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

ペンによるメニュー選択に基づく
日本語入力手法の研究

佐藤 大介

(コンピュータサイエンス専攻)

指導教官　田中 二郎

２００５年 １月

概 要

近年, ペン型デバイスを用いて操作するいわゆるペンコンピュータの数が増加している. しかし, 多くのペンコンピュータはマウス・キーボードのために設計されたインタフェースを, ほぼそのまま利用していた. そのため, キーボードの代わりになるソフトウェアキーボードが他の操作の邪魔になったり, メニューバーなどのインタフェースが画面上に散在しているため, 画面に対して絶対的な位置で操作するペンでは, 画面の大きさに比例してペンを動かす量が増えてしまったりする問題があった.

本研究では, これらの問題に対しフローメニューという円形のポップアップメニューに着目した. フローメニューはポップアップメニューのため必要なときに表示することができる. また, ストロークを使った操作方法により, 文字入力やメニューの実行を一つのインタフェースで素早く行えるため, このメニューを使うことで, 上記の問題を解決できると考えている.

しかし, フローメニューでは日本語のような変換を伴う入力が考慮されていなかった. そこで, フローメニューを拡張することで, 日本語入力を可能にするシステム Popie を提案し, 実装を行った. Popie の特徴は子音を使った入力と, 予測・補完による候補の生成, および候補が多すぎた場合に適宜母音を選択という機能である. またフローメニューの特徴を生かして, 素早い日本語入力を行うことができる.

本研究では日本語入力システム Popie の評価も行い, 子音入力・予測・補完・母音選択による入力手法の有効性を示し, 拡張したフローメニューのインタフェースに関する検証も行った. さらに, 専用のアプリケーションでしか利用できなかったフローメニューの汎用化について検討し, 様々なアプリケーションに対して文字入力やメニュー操作が行えるシステムの実装も行った.

目 次

図 目 次

表 目 次

第1章　序論

コンピュータを操作するためのデバイスはマウス・キーボードが一般的であるが, ペン型のデバイスを使ったコンピュータとのインタラクションは, 1963 年の Ivan Sutherland の Sketchpad[1] が最初である. 近年では, PDA やタブレット PC のようなペン・コンピュータの数が増え, ペンによるコンピュータとのインタラクションがより一般的になってきている. また, プラズマテレビにタッチパネルを組み合わせた大画面の電子ホワイトボードシステムも登場し, 小画面から大画面までペン・コンピュータは多様化している. 今後訪れるユビキタス社会では, 様々な場所にコンピュータがあり, いつでも手軽にコンピュータを利用できるように, 指やペンで操作するコンピュータの重要性は益々増えると考えられる.

しかし, 現在使われているペンで操作するコンピュータは, マウスのクリックとペンのタップを対応させ, キーボードの代わりにソフトウェアによって実現した文字入力のアプリケーションを使用している. これにより, 広く普及したマウス・キーボードのためのインタフェースをそのまま利用することができるが, マウス・キーボードはペンとは性質の異なるデイバスであり, ペンにとっては使いづらいものになっている.

マウス・キーボードのために設計されたインタフェースをペンで操作する場合の具体的な問題点を 3 つ挙げる.

- 図 1.2 に破線で示した, 画面の上下に配置されたメニューバーや, 画面中央のソフトウェアキーボードのように, 操作のためのインタフェースが散在しているため, インタフェースとインタフェースの間でペンを移動させなければならない. さらに, ペンは画面に対して直接的な操作をするため, 画面の大きさに比例して移動量が増えてしまう.

図 1.1: PDA, タブレット PC, タッチパネル付きプラズマテレビ

- 図 1.2 の中央に表示されているソフトウェアキーボードのように, キーボードの代わりになるインタフェースが, 常に一番手前に表示されるため, 文字入力以外の操作の邪魔になってしまう.
- ペンのタップ操作は, 画面が滑りやすいため, ペン先がタップ位置とずれてしまい, 操作ミスや, タップとして認識されないことが多い.

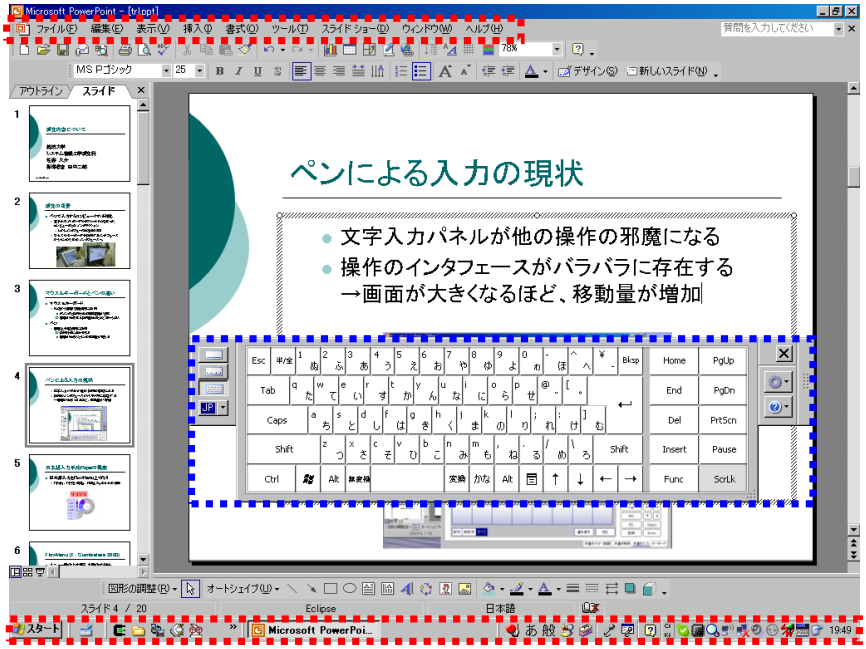


図 1.2: ソフトウェアキーボードを表示した画面の様子

研究の目的

マウス・キーボードのために設計されたインタフェースをペンで操作する場合の問題点を解決し, ペンによるコンピュータとのインタラクションをより使い易くすることが, 本研究の目指す目標である.

そこで, 本研究で注目したのが, FlowMenu[2] というメニューである. このメニューは, メニュー項目の選択にストロークを用いるという特徴を持ち, 複数の項目を連続したストロークで選択することができる. これにより, メニュー操作や文字入力を同じインタフェースとして実現でき, ペンの移動量を少なくすることが可能である. また, このメニューはポップアップ型の円形のメニューであり, 必要なときに表示させれば良く, ストロークを使った操作を行うのでタップ動作をしなくて良い.

しかし, FlowMenu では変換を伴う言語の入力について考慮されていないため, そのままでは日本語入力環境で使うことは難しい. また, FlowMenu は専用のアプリケーションに実装さ

れているため, 利用範囲が限られているが, マウス・キーボードのためのインタフェースを持つペン・コンピュータが増加する現状において, それらのコンピュータにおいて利用できる汎用的なシステムが必要であると考える. これらを踏まえ,

本研究では, フローメニュー上で日本語を入力するシステム Popie の開発と, その評価による有効性の確認ならびに, Popie を含むフローメニューを, 汎用化しペン・コンピュータのインタフェースを改善することを目的としている.

Popie は子音によって日本語を入力システムであり, 入力された子音列からユーザが意図した単語を予測して候補を提示する. 評価では, 子音による入力手法そのものの有効性と, ユーザによる Popie の入力実験, インタフェースの検証を行った. また Windows の様々アプリケーションに対して日本語入力や, メニュー操作を行えるシステムの実装を行った.

本論文の構成

本論文の構成は以下の通りである.

第 2 章では, 本研究に関連するメニューや文字入力などに関する研究について述べる. 第 3 章では, 本研究で提案し実装したペンにより日本語入力を行うシステム Popie について述べる. Popie のインタフェースと操作例, および実装について示す. 第 4 章では, Popie に関する評価について述べる. 第 5 章では, Popie をさまざまなアプリケーションで利用するための汎用化について述べる. 第 6.4 章で日本語入力システム Popie についての考察を行い, 第 7 章でまとめる.

なお, 付録 A では本研究で作成した子音入力エンジンの仕様について説明し, 付録 B に日本語入力手法 Popie を実装したシステム PopiePoint のマニュアルを載せた.

第2章　関連研究

本章では, 本研究で提案する日本語入力システム Popie に関連する, メニュー, アルファベット・かな入力, 日本語入力手法についての関連研究を述べる.

2.1　メニュー

Ben Shneiderman の文献 [3] には, “メニュー選択アプリケーションは 2 つの項目から選択する些細なものから, 何千もの項目を表示する複雑な情報システムに及ぶ” [1]とある. そして, それらは構造的に, 単独のメニュー, 線形に連続するもの, 木構造, 非循環のネットワーク, 循環するネットワーク[2]のような分類ができ, 最も一般的なメニューは木構造だとしている. また, メニューに使われるインタフェースとしては, ラジオボタン, チェックボックス, プルダウン・ポップアップ型のメニュー, 表形式のメニュー (two-dimensional menus), ハイパーテキストのリンク, ツールバーやパレットなどがあり, これらを組み合わせたダイアログボックスなどがある.

