本研究では，以上の問題点を解決するためのインタラクション手法であるショートカットアクションを提案した．ショートカットアクションとは，スタイラスがディスプレイに接していない状態での手の動き，つまりスタイラスを握る手の空中での動きを利用したインタラクション手法である．ショートカットアクションでは人間がペンを握ったまま行うことができ，かつ人間にとって無理のない動作として，ペンを振る，回すなどの動きである３つの動作，rolling，shaking，swinging を入力操作として採用している．また，ショートカットアクションを実現するデバイスとして Context Sensitive Stylus を作成した．Context Sensitive Stylus を用いることで，スタイラスを握る手の空中での動きを取得することができる．

ショートカットアクションの応用例として，Oh! Stylus Series というアプリケーション群を作成した．Oh! Stylus Series には，お絵かきツールである Oh! Stylus Draw や，スクロール支援ツールである Oh! Stylus Scroll などがある．Oh! Stylus Draw では，メニューインタフェースとして FlowButton というインタフェースを作成した．FlowButton はユーザがメニュー呼び出しをしたときに画面上に浮き上がってくるようなメニューである．

Oh! Stylus Series のアプリケーションを用いてショートカットアクションの評価実験を行った．その結果，ショートカットアクションの有用性を示すことができたとともに，スタイラスでメニューバー形式のメニューを操作することの難しさを証明することができた．

本研究により，既存のスタイラスを用いるシステムで無視されていた空中での手の動きをインタラクションに活用することができるようになった．その結果として，スタイラスがユーザの視線を遮断してしまう問題，細かいポインティングに意識や集中力を奪われてしまう問題，大画面ディスプレイでスタイラスを使用するときにユーザが操作のために移動を強いられる問題などの，既存のシステムにおける複数の問題点を解決することができた．

目 次

第 1 章　序論 **1**
1.1 本研究のフォーカス 1
1.2 本研究の目的 4
1.3 本研究の貢献 4
1.4 論文の構成 5

第 2 章　提案手法：ショートカットアクション **6**
2.1 基本アイデア 6
2.2 ショートカットアクション 7
2.3 ショートカットアクションの設計 8
2.3.1 swinging の意味付け 8
2.3.2 バレルボタンの使用について 9
2.3.3 ストロークジェスチャとの組み合わせについて 10
2.3.4 ペンメタファの導入 10
2.4 関連する操作手法 11

第 3 章　ショートカットアクションの実現 **13**
3.1 Context Sensitive Stylus の作成 13
3.2 アクションの認識方法 14
3.2.1 認識に使用するデータ 15
3.2.2 方法１：単純な加速度情報の比較 15
3.2.3 方法２：方法１のアルゴリズムの改善 16
3.2.4 方法３：DP マッチング 18

第 4 章　ショートカットアクションの応用 **21**
4.1 応用に関する議論 21
4.1.1 アクションへのコマンドの割り当て 21
4.1.2 アクションに対するフィードバック 21
4.2 実装に用いた基本技術 22
4.3 Oh! Stylus Series 23
4.3.1 Oh! Stylus Draw 23
4.3.2 Oh! Stylus Scroll 33

4.3.3 Oh! Stylus Firefox Scroll . . . 34
4.3.4 Oh! Stylus 3D Viewer . . . 36
4.3.5 Oh! Stylus OneNote Extension . . . 37
4.4 操作体系 . . . 39

第 5 章 評価 **40**
5.1 評価実験１ . . . 40
5.1.1 目的と観点 . . . 40
5.1.2 評価方法 . . . 40
5.1.3 結果 . . . 43
5.1.4 結果に関する考察 . . . 44
5.2 評価実験２ . . . 44
5.2.1 目的 . . . 44
5.2.2 評価方法 . . . 44
5.2.3 結果と考察 . . . 45
5.3 評価実験３ . . . 46
5.3.1 目的 . . . 46
5.3.2 評価方法 . . . 47
5.3.3 結果と考察 . . . 47
5.4 議論 . . . 51
5.4.1 利き手について . . . 51
5.4.2 ショートカットアクションの習慣化について . . . 51
5.4.3 ディスプレイの角度の違いについて . . . 52

第 6 章 関連研究 **53**
6.1 研究のスコープ . . . 53
6.2 関連研究 . . . 53
6.2.1 人間の動作に着目した研究 . . . 53
6.2.2 端末のコンテキストに着目した研究 . . . 54
6.2.3 スタイラス向けインタフェースに関する研究 . . . 55
6.3 本研究の位置付け . . . 58

第 7 章 結論 **59**

謝辞 **60**

参考文献 **61**

図 目 次

1.1 入力デバイスにスタイラスを用いるさまざまなシステム 2
1.2 WIMP インタフェースの例 3

2.1 ３つのアクション 8
2.2 ペンを持ち替える様子 9
2.3 バレルボタン（グリップ部にあるボタン） 10
2.4 ４色ボールペンメタファ 11

3.1 Cookie と取得できる３軸方向 13
3.2 作成した２種類の CS Stylus 14
3.3 shaking，swinging の動作軸と Cookie の加速度軸 16
3.4 求める θ のイメージ図 17
3.5 DP マッチング手順１ 19
3.6 DP マッチング手順２ 19
3.7 DP マッチング手順３ 19
3.8 DP マッチング手順４ 19
3.9 DP マッチング手順５ 19
3.10 DP マッチング手順６ 19

4.1 ボタンのフィードバック 22
4.2 Oh! Stylus Draw 1 のスクリーンショット 24
4.3 Oh! Stylus Draw 1 のカラーパレットとマウスポインタの拡大図 24
4.4 Oh! Stylus Draw 2 のスクリーンショット 25
4.5 Oh! Stylus Draw 2 のカラーパレットの拡大図 26
4.6 Oh! Stylus Draw 2 swing における FlowButton の表示例 27
4.7 Oh! Stylus Draw 2 swing のメニューバー 27
4.8 Oh! Stylus Draw 2 swing のペン先を切り替える FlowButton の表示例 28
4.9 Oh! Stylus Draw 2 swing の描画モードを切り替える FlowButton の表示例 . . . 29
4.10 Oh! Stylus Draw 3 のスクリーンショット 30
4.11 Oh! Stylus Draw 3 の操作の流れ 31
4.12 Oh! Stylus Draw 3 のトップ FlowButton 32
4.13 Oh! Stylus Draw 3 の色変更 33

4.14 Oh! Stylus Draw 3 のペン先変更 33
4.15 Oh! Stylus Draw 3 の描画モード変更 34
4.16 Oh! Stylus Scroll によるスクロールの様子 35
4.17 Oh! Stylus 3D Viewer のスクリーンショット 36
4.18 OneNote で選択可能なペンの種類 38
4.19 Oh! Stylus OneNote Extension での swinging によるコピー&ペースト 38

5.1 評価実験 1 で用いるタブレット PC 41
5.2 Oh! Stylus Scroll の評価用 Web ページ 42
5.3 実験で用いたタッチスクリーン付き大画面ディスプレイ 45
5.4 タスク 1 で用いる Oh! Stylus Draw 3 のスクリーンショット 48
5.5 評価実験の結果 49
5.6 利き手による使用感の違い 52

6.1 本研究の位置付け 58

表 目 次

3.1 アクションの認識に用いる条件 15
3.2 rolling の認識に用いる条件 16
3.3 DP マッチングに用いる定数 20

5.1 Oh! Stylus Draw 1 の実験結果（単位：個） 43
5.2 Oh! Stylus Scroll の実験結果（単位：秒） 43
5.3 タスクの所要時間 (秒) 46
5.4 タスク 1 の所要時間 (秒) 48
5.5 タスク 2 の所要時間 (秒) 48
5.6 最も操作しやすかった方式の選択 50

第1章　序論

1.1　本研究のフォーカス

我々はコンピュータを操作するインタフェースとして，一般的にマウスとキーボードを用いている一方で，スタイラスを用いることも一般的になりつつある．スタイラスとは，コンピュータ操作に用いるペン型デバイスの一般的な呼称である．スタイラスを用いたコンピュータ操作におけるインタフェース上の最も大きな特徴は，画面の直接操作できることである．スタイラスを用いて画面を直接操作するコンピュータには，図 1.1 のように，PDA やスマートフォンなどの小型情報端末，タブレット PC，タッチスクリーン付き大画面ディスプレイなどがあり，さまざまなところで利用されている．例えばタブレット PC は教育現場や，美術家のアトリエで用いられており，大画面ディスプレイは大学や企業において電子ホワイトボードとして用いられている．

入力デバイスとしてスタイラスを採用する利点としては次のようなことが考えられる．例えば，小型情報端末では端末を小型化，軽量化できることである．また，タブレット PC や大画面ディスプレイでは画面を直接操作することが可能なため，絵を描く，文字を書くといったクリエイティブな作業が行いやすいという利点がある．

このようなインタフェース上の特徴や利点を持ち，マウスとキーボードでは行うことができない直感的なコンピュータ操作が実現できる一方で，スタイラスを用いるコンピュータシステムのユーザビリティは高くなく，現状では必ずしも使いやすいシステムであるとは言えない．このような現状について，次の 2 点が問題であると考えた．

- 入力デバイスとしてスタイラスを用いるシステムにおける WIMP[1] インタフェースの採用
- スタイラスがペン型デバイスであることを活用した入力操作の少なさ

まず，前者について述べる．現在多くのコンピュータのインタフェースには GUI（グラフィカル・ユーザ・インタフェース）と呼ばれるものが採用されている．GUI を用いるインタラクションでは，ユーザはグラフィカルなオブジェクトを操作するため，直感的な操作を行うことができる．GUI の中で，ウィンドウ，アイコン，メニュー，ポインティングデバイスから構成されている GUI を WIMP インタフェース（図 1.2 参照）と呼び，多くの GUI において WIMP インタフェースが採用されている [30].

しかしながら，WIMP インタフェースはポインティングデバイスとしてマウスを利用することを想定してデザインされているため，スタイラスを用いるシステムに WIMP インタフェー

[1] Window，Icon，Menu，Pointing device の頭字語.

(a) PDA　　(b) タブレット PC　　(c) タッチスクリーン付き大画面ディスプレイ

図 1.1: 入力デバイスにスタイラスを用いるさまざまなシステム

スを採用しても，ユーザビリティが高くなることは期待できない．

例えば，メニューバーの操作について考えてみる．メニューバーとは実行可能なメニューが表示されている部分の名称であり，Windows[2]の場合，アプリケーションウィンドウの上部に横一列で表示されており，Mac OS[3]の場合，デスクトップの上部に横一列で常時表示されているが，本質的には両者に大きな差異はない．スタイラスを用いてメニューバーを操作する場合，スタイラスのペン先がメニューバーを隠してしまうことで，ポインティングする場所をユーザが視認することができず，操作ミスを誘発することが少なくない．マウスを用いる場合には，そのような問題は発生しないことは明らかである．メニューバーに限らず，スクロールバーやアイコンなどの操作においても同様のことが言える．

スタイラスでの操作の場合，ペン先がディスプレイに接触したあとのストローク操作は非常に得意としている．しかしながら，ペン先がディスプレイに触れる瞬間の細かい座標の指定は簡単ではない．これにはペン先がディスプレイを隠してしまう問題の他に，視差やタッチパネルの解像度などの問題点が関わってくる．

このように，スタイラスを用いるシステムに WIMP インタフェースを流用することで，スタイラスを用いるシステムのユーザビリティの低下を招くさまざまな問題点を生み出している．

次に 2 つ目の問題点について述べる．スタイラスで行うことができる一般的な入力操作として次のような操作がある．

- タップ
- ストローク
- ダブルタップ
- バレルボタン＋タップ

[2]米 Microsoft 社の開発したオペレーティングシステム
[3]米 Apple 社の開発したオペレーティングシステム

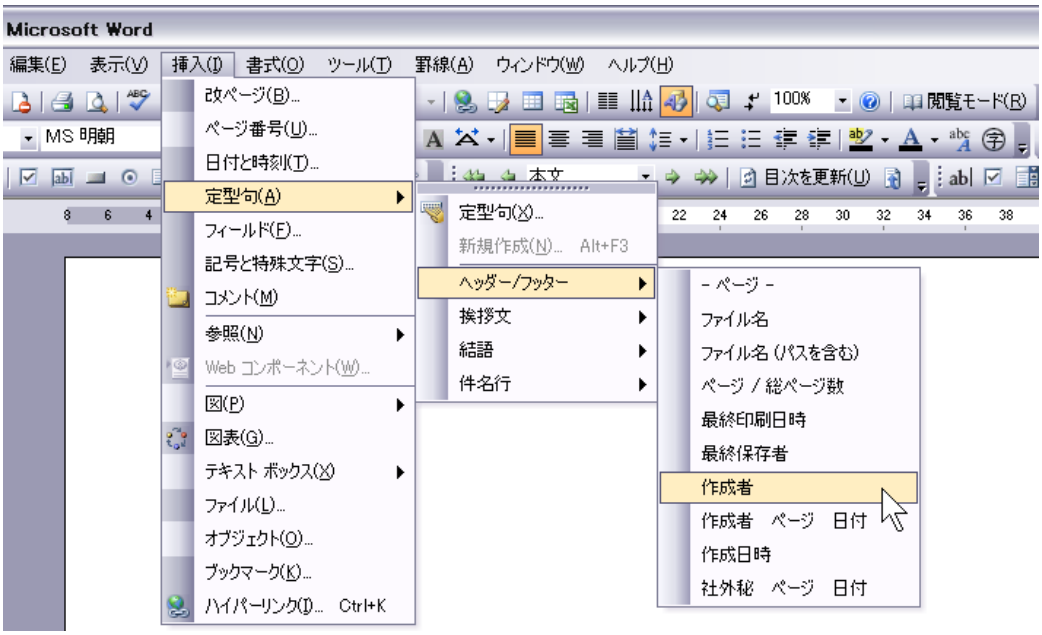


図 1.2: WIMP インタフェースの例

- ホールド

これらの操作の中で，スタイラスがペン型デバイスであることを活用した入力操作はタップとストローク（ペン先がディスプレイに接した状態での移動）のみである．ダブルタップとバレルボタン（スタイラスのグリップに付加しているボタン）＋タップはマウスで使われている操作を単純にスタイラス操作として取り入れようとしたために導入された操作であるため，スタイラスがペン型デバイスであることを活用していない．また，ホールドはバレルボタンの無いスタイラスにおいて，入力操作の少なさを補うために導入された操作であると思われる．スタイラスを用いる場合，一定時間１つの場所をポインティングしておくことは簡単な操作ではない．その上，一定時間スタイラスを留めておくこと自体にフラストレーションを感じるユーザも少なくない．よって，ホールドもペン型デバイスであることを活用していないといえる．

操作例として，コンテキストメニューを表示する操作について考えてみる．マウスを用いる場合，右クリックだけで簡単にコンテキストメニューを表示することができる．一方，スタイラスを用いる場合，タップ&ホールド（ディスプレイをタップし，一定時間ホールドする操作），もしくはバレルボタンのあるスタイラスの場合にはバレルボタンを押下しながらタップをするという煩雑な操作が要求される．スタイラスにもタップ以外の簡単に行うことができる入力手法があれば，このような問題は解決できる．

1.2　本研究の目的

本研究では，スタイラスを用いるコンピュータシステムにおけるインタラクション手段をより豊かにすることで上述した問題点を解決し，システムのユーザビリティを向上させることを目的としている．

一般的なコンピュータの利用シーンを考えると，入力デバイスとしてマウスを接続して利用することや，マウスを用いるシステムをタッチスクリーン付きディスプレイに接続し，入力デバイスとしてスタイラスを利用することも十分に考えられる．よって，広く普及している WIMP インタフェースとの互換性は必要である．

そこで本研究のアプローチとしては，現在用いられている WIMP インタフェースを完全に取り払い，全く新しいインタフェースの構築を目指すのではなく，WIMP インタフェースとの互換性を保持しつつ，スタイラスの特徴を活用した新しいインタラクション手法の構築を目指す．

1.3　本研究の貢献

本研究により，既存のスタイラスを用いるシステムで無視されていた空中での手の動きをインタラクションに活用することができるようになった．その結果として，以下のような問題点を解決することができた．

ユーザの視線遮断の問題

スタイラスをディスプレイに近づけたときにペン先がユーザとディスプレイの間に位置するため，ユーザの視線を遮断してしまう問題．

この問題を解決することで，誤認識などによるユーザのフラストレーションを軽減できる．

細かいポインティング操作に関する問題

タッチスクリーンの解像度，視差などに起因するタッチスクリーンに触れる瞬間の細かいポインティング操作の困難さ，および，このことにより発生する，メニュー操作などの本来の作業とは関係のない作業に意識や集中力を奪われる問題．

この問題を解決することで，スタイラスがタッチスクリーンに近づける際にペン先の位置を意識せずに操作が行えるようになり，本来の作業に意識や集中力を維持することができる．

ディスプレイサイズに起因する問題

50 インチ以上の大画面プラズマディスプレイでタッチスクリーンを用いる場合，ユーザの操作しているときの立ち位置によっては，画面の端まで手が届かないことがある．このよう

な場合，画面の端の配置されているメニュー類に手が届かず，メニュー操作のために立ち位置を移動する必要がある．

例えば，黒板ほどの大きさの大画面タッチディスプレイが導入されたときの将来の学校環境を考える．この問題を解決することで，授業の作業の効率化を図ることができ，その結果として教師は授業に集中することができ，さらに壇上の教師の左右への往来が減るため，生徒が授業へより集中できると考えられる．

1.4 論文の構成

第２章では，1.2 節で述べた本研究の目的を達成するための，本研究で提案するインタラクション手法であるショートカットアクションについて述べる．第３章では，ショートカットアクションを実現するための具体的なデバイスや実装方法について述べる．第４章では，ショートカットアクションを応用するアプリケーションと，その操作体系について述べる．第５章では，ショートカットアクションの評価について述べる．第６章では，本研究に関連する研究と，それに基づく本研究の位置付けについて述べる．そして，第７章では，本論文の結論について述べる．

第2章　提案手法：ショートカットアクション

本章では，筆者が提案するインタラクション手法であるショートカットアクションについて紹介する．まず，ショートカットアクションの基本アイデアについて述べる．そして，ショートカットアクションについて具体的に説明する．

2.1　基本アイデア

1.1 節で述べたように，ペン型デバイスであることを活用した入力操作が少ないことが問題点の１つであった．入力操作を増やす手段としては，例えばスタイラスを握っていない方の手を使用することや，スタイラスとキーボードを併用することなどが考えられる．しかしながら，PDA などの小型端末でスタイラスを用いる場合，スタイラスを握っていない手は小型端末を持っていることが想定される．タブレット PC や大画面ディスプレイでは物理的には可能であることが多いが，片手にスタイラスを持った状態での両手を使ったインタラクションは操作が煩雑になり，ユーザビリティが低下する恐れがある．また，入力デバイスとしてスタイラスを用いるシステムの環境には，キーボードが周りにないことが多いため，スタイラスとキーボードの併用は難しい．

そこで，筆者は入力操作を増やすために，スタイラスを握る手に着目した．スタイラスを握る手に着目すると，手の動きは以下の２つに分けることができる．

- ペン先がディスプレイに接した状態での手の動き
- 空中（ペン先がディスプレイに接してない状態）での手の動き

前者は既存のシステムにおいて入力操作として利用されている手の動きである一方，後者は既存のシステムでは全て無視されている動きである．空中での手の動きはマウスなどの入力デバイスでは行うことができない動きであり，ペン型デバイスの物理的な特徴を活かした動きである．そこで，筆者はスタイラスの特徴を活かしたインタラクション手法として，スタイラスを握る手の空中での動きを利用することを考えた．このインタラクション手法を実現することで，マウスでの操作体系を踏襲した，スタイラスを入力デバイスとして用いる従来のシステムの操作体系をよりペンらしい自然な操作体系に置き換えることができると期待される．

2.2 ショートカットアクション

前節で述べたアイデアを体系化したインタラクション手法として，ショートカットアクションを提案する．ショートカットアクションの基本となるアイデアは，スタイラスを握る手の空中での動きをインタラクションに利用することである．人間がペンを握った状態で行うことができる手の動きにはさまざまなものが存在するが，ショートカットアクションに採用する動作として，以下の３つの制約を満たす動作を採用した．