本節では, 木構造を持つポップアップ型のメニューを紹介し, 比較を述べる.

2.1.1　ポップアップ型メニュー

最も良く使われるポップアップ型のメニューは, 矩形でメニュー項目を縦方向に並べたリストになっており, 沢山のメニュー項目を並べて見せることに適している. しかし, ユーザは常にマウスカーソルとメニューの位置を見て操作を行う必要がある. なぜならば, 各メニュー項目がマウスカーソルの縦方向の移動距離という 1 次元的な情報で特定されるのに対し, ユーザがマウスカーソルを見ずに特定の距離だけそれを移動させることが難しいからである.

この矩形のメニューに対し, 2 次元的な情報によってメニュー項目を特定できる Pie Menu [4] がある. これは円形のメニューであり, 1 度に 8〜12 個程度[3] までのメニュー項目を表示することしかできないが, マウスカーソルの移動方向, 例えば上や右下などでメニュー項目を特定することができる. [6] によれば, Pie Menu では矩形のメニューに比べて, メニュー選択までの時間が短く, エラーも少ないとしている.

これらのメニューでは, マウスボタンのプレスまたはリリースによって, メニュー項目に対

[1]原文は “Menu-selection applications range from trivial choices between two items to complex information systems that offer thousands of displays.”

[2]原文は, Single Menus, Linear Sequence, Tree Structure, Acyclic Network , Cyclic Network

[3][5] の実験によれば, 分割数が増加すると反応時間とエラー回数も増加してしまうため, むやみに分割数を増やすことはできない. またメニュー項目を表示するスペースの制約からも分割数を増やす事ができない.

応するコマンドが実行される. 一方, ペンやマウスのストロークを使うことにより, メニュー項目の選択を行う Marking Menu [7] や FlowMenu [2], Control Menu [8] などのメニューがある.

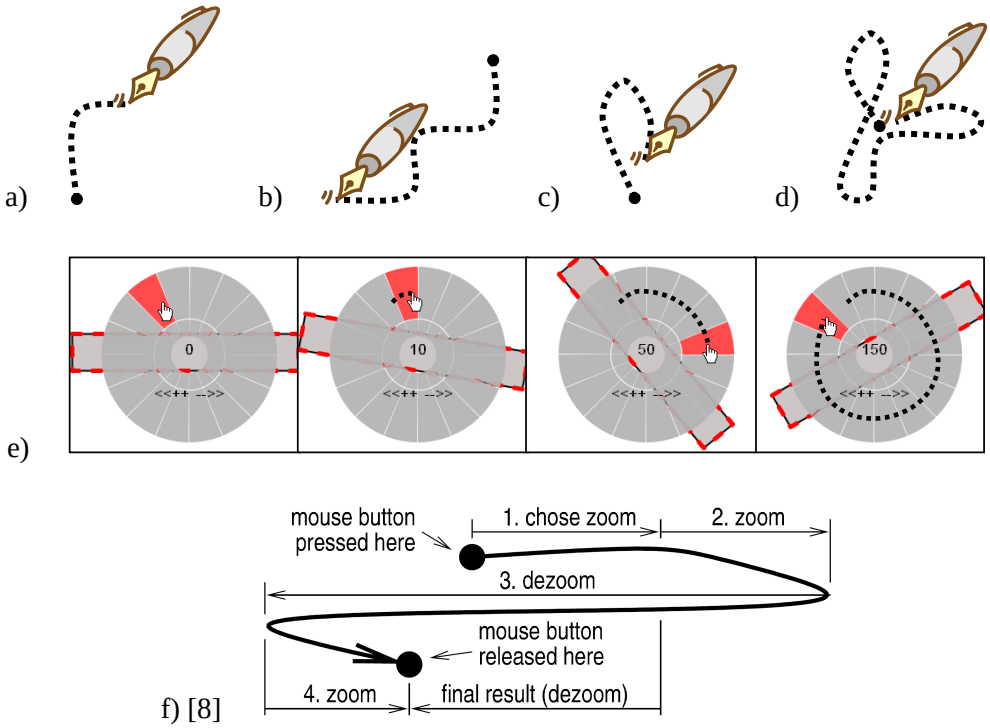


図 2.1: Marking Menu (a,b), FlowMenu (c,d,e), Control Menu (f) におけるペンのストローク

Marking Menu では, “上, 右” (図 2.1 a) のようにストロークを描くとその方向に対応したコマンドが実行される. “上, 右” は 2 階層であるが, “下, 左, 下, 左” (図 2.1 b) のように複数の階層を表現可能である. しかしこのメニューでは, 階層が深くなるほど, 最初の位置から離れたところまでストロークを描かなければならないことがある. それに対して FlowMenu では, ペンが元の位置に戻るようにストロークを描くことで (図 2.1 c), 1 つのコマンドを実行する. このため, 複数のコマンドを連続したストロークで実行可能である (図 2.1 d). また, 円形のストロークを描く事で連続的なプロパティを設定するような操作も可能である (図 2.1 e). Control Menu はストロークによって実行するメニューであるが, ズームや回転などの連続したプロパティを入力することに特化したメニューである. ペンを動かして方向によって変化させたいプロパティを決定したあと, ストロークの起点からペンがどれだけ離れているかでその値を決定している (図 2.1 f).

2.1.2 ポップアップ型メニューの比較

表 2.1 にこれらのメニューの特徴をまとめた. 矩形のメニューと円形のメニューの最も大きな違いは, メニュー選択時に, ペン・マウスの動く方向によってメニュー項目を特定できるか

表 2.1: 矩形のメニューと円形のメニューの特徴

	メニューの実行	特徴
矩形のメニュー	プレス・リリース	○多くのメニューを一度に表示できる ×メニュー項目を見ていないと選択ができない
Pie Menu	プレス・リリース	○方向によってメニューを選択できる ×１度に 8 個程度のメニューしか表示できない
Marking Menu	ストローク	○方向によってメニューを選択できる ×１ストロークに対して１つの操作を実行
FlowMenu	ストローク	○方向によってメニューを選択できる ○１ストロークに対して複数の操作を実行
Control Menu	ストローク	○方向によってメニューを選択できる ×プロパティの設定に特価している

どうかである. この特徴により, 円形のメニューではメニュー項目の位置を覚えることによって, ペンマウスの動きを目で追いかけずにメニューの選択ができ, 選択操作を素早く行うことが可能である.

本研究でフローメニューに注目した理由は, 1 つの連続したストロークによって複数の操作を実行できることと, メニュー項目の位置を覚えることによって, 素早いメニュー選択ができることである.

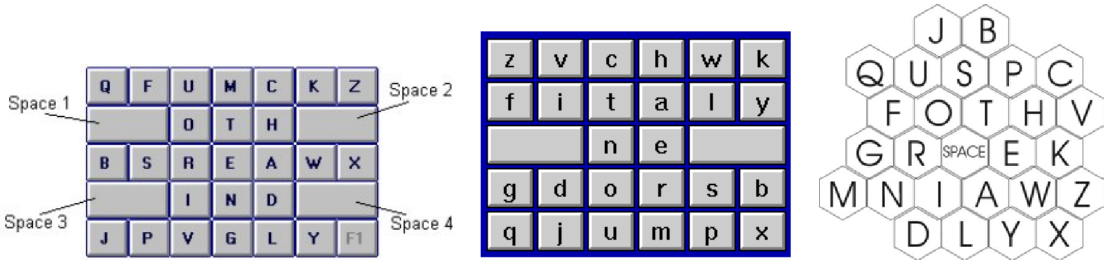
2.2 アルファベット・かな入力

ペンによる文字入力は, ソフトウェアキーボードを用いて入力する方法, ペンで文字に対応するストロークを描き認識することで入力する方法, 円形メニューを使って入力する方法の大きく 3 つに分類することができる.

2.2.1 ソフトウェアキーボードによる入力

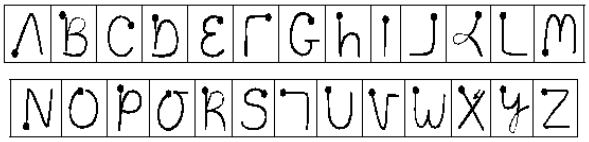
ソフトウェアキーボードによる入力は, ペンで, ソフトウェアキーボード上のボタンをタップすることで, 一文字ずつ入力する. キーの配列としては, 実際のキーボードでも良く使われている Qwerty 配列の他に, ペンの移動が最小限に抑えられるようにキーが配置された, OPTI [9], FITALY [10] や, 六角形のボタンを使った Metropolis [11] などがある (図 2.2). これらのソフトウェアキーボードでは, Qwerty 配列を用いたものに比べて入力速度が向上している.