- ペンを握ったまま行える動作
- 人間にとって無理のない自然な動作
- 手元で行える動作

この３つの制約を課すことの意味について説明する．我々がよく行う動作としては，例えばペンまわし，ペンの上部をノックする動作などがある．これらの動作は空中での動きであるため，全てをインタラクションに利用することも可能である．しかしながら，インタラクションに利用する動作には，ペンを握ったまま持ち替えることなく行うことができる動作に限定するべきである．そして，それらの動作は人間にとって無理のない自然な動作である必要がある．さらに，動作を行うことにより掛かるユーザの身体的な負担はできる限り少なくすることが望ましい．よって，肩や腕を動かす動作は採用せず，手元を動かすだけで完結する動作に限定すべきである．この３つの制約を課すことで，ショートカットアクションに採用する動作を，ペンを扱うことを前提とした動作に限定している．

上記の３つの制約を全て満たす動作として，以下の３つの動作をショートカットアクションの動作に採用した．

(a) ペンをペン軸回りに回転させる動作

(b) ペンをペン軸方向に振る動作

(c) ペンをペン軸に垂直な方向に振る動作

これらの動作のことをスタイラスアクションと呼ぶ．また，それぞれのアクションを以下のように呼ぶ．

(a) rolling（図 2.1(a)）

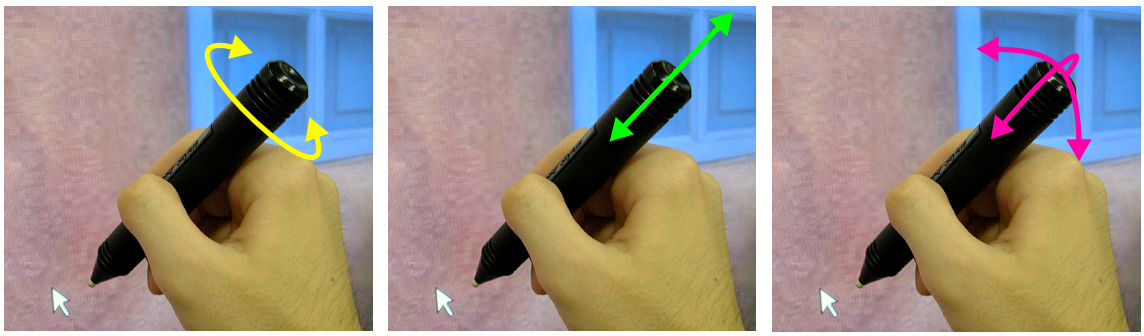
(b) shaking（図 2.1(b)）

(c) swinging（図 2.1(c)）

さらに，rolling には時計回り／反時計回りの２種類，swinging には North, South, East, West の４種類があり，それぞれ別の入力操作として扱うことも想定している．swinging はさらに細分化して５種類以上にすることも可能であるが，細分化することで得られる入力操作の数

とユーザビリティを比較して考えると，４種類が妥当ではないかと考えた．また，後述する４色ボールペンメタファを採用したことも swinging を４種類とした理由の１つである．

ショートカットアクションでは，これら３つのスタイラスアクションとその組み合わせをインタラクションに利用する．



(a) rolling　　(b) shaking　　(c) swinging

図 2.1: ３つのアクション

2.3　ショートカットアクションの設計

2.3.1　swinging の意味付け

swinging はペンをペン軸に垂直な方向に振る動作であるが，これを設計するにあたって以下の２通りの設計方針が考えられる．

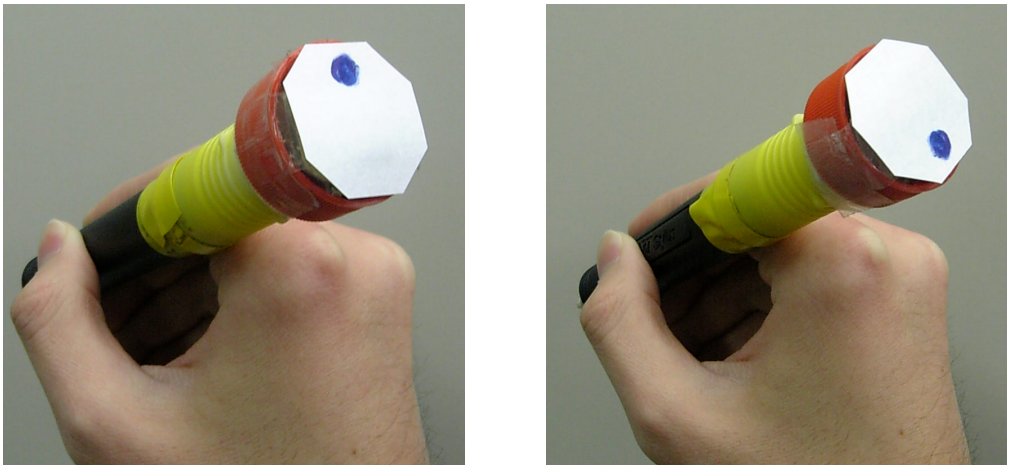
- スタイラス自体の方向に意味を持たせる設計
- スタイラスを振る方向に意味を持たせる設計

まず，前者の設計方針について説明する．前者はスタイラスを中心にした設計方針である．ユーザが swinging を行ったときの絶対的な方向に意味を持たせず，ユーザが swinging を行ったときのスタイラス自体の方向に意味を持たせる．よって，ユーザはスタイラスの向きを意識して swinging 動作を行わなければならない．例えば，ユーザが初め図 2.2(a) のようにペンを持っており，図 2.2(b) のようにペンを持ち替えた状況を考える．初めの状態でユーザが前方（図 2.2(a) の青色の方向）に swinging をしたときにコマンド N が起動すると仮定すると，ペンを持ち替えたあとでコマンド N を起動するときには初めとは異なる方向に swinging を行う必要がある．ペンを持ち替えたあとで初めと同じ方法（図 2.2(b) の青色と反対方向）に swinging を行うと，別のコマンドが起動することになる．

次に，後者の設計方針について説明する．こちらは前者とは対照的に，ユーザが swinging を行ったときの絶対的な方向に意味を持たせるため，スタイラス自体の方向は意味を持たな

い．よって，ユーザはスタイラスの向きを意識して swinging 動作を行う必要はない．例えば，ユーザが初め図 2.2(a) のようにペンを持っており，図 2.2(b) のようにペンを持ち替えた状況を考える．初めの状態でユーザが前方（図 2.2(a) の青色の方向）に swinging をしたときにコマンド N が起動すると仮定すると，ペンを持ち替えたあとも初めと同様にユーザの前方（図 2.2(b) の青色と反対方向）に swinging を行うことでコマンド N が起動する．

これら２つの設計方針にはどちらにも一長一短があるが，ここでは前者を採用した．前者を採用した理由としては，後述する４色ボールペンメタファにも関連するが，ペン自体の方向に意味を与えた方がユーザはより実世界に近い感覚を得ることができると考えたためである．



(a) ペンを持ち替える前の状態　　(b) ペンを持ち替えた後の状態

図 2.2: ペンを持ち替える様子

2.3.2　バレルボタンの使用について

スタイラスのグリップ部にはバレルボタン（図 2.3）というスイッチが付加されているものがある．1.1 節で述べたように，バレルボタンは単独では機能せず，タップとの組み合わせて操作することで初めて機能する．つまり，スタイラス操作の補助的な役割を担っている．スタイラス＋タップにより，マウス操作の右クリックに相当する操作を行うことができる．バレルボタンを用いることで通常のタップ操作とバレルボタン＋タップ操作という２種類の入力ができるようになり，入力操作の増加が期待できる．しかしながら，バレルボタンを押しながら操作するためにはボタンがちょうど指の位置にある必要があり，ユーザのペンの持ち方に制約を与えざるを得ない．ユーザビリティを考慮に入れたインタフェース設計をするとそのような制約を課すことは望ましくないため，本研究ではバレルボタンを使用しない設計としている．

図 2.3: バレルボタン（グリップ部にあるボタン）

2.3.3　ストロークジェスチャとの組み合わせについて

既存のスタイラス操作の１つとして，ストロークジェスチャによる操作がある．ジェスチャとは，ポインタで特定の軌跡（ストローク）を描くことで，それに応じたコマンドや機能が呼び出されるという仕組みである．ジェスチャはポインタの移動をインタラクションに利用しているため，ポインタの移動を通常のポインティング操作とジェスチャ操作に切り分ける必要がある．

ジェスチャはマウス操作でもマウスジェスチャとして利用されており，マウスジェスチャ機能を提供している多くのアプリケーションでは，右クリックを用いて通常の操作とジェスチャ操作の切り分けを行っている．一方，スタイラスではバレルボタンを用いて切り分けを行っている．前述したように，バレルボタンを使用することはユーザビリティの低下を招く恐れがあるため使用するべきではない．よって，ショートカットアクションにストロークジェスチャを用いない方針とした．

しかしながら，ジェスチャ自体は入力操作を増やすことができる有用な手段であるため，スタイラス操作において通常のポインティングとジェスチャの切り分けをショートカットアクションによって行えるようにすることは，ショートカットアクションの用途として考慮に入れるべきである．

2.3.4　ペンメタファの導入

振り子式シャープペンシルメタファ

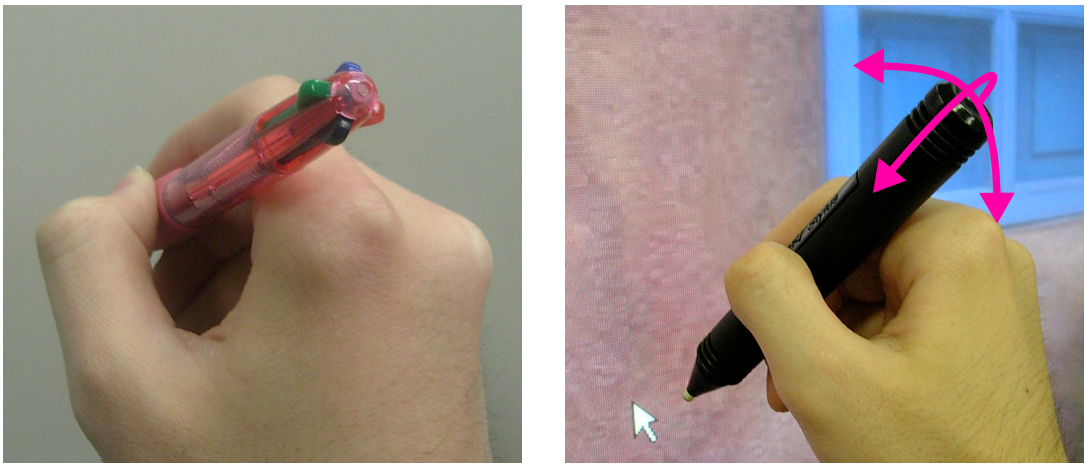
実在するペンに振り子式シャープペンシルというものがある．このシャープペンシルはペンを上下に振るだけで芯を出すことができる機構を備えているが，これはペンを上下に振る

ことに意味付けがされていると考えることができる．本研究では，この特徴を振り子式シャープペンシルメタファとして取り入れ，shaking の設計に応用している．

また，振り子式シャープペンシルはすでに普及しており，多くの人が利用した経験があると考えられる．さらに，このシャープペンシルを使い続けてペンを振って芯を出すことを繰り返すことにより，この行為が習慣化され，無意識的に行えるようになる．このような事象から，ショートカットアクションも使い続けることにより無意識的なインタラクションを実現できることが期待される．

４色ボールペンメタファ

実在するペンに４色ボールペン（図 2.4(a)）というものがある．４色ボールペンとはペンの上部に４つのノック部分があるものであるが，これはペンの４方向に “色” という意味付けがされていると考えることができる．本研究では，この特徴を４色ボールペンメタファとしてショートカットアクションの設計に取り入れている．具体的には，swinging（図 2.4(b)）を４方向に決定する際に用いている．



(a) ４色ボールペン　　(b) swinging の４方向

図 2.4: ４色ボールペンメタファ

2.4　関連する操作手法

我々が提案するスタイラスによるショートカットアクションは，キーボード操作におけるショートカットキー導入のアナロジーで考えることができる．

キーボードには独自のインタラクション方法として，ショートカットキーと呼ばれるものが用意されている．ショートカットキーでは，Ctrl+C や Ctrl+V などで利用頻度の高い機能を呼び出すことができるが，操作を行うためにマウスとキーボードを併用しなければならない．

一方，スタイラスによるショートカットアクションでは，スタイラスのみで操作することができるため，操作体系としてはショートカットアクションの方がシンプルであり優れている．

第3章　ショートカットアクションの実現

本章では，2 章で述べたショートカットアクションの実現方法について説明する．まず，空中での手の動きを取得するためのデバイスについて述べる．そして，取得した手の動きから rolling，shaking，swinging の 3 つのアクションを識別する方法について述べる．

3.1　Context Sensitive Stylus の作成

スタイラスを握る手の空中での動きを取得するために，スタイラスに加速度センサを付加した．加速度センサとは，物体にかかる加速度を推定することができるセンサであり，物体の傾き検知などに利用されている．本研究では，加速度センサに 3 軸の加速度情報が取得できる Cookie センサ（以下，Cookie）を利用した．Cookie と取得できる 3 軸方向を図 3.1 に示す．

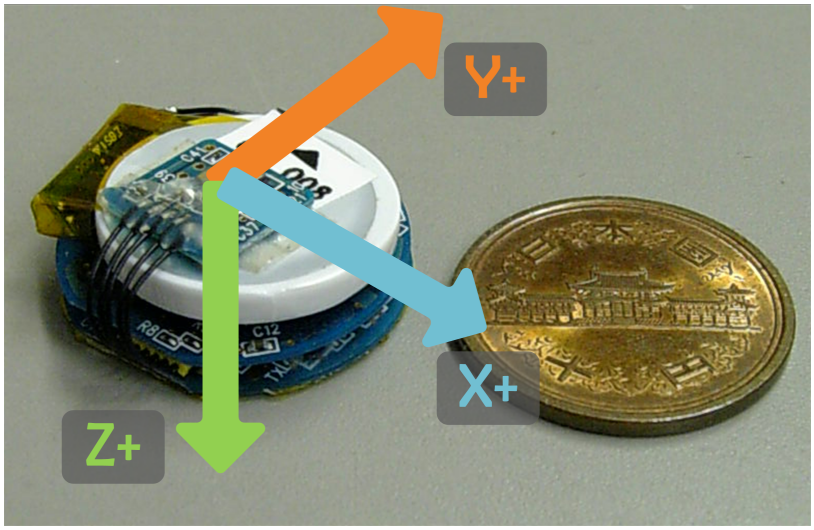


図 3.1: Cookie と取得できる 3 軸方向

Cookie は Nokia Research Center Tokyo が開発したコイン型センサである．Cookie は複合センサとなっており，3 軸加速度センサ，3 軸地磁気センサ，環境光センサなどさまざまなセンサが搭載されている．本研究で利用した Cookie の加速度センサは 1mG 単位で-3000mG ～+3000mG の加速度を時間分解能 10Hz で取得することができる．Cookie はバッテリーで駆動し，コンピュータとの通信には Bluetooth v1.1 Serial Port Profile を用いており，ワイヤレス

な通信を行うことができる．よって，スタイラスに付加してもスタイラスから電源ケーブルや通信ケーブルを延ばす必要がなく，物理的にスマートな加速度センサ付きスタイラスを作成することができる．また，Cookie をスタイラスに直接付加するのは物理的に困難であるため，ペンのキャップとペットボトルのキャップを用いて Cookie ホルダを作成した．

作成した加速度センサ付きスタイラスを Context Sensitive Stylus (以下，CS Stylus) と呼ぶ．本研究では，タブレット PC 用の CS Stylus（3.2(a)）と大画面用ディスプレイ用の CS Stylus（図 3.2(b)）を作成した．

(a) タブレット PC 用 CS Stylus

(b) 大画面ディスプレイ用 CS Stylus

図 3.2: 作成した 2 種類の CS Stylus

図 3.2 に示す CS Stylus はプロトタイプであるため，スマートな形状であるとは言えない．また，スタイラスの上部に加速度センサを付加したことでスタイラスの重量バランスが変化することが懸念されるが，Cookie の重量はそれほど重くないためユーザの使用感に悪い影響を与えることはないと思われる．今回作成した CS Stylus において最も重要なポイントはコード類が一切ないことである．仮に重要バランスに多少の変化があったとしても．コード類がないためユーザは通常のスタイラスと同等の使用感を得ることができる．

3.2 アクションの認識方法

CS Stylus によりスタイラスにかかる 3 軸の加速度情報を取得できるようになったが，3 軸の加速度情報から rolling，shaking，swinging の 3 つのアクションを認識する必要がある．本研究ではこのアクションの認識方法について試行錯誤を行い，アクション認識の再現率の向上を図った．

3.2.1 認識に使用するデータ

3.1 節で述べたように，Cookie には加速度センサのほかに地磁気センサも内臓している．地磁気センサとは地球の磁場を計測することができるセンサであり，方位を調べることができる．よって，地磁気情報を用いるとスタイラスのペン軸回りの回転を比較的容易に識別することができる．しかしながら，地磁気センサを使う際には初めに必ずキャリブレーションをする必要があるため，本研究におけるアクションの認識には，基本的に加速度情報のみを利用する．

3.2.2 方法１：単純な加速度情報の比較

最も簡単な認識方法として単純な加速度情報の比較と，２軸の加速度情報から算出できる擬似的な回転角度を用いた認識アルゴリズムを実装した [24].

まず，shaking と swinging の認識アルゴリズムについて述べる．shaking と swinging の動作軸は，加速度センサから取得できる加速度軸と同軸である（図 3.3）．そのため，ある時刻 τ（τ は 100ms 単位の離散的な時刻）における X 軸方向の加速度 X_τ(もしくは Y 軸方向の加速度 Y_τ，Z 軸方向の加速度 Z_τ) と，その 100ms 前の加速度 $X_{\tau-1}$(もしくは $Y_{\tau-1}$, $Z_{\tau-1}$) を比較することで認識できる．認識に用いた閾値は shaking では 2000mG，swinging では 1500mG とした．これらのアクション認識の条件をまとめると，表 3.1 のようになる．

表 3.1: アクションの認識に用いる条件

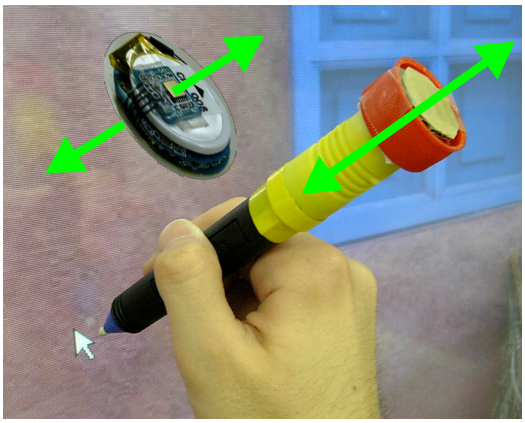
アクション	条件
shaking	$\lvert Z_\tau - Z_{\tau-1}\rvert > 2000$
swinging (North)	$X_\tau - X_{\tau-1} > 1500$
swinging (South)	$X_\tau - X_{\tau-1} < -1500$
swinging (East)	$Y_\tau - Y_{\tau-1} > 1500$
swinging (West)	$Y_\tau - Y_{\tau-1} < -1500$

次に，rolling の認識アルゴリズムについて述べる．rolling の認識には X 軸と Y 軸の２つの加速度情報から算出できる擬似的な角度 θ（図 3.4）を用いた．ある時刻 τ における擬似的な角度 θ_τ は以下の式 3.1 で表すことができる．

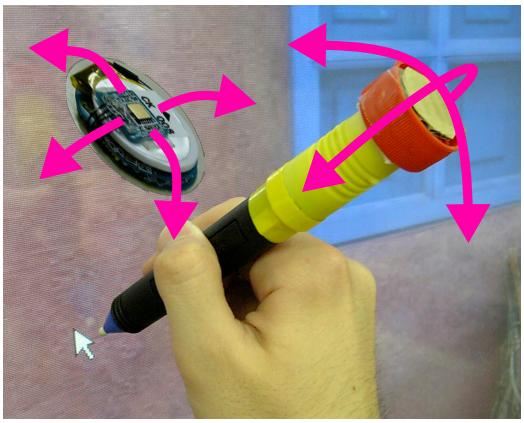
$$\theta_\tau = \tan^{-1}(Y_\tau / X_\tau) \tag{3.1}$$

時刻 τ の 100ms 前の角度 $\theta_{\tau-1}$ も同様の式で表すことができる．時刻 τ と時刻 $\tau-1$ との角度差分 δ_τ は以下の式 3.2 で表すことができる．

$$\delta_\tau = (\theta_\tau - \theta_{\tau-1}) \bmod 360 - 180 \tag{3.2}$$



(a) shaking の動作軸と Cookie の加速度軸



(b) swinging の動作軸と Cookie の加速度軸

図 3.3: shaking，swinging の動作軸と Cookie の加速度軸

rolling を認識する角度差分 δ_τ の閾値は 50 度とした．よって，表 3.2 に示す条件を満たすときに rolling が発生したと認識する．

表 3.2: rolling の認識に用いる条件

アクション	条件
rolling (時計回り)	$\delta_\tau > 50$
rolling (反時計回り)	$\delta_\tau < -50$

3.2.3 方法２：方法１のアルゴリズムの改善

方法１の rolling 認識アルゴリズムでは，多くの誤認識が発生した．この誤認識の原因は rolling 以外の動作により発生する，ユーザの意図しない X 軸と Y 軸の加速度の変化であった．そこで，rolling 認識の再現率を向上させる方法として，角度差分 δ_τ を用いた以下の３つのアルゴリズムを考案し，実装を行った [23].