ソフトキーボードは, キーボードを使用している人にとって利用しやすいものであるが, タップを繰り返す必要があり, 入力したい文字のボタンを常に目で追わなければならない. また, ソフトキーボードは表示／非表示の切り替えに時間がかかったり, ボタンを小さくするとエラーが増えるので, 画面を占有する領域が大きくなって他の作業の邪魔になるなどの問題がある.

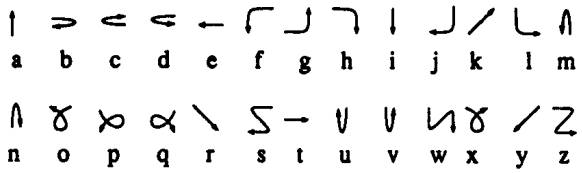


OPTI [9]　　FITALY [10]　　Metropolis [11]

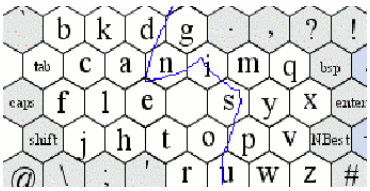
図 2.2: ペンのためのソフトウェアキーボード



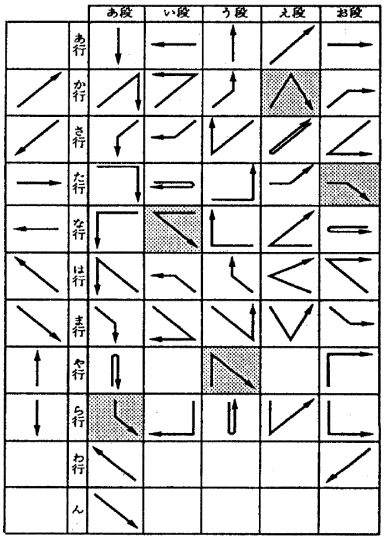
a) Graffiti [12]



b) Unistroke [13]



d) SHARK^2 (“using” に対応) [14]



c) かな Unistrok [15]

図 2.3: Graffiti, Unistroke, かな Unistroke, SHARK^2 のストロークと文字の対応

このため, ペンで利用するにはあまり向いていないと考えられる.

2.2.2　ストロークの認識による入力

ストロークの認識により入力する方法には, Graffiti [12], Unistroke [13], SHARK^2 [14] などがある.

Graffiti, Unistroke は 1 ストロークで 1 文字を入力する. Graffiti ではアルファベットの形に近いストロークを使い (図 2.3 a), Unistroke ではストロークを短くして高速に入力することを目

的としているため, アルファベットの形と対応のないストロークが多くなっている (図 2.3 b). また Unistroke を平仮名に対応させた, かな Unistroke [15] が提案されているが, 仮名の子音と母音にストロークの方向を対応させ, その組合せによりストロークが定義されるので, かなの形とは全く異なるストロークになってしまう (図 2.3 c).

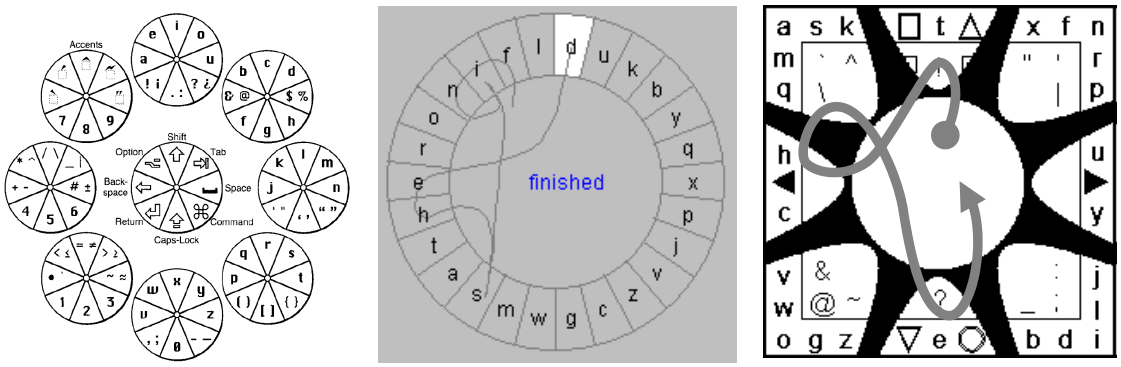
一方 SHARK^2 は, 1 ストロークで 1 単語を入力する. ソフトウェアキーボード上にストロークを描くことで入力を行うが, 入力される単語は, ストロークが通る位置にあるキーから適当なものを推測している (図 2.3 d).

これらの方法では, 似たストロークがあると誤認識を起こし易いという問題点があり, SHARK^2 では特に顕著になる.

2.2.3 円形メニューによる入力

円形メニューを使って入力する方法には, アルファベットを入力対象とした, T-cube [16], Cirrin [17], Quikwriting [18] や, T-cube を仮名にアレンジした, かな T-cube [15] などがある. T-cube, かな T-cube では 1 ストロークで 1 文字を入力するもので, ストロークの始まりの位置と, ストロークを描く方向によって入力される文字が決定される (図 2.4 a).

一方 Cirrin, Quikwriting は 1 ストロークで複数文字を入力する方法である. Cirrin では, 円周を 26 分割してそれぞれにアルファベットを割り当て, ペンが横切った部分のアルファベットを入力することができるが, 分割が細かくなりすぎて, 選択ミスが増えると予想できる. 図 2.4 b は, “finished” と入力した場合の軌跡を示している. Quikwriting では入力後にペンが中心に戻るようなストロークになっているため, FlowMenu と親和性が高い. 外周に表示されている文字のうち中央の文字は, 表示のある領域にペンを移動させ中央に戻すことで入力できる. 両隣の文字は, 表示のある領域から対応する隣の領域にペンを動かしてから中央に戻すことで入力できる. 図 2.4 c は, “the” と入力した場合の軌跡を示している.



a) T-cube [16]　　b) Cirrin [17]　　c) Quikwriting [18]

図 2.4: T-cube, Cirrin, Quikwriting のシステムイメージ

2.2.4 アルファベット・かな入力の比較

表 2.2 に, これらの文字入力手法の特徴をまとめた. ソフトウェアキーボードによる入力は, キーボードの代わりとして位置づけられる方法だと考えられる. 一方, ストロークの認識による入力と円形メニューによるの入力方法は, どちらもストロークを用いてペンの特性を生かしている.

また, ペンのストロークには, 断続的なストロークを用いるものと, 連続的なストロークを用いるものとがある. 両者を比較した文献はないが, それぞれの手法による実験での入力速度を比較する限りでは, 断続的なストロークを用いる方が速い結果が出ているが, 大きな差ではないと考えられる [4].

しかし, 断続的なストロークでは, ペンを画面に付けたり離したりを繰り返すため, 連続的なストロークを用いる方法よりも疲れることが考えられる.

表 2.2: 文字入力手法

分類		特徴
ソフトウェアキーボード	Qwerty 配列	一般的なキーボードの配列. 最も普及しているが, 最適な配列でないことが知られている.
	Metropolis [11] OPTI [9] FITALY [10]	ペンを用いて入力するために最適化した配列. [11] では六角形のボタンを使っている.
ストロークの認識	Graffiti [12]	1 ストロークで 1 つのアルファベットを入力する. ストロークはアルファベットの形に近い.
	Unistroke [13]	Graffiti よりもストロークが短いが, アルファベットとは異なる形のストロークが多い.
	かな Unistroke [15]	規則的なストロークに仮名を割り当てたもの. 仮名の形とストロークの対応は全く無い.
	SHARK^2 [14]	ソフトウェアキーボード上にストロークを書き, ストロークが通るキーから適当な単語を入力する. 誤認識が多くなる.
円形メニュー	T-Cube [16] かな T-cube [15]	Pie Menu を使った文字入力手法, 1 文字を 1 ストロークで入力する.
	Quikwriting [18]	Flowmenu を使った文字入力手法, 複数文字を 1 ストロークで入力する. 円は 8 分割されている.
	Cirrin [17]	Quikwriting と同じように複数文字を 1 ストロークで入力するが, 円が 26 分割されて細かくなり, 選択がしにくい.

[4] [19] によれば, graffiti では熟達者が平均 21wpm, 初心者が平均 7wpm の入力速度であり, [16] によれば, T-Cube では入力速度は 12wpm〜21wpm に分布している. また [20] によれば, Quikwriting では平均 16wpm の入力速度である.

2.3 日本語入力手法

日本語の文章は平仮名と漢字から構成されるという特徴を持ち, アルファベットに比べ非常に文字数が多い. 入力手法としては, 平仮名と漢字を直接入力する方式と, 文章の読みを平仮名で入力してからかな漢字に変換する, という 2 段階の方式に分けられる.

1 つ目の直接入力する方式として挙げられるのは, 文字認識を用いた手書きによる入力や, 複数のキーの組合せに平仮名や漢字を 1 文字ずつ割り当てて入力する, かな漢字直接入力手法である [21] [22].

2 つ目の 2 段階の方式において, 読みを入力する方法としては, ローマ字入力やひらがな 1 文字を 1 個のキーで入力する JIS かな入力などがある. この他, 日本語入力の特徴を考慮してキーを配置したものがある [23]. かな漢字変換を行う方法としては, 最後まで読みを入力してから変換する方法と, 読みの入力途中に予測された候補を利用する変換方法がある [24] [25]. また, 子音のみの入力から変換を行う方法がある [26] [27].

以下に予測を用いた日本語入力手法と, 子音を用いた日本語入力手法について説明する.

2.3.1 予測を用いた日本語入力

POBox [24], kukura [25] などの予測を用いたシステムは, 主に携帯電話や PDA などの小型のデバイスにおける入力手法として採用されている.

これらのシステムはユーザの入力に対して, 入力よりも読みの長い単語を提示したり, 次に入力されると思われる単語を提示したりすることで, ユーザが入力をする回数を少なくし, 入力の効率を上げようとするものである.

例えば, ユーザがすでに “よろしく” と入力を済ませていた場合を考える. 次にユーザが ‘お’ と入力したら “お願いします”, “お願い致します” とユーザが入力するものと予測して候補にこれらの単語を表示するという仕組みである. これらのシステムでは, ユーザが沢山の入力をすることによって, ユーザに適応した辞書を作ることができるため, 入力効率の向上が期待できる.

2.3.2 子音を用いた日本語入力

Touch Me Key [28], T9 [26] は子音による入力から候補を予測し日本語を入力するシステムである.

子音による入力では, 例えば “走る” と入力するためにローマ字では “HaSiRu” と入力するが, 母音を取り除いた “HSR” を入力すれば良い. この方法では, 便宜的に “あ行” の子音は ‘A’ とし, ‘ん’ や ‘ー’ は “わ行” に含め, 濁点や半濁点の区別には別のキー (ここでは ‘*’) を用意しており, “安堵” と入力したい場合は “AWT*” と入力する.

この方法は 2 種類に分類できる. 1 つは T9 のような “HSR” という子音の入力に対して “はしる” や “はしら” など平仮名の候補を示し, ユーザに選択させてから, 通常のかな漢字変換をするという方法である. もう 1 つは, Touch Me Key のように “HSR” という子音の入力に対し

て “走る” や “柱” のような, かな漢字交じりの候補を示す方法である. 後者の方法では候補数がより多くなってしまうことが問題になるが, 単語の使用頻度などの情報を用いて候補を並び替え, 使われる確率の高い候補を上位に表示することで解決できるとされている [28].

2.3.3 日本語入力手法の比較

日本語入力における, 各手法の機能の位置づけを表 2.3 に示す. またそれぞれの特徴を表 2.4 にまとめた.

ローマ字入力やかな入力は変換のためのキーを入力する機能だけを持つ. そして, かな漢字変換は入力されたキーに対応する候補を提示し, 漢字かな交じりの文を入力する機能を持つ. また, 子音入力を用いる場合, かな変換を経て, かな漢字変換によって文を入力する方法と, 子音から直接かな漢字変換により文を入力する方法がある. 子音からかな漢字変換を行う場合, 母音の入力を補助していると考える事ができる. さらに, 予測を用いた変換手法を用いることでもキーの入力を補助することができる. 入力・変換というプロセスではないが, 手書き文字認識やかな漢字直接入力は, 漢字かな交じりの文を単体の手法で入力可能である.

表 2.3: 日本語入力における機能の位置づけ

	キー入力	変換
ローマ字入力	○	
かな入力	○	
子音入力	○	
かな変換		○
子音かな漢字変換	○	○
予測変換	○	○
手書き文字認識	○	○
複数キーを用いた直接入力	○	○
かな漢字変換		○

ペンを用いる場合の日本語入力の方法としては, ソフトウェアキーボードを使ってローマ字入力やかな入力を行い, かな漢字変換により入力する方法と, 文字認識によって入力する方法が多く用いられる. しかし, 第 2.2.1 節に示したのように, ソフトウェアキーボードはペンで利用するには向いていない. 文字認識は自然な入力方法ではあるが, 日本語には漢字を含む多数の文字があるため, 崩し字や続け字に対して認識精度が高くない [29]. このため, 丁寧に文字を書く必要があり入力速度はあまり速くない.

表 2.4: 日本語入力手法とその特徴

入力手法	特徴
ローマ字入力	最も一般的な方式. 入力したい文の読みをローマ字で入力するため, 仮名 1 文字に対して約 2 個のキーを入力する.
かな入力 (キー 1 個)	仮名 1 文字に対して約 1 個のキーを入力する. ローマ字入力よりも覚えなければならないキーが多い. め入力速度が上がる.
かな入力 (キー 2 個) [23]	かなを 2 個のキータイプで入力するために工夫された配列. ローマ字入力よりも左右交互に打鍵する確率が増え, 高速な入力が可能になる.
子音入力 + 2 段階変換 [26]	子音の列からひらがなの単語に変換し, その後かな漢字変換を行う. 変換操作に時間がかかる.
子音入力 + 1 段階変換 [28]	子音の列からかな漢字変換を行う. 候補が増えるため, 候補の選択に時間がかかることがある.
各種入力 + 予測変換 [24] [25]	入力に対して, ユーザが入力しようとしている単語を予測して提示する. 入力を減らす効果があり, 文の入力速度も向上する.
文字認識	紙に書くように画面に書いた文字を認識して入力する方法. 崩し字や続け字のような文字に対しては, 認識性能が十分ではないため, 入力に時間がかかってしまう.
かな漢字直接入力 [21] [22]	かなと漢字を 2 個以上のキーの組合せを用いて直接入力するための方法. 1000 文字以上のキーの組合せを覚えるのは非常に大変である.

第3章　日本語入力システムPopie

本章では本研究で開発した [30], 日本語入力システム Popie について述べる. 最初に Popie の入力方式の特徴を説明し, 次に操作例を用いたインタフェースの紹介をする. 最後に Popie の実装について述べる.

3.1　Popie の特徴

Popie はフローメニュー上で日本語入力を行うシステムである. 主な特徴は

1. 全ての操作をフローメニュー上で実行でき, その特徴を生かして素早い入力ができる.
2. 入力には基本的に子音 (10 個のキー) を用いる.
3. 入力途中の子音列から単語を補完したり, 入力された単語から次の単語を予測する.
4. 候補が多すぎる場合など, 適宜子音に対応する母音を選択することで絞込みが行える.

の 4 点である. 以下にそれぞれの特徴に関する詳細を順に述べる.

3.1.1　フローメニュー上で入力を行う利点

Popie では, 日本語入力における全ての操作をフローメニュー上で行うことができる. 全ての操作を連続したストロークで表現できるため, キー入力と候補選択の操作をシームレスに行う事が行える. したがって変換を伴う入力を必要とする日本語入力に適していると言える.

また, ソフトウェアキーボードなどボタンをインタフェースとする入力方法では, 常にユーザがペン先とボタンを目で追いかける必要があったのに比べ, フローメニューでは 10 個の入力キーの位置を覚えることで, ユーザはペンなどを目で追いかける必要がなくなり, 素早い入力が可能になる.

3.1.2　子音入力方式

Popie では子音による入力方式を採用している. 以下に他の入力方式との比較を述べ, 子音入力方式を採用した理由を示す.

日本語入力システムは様々な方式のものが開発されているが, 入力に使うキーの組合せ数と入力の曖昧さの間でトレードオフの関係がある (表 3.1). 入力に使うキーの組合せ数が増えれ

ば曖昧さは減少するが, 入力するユーザの負担が増大し, 入力に使うキーの組合せ数が減れば曖昧さは増加するが, 入力するユーザの負担は減少する.

表 3.1: 入力キーの数と曖昧さ

入力方式	ユーザへの負担			曖昧さ	
	キーの数	組合せ数			
かな漢字直接入力	26〜50 個	文字数 (数千種類)	多	なし	少
かな入力	50 個	ひらがなの数	↑	同音異義語	↑
ローマ字入力	26 個	ひらがなの数	↕	同音異義語	↕
子音入力	10 個	子音の数	↓	母音+同音異義語	↓
母音入力	5 個	母音の数	少	子音+同音異義語	多

我々は, ペンで入力を行う場合には, ユーザの負担をなるべく減らすことが重要であると考えている. ユーザの負担は, 入力に使うキーの組合せ数に比例して大きくなると考えられるため, 数千種類もの組合せを覚える必要のある, かな漢字直接入力方式は使えない.