差分同期法

差分同期法とは，スタイラスを回転させたときの回転角度の差分 δ_τ が一定値以上であれば，その操作が rolling であるとすぐに認識する方法である．メリットは，ユーザの操作からアプリケーションの反応（フィードバック）まで時間的にギャップがないことであるが，手ぶれによる rolling の誤認識が多発するというデメリットがある．

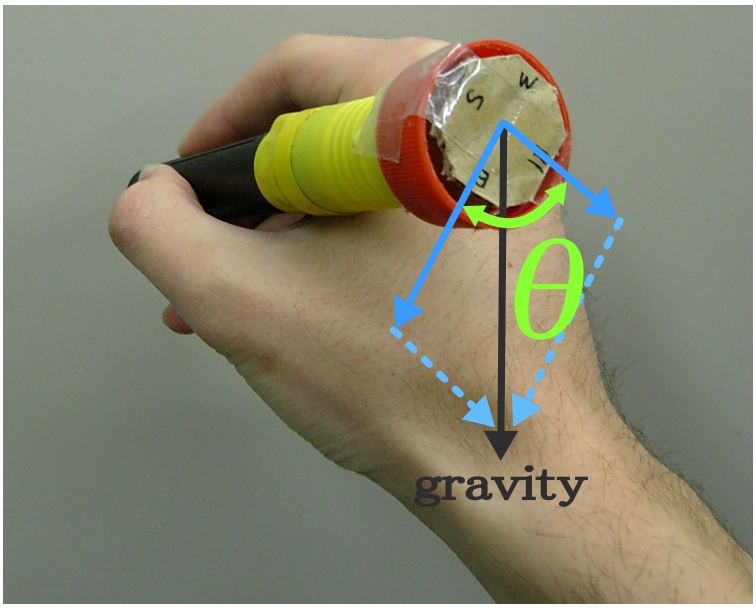


図 3.4: 求める θ のイメージ図

回転時間法

時間同期法とは，スタイラスを回転させ始めてから 500ms 間同じ方向の回転，つまり，500ms 間，角度差分 δ_τ が一定値以上である状態が続けば，500ms 以降の同じ方向の回転を rolling であると認識する方法である．この方法は，差分同期法と比較して手ぶれによる誤認識が発生しにくいというメリットがあるが，rolling を認識しにくくなるというデメリットがある．

要素比較法

要素比較法は，ノイズを除去するためのアルゴリズムである．一定方向への rolling 条件を満たしているときに，角度差分 δ_τ がその方向の rolling 条件を 3 回連続して満たさなくまで rolling であると認識する．つまり，2 回連続したノイズを除去することができる．この方法は，手ぶれなどによる値のノイズを除去することができるというメリットがあるが，回転動作を止めたときに最大 2 回分の rolling を余計に認識してしまうことがあるというデメリットも持つ．

アルゴリズムの組み合わせ

各アルゴリズムを検証した結果，回転時間法と要素比較法とを組み合わせて rolling の認識を行うことで，手ぶれの影響を抑えることができる精度の高い認識ができるということがわかった．方法 1 における rolling 認識の再現率と比較して，このアルゴリズムを組み合わせでは再現率の向上が確認できた．

3.2.4 方法３：DP マッチング

方法３では前述した方法１，２とは全く異なるアプローチをとった．方法１，２では閾値を用いた認識アルゴリズムであったが，方法３ではパターンマッチングを利用した．パターンマッチングにはさまざまな手法が存在するが，本研究では汎用性が高く，高精度のマッチングを期待することができる DP マッチングを用いた．DP マッチングとは，２つのパターンの類似度を動的計画法を用いて計算する方法である．文字列のマッチングなどにおいて完全一致を求めることを前提とすると，必要な定数はパターンとの一致／不一致を判定する距離の値のみである．しかしながら，本研究では数値のマッチングであるため，パターンとの一致／不一致を判定する距離の値（lpj）に加えて，要素値（個々の加速度値）の照合許容誤差（e），パターン長（pl），パターンの照合に用いる最大データ長（dlmax）と最小データ長（dlmin）が必要である．照合データ長を長くすれば照合できる可能性があがるが，長くすればするほど計算量が膨大になるため，最大のデータ長が必要になる．

計算の手順について述べる．例えば登録パターン A={a1, a2, a3}，照合データ M={m1, m2, ..., m10} とし，各定数を lpj=5，e=100，pl=3，dlmax=5，dlmin=3 とする．

まず，dlmin が 3 であるので，登録バターンの３要素 a1, a2, a3 と照合データの初めの３要素 m1, m2, m3 でマッチングを行う．そのとき，各要素の誤差が照合許容誤差 e よりも小さければ，その要素同士は照合すると見なす．これを全ての要素について行い，最終的な距離を算出する．最終的な距離が lpj よりも近ければマッチングと見なす．ここでマッチングされなかった場合，照合データの要素を１つ追加して，登録バターンの３要素 a1, a2, a3 と照合データの初めの４要素 m1, m2, m3, m4 を用いて，上記と同様の手順でマッチングを行う．マッチングに用いる照合データの要素数が dlmax になるまでこれを繰り返す．この例では a1, a2, a3 と m1, m2, m3, m4, m5 のマッチングまで行う．そして，照合データの要素数が dlmax になるまでこれを繰り返したあとは，照合データを１つずらして２番目以降の３要素 m2, m3, m4 でマッチングを行う．そのあとの処理は上記と同様の手順である．このように，DP マッチングを用いて部分一致を求めている．この計算手順を図で表したものが図 3.5〜3.10 である．

この計算方法を用いてアクションの認識を行っている．まず，ユーザがあらかじめ各アクションのパターンを登録しておく．登録するパターンは以下の７パターンである．

- rolling（時計回り）
- rolling（反時計回り）
- shaking
- swinging（North）
- swinging（South）
- swinging（East）
- swinging（West）

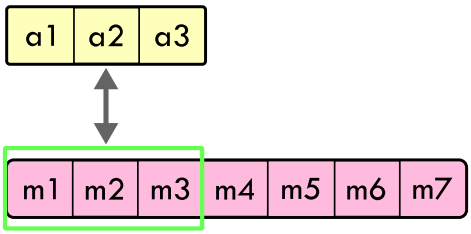


図 3.5: 手順 1 : 登録バターンの 3 要素と照合データの初めの 3 要素をマッチングする

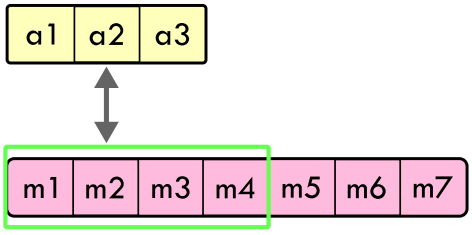


図 3.6: 手順 2 : 照合データ長を 1 増やし，登録バターンの 3 要素と照合データの初めの 4 要素をマッチングする

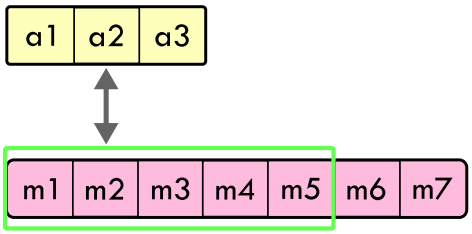


図 3.7: 手順 3 : 照合データ長をさらに 1 増やし，登録バターンの 3 要素と照合データの初めの 5 要素をマッチングする

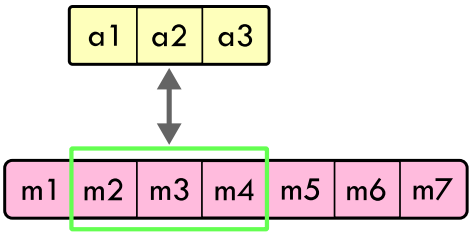


図 3.8: 手順 4 : 照合データの始点を 1 つ進めて照合データ長を 3 に戻し，登録バターンの 3 要素と照合データの 2 番目からの 3 要素をマッチングする

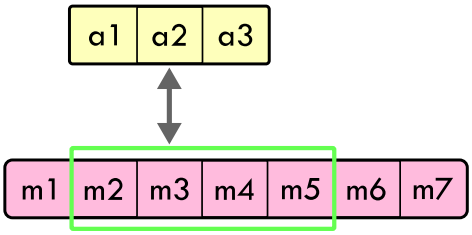


図 3.9: 手順 5 :照合データ長を 1 増やし，登録バターンの 3 要素と照合データの 2 番目からの 4 要素をマッチングする

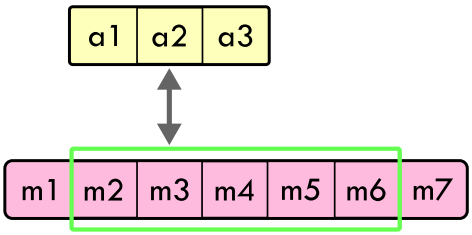


図 3.10: 手順 6 : 照合データ長を 1 増やし，登録バターンの 3 要素と照合データの 2 番目からの 5 要素をマッチングする．その後は照合データの始点を 1 つずつ進めながら同様の処理を繰り返す

次に，１軸の加速度情報（Z 軸）と１つのパターン（shaking）を用いてマッチングに必要な定数を決定する実験を行った．その結果，各定数の値は表 3.3 のように決定した．

表 3.3: DP マッチングに用いる定数

定数	値
lpj	5
e	100〜300
pl	5
$dlmax$	7
$dlmin$	5

この定数と登録した７つのパターンを用いて DP マッチングを行った結果，shaking の再現率は 90 %以上であった．rolling と swinging の再現率は shaking ほど高くはなかったが，アクション認識の再現率は方法１，２と比較して向上したことが確認できた．

第4章　ショートカットアクションの応用

本章では，ショートカットアクションの応用例について述べる．まず，ショートカットアクションの応用に関する議論として，各アクションへのコマンドの割り当てとフィードバックについて説明する．そして，ショートカットアクションを応用した具体的なアプリケーションについて述べる．

4.1　応用に関する議論

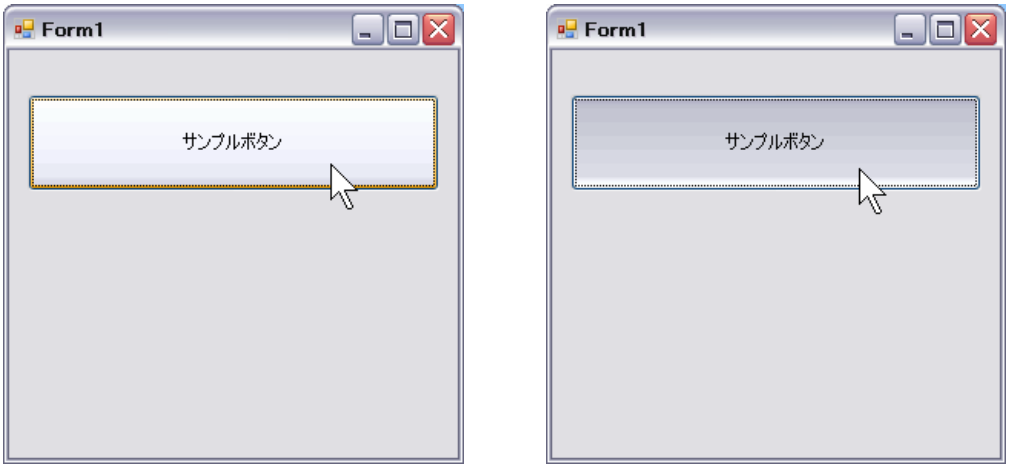
4.1.1　アクションへのコマンドの割り当て

ショートカットアクションを利用するには，３つのアクション（rolling，shaking，swinging）をそれぞれ特定のコマンドに割り当てる必要がある．ショートカットアクションを使い心地の良いインタラクション手法にするためには，割り当ては重要であると思われる．ショートカットアクションの設計を始めた当初は割り当ての方法として,「人間が実際にペンを用いるときの動作を割り当てる」という方法を考えた．この方法のメリットは，ユーザにとって操作が直感的になることであった．しかしながら，この方法では応用できるアプリケーションやコマンドの種類が非常に限られてしまう．本研究では，入力デバイスとしてスタイラスを用いるシステム全体のユーザビリティの向上を目指しているため，あらゆるアプリケーションに応用できることが望ましい．

そこで，本研究ではそれぞれのユーザが自分の好みに合わせたアクションとコマンドの割り当てを行えるようにした．初期の割り当ては一般的に使用頻度の高いと思われるコマンドに対してアクションを割り当てている．

4.1.2　アクションに対するフィードバック

インタフェースに関する研究においてフィードバックは非常に重要である．ここでは，フィードバックをハードウェアによる物理的なフィードバックとソフトウェアによる電子的なフィードバックの２種類に分類して考える．物理的なフィードバックとは人間の触覚に対する働きかけるものであり，電子的なフィードバックとはコンピュータの出力であり，人間の視覚や聴覚などに働きかけるものである．例えば，マウスにおける物理的なフィードバックはマウスを押下したときのクリック感であり，電子的なフィードバックはディスプレイ上のボタンをクリックしたときのボタンが凹むアニメーション（図 4.1）などである．



(a) カーソルをのせたときのボタン　　(b) 押下したときのボタン

図 4.1: ボタンのフィードバック

ショートカットアクションでは，アクションに対するフィードバックとして電子的なフィードバックを用いた．例えば，電子音やディスプレイ上に表示されるグラフィカルなフィードバックである．ショートカットアクションを応用した個々のアプリケーションにおけるフィードバックについては，それぞれ後述する．

4.2　実装に用いた基本技術

アプリケーションの作成には以下の言語や API を用いている．

- Java™Platform, Standard Edition 6
- Java™Communications API
- Java™APIs for Bluetooth (BlueCove)
- Java™Binding for the OpenGL®API (JOGL)
- The Standard Widget Toolkit (SWT)

全てのアプリケーションの作成にはプラットフォームとして Java を採用している．Java の標準ライブラリの他に，Java™Communications API と BlueCove は Cookie に搭載されたファームウェアから Bluetooth による通信で加速度データを受け取る際に用いている．JOGL と SWT は 3D 描画や特殊なデスクトップアプリケーションを作成するために用いている．

4.3 Oh! Stylus Series

ショートカットアクションを応用するアプリケーションとして，お絵かきツールや既存アプリケーションのインタフェース拡張ツールなど，さまざまな種類のアプリケーションを開発した．これらショートカットアクションを応用するアプリケーション群を Oh! Stylus Series と呼ぶ．ここでは Oh! Stylus Series に含まれるアプリケーションについて説明する．

4.3.1 Oh! Stylus Draw

コンピュータの入力デバイスとしてスタイラスを用いる場合の面白さやメリットを手軽に享受できるアプリケーションは手書きで文字や絵を描くことができるお絵かきツールである．そこで，お絵かきツール Oh! Stylus Draw を開発した．Oh! Stylus Draw は数回のユーザスタディを経て，機能の拡張とインタフェースの改良を繰り返した．

Oh! Stylus Draw 1

Oh! Stylus Draw 1[38] は最初に作成したものであり，主に以下の２つの特徴を持つ．

(i) rolling によってカラーパレット内の４色を選択できる

(ii) 選択中の色と描画モードをマウスポインタ周辺に表示させる

Oh! Stylus Draw 1 を実行した様子を図 4.2 に示す．左端に配置してあるパネルはツールパネルであるが，このパネルはユーザが自由に場所を移動できるようになっている．ツールパネルの内，上の４つは描画モードを表している．上からそれぞれ自由線，直線，矩形，楕円である．描画モードの各パネルをタップすることで，描画モードを切り替えることができる．

また，ツールパネルの最も下にある円形パネルはカラーパレットを表している．このパネルには常に４色を置いておくことができ，色を変更したい場合には４色のうちのどこかをタップすることで新しい色をセットすることができる．rolling 操作に同期してカラーパレットの周りを小さな円が回る（図 4.3 左参照）．この小さな円がある場所の色が選択中の色として扱われる．この機能により，色選択を行う度にカラーパレットを選択する煩わしさを低減できる．

特徴 (ii) のマウスポインタ周辺の表示について，4.3 右に示す．マウスポインタの右にある表示は，選択中の描画モードを表しており，マウスポインタの下にある表示は，選択中の色を表している．この表示により，特徴 (i) の色選択機能のフィードバックを得やすくなる．

Oh! Stylus Draw 2

Oh! Stylus Draw 2[24] は Oh! Stylus Draw 1 を用いたユーザスタディを基にカラーパレットの機能と操作インタフェースを改良したものである．Oh! Stylus Draw 2 では実験的な試みとしてバレルボタンを操作体系に取り入れた．主な特徴は以下の３点である．

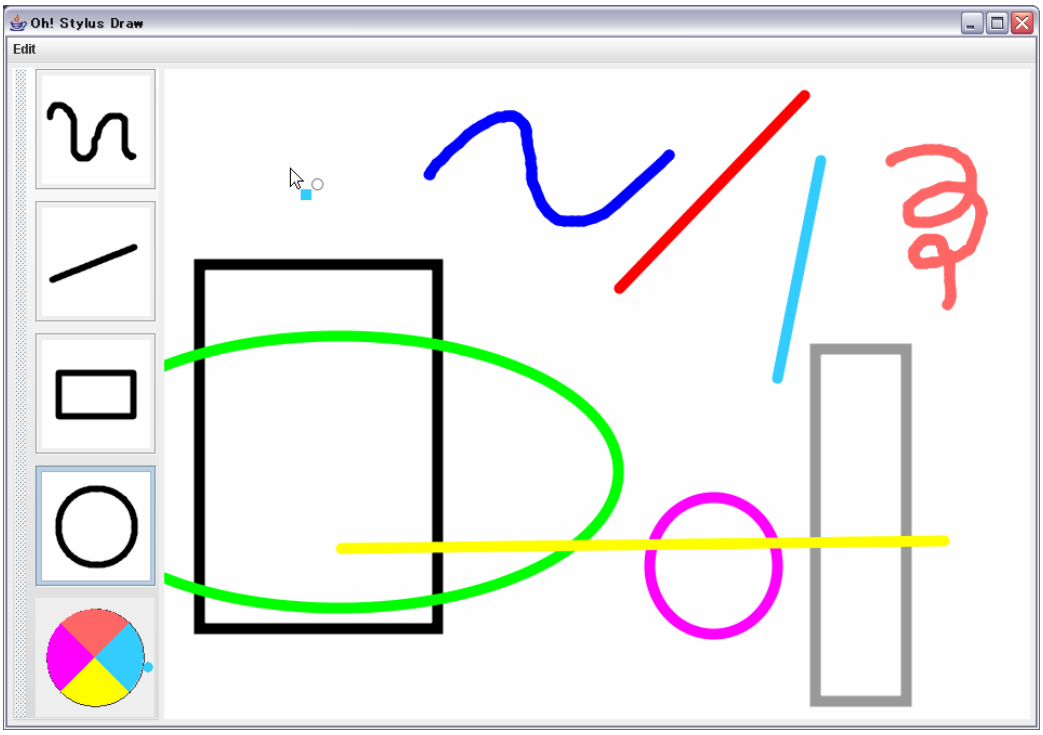


図 4.2: Oh! Stylus Draw 1 のスクリーンショット

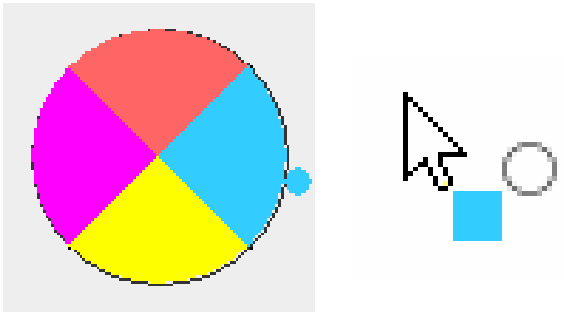


図 4.3: Oh! Stylus Draw 1 のカラーパレットとマウスポインタの拡大図

(i) rolling による描画色の変更

(ii) shaking によるカラーパレットの切り替え

(iii) バレルボタン＋タップ＋ rolling による描画モードの切り替え

Oh! Stylus Draw の実行例を図 4.4 に示す．左端に配置してあるパネルはツールパネルであるが，このパネルはユーザが自由に場所を移動できるようになっている．ツールパネルの内，上の４つは描画モードを表している．上からそれぞれ自由線，直線，矩形，楕円であり，こ