またメニュー項目の数による制限についても考える必要がある. フローメニューのメニュー階層は理論上無限であるが, 文字入力操作の速度, 操作性を考慮して 2 階層までに制限すると, メインメニューが 8 項目, サブメニューがそれぞれ 8 項目で 64 通りの入力が行える.

しかし, “中心から上側にペンを移動し, 円の下側までペンをぐるっと移動させ中心に戻す” というような操作は時間がかかる. したがって, サブメニューはメインメニュー項目の位置の, 両側 2 つ隣までの 5 項目を使うべきであり, キーの数は $8 \times 5 = 40$ 通りに制限される. よって, キーの数が約 50 個であるかな入力は除外した.

次にローマ字入力と子音入力を比較すると, 日本語ではほぼ子音と母音を交互に打つので, 子音入力ではキーの入力回数が約半分になる. 子音入力では, 母音が抜ける分だけかな漢字変換の際の曖昧さが増えるが, 子音入力システム Touch Me Key[28] では, 使用頻度の高い候補を上位に表示することで, この問題を改善している.

さらに, 子音入力よりもキーの数が減る母音入力が考えられる. しかし, 10 個が半分の 5 個になるだけで, キーの入力回数はほぼ変わらないので, 子音が抜ける分の曖昧さは不利になるだけだと考えられる. 母音入力のシステムを試作したが, 子音入力と同じ 10 万単語レベルのコーパスではとても使いものにならなかった. 母音入力を導入するのであれば, パーソナライズするなどの工夫が必要だろう.

以上より, Popie には子音入力の方式を用いることとした. 入力には “AKSTNHMYRW” の 10 個のキーを用い, 子音のない “あ行” は便宜的に子音を ‘A’ とし, 促音 ‘ん’ や長音 ‘ー’ は ‘W’ に含めた. また濁点や半濁点, 拗音や促音に特別な子音は用意せず, “が” は ‘K’, ‘ぱ’ は ‘H’, ‘っ’ は ‘T’ のようにした. 子音キーとひらがなの対応を表 3.2 に示す

表 3.2: 子音キーとひらがなの対応

子音キー	A	K	S	T	N	H	M	Y	R	W
ひらがな	あ ぁ い ぃ う ぅ ヴ え ぇ お ぉ	か が き ぎ く ぐ け げ こ ご	さ ざ し じ す ず せ ぜ そ ぞ	た だ ち ぢ つ づ っ て で と ど	な に ぬ ね の	は ば ぱ ひ び ぴ ふ ぶ ぷ へ べ ぺ ほ ぼ ぽ	ま み む め も	や ゃ ゆ ゅ よ ょ	ら り る れ ろ	わ ゐ ん ゑ ー を

3.1.3 予測と補完

さらにユーザの負担を減らすため, 補完や予測による候補の提示も導入した. 補完とは, 単語の途中までの入力で単語を推測することであり, 例えば “こいず” という入力に対して “小泉” などの候補を提示することである. 予測とは, 入力された文章から次に入力されるよ考えられる単語を候補として提示する機能であり, “小泉” と選択した直後に “首相” や “内閣” といった候補を提示する. これらの機能は POBox [24], kukura [25] などのシステムで実現されており, ユーザの負担を減らすのに大きな効果を得ている. POBox, kukura ではかな入力やローマ字入力を用いているが, 子音入力でも, これらの機能を導入することは可能である.

3.1.4 母音による絞込み

Touch Me Key [28] では, 子音入力において曖昧さが増すことにより候補数が増すという問題点は, 文章を推測する方法を改良することで, 改善できるとしている. しかし, Touch Me Key ではコーパス中の単語のみを入力することが前提で, コーパスにない未知語を入力できない. また, 実際の入力においては出現頻度の低い単語も入力できなければならないが, 子音だけしか入力できない場合, 候補を選択するのに時間がかかってしまう.

表 3.3 は Popie の辞書において, n 文字の子音の入力に対して, 先頭から m 文字の母音を確定した場合の, 候補数の平均値を示している. 候補数の平均は以下のように定義した.

$$\text{候補数の平均}\ (n,m) = \frac{\text{読みの長さが } n \text{ 文字の単語数}}{\text{入力可能な子音 } n \text{ 個・母音 } m \text{ 個の組合せのうち候補数が 1 以上の数}}$$

表 3.3 の m=0 の列は, 子音のみを入力した状態での候補数を示している. m=0 とき, 文字数 n=1 で 194 個の候補, n=2 で 129 個の候補と, 非常に候補数が多いことが分かる. しかし, m=1 の状態, つまり母音を 1 つ確定するだけで, 候補を相当数減らせる事が分かる. そこで, ユーザが必要に応じて, 子音に対応する母音を確定できる “母音選択” 機能を Popie に実装した. Popie ではフローメニューを拡張することで, “母音選択” の機能を提供する.

表 3.3: 子音 n 個のうち先頭から m 個に対応する母音を確定した時の候補数の平均

文字数 n	確定文字数 m						単語数
	0	1	2	3	4	5	
1	194.10	23.40					1941
2	129.42	20.05	4.99				12813
3	34.63	7.26	3.04	1.95			32069
4	8.06	3.10	2.19	1.60	1.46		49109
5	2.31	1.61	1.47	1.37	1.27	1.20	33969

3.2 Popie のインタフェース

Popie のインタフェースを図 3.1 に示す. この図は, “Popie”(ポパイ) と入力するために “HHA” と子音を入力したところである.

Popie のインタフェースは, ユーザが操作を行うフローメニューの部分と, 候補や入力文字の表示部分に分けられる. フローメニューは 8 つのオクタントと, 中心のレストエリアからなる.

8 つのオクタントのうち, 上, 右上, 右, 右下, および下の 5 つのオクタントでは, 主に子音入力, 母音選択の操作を行い, 改行や, アンドゥ・リドゥの操作を行う. 残りの 3 つのオクタントでは候補の選択操作を行う.

上部に表示されている “HHA” にはユーザが入力した文字を表示する. 左側のリストは, ユーザの入力に対してシステムが提示した候補を表示している. リストの横に表示されている円は, 候補の量を示しており, 候補数によって円の大きさが変化する.