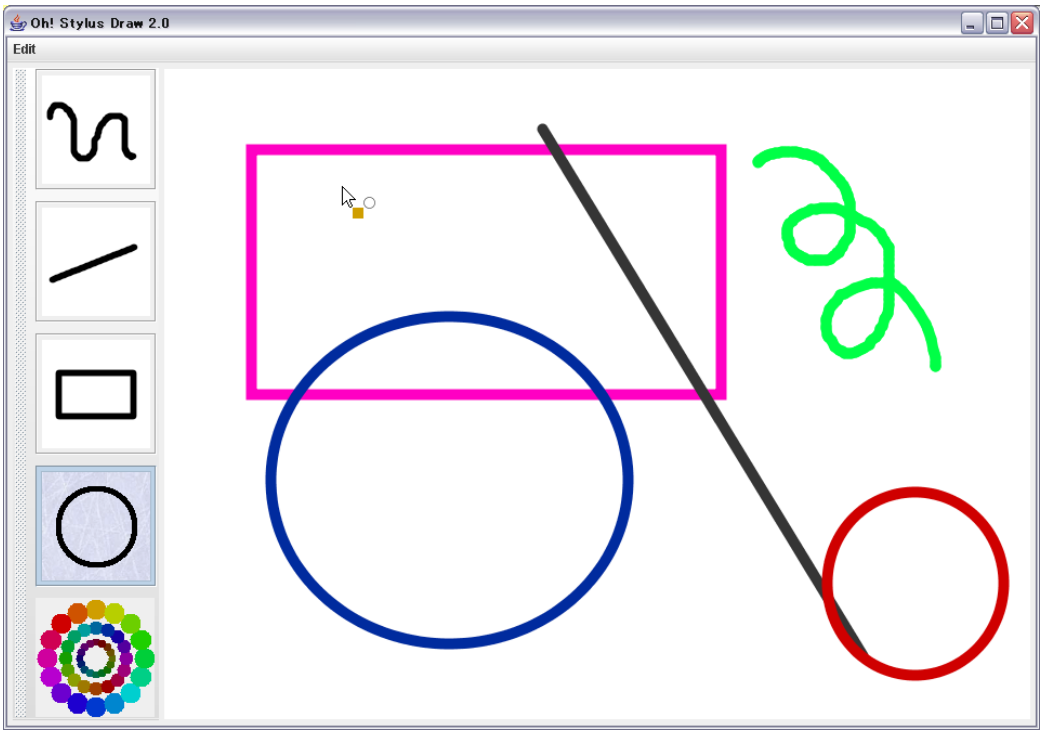


図 4.4: Oh! Stylus Draw 2 のスクリーンショット

れは Oh! Stylus Draw 1 と同様のインタフェースである．

Oh! Stylus Draw 1 と同様に，描画モードの各パネルをタップすることで，描画モードを切り替えることができる．これに加えて，Oh! Stylus Draw 2 では，バレルボタンの押下＋タップ＋ rolling によって描画モードを切り替えることができる．ユーザがスタイラスを時計回りに rolling すると，描画モードは 1 つ下に移動する（現在の描画モードが最も下の場合は最も上に移動する）し，ユーザがスタイラスを反時計回りに rolling すると，描画モードは 1 つ上に移動する（現在の描画モードが最も上の場合は最も下に移動する）．描画モードを切り替えた際にはフィードバックとして音が鳴る仕組みも導入している．

最も下にあるパネルはカラーパレットパネルである．カラーパレットパネルには，図 4.5 に示すように 3 つのゾーン（first zone, second zone, third zone）がある．カラーパレットは全部で 5 つあり，shaking によりこれらが順番に入れ替わる．first zone にあるカラーパレットはアクティブであり，ここから描画色を選択する．shaking を行うと second zone にあるカラーパレットは first zone に移動し，アクティブになる．同時に，third zone にあるカラーパレットは second zone に移動する．third zone には次のカラーパレットが表示される．カラーパレットが入れ替えられる際の動きはアニメーションになっており，カラーパレット入れ替えのフィードバックがわかりやすくなっている．また，カラーパレットが入れ替わる際にも音によるフィードバックを行っている．

１つのカラーパレットは１６個の円で構成されており，それぞれが描画色になっている．カラーパレットは全部で５つなので，描画色数は１６×５＝８０色である．好みの描画色がカラーパレットにない場合には，小さな円をタップすることでその色と好みの色とを入れ替えることができる． rolling によってカラーパレット内から描画色を選択することができる．描画色はカラーパレットの最上部に表示されている色である．スタイラスを時計周りに rolling させたときにカラーパレットが時計回りに回転して描画色が変更され，反時計回りに rolling させたときにカラーパレットが反時計回りに回転して描画色が変更される．カラーパレットが回転する際の動きはアニメーションになっており，描画色変更のフィードバックがわかりやすくなっている．また，カラーパレットが回転する際にも音によるフィードバックを与える仕組みを導入している．

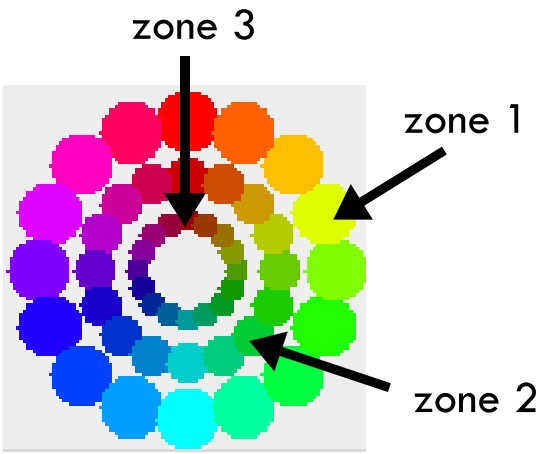


図 4.5: Oh! Stylus Draw 2 のカラーパレットの拡大図

Oh! Stylus Draw 2 swing

Oh! Stylus Draw 2 swing[23, 39] は Oh! Stylus Draw 2 のカラーパレット以外のインタフェースを一新したアプリケーションである．Oh! Stylus Draw 1,2 で左端にあったツールパネルを廃止し，新しいインタフェース “FlowButton” を採用した．FlowButton とは図 4.6 のようなパネル型メニューであり，ユーザによってメニューが呼び出されたときに画面上に浮き上がってくる．また，Oh! Stylus Draw 2 swing では rolling と shaking に加えて，操作に用いるアクションに swinging を導入した．

Oh! Stylus Draw 2 swing の主な機能は以下の４点である．

(i) 描画色の変更

(ii) 描画モードの切り替え

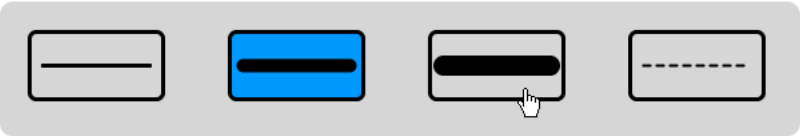


図 4.6: Oh! Stylus Draw 2 swing における FlowButton の表示例

(iii) ペンの太さの切り替え

(iv) UNDO

Oh! Stylus Draw 2 swing でこれらの機能を選択する方法として，階層メニューから選択する方法と，ショートカットアクションから選択する方法の 2 通りがある．階層メニューから選択する方法は既存の多くのアプリケーションと同様の操作である．まず，ウィンドウの上部に配置されているメニューバーから項目を選択し，次にその項目の中から起動したいコマンドなどを選択するという操作の流れである（図 4.7）．

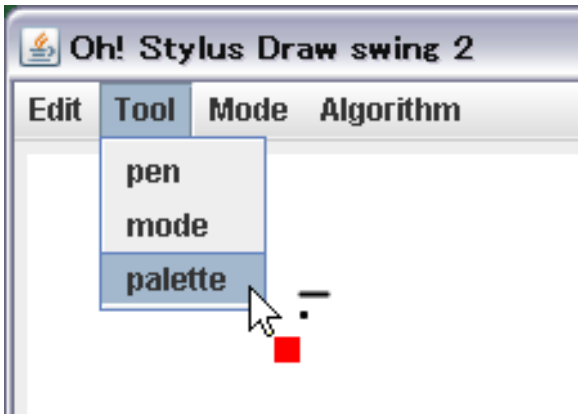


図 4.7: Oh! Stylus Draw 2 swing のメニューバー

ショートカットアクションによるメニューの選択では，swinging により FlowButton の呼び出しを行い，rolling と shaking により描画色の変更やパレット切り替えを行う．swinging によって呼び出される FlowButton はスタイラスで操作しやすい大きめのパネルに配置されてユーザに提示される．WINP インタフェースを採用している多くのアプリケーションが持つアイコンメニューとは異なり，スタイラスでの操作を考慮に入れたインタフェースデザインになっている．

各アクションとメニュー呼び出しの詳細は以下のようになっている．

- rolling による描画色の選択
- shaking によるパレットの切り替え

- swinging による

    1. 描画モードを切り替えるパネルの表示
    2. ペン先を切り替えるパネルの表示
    3. UNDO
    4. カラーパレットの表示／非表示の切り替え

図 4.8 に swinging でペン先を切り替える FlowButton を表示している例を示す．swinging により，図 4.8 の画面中央に表示されているようなパネルが表示され，そのパネルの中から好みのペン先をタップ操作によって選択する．

また，図 4.9 に swinging でペン先を切り替える FlowButton を表示している例を示す．ペン先を切り替えるパネルと同様の動作で，描画モードを切り替える．

このように，呼び出された FlowButton がディスプレイの中央に大きく表示されるため，その中から好みの機能を選択することができ，メニュー操作のために意識を大きく奪われることがなくなる．よって，本来の作業に集中することができるようになる．

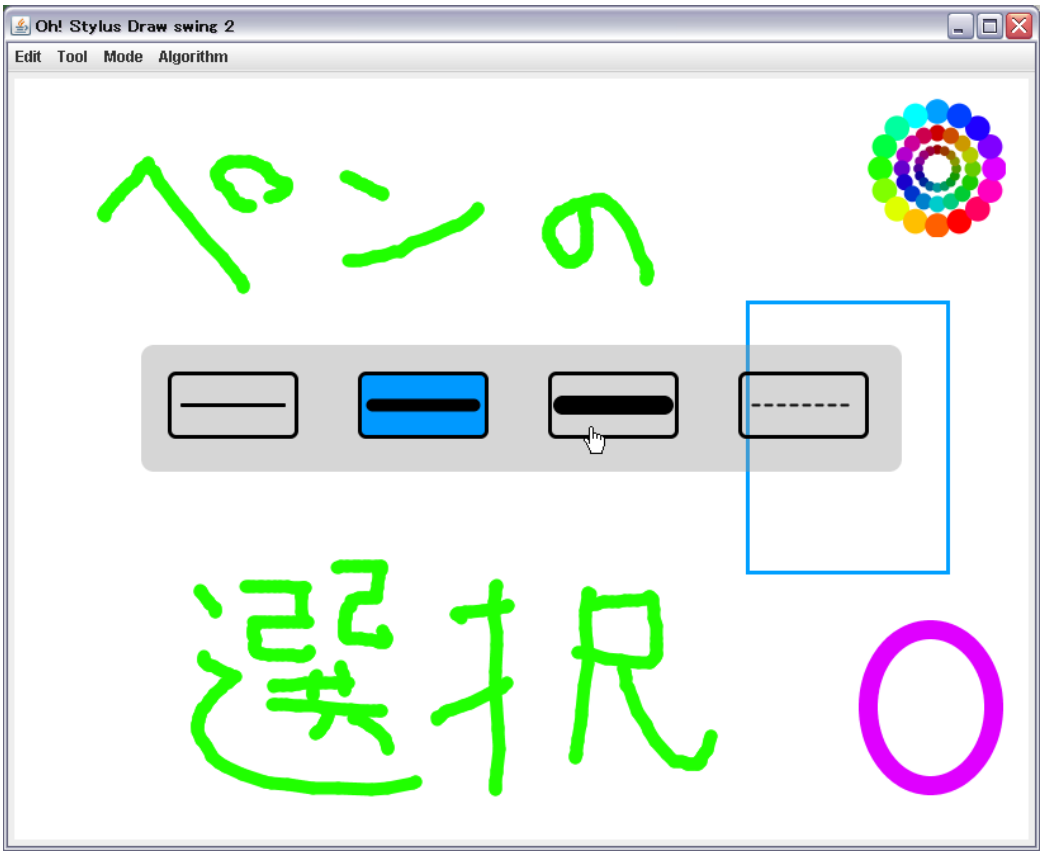


図 4.8: Oh! Stylus Draw 2 swing のペン先を切り替える FlowButton の表示例

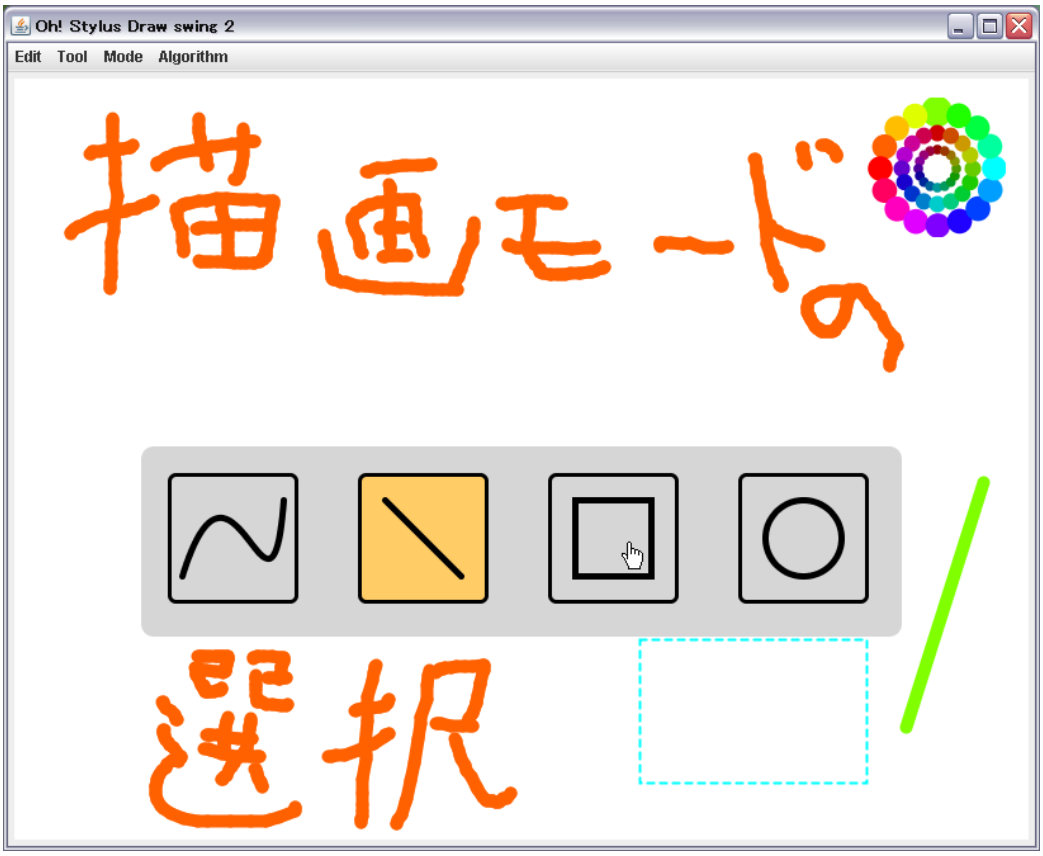


図 4.9: Oh! Stylus Draw 2 swing の描画モードを切り替える FlowButton の表示例

Oh! Stylus Draw 3

Oh! Stylus Draw 3 は Oh! Stylus Draw 2 swing までの開発で得た，お絵かきツールに必要なインタフェースや機能，ノウハウなどを活用したお絵かきツールである（図 4.10）．操作インタフェースには，Oh! Stylus Draw 2 swing で導入した FlowButton を採用した．FlowButton の表示位置は，画面の中央ではなくポインタを中心とした位置に変更した．また，マウスでの操作との比較を行えるようにメニューバー形式のメニューも採用した．カラーパレットは Oh! Stylus Draw 2 swing などで採用していた円形パレットを廃止し，これにも FlowButton を採用した．これは，アプリケーション全体の操作の一貫性を保つためである．さらに，描いた図形の扱いとして Oh! Stylus Draw 2 swing までは描いた図形をラスタ画像として扱っていたが，Oh! Stylus Draw 3 ではベクター画像として扱うように変更した．これにより，一度描いた図形を移動，回転などをアフィン変換を情報を失うことなく容易に行うことができる．また，新しい機能として Oh! Stylus Draw 3 ではレイヤー（図 4.10 の左端のオレンジ部分）を導入した．これにより，１つの絵を複数のレイヤーに分割して構成することができるようにした．さらに，一般的な形式の画像ファイルの読み込みや，描いた図形を Scalable Vector Graphics(SVG)

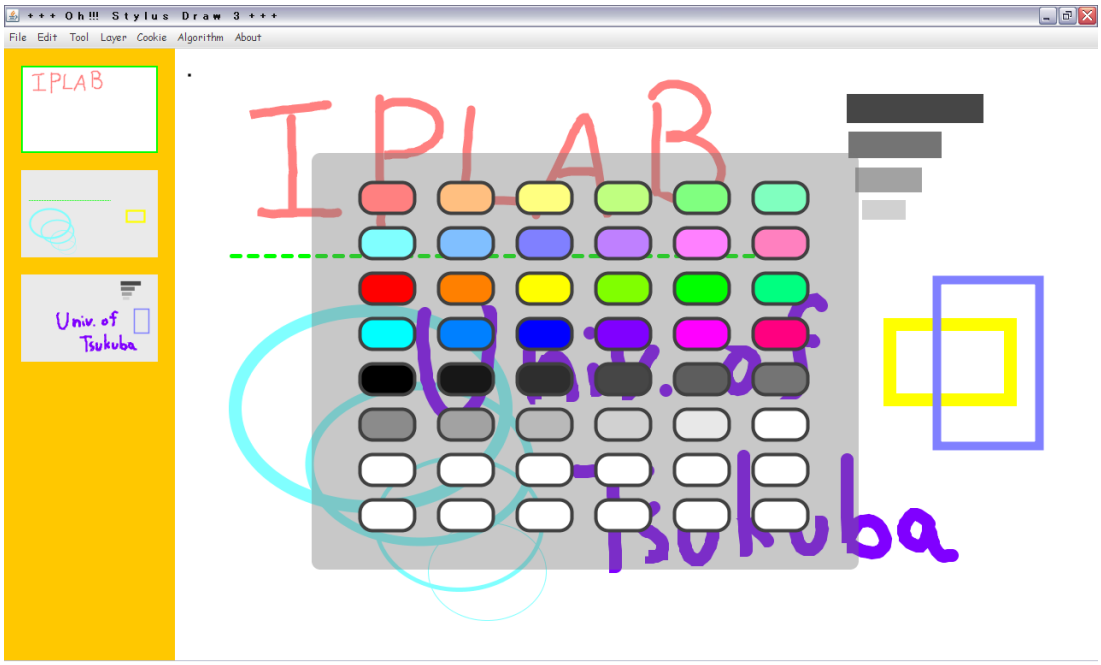


図 4.10: Oh! Stylus Draw 3 のスクリーンショット

形式で保存することも可能である．

Oh! Stylus Draw 3 における各アクションとそれに対応するメニューやコマンドは以下の通りである．

- rolling（時計回り）…カラーパレット FlowButton の表示
- rolling（反時計回り）…ペンを切り替える FlowButton の表示
- shaking…トップメニュー FlowButton の表示
- swinging（North）…コピー
- swinging（South）…ペースト