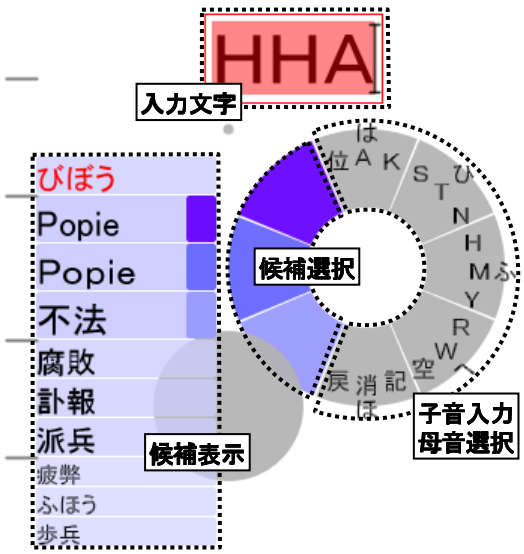


図 3.1: Popie のインタフェース

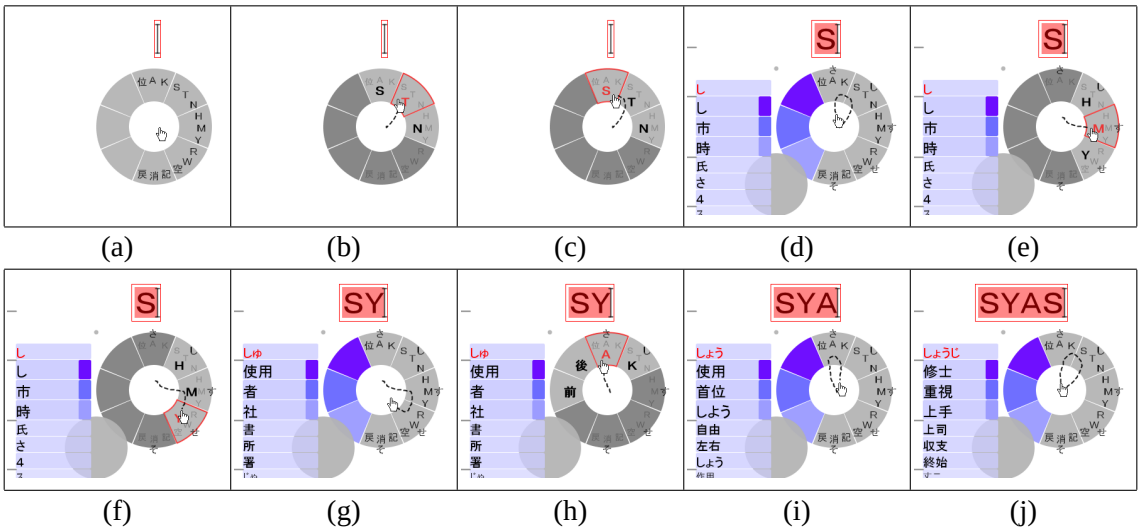


図 3.2: Popie において, 子音列 “SYAS” を入力する例

3.2.1 子音の入力

図 3.2 に Popie を使って子音 “SYAS” を入力する操作を示す. まず初期状態から (a), “STN” とラベルの付いた右上側のオクタントにペンを移動し, サブメニューを表示させる (b). 続けて, サブメニューの上側のオクタント ‘S’ の位置にペンを移動させ (c), レストエリアにペンを戻すことで ‘S’ を入力する (d). 続けて, “HMY” とラベルの付いた右側のオクタントにペンを移動し, サブメニューを表示させる (e). 続けて, サブメニューの右下側のオクタント ‘Y’ の位置にペンを移動させ (f), レストエリアにペンを戻す事で ‘Y’ を入力する (g). さらに, “位 AK” とラベルの付いた上側のオクタントにペンを移動させ (h), そのままレストエリアに戻すことで ‘A’ を入力し (i), 先ほどと同様に ‘S’ を入力する (j).

図中では線が途切れているが, ここで示した 4 つの子音を入力する操作は 1 つのストロークで実行されている. 各子音はそれぞれ対応するストロークによって入力される. ユーザは子音キーの位置を覚えることで, 素早い入力を行うことができる.

3.2.2 母音の選択

母音選択は, 母音が確定していない子音を先頭から順番に母音を選択する操作である. ここでは, 母音選択の機能を子音入力と両立させて実現するため, オクタントの外側の領域にペンを移動したときにも, アクションが起こるようにフローメニューを拡張した. その動作は以下の通り.

- 各オクタントとその外側をペンが往来する場合, オクタントから外側にペンが横切った場合だけアクションが実行される.
- 一旦オクタントの外側にペンを出すと, レストエリアに戻る際のアクションを実行しない.

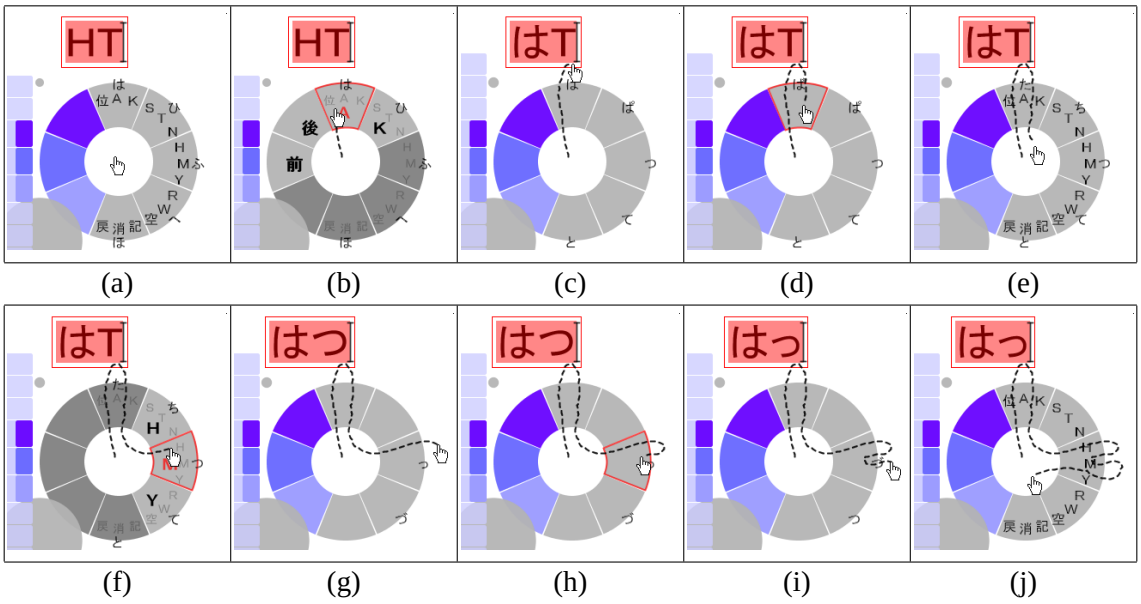


図 3.3: Popie において, 母音を選択する例

図 3.3 に子音列 “HT” の母音を選択する方法を示す. まず, “HT” が入力された状態で (a), ペンを上側のオクタントを通らせ (b), ‘は’ を横切るようにペンをオクタントの外側に出す (c). その後, ペンをレストエリアに戻すと ‘は’ が確定する (d, e). 次に ‘っ’ を確定するために, まず ‘つ’ を選択する (f, g). 次に ‘っ’ を選択するため, 一度オクタントまでペンを戻した後 (h), もう一度オクタントの外側にペンを移動させる (i). その後, ペンをレストエリアに戻すと ‘っ’ が確定する (j).

母音はオクタント上側から順にあいうえお順に配置されており, オクタントの外側の境界上に, 確定される仮名が表示されている.

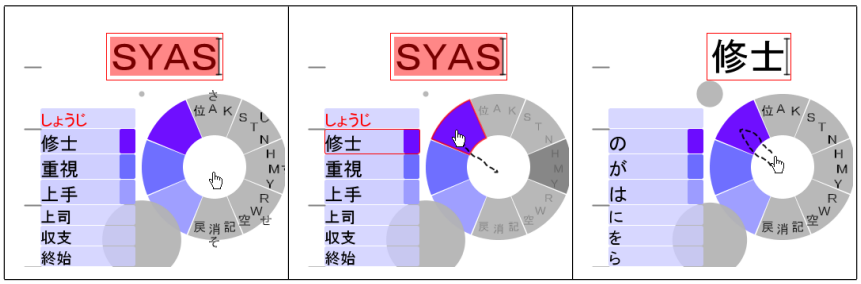
3.2.3 候補の表示と選択

ユーザの子音入力や母音選択に対して, 縦長のリストに候補が表示される. それぞれの候補は, 黒色と赤色のどちらか, もしくは両方の色の文字で表示される. 黒色の文字は確定, 赤色の文字は未確定のまま変換することを示している.

Popie では, 左側 3 つのオクタントを使って候補を選択する. よって, 上位 3 候補はスクロールすることなく選択が可能である. 上位 3 候補にない場合は, 母音選択により絞り込むか, スクロールすることで候補を選択する.

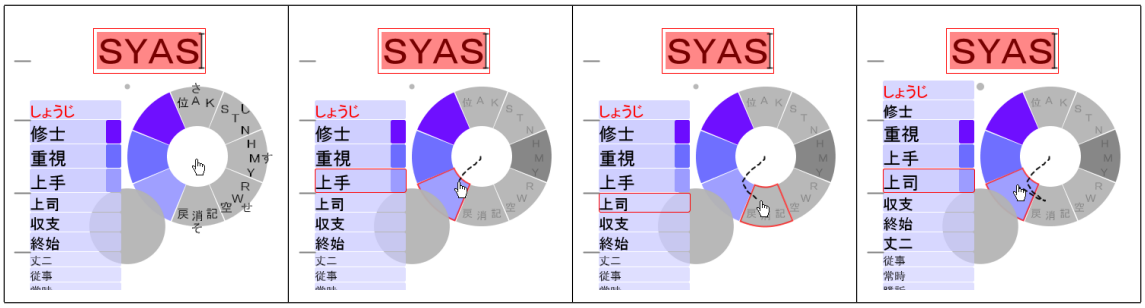
図 3.4 に “修士” を選択する例を示す. この例では “SYAS” がすでに入力された状態である (a). ユーザは “修士” に対応した左上側のオクタントにペンを移動し (b), レストエリアにペンを戻すことで “修士” を選択することができる (c).

図 3.5 に候補のリストをスクロールする例を示す. ペンを下側のオクタントから, 左下側の



(a) (b) (c)

図 3.4: Popie において, ‘修士’ を選択する例



(a) (b) (c) (d)

図 3.5: Popie において, 候補のリストをスクロールする例

オクタントに移動させると, 候補のリストをスクロールする. 初期状態から (a), 左側 3 つの候補選択のためのオクタントを通り (b), 下側のオクタントまでペンを移動する (c), そこから左下側のオクタントにペンを戻すとリストがスクロールされる (d). リストをスクロールすると, 候補選択のためのオクタントに対応する候補が変化し, 対応するオクタントからレストエリアにペンを戻すことで単語が確定される.