操作体系について説明する．Oh! Stylus Draw 3 はメニューバー形式のメニューを用いる場合でも全ての操作が可能であり，操作体系は既存の多くのアプリケーションと同様であるため，詳細な説明は省略するが，基本的な流れはメニューバー，サブメニュー，コマンドの順のクリック操作である．後述する各機能を呼び出す図（図 4.13〜4.15）の (a) において，メニューバー形式のメニューのスクリーンショットを示す．

ショートカットアクションを用いる場合は，３つのアクションとタップを組み合わせて操作を行う．全体的な操作の流れは図 4.11 のようになり，基本的な操作手順は以下の通りである．

1. shaking でトップメニュー FlowButton（図 4.12）を表示

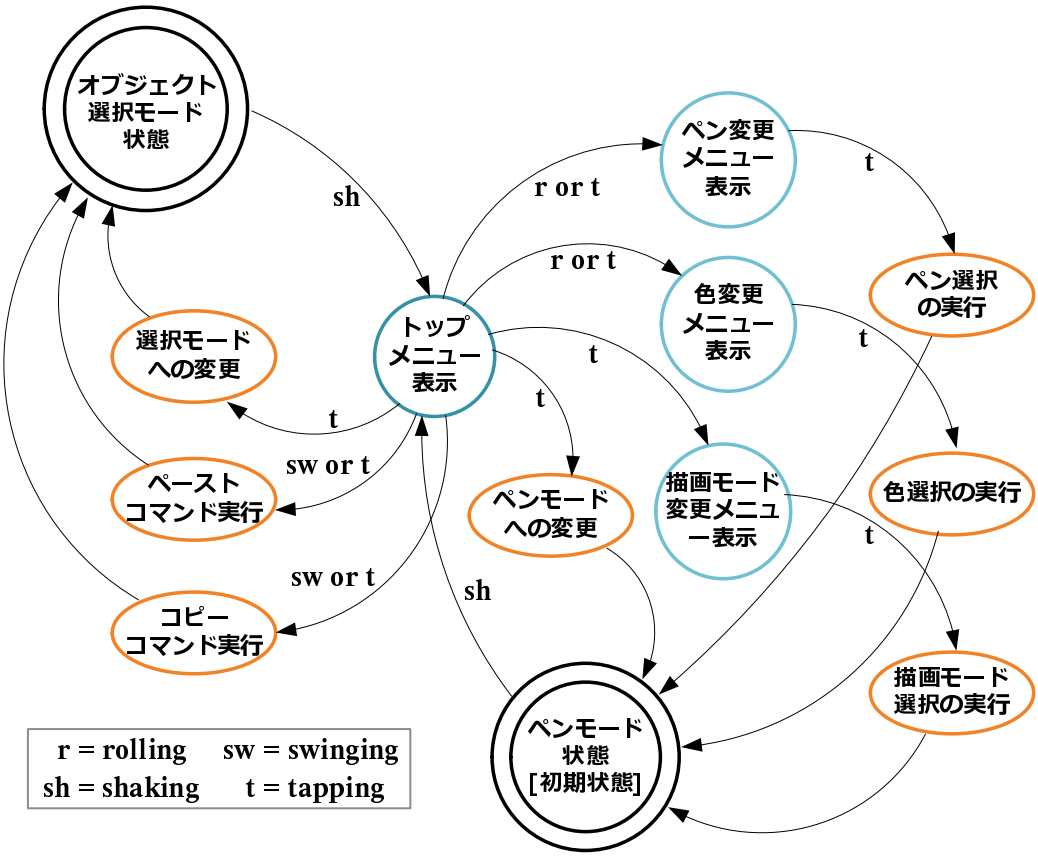


図 4.11: Oh! Stylus Draw 3 の操作の流れ

2. rolling，もしくはタップでサブメニューを表示
3. タップによるコマンド実行

まず，色の変更について説明する．色の変更を行う手順は以下の通りである．

1. shaking でトップメニュー FlowButton（図 4.12）を表示
2. 時計回りの rolling，もしくは COLOR ボタンのタップによるカラーパレット FlowButton（図 4.13(b)）の表示
3. タップによる色の選択

カラーパレットはパステル，ビビッド，グレー，カスタムに分かれており，それぞれが 12 色を持つ．パステル，ビビッド，グレーにはあらかじめ決められた色が配置されているが，カスタムにはユーザの好みの色を配置することができる．

次に，ペン先の変更について説明する．ペン先の変更を行う手順は以下の通りである．

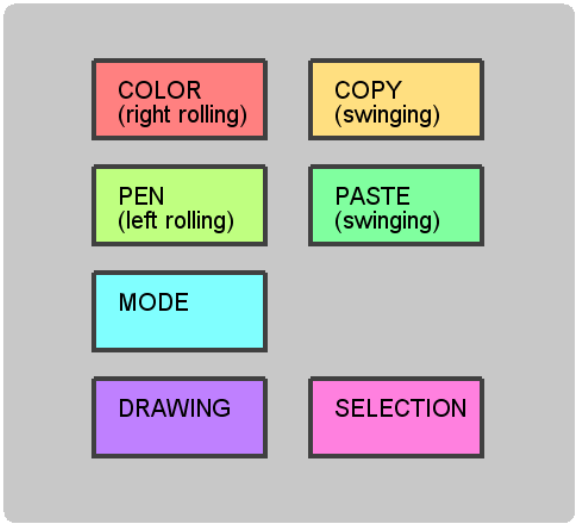


図 4.12: Oh! Stylus Draw 3 のトップ FlowButton

1. shaking でトップメニュー FlowButton（図 4.12）を表示
2. 反時計回りの rolling，もしくは PEN ボタンのタップによるペン先 FlowButton（図 4.14(b)）の表示
3. タップによるペン先の選択

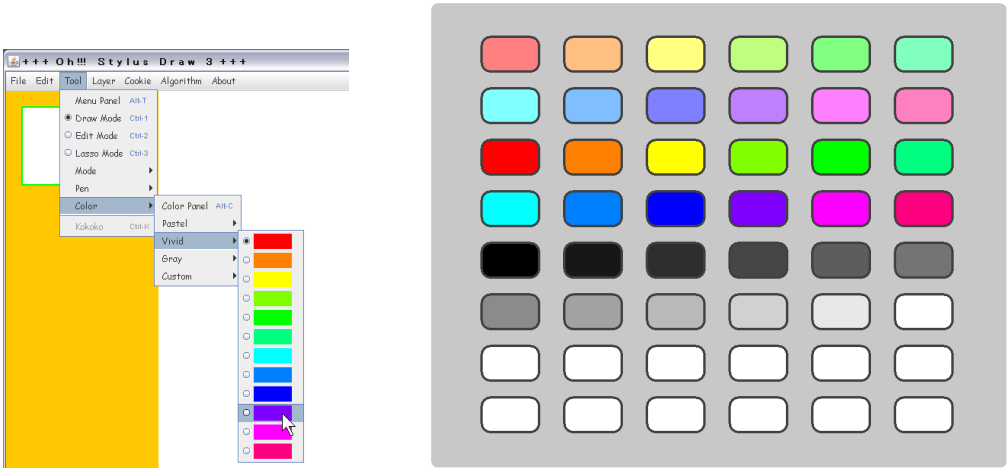
ペン先は通常の線と破線に加えて，それぞれにおいて Extra-fine，Fine，Normal，Thick，Extra-thick の 5 種類の太さがある．

次に，描画モードの変更について説明する．描画モードの変更を行う手順は以下の通りである．

1. shaking でトップメニュー FlowButton（図 4.12）を表示
2. MODE ボタンのタップによる描画モード FlowButton（図 4.15(b)）の表示
3. タップによる描画モードの選択

描画モードには，自由線，直線，矩形，塗りつぶし矩形，楕円，塗りつぶし楕円の 6 種類がある．

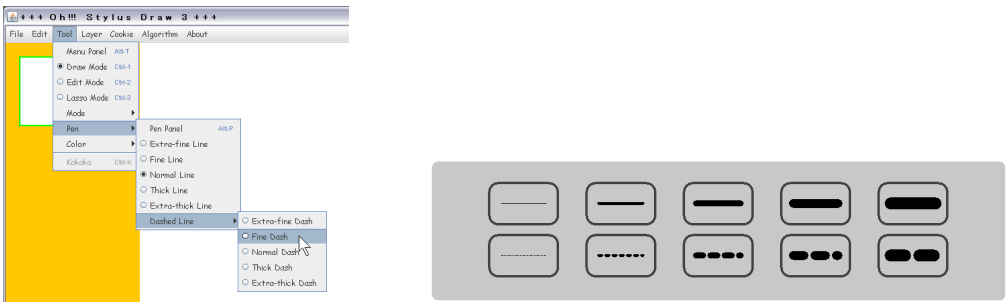
前述したように，Oh! Stylus Draw 3 では FlowButton を画面中央からポインタを中心とした場所に表示されるようにしているため，大画面ディスプレイでの操作においてユーザは立ち位置を気にする必要はなく操作できるようになっている．その結果，メニュー操作に意識や集中力が奪われず，本来の作業に集中することができるようになった．



(a) メニューバー形式の色変更メニュー

(b) カラーパレット FlowButton

図 4.13: Oh! Stylus Draw 3 の色変更



(a) メニューバー形式のペン先変更メニュー

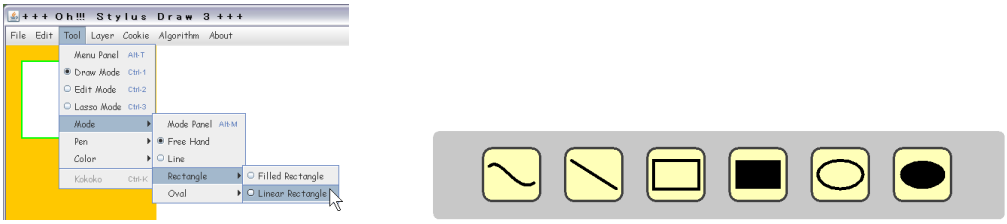
(b) ペン先変更 FlowButton

図 4.14: Oh! Stylus Draw 3 のペン先変更

4.3.2 Oh! Stylus Scroll

スタイラスによるスクロールを行うためにはスクロールバーを操作する必要があるが，この操作はユーザに細かいポインティングが要求する．また，大画面ディスプレイの場合，ユーザの立ち位置によってはスクロールバーのある位置までユーザ自身が移動する必要がある．このように，スタイラスによるスクロール操作のユーザビリティは良いとは言えない．

Oh! Stylus Scroll[24, 38] はスタイラスでのスクロール操作をより簡単にするためのスクロール支援アプリケーションである．特徴は以下の 3 点である．



(a) メニューバー形式の描画モード変更メニュー

(b) 描画モード変更 FlowButton

図 4.15: Oh! Stylus Draw 3 の描画モード変更

(i) rolling による上下（左右）方向のスクロール

(ii) shaking によるコンテキストメニューの表示

(iii) マウスホイールでの操作に対応したアプリケーションであればどんなアプリケーションにでも対応

Oh! Stylus Scroll の実行した様子を図 4.16 に示す．スクロールバーが垂直方向に配置されている場合，スタイラスを時計回りに rolling させると下方向へのスクロールとなり，反時計回りでは上方向へのスクロールとなる．スクロールバーが水平方向に配置されている場合，スタイラスを時計回りに rolling させると右方向へのスクロールとなり，反時計回りでは左方向へのスクロールとなる．スクロールバーが垂直方向と水平方向の両方に存在する場合は，垂直方向のスクロールが行われる．Oh! Stylus Scroll を用いると，ユーザはスクロールバーを直接操作しないので立ち位置を気にせずにスクロール操作を行うことができる．

また，shaking によりコンテキストメニューを表示することができる．Oh! Stylus Scroll を用いない場合，コンテキストメニューを表示するにはスタイラスをホールドさせる，もしくはバレルボタン＋タップという煩雑な操作をする必要があるが，Oh! Stylus Scroll を用いる場合スタイラスを振るだけでコンテキストメニューを表示することができる．

さらに，このアプリケーションはスクロールバーに限らず，スライダーやページ送りの操作などのマウスホイールによって操作できるものの多くを操作することができる．このアプリケーションはマウスで使うことを前提としてデザインされた GUI を，スタイラスでも容易に操作できるようなインタフェースをユーザに提供する．

4.3.3 Oh! Stylus Firefox Scroll

Oh! Stylus Firefox Scroll は Web ブラウザの Mozilla Firefox[1] （以下，Firefox）におけるスクロール操作を行いやすくするアプリケーションである．Firefox の拡張機能（アドオン）に

[1] http://www.mozilla-japan.org/products/firefox/

図 4.16: Oh! Stylus Scroll によるスクロールの様子

Grab and Drag[2]がある．これは Web ページをスクロールするときにスクロールバーをドラッグするのではなく，Web ページ自体を掴んでスクロールできるようにする拡張機能である．スクロールバーを用いて操作する場合，スクロールバーを下へドラッグするとコンテンツは上方向に移動するので，手の動きとスクロール方向が逆になる．一方，掴んで操作する場合，ページを掴んで下へドラッグするとコンテンツも下に移動するので，手の動きとスクロール方向が一致する．よって，スタイラスでのスクロール操作がより直感的かつ容易になるが，問題点も存在する．それは Grab and Drag でのスクロールモードと通常のブラウジングモードの切り替えである．通常，このモード切り替えはブラウザの切り替えアイコンをクリックすることで実現するが，Oh! Stylus Firefox Scroll では shaking によりこれを実現することができる．このように，Oh! Stylus Firefox Scroll と Grab and Drag を併用することで，Oh! Stylus Scroll とは違ったアプローチのスクロール操作が行えるようになる．

[2]http://grabanddrag.mozdev.org/

4.3.4 Oh! Stylus 3D Viewer

Oh! Stylus 3D Viewer は独自の 3D 操作マッピングを行った 3D ビューアである．Oh! Stylus 3D Viewer の実行例を図 4.17 に示す．

3D オブジェクトの閲覧方法について説明する．3D オブジェクトを Y 軸を軸とした方向に回転させる場合，垂直方向へのストローク操作により行え，X 軸を軸とした方向に回転させる場合，水平方向へのストローク操作により行うことができる．これは従来の 3D ビューアと同様の操作方法であると思われる．

また，3D オブジェクトを Z 軸を軸とした方向に回転させる場合，rolling により行うことができる．時計回りの rolling により 3D オブジェクトを時計回りに回転させることができ，反時計回りの rolling により反時計回りに回転させることができる．従来の操作方法の場合，3D オブジェクトの端をつまんで回転させるような操作方法が多く，これらの操作は 3D オブジェクトに対して直感的な操作方法ではない．一方，rolling による操作は rolling の回転方向と 3D オブジェクトの回転方向が同じであるため，直感的に理解しやすい操作方法である．

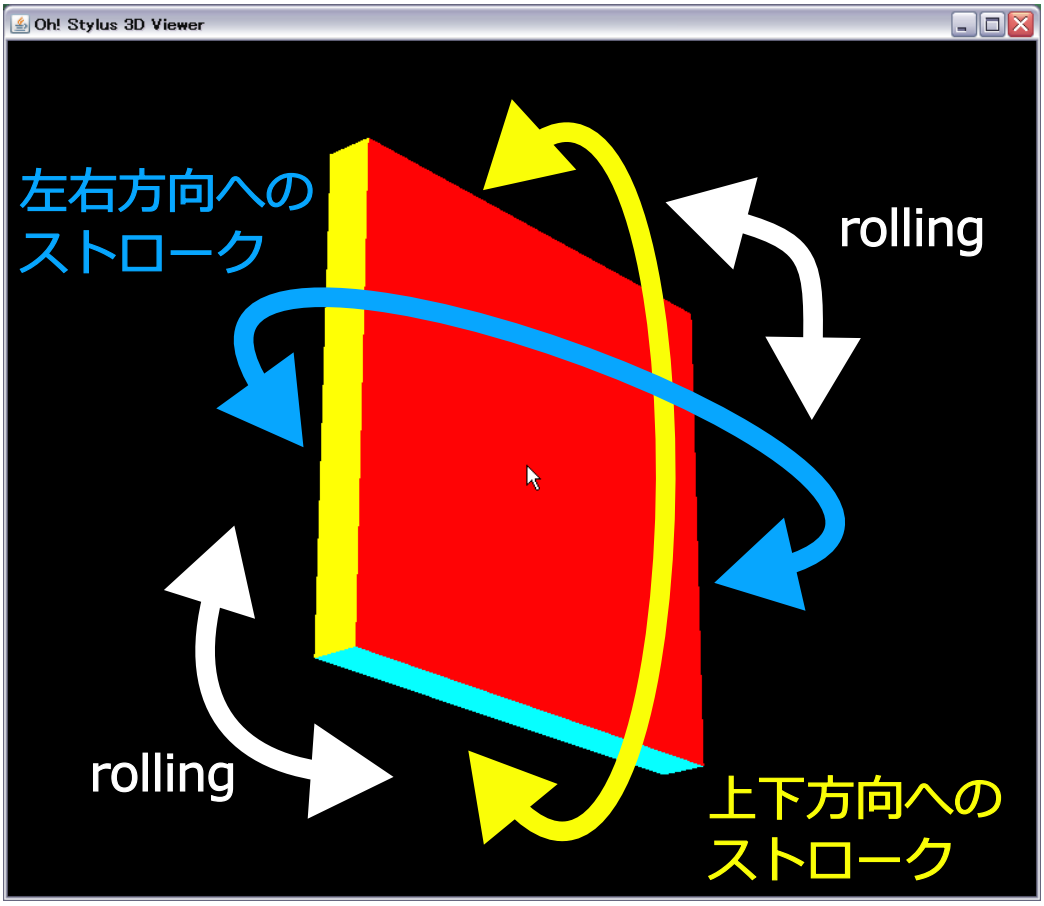


図 4.17: Oh! Stylus 3D Viewer のスクリーンショット

4.3.5 Oh! Stylus OneNote Extension

電子的にメモを取るためのツールとしては，以前はディスプレイ上に手書きで文字や図形を書くことができるだけの単純なツールが多かったが，最近では Microsoft Office OneNote[3]（以下，OneNote）のようなデジタルノート用のツールが開発されている．

OneNote では手書きのメモだけでなく，画像や音声などのディジタル情報を付加することができ，それらを１ヶ所に収集・管理することができる．これにより，すべてのディジタル情報を持つリポジトリを作成することができ，すべてのディジタル情報の再利用が可能になる．その結果，創造活動の作業の生産性が向上する．このように，OneNote はメモを取るなどの創造活動を積極的に支援する機能を持ったツールとなっている．

OneNote ではマウスを用いて操作することも可能であるが，手書き文字認識やデジタルインクなどのペン操作を想定した機能が多く備わっている．しかしながら，インタフェースとしては従来の WIMP インタフェースをベースとしているため，必ずしも創造活動に適したインタフェースであるとは言えない．例えば，メニューの選択にはウィンドウ上部にあるメニューバーからツリー構造になっている階層メニューを辿っていくという操作や，ツールバーにある小さいアイコンをクリックするという操作が必要である．このような操作はディスプレイを注視しながら細かいカーソル移動操作を行わなければならないため，操作に意識を奪われる．さらに，スタイラスはディスプレイ上の操作対象を直接ポインティングする必要があるため，ディスプレイが大きくなればなるほど操作が困難になる．これらは創造活動の障害となっている．

Oh! Stylus OneNote Extension[23, 39] は既存のディジタルノートである OneNote のインタフェースをスタイラスでも使いやすいものにするインタフェースの拡張ツールであり，OneNote の操作体系にショートカットアクションを追加するものである．

各アクションに割り当てた OneNote の操作は以下の通りである．

- rolling…OneNote に標準で備わっている 13 種類のペン（図 4.18）を変更することができる．時計回りに rolling をすると現在選択しているペンの１つ下のペンが選択される．一方，反時計回りに rolling をすると１つ上のペンが選択される．

- shaking…ペンモード／選択モードを切り替えることができる．ペンモードとはストロークにより文字や図形などが描けるモードであり，選択モードとはストロークによりバウンディングボックスが表示されてノート上のオブジェクトを選択できるモードである．現在のモードが選択モードの場合，１度 shaking を行うとペンモードになり，さらにもう１度 shaking を行うと選択モードになる．

- swinging…コピー／ペーストを行うことができる（図 4.19）．swinging は４方向の識別が可能であるが，その中の２方向にコピー機能とペースト機能を割り当てた．

[3]Microsoft Office OneNote
http://office.microsoft.com/en-us/onenote/default.aspx

図 4.18: OneNote で選択可能なペンの種類

shaking と swinging により，従来とは異なるオブジェクトのコピー&ペースト操作やペンの変更操作が可能となる．コピー&ペーストの場合，従来のインタフェースでは，まず対象オブジェクトを選択し，メニューバーからコピーやペーストを選択する必要があった．一方，ショートカットアクションを用いると，対象オブジェクトを選択した後は swinging するだけでコピーやペーストを行うことができる．

Oh! Stylus OneNote Extension を利用することで，メニューバーや小さいアイコンを操作しなくても OneNote の利用頻度の高い機能やコマンド選択を実行しやすくなる．その結果，煩わしい操作に伴う思考の中断が減り，創造活動の効率性向上が期待できる．

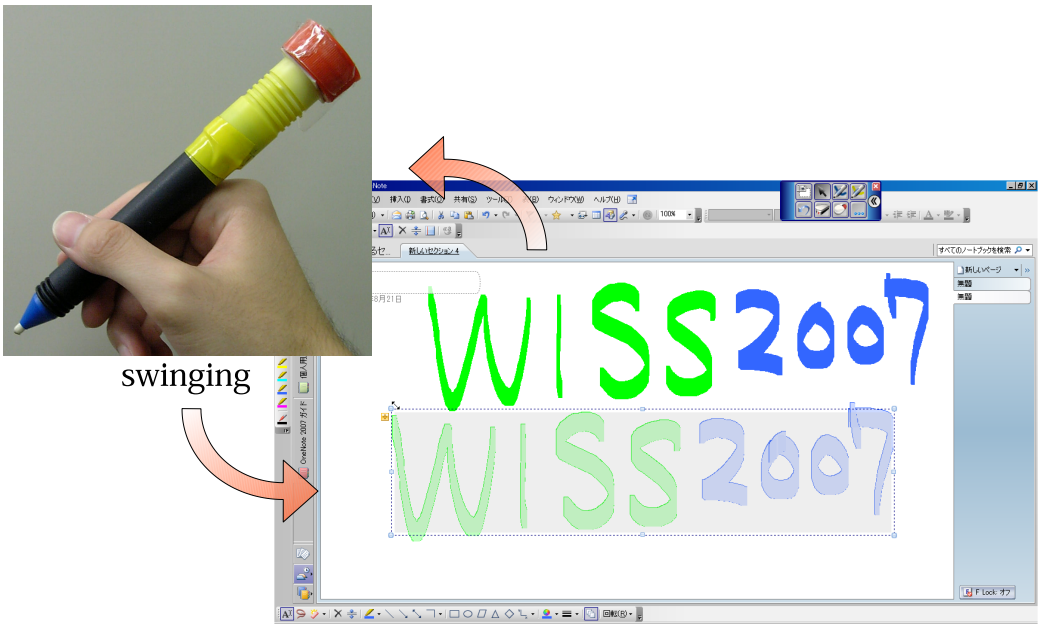


図 4.19: Oh! Stylus OneNote Extension での swinging によるコピー&ペースト

4.4 操作体系

4.3 節で述べたアプリケーションではさまざまな操作体系を取り入れた．特に Oh! Stylus Draw の各バージョンは，前のバージョンを使ったユーザスタディの結果を基にしてインタフェースや機能と共に操作体系も変更した．その結果，ショートカットアクションだけでコマンド実行までの全ての操作ステップを行うのではなく，既存のインタラクション手法であるタップをうまく組み合わせるという操作体系に達した．

第5章　評価

本章では，本研究で行った３つの評価実験について述べる．３つの評価実験はそれぞれ実施時期が異なるため，アクションの認識アルゴリズムや使用したアプリケーションが異なるので３つの評価結果を単純に比較することは好ましくないが，それらを通してアクション認識アルゴリズムやアクションとコマンドの割り当てについての知見を得ることができた．

5.1　評価実験１

この評価実験 [24] には Oh! Stylus Draw 1，Oh! Stylus Draw 2 と Oh! Stylus Scroll を用いた．アクションの認識には 3.2.2 節で説明したアルゴリズムを用いた．

5.1.1　目的と観点

評価の目的は以下の２点である．

(i) 提案手法の使いやすさについての知見を得る

(ii) 提案手法と従来手法とのユーザビリティの比較

(ii) は，Oh! Stylus Draw 1 と Oh! Stylus Scroll を利用し，条件やタスクを与えることによるユーザビリティの簡易的な評価により行った．また (ii) の実験後に，(i) のためのアンケート調査を実施した．さらに，Oh! Stylus Draw 2 を使ったアンケート調査も行った．

5.1.2　評価方法

共通事項

ショートカットアクションを応用したアプリケーションと，従来手法を用いたアプリケーションの２種類を用いて実験を行った．被験者は８人で，全員が大学生もしくは大学院生である．彼らはそれぞれ４人のグループ A とグループ B に無作為に分けられた．また，全員利き手は右手であったため，全員がスタイラスを右手で握って操作した．

実験にはタブレット PC（図 5.1）を用いたが，ディスプレイサイズの違いによる使いやすさに違いを検証するために，ディスプレイを全画面使ったタブレット PC サイズと，ディスプレイの一部を使った PDA サイズの２種類のディスプレイサイズを用意した．また，Oh! Stylus Draw 2 を使ったアンケート調査はタブレット PC のみで行った．

図 5.1: 評価実験 1 で用いるタブレット PC

Oh! Stylus Draw 1 の評価方法

まず，自由線，直線，矩形，楕円の４つの描画モードの中からあらかじめ１つを決めておき，被験者は一定時間内に指定された描画モードを使って，指定された文字や図形を色を変えながらできる限り多く描くというタスクを行い，描けた文字や図形の数で評価を行った．どの順番で色を変えていくかは指定されている．また，今回の実験では制限時間を 10 秒とした．これを回転操作による色選択を行う提案手法と，パレット選択による色選択を行う従来手法の２つの手法で行い，それらの結果を比較した．これを各描画モードで行った．

各描画モードで指定される文字や図形は，自由線モードでは「永」という漢字，直線モードでは「×」という図形，矩形モードではできる限り正方形に近い四角形，楕円モードではできる限り真円に近い円である．自由線モードで「永」という漢字を選んだ理由としては，この漢字には漢字の字体を構成する最小単位の要素である筆画が全て含まれているためである．

被験者には，描いてもらう文字や図形の大きさや丁寧さについて，厳密でなくて良いができるだけ一定にするように事前に伝えておいた．

２つのディスプレイサイズに対応できるように，アプリケーションのウィンドウの大きさがタブレット PC 用で 800×600 ピクセル，PDA 用で 320×240 ピクセルになるように評価用アプリケーションを作成した．

また，グループ A では最初に提案手法での実験を行い，次に従来手法での実験を行った．グループ B はその逆順で実験を行った．

Oh! Stylus Scroll の評価方法

まず被験者は，10 個のボタンがランダムに配置されている Web ページにブラウザでアクセスする．全てのボタンのラベルは “∗” になっている．この Web ページの左上には start ボタンが配置されており，それをクリックするとボタンのラベルが 1〜10 の文字に変わる．図 5.2 にこの Web ページを示す．被験者はこのボタンを 1 から順番にクリックしていき，10 がクリックされるまでの時間を計測する．ボタンの配置は，次のボタンをクリックするには上下のどちらかに必ずスクロールしなければならないように意図的に配置されている．これを回転操作によるスクロールを行う提案手法と，スクロールバーのドラッグによるスクロールを行う従来手法の 2 つの手法で行い，それらの結果を比較した．

Oh! Stylus Scroll の汎用性を活かして，Web ブラウザには Firefox を利用した．また， 2 つのディスプレイサイズに対応できるように，Firefox のウィンドウの大きさをタブレット PC で 800×600 ピクセル，PDA 用で 320×240 ピクセルに設定した．作成した評価用の Web ページのサイズは，タブレット PC が 800×1200 ピクセル，PDA 用が 320×480 ピクセルになっている．

また，グループ A では最初に提案手法での実験を行い，次に従来手法での実験を行った．グループ B はその逆順で実験を行った．

図 5.2: Oh! Stylus Scroll の評価用 Web ページ

5.1.3 結果

実験結果を表 5.1 と表 5.2 に示す．アンケートから得た各アプリケーションに関する主な意見は以下の通りである（括弧は Oh! Stylus Draw のバージョン）.

表 5.1: Oh! Stylus Draw 1 の実験結果（単位：個）

	提案手法 (tabletPC)	従来手法 (PDA)	提案手法 (tabletPC)	従来手法 (PDA)
自由線	3.0	3.5	2.75	3.625
直線	3.625	5.5	4.5	5.125
矩形	4.375	6.0	5.125	6.125
楕円	5.375	7.125	5.5	6.625

表 5.2: Oh! Stylus Scroll の実験結果（単位：秒）

	提案手法 (tabletPC)	従来手法 (PDA)	提案手法 (tabletPC)	従来手法 (PDA)
合計時間	34.823	25.011	36.286	22.496

Oh! Stylus Draw 1, Oh! Stylus Draw 2

- shaking は非常に使いやすい (1,2)
- １ストロークで描ける図形は描きやすい (1,2)
- 音によるフィードバックはわかりやすい (2)
- 色選択のときに似た色相が隣にあるため，グラデーションが描きやすい (2)
- キャンバスにスタイラスを近づけたときに rolling で選択した色が変わってしまう (1,2)
- パレットの反対側にある色を選択するのに時間がかかる (2)

Oh! Stylus Scroll

- スタイラスのポインティング位置を変えずにスクロールできるので便利である
- 慣れていないインタフェースなので，rolling とスクロール幅との相関関係を把握するのに時間がかかった
- リンクをタップしようとしたときに稀にスクロールしてしまう

5.1.4 結果に関する考察

表 5.1 より，提案手法によって描くことができた図形や文字の数は，従来手法のそれと比較してタブレット PC の場合におよそ 75%，PDA の場合におよそ 83%であることがわかった．表 5.2 より，提案手法によるスクロールでかかった合計時間は，提案手法のそれと比較してタブレット PC の場合におよそ 139%，PDA の場合におよそ 161%であることがわかった．Oh! Stylus Scroll を用いた実験では提案手法での試行の方が時間がかかったが，これは rolling 後のフィードバックの遅延や rolling の認識精度の悪さが主な原因であると考えられる．しかしながら，成熟している既存の操作方法と比較して，操作スピードに関して十分に対抗できることを確認できた．

さらに，アンケートの結果から，加速度センサ付きスタイラスの操作として shaking はユーザにとって使いやすいということがわかった．これは shaking がシャープペンシルで芯を出す際のメタファを採用しているからであると考えられる．rolling は Oh! Stylus Scroll では使いやすく便利であるという意見をもらった一方，Oh! Stylus Draw1,2 では色の選択に rolling が適切でないという意見もあった．これは我々がペンを使うときの振る舞いではなく，この動作に慣れていないからであると思われる． rolling を使用するアプリケーションによって rolling に対する評価が異なることから，rolling を適切なアプリケーションの適切な操作に割り当てるとインタフェースとして使いやすいものになるのではないかと考えられる．また，多くのポシティブな意見をもらった一方，ネガティブな意見もあった．しかしながら，それらの意見の多くはインタフェース自体を否定するものではなく，実装の問題点を指摘するものであった．

5.2 評価実験２

この評価実験 [23, 39] には OneNote と Oh! Stylus OneNote Extension を用いた．アクションの認識には 3.2.3 節で説明したアルゴリズムを用いた．

5.2.1 目的

この評価実験は，研究の目的である入力デバイスとしてスタイラスを用いたシステムのユーザビリティの向上を達成するために必要な知見を得るための予備実験として行った．

5.2.2 評価方法

被験者は，著者が所属する研究室の大学院生 3 名＋アプリケーションの開発者である著者の合計 4 名で行った．開発者自身は何度も CS Stylus を扱っているので，入力デバイスとして CS Stylus を使い慣れているといえる．そのため，開発者自身も被験者となることで，デバイスに対する “慣れ” の操作性に及ぼす影響についても知見を得ることができる．

評価実験は，OneNote と Oh! Stylus OneNote Extension，タッチスクリーン付き大画面プラズマディスプレイパネル（スマートボード[1]），CS Stylus，通常のスタイラスを用いて行った．デバイスにはスマートボードではなく，タブレット PC を用いることも可能であった．しかしながら，著者の使用経験より，大画面用 CS Stylus とタブレット PC 用 CS Stylus を比較すると，後者の方が重量バランスが優れていることが判明していたため，実験に用いるデバイスとして大画面ディスプレイを採用した．プラズマディスプレイパネルには，画面サイズが 50 インチ，パネル解像度が 1280*768 ドットのものを使用した（図 5.3）．

評価実験のタスクは，被験者がそれぞれ自分の名前を OneNote 上に書き，それをコピー&ペーストで 1 つ複製するというシンプルなものである．被験者に書いてもらう内容によって，被験者間に有利不利が生じないように，最も書き慣れているであろう各自の名前を書くというタスクを設定した．

このタスクを CS Stylus と通常のスタイラスの 2 種類のスタイラスを用いてそれぞれ 3 回ずつ行った．4 人の被験者のうち，被験者 A，D は最初に CS Stylus を用いた実験を行い，次に通常のスタイラスを用いた実験を行った．被験者 B，C はその逆順で実験を行った．

また本実験では，2 種類のスタイラスにおけるタスク達成時間の差を検証するため，被験者によって名前の画数の違いによるタスク達成の時間の違いは考慮に入れない．

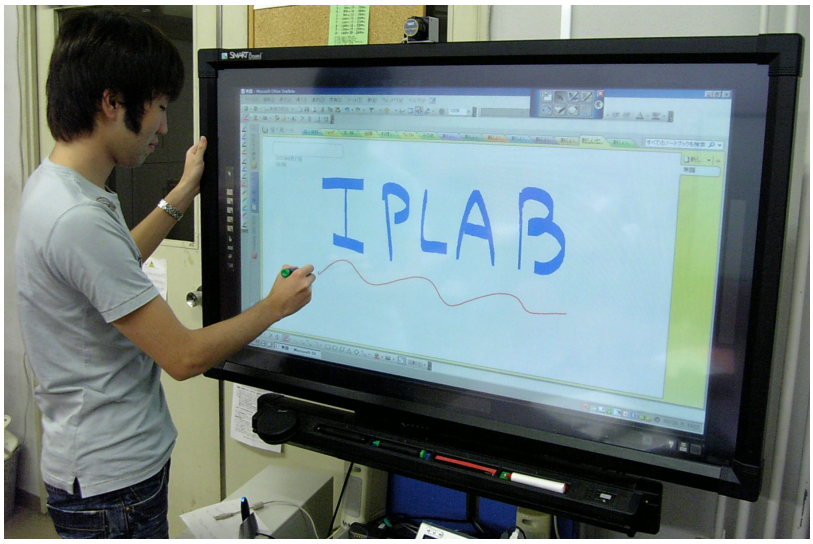


図 5.3: 実験で用いたタッチスクリーン付き大画面ディスプレイ

5.2.3 結果と考察

実験の結果を表 5.3 に示す．被験者全員のタスクにかかった時間の差から，ほとんどのタスクで CS Stylus による操作の方が通常のスタイラスによる操作よりも良い結果を残しているこ

[1]http://www.smartboard.co.jp/

表 5.3: タスクの所要時間 (秒)

	被験者 A			被験者 B		
	CS Stylus	Stylus	差	CS Stylus	Stylus	差
1 回目	10.2	10.9	-0.7	20.9	26.0	-5.1
2 回目	16.9	10.8	6.1	13.6	23.4	-9.8
3 回目	9.0	10.4	-1.4	14.8	28.3	-13.5
	被験者 C			被験者 D		
	CS Stylus	Stylus	差	CS Stylus	Stylus	差
1 回目	13.0	15.5	-2.5	13.2	15.0	-1.8
2 回目	16.8	16.0	0.8	13.1	14.3	-1.2
3 回目	12.8	15.8	-3.0	12.0	14.0	-2.0

とがわかる．

また，通常のスタイラスによるタスク達成の時間を比較すると，それぞれの被験者は 3 回の試行をほとんど同じ時間でこなしたことがわかる．これは，被験者全員が WIMP インタフェースという一般的に採用されている GUI に使い慣れているためであると考えられる．

一方，CS Stylus によるタスク達成の時間を比較すると，被験者 A と被験者 C の 2 回目の試行結果を無視すれば，全員が 1 回目の試行よりその後の試行の方が速いと言える．被験者 A, C の 2 回目の試行は，彼らが CS Stylus の扱いに慣れていなかったために起こった操作ミスであった．開発者の経験からすると，CS Stylus の扱いに慣れることによってこのような操作ミスは軽減できると考えられる．開発者である被験者 D は CS Stylus の扱いに十分慣れており，操作ミスはなく 3 回それぞれの試行でかかった時間はほとんど同じであった．実験結果より，CS Stylus の扱いに慣れることで，通常のスタイラスよりも操作性が向上すると期待できる．

5.3 評価実験３

この実験には Oh! Stylus Draw 3 を用いた．アクションの認識には 3.2.4 節で説明したアルゴリズムを用いた．また，加速度情報と併せて地磁気情報を rolling の認識の補助として用いている．

5.3.1 目的

この評価実験は FlowButton の有用性や以下の３つのメニュー操作方法の操作性を比較するために行った．

- 複数回のショートカットアクションと１回のタップを組み合わせてメニュー操作を行う方式（以下，ショートカットアクション方式１）

- １回のショートカットアクションと複数回のタップを組み合わせてメニュー操作を行う方式（以下，ショートカットアクション方式２）
- メニューバーからメニュー操作を行う方式（以下，メニューバー方式）

5.3.2 評価方法

被験者は５名であり，いずれも大学生もしくは大学院生であった．評価のタスクは大きく分けて２種類あり，それぞれのタスクをショートカットアクション方式１，ショートカットアクション方式２，メニューバー方式の３つのメニュー操作方法を用いて行った．１つ目のタスク（以下，タスク１）は，あらかじめ描かれた２つで１組の矩形を，それと同じ色の線で繋ぐというものである．矩形の組は全部で５組あり，それぞれは色で分けられている（図 5.4）．２つ目のタスク（以下，タスク２）は，あらかじめ描かれた２つの図形をそれぞれ複製するというものである．これらのタスクでは，shaking，rolling，rolling の３つのアクションと，それらと既存の入力手法の組み合わせを使用するので，ショートカットアクションで行える操作を全て評価することができる．それぞれのタスクにおいて，タスク達成に要した時間を比較することで，この評価実験の目的を達成する．

評価実験の手順としては，まず被験者には Oh! Stylus Draw 3 の使い方やインタフェースの説明を行った上で，３分程度自由に使ってもらった．そして，タスク１，タスク２という順番で実験を行ってもらった．被験者は２つのタスクを１つのメニュー操作方法につき１回ずつ行う．また，それぞれのタスクで行う３つのメニュー操作の順番はあらかじめ被験者に指示しており，被験者ごと順番が異なるようにした．実験が終了したあとに，実験内容やショートカットアクションなどに関するアンケートに回答してもらった．