さらにスクロールする場合は, (c) と (d) の操作を繰り返し行うことで, 順にスクロールさせることができる. 逆にスクロールさせたい場合は, (c) と (d) の操作と同様の操作を, 上側のオクタントと, 左上側のオクタントで行う.

3.2.4 その他の操作

Popie ではこれ以外に, 文字入力に必要な機能を提供している.

スペース, タブ, 改行, バックスペース

専用のキーを用意し, 空白や制御文字を入力することができる.

キャレットの位置を前後に移動する

編集対象の文字列上でキャレットを自由に移動させることができる. 子音列の入力中に未確定の文字列の間にキャレットを移動しようとした場合には, 単語を明示的にキャレッ

トの位置で区切って変換を行う.

アンドゥ・リドゥ機能

入力中の全ての操作を記録し, いつでも前の状態に戻ったり, 戻った後にやり直すことができる. アンドゥをしたのちに別の操作を行った場合は, 新しい操作で上書きされる.

記号の入力補助

記号は “KKA” と子音を入力することで入力が可能だが, 記号用のキーを用意した. このキーが入力されると “きごう” という未変換の文字列が挿入される.

3.2.5 入力キーの配置

Popie の入力キーの配置を表 3.4 に示す. 縦軸が初期状態のメインメニュー項目の方向を示し, 横軸がそのメニューに対するサブメニューである.

メインメニューの位置からサブメニューの位置が離れてしまうと, そのメニューの選択に手間がかかるため, サブメニュー項目の位置は, メインメニュー項目の位置からあまり離れないように配置している. 逆に, あまり使わない, もしくは間違って選択してしまうと困るサブメニュー項目の位置はメインメニュー項目の位置から離して配置するべきである.

表 3.4: Popie の入力キーの配置

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="8">サブメニュー</th></tr>
<tr><th>上</th><th>右上</th><th>右</th><th>右下</th><th>下</th><th>左下</th><th>左</th><th>左上</th></tr>
<tr><td rowspan="8">メインメニュー</td><td>上</td><td>あ行</td><td>か行</td><td></td><td></td><td></td><td></td><td>キャレット前</td><td>キャレット後</td></tr>
<tr><td>右上</td><td>さ行</td><td>た行</td><td>な行</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>右</td><td></td><td>は行</td><td>ま行</td><td>や行</td><td></td><td></td><td></td><td></td></tr>
<tr><td>右下</td><td></td><td></td><td>ら行</td><td>わ行</td><td>空白</td><td>改行</td><td>タブ</td><td></td></tr>
<tr><td>下</td><td>終了</td><td></td><td>リドゥ</td><td>アンドゥ</td><td>消去</td><td>記号</td><td></td><td></td></tr>
<tr><td>左下</td><td rowspan="3">スクロール上</td><td></td><td></td><td></td><td rowspan="3">スクロール下</td><td colspan="3" rowspan="3">候補選択</td></tr>
<tr><td>左</td><td></td><td></td><td></td></tr>
<tr><td>左上</td><td></td><td></td><td></td></tr>
</table>

3.2.6 左利き用のインタフェース

本章で説明した Popie は右利きのユーザを想定している. 左利きの利用者には左右が反対になるインタフェースを用意した（図 3.6）. これはペン型デバイスを使用すると操作する画面をユーザ自身の腕で隠してしまうためである. 子音入力や母音選択の部分は覚えてしまえば隠れてもさほど問題がないが, 表示位置が変化する候補選択の部分は隠れてしまうと操作しづらいため, ユーザ自身の腕で隠れないように子音キーを左側に並べるような配置にした.

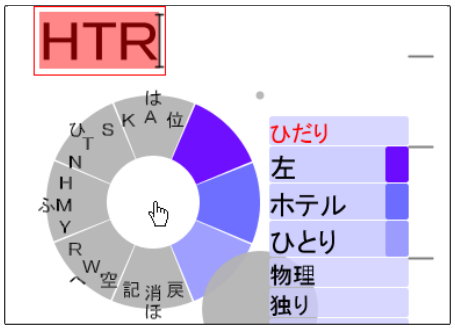


図 3.6: Popie の左利き用のインターフェース

3.3 Popie の実装

3.3.1 システム構成

システムの実装に使用した言語は Java で, Java™2 SDK, Standard Edition バージョン 1.4.2 を使用した. システム全体はプレゼンテーション機能を持ったアプリケーション PopiePoint として実装している. PopiePoint では, 図形の描画や文字列の表示, 画像ファイルの挿入などが行える. 詳しい使い方は, 付録 B PopiePoint のマニュアルを参照.

FlowMenu[2] はソースコードが公開されていないので, フローメニューの基本部分から実装し, Popie インタフェースはフローメニューの基本部分の派生として実現した.

子音入力エンジンは, 辞書データと候補を生成する部分からなる. 子音入力エンジンは Popie と切り離して実装してあるため, 異なるインタフェースを持つシステムからこのエンジンを容易に利用可能である.

プレゼンテーション機能を持つ PopiePoint のシステム全体ではプログラムはおよそ 14000 行である. Popie のインタフェースと子音入力エンジンのプログラムはおよそ 6000 行である.

3.3.2 Popie インタフェースの実装

Popie インタフェースはフローメニューの基本部分から派生して実装している. フローメニューの基本部分では, 8 等分されたドーナツ状のインタフェースを表示する機能と, ペンがインタフェース状のどの領域に含まれるかを判定し, 移動によってペンを含む領域が変化した際に, イベントを発行する機能を持っている.

フローメニューの各派生では, 基本的なインタフェースの表示を継承するか, 独自のインタフェース (例えば 4 等分の表示インタフェース) を定義することができる. また, イベントの発行に対してどのような操作を実行するかは各派生において定義する. 具体的な操作としては, “オブジェクトの色を変化させる”, “別のメニューに遷移する” などである.

Popie インタフェースでは, フローメニューの基本部分の 8 等分されたドーナツ状のインタフェースに加え, 候補のリスト表示を行っている. 候補のリストは, 後述の子音入力エンジンから取得し, フローメニューの中心に近いほど大きく, 遠いほど小さい文字で表示している. ま

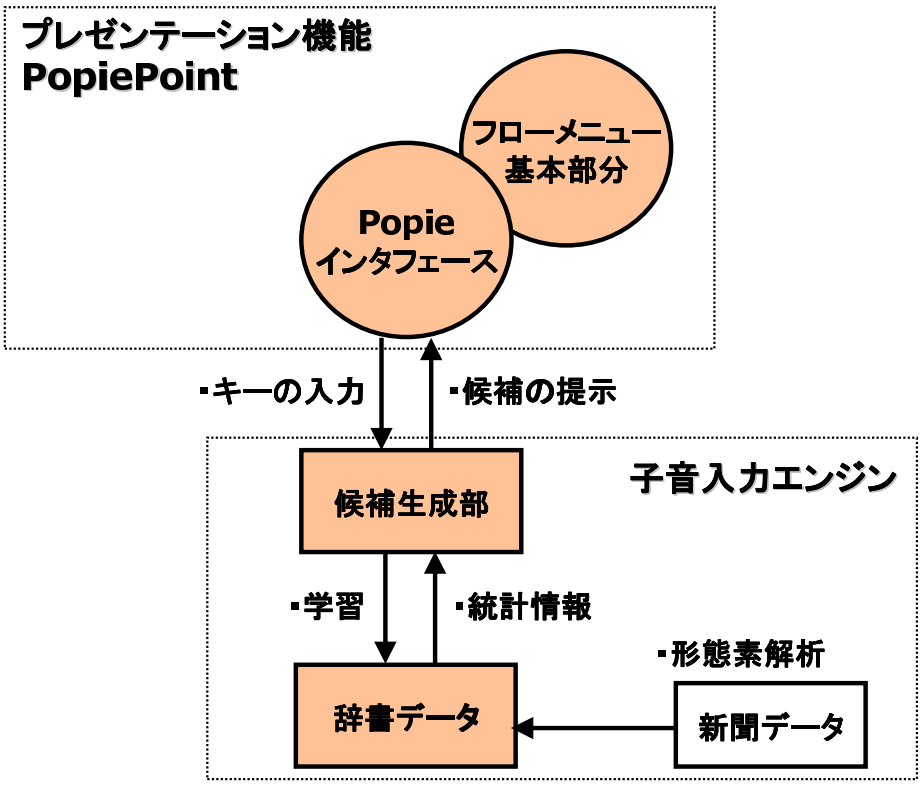


図 3.7: Popie のシステム構成

た, 入力された文字列は, PopiePoint の文字列オブジェクトとして表示されている.

入力の状態遷移

図 3.8 に, Popie の入力の状態遷移図を示す. Popie では, ペンが含まれる領域によって状態が変わるため, 状態とインタフェースの位置の関係が深い. 状態の遷移に入力が起こる場合には, 線の横に入力の種類が示され, それがない場合には状態の遷移に伴って何も入力がされないことを意味する.