この評価実験には Oh! Stylus Draw 3 の他に，タッチスクリーン付き大画面プラズマディスプレイパネル（スマートボード），CS Stylus，通常のスタイラスを用いて行った．プラズマディスプレイパネルには，画面サイズが 50 インチ，パネル解像度が 1280*768 ドットのものを使用した（図 5.3）．

5.3.3 結果と考察

タスクの結果

被験者全員の各タスク達成に所要した時間と，その平均値をそれぞれ表 5.4，表 5.5 に示す．また，各タスクの所要時間の平均値を図 5.5(a) と図 5.5(b) に示す．

まず，タスク１の所要時間の平均を比較すると，ショートカットアクション方式２での操作が最も素早くタスクをこなせたことがわかる．また，ショートカットアクション方式２とショートカットアクション方式１とのタイム差はそれほど大きくない．よって，これはこの２つの方式はそれほど操作性に差がないことがわかる．一方，メニューバー方式は他の２つの方式よりもタスクの所要時間の平均が大きくなっている．

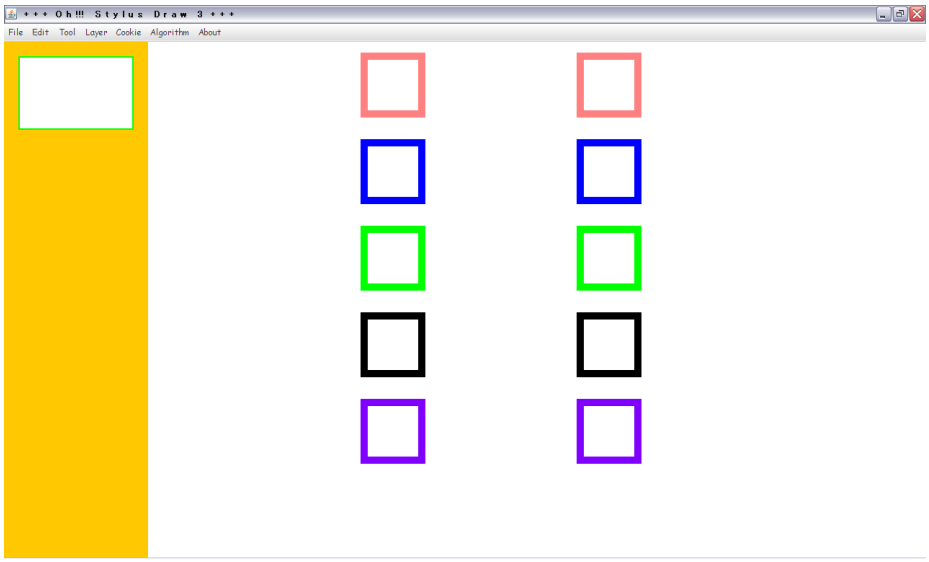


図 5.4: タスク 1 で用いる Oh! Stylus Draw 3 のスクリーンショット

表 5.4: タスク 1 の所要時間 (秒)

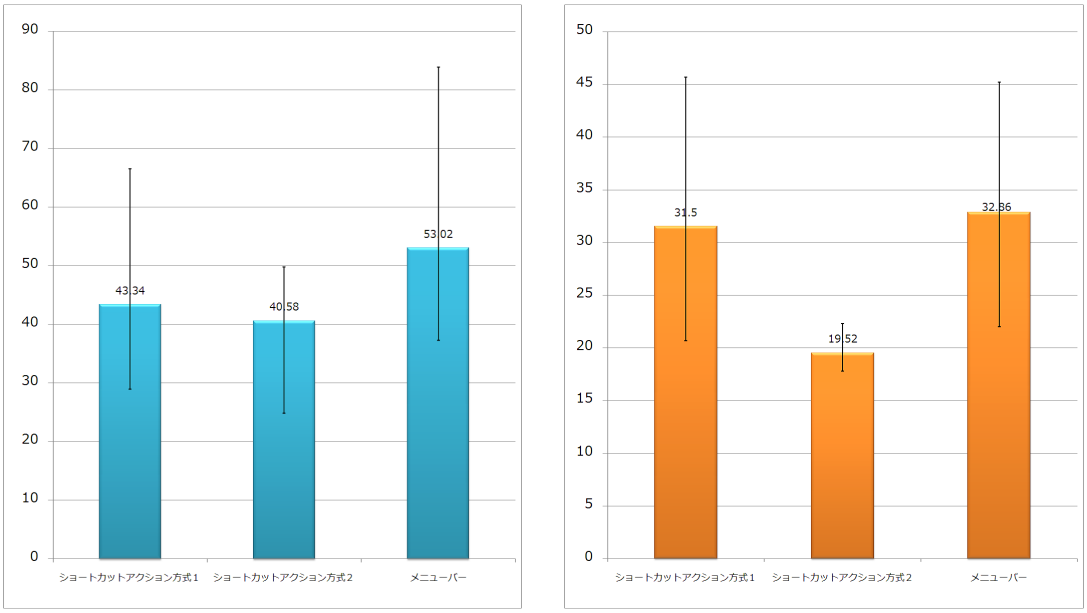
	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	平均
ショートカットアクション方式 1	28.9	35.3	55.4	66.5	30.6	43.3
ショートカットアクション方式 2	24.8	48.8	49.8	49.0	30.5	40.6
メニューバー方式	37.3	83.9	45.7	51.7	46.5	53.0

表 5.5: タスク 2 の所要時間 (秒)

	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	平均
ショートカットアクション方式 1	20.7	34.8	33.1	45.7	23.2	31.5
ショートカットアクション方式 2	20.2	22.3	17.9	19.4	17.8	19.5
メニューバー方式	26.4	45.2	32.6	38.1	22.0	32.9

次に，タスク 2 の所要時間の平均を比較すると，タスク 1 と同様にショートカットアクション方式 2 での操作が最も素早くタスクをこなせたことがわかる．しかしながら，ショートカットアクション方式 1 の所要時間の平均はショートカットアクション方式 2 よりも大きくなっている．また，メニューバー方式の所要時間の平均はショートカットアクション方式 1 とほとんど同じであった．

ショートカットアクション方式 1 はほとんどの操作をアクションだけで行う方式であるが，被験者は CS Stylus の扱いになれていないため，アクションを使いこなすことが困難であっ



(a) タスク 1 でかかった時間の平均　　(b) タスク 2 でかかった時間の平均

図 5.5: 評価実験の結果

たと思われる．よって，1 回のショートカットアクションと複数回のタップを組み合わせた，ショートカットアクション方式 2 が最も良い成績を収めたのではないかと推測できる．また，CS Stylus を用いてショートカットアクションを使うトレーニングを行うことで，ショートカットアクション方式 1 がより良い結果を収めることが期待できる．

メニューバーを使った操作は普段のメニュー操作であるにも関わらず良い結果とならなかった．これは既存の多くのシステムで採用されている WIMP インタフェースが FlowButton によるメニュー表示と比較して，スタイラスではうまく操作できないことを示している．操作に用いるデバイスごとに操作しやすいインタフェースデザインが存在し，少なくともスタイラスにはメニューバー形式などの細かいポインティング操作が要求されるインタフェースは向いていないことがわかった．

アンケートの結果

アンケートでは，それぞれのタスクにおいて最も操作しやすかった操作方式と，その理由を回答してもらった．その結果を表 5.6 に示す．ほとんどの被験者がショートカットアクション方式 2 を選択したことがわかる．選択した理由として，主に以下のような回答（括弧は該当するタスク番号）を得た．

- ショートカットアクション方式 1 とショートカットアクション方式 2 はメニューが目の

表 5.6: 最も操作しやすかった方式の選択

操作方式	タスク１	タスク２
ショートカットアクション方式１	1	0
ショートカットアクション方式２	4	5
メニューバー方式	0	0

前に出現するので選択しやすい．(1, 2)

- ショートカットアクション方式１の rolling と swinging は認識しにくいことがあるため，ショートカットアクション方式２の方がフラストレーションを感じずに操作できた．(1, 2)
- ショートカットアクション方式１とショートカットアクション方式２はメニュー項目が一覧できるので項目を選びやすい．(1, 2)
- ショートカットアクション方式１とショートカットアクション方式２は操作に慣れると，リズミカルに操作することができる．(1, 2)
- ショートカットアクション方式１とショートカットアクション方式２は目線と立ち位置の移動をほとんど行わずに全ての操作を完了できる．(1, 2)
- ショートカットアクション方式１は手元でほとんどのメニュー操作が完了するので操作しやすい．(1)
- タッピングでペーストする方が操作として自然な感じがする (2)
- メニューバーが存在する方が安心感がある (1, 2)
- メニューバー方式は個々のメニュー項目が小さいため内容を確認しづらい．(1,2)
- メニューバー方式は正確にメニュー項目をタップすることが難しい．(1, 2)

アクションの認識精度はさらに改善の余地があるため，認識精度に対するネガティブな意見は今後の研究に役立てたい．しかしながら，現在の認識精度でもショートカットアクション方式１とショートカットアクション方式２が使いやすいという意見があるので，認識精度を向上させることで，操作性の更なる向上も大いに期待できる．また，筆者は 1.1 節において，入力デバイスとしてスタイラスを用いるシステムにおけて WIMP インタフェースが採用されていることがシステムのユーザビリティを低下させている原因であると主張しているが，このアンケート結果からもその主張が正しいことがわかった．

アンケートでは，さらに Oh! Stylus Draw 3 に関する意見ももらった．主な意見は以下の通りである．

- FlowButton が表示されるときのアニメーションがショートカットアクションの動作に合っている．
- FlowButton で色やペン先の形状が一覧できるため，選択しやすかった．
- 音のフィードバックはわかりやすかった．
- swinging に対するコマンドの割り当てはもっと工夫が必要である．
- FlowButton が表示されたときに，その後ろに描かれた図形が隠れてしまうことがある．
- 操作に対してレスポンスが遅れることがあった．

フィードバックやアニメーションなどに関しては多くのポジティブな意見をもらった一方で，ネガティブな意見ももらった．しかしながら，ネガティブな意見の多くはレスポンスや認識精度に関する意見であり，インタフェースデザインに関するものはほとんどなかった．唯一，インタフェースデザインに関するネガティブな意見として，FlowButton の後ろに描かれた図形が隠れることがあるという意見をもらったが，FlowButton 表示を全て透過にすることで容易に対応できる．

5.4 議論

ここでは，３つの評価実験を通じて行われた議論について述べる．

5.4.1 利き手について

既存の GUI は右利きの人にとって使いやすい．例えば，スクロールバーを操作する場合，右手で操作する場合は図 5.6(b) のように操作中であってもディスプレイを確認しやすいが，左手で操作する場合，図 5.6(a) のように手がディスプレイを隠してしまう．このように現在の多くの GUI では，右利きの人が使いやすいようになっている場合が多く，左利きの人は右利きの人と比較して作業効率が悪くなってしまうおそれがある．しかしながら，ショートカットアクションを利用することで，このような問題を解決することができる．

5.4.2 ショートカットアクションの習慣化について

実世界においてシャープペンシルを使用するときに，芯を出すためにペンの上部をノックしたり，ペンを上下に振ったりするが，これらの行為は習慣化されており，芯を出すことを意識しなくても行える．

筆者の CS Stylus の使用経験からすると，CS Stylus は空間マウスなどの空中で操作する他のデバイスと比較して慣れるまでの時間が早い．これは CS Stylus も “ペン” であるからだと思われる．

(a) 左手での操作　　(b) 右手での操作

図 5.6: 利き手による使用感の違い

したがって，我々が提案するショートカットアクションも容易に習慣化でき，シャープペンシルと同等の効果が期待できる．そして，スタイラスを使ったインタラクションでもメニュー選択などに意識を奪われない操作が可能になることが期待できる．

5.4.3　ディスプレイの角度の違いについて

大画面ディスプレイとタブレット PC とでは，ユーザに対するディスプレイの角度が全く異なるため，ユーザのスタイラスの持ち方が異なる．大画面ディスプレイの場合にはスタイラスを水平に持ち，タブレット PC の場合には垂直に持つ．この違いによって，大画面ディスプレイとタブレット PC との操作性に違いが生じることも予想されるが，筆者のタブレット PC ＋ CS Stylus の使用経験では，操作性に大きな違いを感じることはなかった．

第6章　関連研究

本章では，まず，HCI（ヒューマン・コンピュータ・インタラクション）研究における本研究のスコープについて述べる．そして，本研究に関連する研究について紹介する．さらに，スコープと関連研究に基づき，本研究の位置付けを明確化する．

6.1　研究のスコープ

本研究では，HCI 分野での研究としてペン入力インタフェースに着目している．そして，入力デバイスとしてスタイラスを用いる既存のシステムにおけるメニュー操作のユーザビリティの低さや，スタイラスがペン型であることを有効に活用した入力操作の少なさを問題点として捕らえている．これらを解決する手段として，実世界指向という観点で問題の解決に取り組んでいる．

6.2　関連研究

本研究に関連する研究をいくつかのカテゴリに分類して紹介する．6.2.1 節では実世界指向インタフェースにおける人間の動作に着目した研究について，6.2.2 節では実世界指向インタフェースにおける端末のコンテキストに着目した研究について述べる．6.2.3 節ではスタイラス向けのインタフェースに関する研究について述べる．

なお，複数のカテゴリにまたがるような関連研究については，１つのカテゴリにのみ分類している．

6.2.1　人間の動作に着目した研究

椎尾らは文鎮メタファを利用したインタラクション手法 [28] を提案している．彼らは人間が紙に文字を書く際に紙を手のひらで押さえながら書くという動作に着目している．ユーザの手のひらが PDA やタブレット PC のディスプレイ下部に触れているかどうかをモード切り替えなどの操作に利用し，スタイラスのユーザビリティ向上を狙っている．手のひらの検出にタッチセンサを用いている．

Harrison らの Squeeze Me, Hold Me, Tilt Me![7] は，人間がハンドヘルド PC を扱うときに端末を握ったり，傾けたりする動作に着目したインタラクション手法を提案している．端末を

握る動作によるページ送りや，傾ける動作によるスクロールなどを実現している．動作の検出には加速度センサや圧力センサを用いている．

伊賀らは人間の呼気･吸気をインタラクションに利用する kirifuki[34] を提案している．kirifuki は彼らが提案したやわらかインタフェース [35] という人間の行為に密着したインタラクションスタイルを実現したシステムである．例えば，ディスプレイに向かって息を吹きかけたり，吸い込んだりすることで複数のウィンドウを集散する操作や，息を吸い込むことでオブジェクトをカットし，息を吹き出すことでオブジェクトのペーストを行うという操作が可能である．呼気・吸気の検出にはマイクからの音の信号を用いている．

矢谷らの Toss-It[31] は，PDA などのモバイル端末間の情報の移動を直感的に実現するインタフェース技術であり，人間がモノを渡すときに投げ渡す（トスをする）という行為に着目している．モバイル端末をトスする動作により，情報を別のモバイル端末に移動させることができる．加速度センサや角速度センサを用いて人間がトスする強さや動作の軌跡を検出し，システムを実現している．

Rekimoto の GestureWrist and GesturePad[20] では，ウェアラブルコンピュータのための 2 つの入力デバイスを紹介している．GestureWrist はリストバンド型のデバイスであり，手の動きを認識することができる．この手の動きをコンピュータとのインタラクションに利用している．GesturePad は着用している服に内蔵することができるモジュールである．センサが内蔵されている部分に触れることで特定の入力操作が行える．

Bang らは 3 次元の手の動きを再現するペン型入力デバイスに関する研究を行っている [3]. 彼らは加速度センサ，ジャイロセンサ，ジャイロスコープなどを用いてディスプレイ面での 2 次元の手書きストロークの軌跡，もしくは空中での 3 次元の手書きストロークの軌跡を認識できるデバイスと，軌跡の認識に必要なアルゴリズムを開発している．

塚田らは日常的な行為を活用することなどをコンセプトとした，AfterTouch[32] を提案している．AfterTouch ではユーザの日常的な行為と ID ベースのインタラクション技法を組み合わせることで，なめらかな粒度を持つ情報操作を実現できる．例えば，Touch&Turn ではノブなどの回すインタフェースに指紋センサを組み合わせて，ユーザがノブを握る際に指紋によるユーザ認証を行い，ユーザごとに異なる操作を行うことができる．

6.2.2 端末のコンテキストに着目した研究

Rekimoto は加速度センサをコンピュータとのインタラクションに利用することの有用性を述べた [18]．彼は端末の姿勢を利用するアプリケーションのプロトタイプとして，端末の傾きによって円柱メニューとパイメニューからメニューを選択する TILT MENUs と，端末に付いたボタンと端末の傾きによって地図の支店を制御する NAVIGATING A LARGE 2D SPACE を作成した．この研究が発表されて以降，加速度センサをコンピュータとのインタラクションに用いる研究が多くなされている．

Miura らは RodDirect[15] を提案している．RodDirect はスタイラスをペン以外のものに見立てたインタラクション手法である．通常スタイラスは手に持って扱うデバイスであるが，RodDirect ではスタイラスを格納場所に挿した状態で利用する．スタイラスを格納場所から出

し入れするときの平行移動量と，格納場所でスタイラスを回転させるときの回転量をスクロール操作などのインタラクションに利用する．RodDirect では，実世界メタファとしてスタイラスを巻き物の芯に見立てたメタファなどを用いている．

野中らはペン型の入力デバイス，Tilt-Stick&Spin-Stick[33] を開発した．彼らはペン型デバイス Tilt-Stick&Spin-Stick をペンらしく使うのではなく，ペンでは通常行われない動作をインタラクションに利用する．例えば Tilt-Stick&Spin-Stick をジョイスティックとして用いる場合，デバイスを倒立させているときに電源 OFF 状態にし，正立したときに電源 ON 状態する．電源が ON の状態でデバイス下部を何からの物体に近接させるとジョイスティックとして機能する．

Wilson らはリビングルームにあるような電子機器を統一的に操作することができるデバイス XWand [26] を開発した．XWand には加速度センサやジャイロセンサなどが付いており，XWand を振ったりねじったりすることで，それが指している方向にある電子機器を操作することができる．例えば，電灯の方向に XWand を向けてボタンを押下すると，その電灯の ON/OFF を切り替えることができる．

Hinckeley らはジェスチャによりモバイル端末を操作する手法について述べた [9]．彼らはモバイル端末に加速度センサやタッチセンサなどを付加し，端末の傾きを利用したスクロール操作や，ディスプレイの周囲に触れることによるページ送りなどを実現している．

Siio の開発した Scroll Display[21] や，Fällmana らの開発した ScrollPad[5] は実空間におけるモバイル端末の平面上の移動量を利用したインタラクション手法である．モバイル端末の移動量により画面のスクロールやメニュー操作を行うことができる．

Yee らはペン入力と端末の移動を使ったインタラクション手法である Peephole Display[27] を提案している．Peephole Display では PDA の３次元の位置や移動量を用いたインタラクションが可能である．例えば，PDA の平面上の移動を利用して PDA の小画面から仮想的な大画面の操作を行ったり，PDA を持ち上げることによるズームアウトなどが行える．

Wigdor らは携帯電話の傾きを利用した文字入力手法として TiltText[25] を提案している．TiltText では，携帯電話の傾きとテンキーを用いて文字を入力する．携帯電話の傾いている方向と傾き具合によって，入力される文字の種類が決定される．例えば，アルファベットの ‘s’ を入力する場合，通常の方法を用いるとテンキーの ‘７’ を４回押下する必要があるが，TiltText を用いると端末を下に少し傾けてテンキーの ‘７’ を１回押下するだけで入力が行える．

6.2.3　スタイラス向けインタフェースに関する研究

ペン入力向けのメニューはさまざまな研究が行われている．Hopkins はペン型デバイスで扱いやすいメニューとして，Pie Menu[10] を提案している．従来のポップアップメニューはメニュー項目が線形に並んでいるのに対し，Pie Menu ではメニュー項目が放射状に配置されている．従来の線形に並んだメニューではユーザはメニュー項目の位置を確認して選択する必要があったが，Pie Menu ではユーザは位置ではなく方向でメニュー項目を選択することができる．よって，操作に慣れてくるとメニュー表示を見ることなく項目を選択できるようにな

る．Pie Menu と線形メニューの比較実験では，選択時間やエラー数に関して Pie Menu の方が有利であるという結果が示された [4].

さらに，Kurtenbach らは Pie Menu のメニュー表示がなくても，ペンの動きだけでメニュー項目を選択できる手法，Marking Menu[11, 12] を提案している．階層化された Marking Menu を操作する場合，ジェスチャのような感覚でメニュー項目を選択できる．Marking Menu の問題点としては，ジェスチャとメニュー項目とのマッピングを記憶する必要があるため，初心者にはほとんど使えないという点である．

Guimbretiére らはマウスストロークを用いるメニューとして FlowMenu[6] を提案している．FlowMenu は Pie Menu と同様に放射状にメニュー項目を配置したメニューであるが，選択方法が Pie Menu とは異なる．Pie Menu ではメニュー項目をタップにより選択するが，FlowMenu ではメニュー項目を選択した後に開始した位置まで戻すようなストロークにより選択する．また，表示されるメニュー上で円形のストロークを描くことでパラメータ調整なども行える．

Pook らは FlowMenu と同様にマウスストロークを用いたメニューとして Control Menu[17] を提案している．Control Menu はメニュー項目の選択と，選択したメニューのパラメータ変更が１ストロークで行うことができる．変更されるパラメータはストロークを開始した点からの距離によって決定される．

これらのメニューを応用した研究も行われている．Marking Menu を応用した研究として，Hinckley らは Pigtail を用いて Marking Menu を表示するシステム，Scriboli[8] を開発した．Pigtail とはペンストロークの最後にブタのしっぽのような形状であり，これを描くことで操作を行うテクニックである．Scriboli では操作対象を囲むストロークの後に Pigtail を描くことで，Marking Menu を表示させ，実行するコマンドを選択する．

FlowMenu の応用例として，Perlin が開発した Quikwriting[16] がある．Quikwriting は円形メニューの周囲に配置されたアルファベットなどを FlowMenu と同じ操作方法で選択して文字を入力する手法である．さらに FlowMenu を日本語入力インタフェースに応用した研究として，佐藤らの Popie[36] がある．Popie では円形メニューの周囲に配置された子音を FlowMenu と同じ操作方法で選択し，提示されるいくつかの候補の中から適当なものを選択することで文字入力を行う．キーがキーボードのように並んだソフトキーボードでは１文字ごとにペン先をディスプレイから離す必要があるが，これらの手法を用いるとペン先をディスプレイから離すことなく，１ストロークで１つの文章を入力することができる．

メニュー以外のスタイラス向けインタフェースに関する研究もさまざまなものがある．GUI においてコマンドを実行する操作として，Accot らの提唱したクロッシング [1] という手法がある．クロッシングとはターゲットを横切る操作である．マウスのクリックに相当する操作として，スタイラスではターゲットを１点でポインティングする手法であるタッピングが標準となっている．しかしながら，ペン先をディスプレイに接触させるときに正確な位置にポインティングできないことが多いため，小さなターゲットを狙ったタッピングは非常に難しい．一方，クロッシングはペンをディスプレイに接触させた後の動きにより操作を行うため，タッピングに比べて正確な操作を行いやすい．

クロッシングを応用した研究として，Apitz らはあらゆる操作をクロッシングにより行うド

ローイングツール，CrossY[2] を開発した．CrossY では，ラジオボタンやスクロールバーなどの GUI 部品をクロッシングによって操作することができる．ドローイングツールの機能としては，描画色の選択，ペンやブラシの選択，スクロール操作などがある．

また，Smith らは The radial scroll tool[22] を提案している．これは，ディスプレイ上をスタイラスでタップするとその位置にガイドが表示され，ガイド上で円を描くジェスチャを行うことでスクロール操作が行えるというツールである．タップした位置がガイドの中心位置となり，そこから円を描くジェスチャを行う場所までの距離によってスクロール量が変化する．ガイドの中心付近でジェスチャを行うとスクロール量が多くなり，ガイドから離れた場所でジェスチャを行うとスクロール量が少なくなる．

さらに，Smith らは Curve Dial[13] を提案している．Curve Dial は the radial scroll tool と同様に円を描くジェスチャでスクロールを行うツールであるが，ジェスチャの描き方に違いがある．the radial scroll tool では最初にタップした位置を常に中心にして円を描くジェスチャを行う必要があったが，Curve Dial では円の中心を移動させながら円を描くジェスチャを行うことができる．よって，Curve Dial ではジェスチャ操作に視線を向ける必要がなくなる．

Matsushita らの DualTouch[14] は，PDA を複数の指で操作するインタフェースである．彼らは抵抗皮膜型タッチパネルの特性を利用して，市販のシングルタッチインタフェースをマルチタッチインタフェースにする技術を開発した．DualTouch では PDA を持つ手の親指とスタイラスを用いたマルチタッチ操作，もしくは PDA を持たない手の 2 つの指を用いたマルチタッチ操作を想定しており，地図の回転や文字入力を行うアプリケーションの試作を行っている．

Rekimoto はペンを使って複数のコンピュータ間でデータをやり取りする手法として Pick-and-Drop [19] を提案している．Pick-and-Drop とは，ドラッグ・アンド・ドロップの概念を発展させたもので，ペンを持ち上げることでデータをつまみ上げ，ペンを下ろすことでデータを落とすというイメージである．例えば，Pick-and-Drop を用いると複数のコンピュータ間でコピー&ペーストを容易に行うことができる．

平岡らは Behind Touch[37] を提案している．Behind Touch は携帯端末を対象とした入力デバイスとユーザインタフェースであり，携帯端末の背面に配置されたキーを操作する．ユーザはキーを直接視認することができないが，表面のディスプレイに指の位置を表示することによって直接的な操作が行える．

廣木らは Hybrid Touch[29] を提案している．Hybrid Touch はモバイルデバイスの表裏を使ったインタラクション手法である．通常のモバイルデバイスではタッチディスプレイ上でし入力を行うことができないが，彼らはモバイルデバイスの裏面にタッチパッドを取り付けることで，裏面からの入力を可能にした．Hybrid Touch の応用として，地図ビューアやドローツールなどを開発している．Hybrid Touch を用いることでディスプレイの小さなデバイスでも煩雑な操作が可能になる．

6.3　本研究の位置付け

6.1 節で述べたように，本研究ではスタイラスを用いるシステムにおけるメニュー操作や，スタイラスがペン型であることを有効に活用した入力操作の少なさを問題点として捕らえており，これらを解決する手段として，実世界指向というアプローチをとっている．具体的には，空中での手の動きに着目したインタラクション手法であるショートカットアクションの提案と，ショートカットアクションを用いたメニューに関する研究を行っている．

よって，本研究は人間の動作に着目した研究であり，かつインタラクション手法とメニュー操作について研究である．HCI 分野における本研究の位置付けとしては図 6.1 で表すことができる．

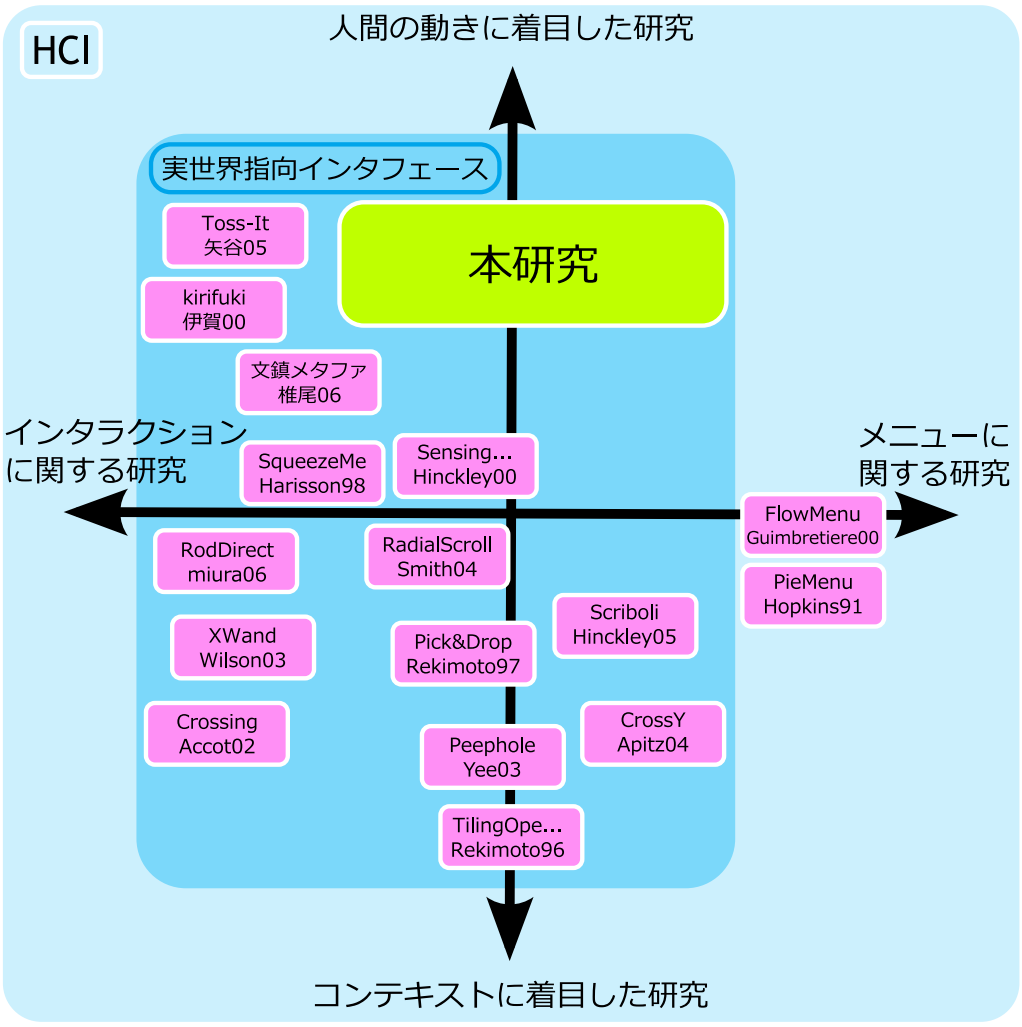


図 6.1: 本研究の位置付け

第7章　結論

本研究では，入力デバイスとしてスタイラスを用いるシステムにおける問題点として，メニュー操作のユーザビリティの低さや，スタイラスがペン型であることを有効に活用した入力操作の少なさに着目した．問題点を解決するために，人間の動きに着目したインタラクション手法を導入した．そして，人間の動きに着目したインタラクション手法として，空中での手の動きを利用したインタラクション手法であるショートカットアクションを提案した．ショートカットアクションを実現するためのスタイラスとして，加速度センサの付加したスタイラス，Context Sensitive Stylus を作成した．さらに，ショートカットアクションを応用するアプリケーションである Oh! Stylus Draw をはじめとするアプリケーション群，Oh! Stylus Series を開発した．Oh! Stylus Series のアプリケーションを用いた評価実験を行い，ショートカットアクションの有用性を示すことができた．

本研究により，今まで有効活用されてこなかった空中での手の動きを，新しいインタラクション手法として有効活用できるようになった．それにより，ユーザの視線遮断に関する問題，細かいポインティング操作に関する問題，ディスプレイサイズに起因する問題などの，入力デバイスとしてスタイラスを用いるシステムにおける諸問題を解決することができた．

謝辞

本研究を行うにあたり，指導教員である田中二郎教授には終始ご指導とご助言を頂きました．さらに，学外にて研究発表を行う機会を与えていただき，充実した研究室生活を送ることができました．この場を借りて深く御礼を申し上げます．

副指導教員である三末和男准教授には研究を進め方，論文の執筆等において上で大変貴重なご意見を頂きました．心からの感謝を申し上げます．

また，筑波大学システム情報工学研究科コンピュータサイエンス専攻の高橋伸講師，志築文太郎講師にはゼミ等を通じて多くのアドバイスや激励を頂きました．心から感謝致します．

本研究を行うにあたり，Nokia Research Center Tokyo の方々には Cookie センサを貸与して頂いただくと共に，Cookie センサに関する技術的なご指導を頂きました．ここに深く感謝致します．

そして，筑波大学システム情報工学研究科コンピュータサイエンス専攻インタラクティブプログラミング研究室のメンバーには，公私共に大変お世話になりました．特に，インタラクティブプログラミング研究室 NAIS チームの酒井慎司君，佐藤修治君，中園長新君には多大なサポートをして頂きました．また，インタラクティブプログラミング研究室の先輩である石原正樹さんには研究に対するさまざまなご指導を頂きました．心から感謝致します．

最後に 2 年間の研究生活を行うにあたって，精神面や金銭面で多大なサポートをしてくれた家族，そして全ての友人に心より感謝を申し上げます．有難うございました．

参考文献

[1] Johnny Accot and Shumin Zhai. More than dotting the i's - foundations for crossing-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'02)*, pp. 73–80, 2002.

[2] Georg Apitz and François Guimbretiére. Crossy: A crossing-based drawing application. In *Proceedings of the 17th annual ACM Symposium on User Interface Software and Technology (UIST'04)*, pp. 3–12, 2004.

[3] Won-Chul Bang, Wook Chang, Kyeong-Ho Kang, Eun-Seok Choi, Alexey Potanin, and Dong-Yoon Kim. Self-contained spatial input device for wearable computers. Proceedings of the Seventh IEEE International Symposium on Wearable Computers (ISWC ð3), pp. 26–34, 2003.

[4] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'88)*, pp. 95–100, 1988.

[5] Daniel Fällmana, Andreas Lund, and Mikael Wiberg. Scrollpad: Tangible scrolling with mobile devices. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, p. 90286c, 2004.

[6] François Guimbretiére and Terry Winograd. Flowmenu: Combining command, text, and data entry. In *Proceedings of the 13th annual ACM Symposium on User Interface Software and Technology (UIST'00)*, pp. 213–216, 2000.

[7] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'98)*, pp. 17–24, 1998.

[8] Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, and François Guimbretiére. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'05)*, pp. 451–460, 2005.

[9] Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM Symposium on User Interface Software and Technology (UIST'00)*, pp. 91–100, 2000.

[10] Don Hopkins. The design and implementation of pie menus. In *Dr. Dobb's Journal*, Vol. 16, pp. 16–26, 1991.

[11] Gordon P. Kurtenbach, Abigail J. Sellen, and William A.S. Buxton. An empirical evaluation of some articulatory and cognitive aspects of marking menus. In *Journal of Human Computer Interaction*, Vol. 8, pp. 1–23, 1993.

[12] Gordon Kurtenbach and William Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'93)*, pp. 482–487, 1993.

[13] m. c. schraefel, Grham Smith, and Patrick Baudisch. Curve dial: Eyes-free parameter entry for guis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'05)*, pp. 1146–1147, 2005.

[14] Nobuyuki Matsushita, Yuji Ayatsuka, and Jun Rekimoto. Dual touch: a two-handed interface for pen-based pdas. In *Proceedings of the 13th annual ACM Symposium on User Interface Software and Technology (UIST'00)*, pp. 211–212, 2000.

[15] Motoki Miura and Susumu Kunifuji. Roddirect: Two-dimensional input with stylus knob. In *Proceedings of the 8th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI'06)*, pp. 113–120, 2006.

[16] Ken Perlin. Quikwriting: Continuous stylus-based text entry. In *Proceedings of the 11th annual ACM Symposium on User Interface Software and Technology (UIST'98)*, pp. 215–216, 1998.

[17] Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Control menus: Excecution and control in a single interactor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'00)*, pp. 263–264, 2000.

[18] Jun Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of the 9th annual ACM Symposium on User Interface Software and Technology (UIST'96)*, pp. 167–168, 1996.

[19] Jun Rekimoto. Pick-and-drop: A direct manipulation technique for multiple computer environments. In *Proceedings of the 8th annual ACM Symposium on User Interface Software and Technology (UIST'97)*, pp. 31–39, 1997.

[20] Jun Rekimoto. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *Proceedings of the Fifth International Symposium on Wearable Computers (ISWC'01)*, pp. 21–27, 2001.

[21] Itiro Siio. Scroll display: Pointing device for palmtop computers. In *Proceedings of the Third Asian Pacific Computer and Human Interaction (APCHI'98)*, pp. 243–249, 1998.

[22] G. M. Smith and m. c. schraefel. The radial scroll tool: scrolling support for stylus- or touch-based document navigation. In *Proceedings of the 17th annual ACM Symposium on User Interface Software and Technology (UIST'04)*, pp. 53–56, 2004.

[23] Yu Suzuki, Kazuo Misue, and Jiro Tanaka. Interface for digital notes using stylus motions made in the air. In *Proceedings of the 2nd International Conference on Knowledge, Information and Creativity Support Systems (KICSS2007)*, pp. 104–109, 2007.

[24] Yu Suzuki, Kazuo Misue, and Jiro Tanaka. Stylus enhancement to enrich interaction with computers. In *Proceedings of 12th International Conference on Human-Computer Interaction (HCI International 2007)*, LNCS4551, pp. 133–142, 2007.

[25] Daniel Wigdor and Ravin Balakrishnan. Tilttext: Using tilt for text input to mobile phones. In *Proceedings of the 16th annual ACM Symposium on User Interface Software and Technology (UIST'03)*, pp. 81–90, 2003.

[26] Andrew Wilson and Steven Shafer. Xwand: Ui for intelligent spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*, pp. 545–552, 2003.

[27] Ka-Ping Yee. Peephole displays: Pen interaction on spatially aware handheld computers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03)*, pp. 1–8, 2003.

[28] 椎尾一郎, 辻田眸. 文鎮メタファを利用した小型情報機器向けインタフェース. 情報処理学会論文誌, 第 48 巻, pp. 1221–1228, 2007.

[29] 廣木桂一, 矢谷浩司, 杉本雅則. モバイルデバイスの表裏両面からの操作を連携させることによる効率的な入力手法. インタラクション 2005 論文集, pp. 61–62, 2005.

[30] 岡田謙一, 西田正吾, 葛岡英明, 仲谷美江, 塩澤秀和. ヒューマンコンピュータインタラクション. オーム社, 2002.

[31] 矢谷浩司, 田村晃一, 広木桂一, 杉本雅則. toss-it: モバイルデバイスにおける「トス」や「振り」の動作を用いた情報の移動を直感的に実現するインタフェース技術. インタラクション 2005 論文集, pp. 151–158, 2005.

[32] 塚田浩二, 増井俊之. Aftertouch: 日常的な動作と id システムの相補的拡張. インタラクション 2006 論文集, pp. 53–55, 2006.

[33] 野中秀俊, 栗原正仁. Tilt-stick & spin-stick: 特定アフォーダンスに基づくペン入力デバイス. インタラクション 2006 論文集, pp. 43–44, 2006.

[34] 伊賀聡一郎, 伊藤英一, 安村通晃. Kirifuki: 呼気・吸気を利用した gui 操作環境の提案. 情報処理学会 HI 研究会 2000-HI-87(情処研報 Vol.2000,No.12), pp. 49–54, 2000.

[35] 伊賀聡一郎, 新西誠人, 明関賢太郎, 樋口文人, 安村通晃. やわらかインタラクション: 「やわらかい世界」を対象にしたユーザインタフェース. インタラクティブシステムとソフトウェアに関するワークショップ VI (WISS'98), pp. 137–142, 1998.

[36] 佐藤大介, 志築文太郎, 三浦元喜, 田中二郎. Popie：フローメニューに基づく日本語入力手法. 情報処理学会論文誌, 第 47 巻, pp. 2305–2316, 2006.

[37] 平岡茂夫, 宮本一伸, 富松潔. Behind touch: 携帯電話のための背面・触覚操作インターフェースによる文字入力. インタラクション 2003 論文集, pp. 131–138, 2003.

[38] 鈴木優, 三末和男, 田中二郎. ペンによる自然な振る舞いを利用したインタラクション手法. 第 14 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2006), pp. 141–142, 2006.

[39] 鈴木優, 三末和男, 田中二郎. ユーザに優しいディジタルノート向けスタイラスインタフェース. 第 15 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2007), pp. 69–74, 2007.