レストエリアから, 子音入力のオクタント, もしくは候補選択のオクタントに移動し, レストエリアに戻ると, それぞれの入力が発生する. 子音入力のオクタントから外側に移動すると, その時点で母音選択が発生し, 外側, オクタントと行き来することで, 同じ子音の濁点, 半濁点などの選択が順次行える. レストエリアに戻るときは, 別の状態を経由してレストエリアに戻るため, 入力は発生しない. またこのとき, 1 文字分の子音に対して母音選択が終了したことが子音入力エンジンに通知される.

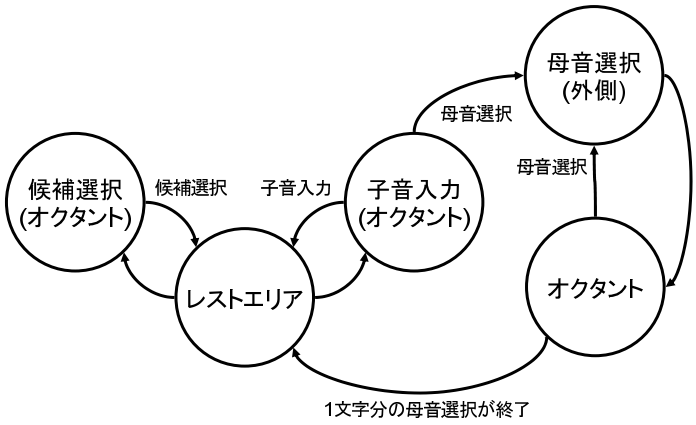


図 3.8: 入力の状態遷移図

3.3.3 子音入力エンジンの実装

子音入力エンジンは, 候補生成部がキーの入力に対して, 辞書データに基づいて候補を生成するという機能と, 一連の入力を学習し, 候補生成に反映する機能を持つ. 以下に, 辞書データと候補生成部に関する説明をする. また付録 A に, 子音入力エンジンを利用するために必要な仕様に関する情報を載せた.

辞書データ

辞書は単語の出現数から成る単語辞書と, 2 つの単語が続けて出現した回数から成る単語間関係辞書の 2 つの静的辞書と, ユーザからの入力を記録するユーザ辞書の 3 つで構成される. ユーザ辞書は, 単語単体と単語間関係のデータを両方持っている. それぞれの辞書のデータ形式は表 3.5 の通りである.

辞書データは CD-毎日新聞 2001 年度版 [31] のテキストデータを, 形態素解析システム茶筌 [32] を用いて単語ごとに切り分け, それぞれの単語の出現数と, 2 つの単語が続けて出現する回数を計測し作成した. テキストデータは約 3600 万単語からなり, 異なり語数が約 28 万単語であったが, そのうち, 1 回しか出現しない単語を辞書から削除し, 約 11 万単語を使って単語数をスコアとする単語辞書を作成した. しかし, 新聞のテキストだけでは語彙が少ないため, SKK[33] の辞書を用い, 辞書データにない単語を補った. 補った単語のスコアは 1 とした. また単語間関係辞書も同様に出現回数が 50 回より少ないものを削除し, 約 10 万組のデータを作成した. この場合もスコアは出現回数としている.

ユーザ辞書の単語は使用回数が多いほどスコアが高くなるようにし, 使われないとスコアが少しずつ下がるようにしている. スコアが 0 になった単語は, 静的辞書にエントリーがある場合削除される.

表 3.5: 辞書データの形式

辞書	データの形式
単語辞書	スコア, 読み, 単語
単語間関係辞書	スコア, 直前の単語, 単語, 読み
ユーザ辞書	スコア, 読み, 単語, 直前の単語

表 3.6: 検索方法と検索辞書に対するスコア

検索する辞書	単語辞書	単語間関係辞書	ユーザ辞書
子音の長さと候補の読みの長さが一致	20000	10000	20000
子音の長さより候補の読みの長さが長い	5000	20000	30000

候補生成部

候補生成部は子音や母音のキーが入力されるたびに候補を生成する. 辞書データから候補の検索をする際, 入力された子音の数と読みの長さが同じ単語 (例えば “TNK” という入力に対して “田中”) と, 入力された子音の数よりも読みの長さが長い単語 (例えば “TNK” という入力に対して “トナカイ”) に対して別々の重みを付ける. また, この検索を 3 つの辞書ごとに行うので, $2 \times 3 = 6$ つの重みパラメータを持つ.

検索された候補は辞書が持つスコアと, 重みをもとにソートされて提供される. これまでの実装では, 実際にシステムを使った経験から重みパラメータを設定している. 子音の長さと候補の読みの長さが一致する候補に対するパラメータを高くすると, ユーザが入力した子音の長さに合うような候補が上位に表示される. 逆に, 子音の長さより候補の読みの長さが長い候補に対するパラメータを高くすると, ユーザの意図しない候補が多く表示される可能性があるが, 少ない入力でより長い単語を入力できるようになる. 本実装では、単語辞書に関して, 読みの長さに一致する単語を優先的に上位の候補となるように, 単語間関連辞書とユーザ辞書に関して、より長い単語を優先的に上位の候補となるようなパラメータの設定にした (表 3.6)。

これは、ユーザ辞書が、ユーザからの入力の学習によって作られた辞書であるため、ユーザの良く入力する単語を多く含む可能性があり、また、単語間関連辞書は、ある単語とある単語の結びつく頻度を収めた辞書であるため、ユーザの入力よりも読みの長い単語であっても、ユーザが意図した単語である可能性が高いからである。逆に単語辞書からは候補の読みと長さを一致する検索を優先させ、より確実に変換を行えるようにした。

第4章　Popieの評価

本章では, 日本語入力システム Popie の評価について述べる. はじめに評価の概要について説明し, 各評価の方法と結果を示す.

4.1　評価の概要

子音による日本語入力システム Popie は, 子音による入力手法を用い, フローメニューによって, 日本語入力を行うシステムである. したがってシステムのを評価するためには, 複数の要素を考慮し, 複数の観点から評価を行う必要があると考えられる.

そこで本研究では, 大きく 3 つの観点から評価を行った

- Popie の子音入力方式そのものに関する評価
- Popie の日本語入力システム全体としての評価
- Popie のインタフェースに関する評価

子音入力方式に関する評価としては, まずローマ字入力方式との比較により, 子音入力方式の有効性を調べた. また, 母音選択と予測・補完の機能の子音入力方式への効果についても検証を行った. さらにこれまで経験的に設定されていた候補生成に用いるパラメータの, 適切な値について検討を行った.

日本語入力システム全体としての評価としては, ユーザによる入力実験を行い, 入力速度に関する評価を行った.

インタフェースに関する評価としては, フローメニューの拡張をした候補選択の部分の評価, および子音キーの配置に関する評価を行った.

4.2　子音入力方式の評価

これまでにも , 子音入力方式の評価を行った研究がある [28]. しかし, この研究では子音入力方式の評価対象として携帯電話の入力方式[1] を取り上げており, 一般的に使われているローマ字入力方式との比較はされていなかった.

[1]“さとう” という入力に対して携帯電話のボタンを “3 4444 111” のように, 子音に対応するボタンを母音に対応する回数だけ押して入力する方式.

そこで我々は, 子音入力方式とローマ字入力方式の比較を行った. 考えられる比較の方法としては, ユーザが実際に入力を行う方法と, 計算機により入力をシミュレーションするという方法がある. ここでは, 経験や学習によって影響されず, より入力方式を定量的に評価することができる計算機による方法を選択した.

また, Popie の子音入力方式では, 候補が多数存在する場合に適宜母音を選択する機能や, 入力途中で入力を補完したり, 次の単語を予測するといった機能を持っている. 1) 子音だけを入力する方式, 2) 母音選択の機能を追加した方式, 3) 予測・補完の機能を追加した方式, についても入力をシミュレーションすることで, それぞれの機能にどの程度効果があるかを検証した.

4.2.1 自動入力プログラム

計算機によって子音入力方式とローマ字入力方式での入力をシミュレーションするため, 各方式に対応した自動入力プログラムを作成した. 入力するテキストデータは, CD-毎日新聞 2001 年度版 [31] の社会面の記事を用いた.

子音入力方式とローマ字入力方式では同じ Popie の変換エンジンを用いた. この変換エンジンは入力された単語から次の単語を予測したり, 入力途中で補完する機能を持っている. 基本的には子音入力方式のためのエンジンであるが, ローマ字によるの入力を, 変換エンジンの内部で子音を入力し直後に母音を選択するという操作に変換することで, ローマ字入力方式を実現している. ただし, 子音入力方式では促音や拗音などの文字も一つの子音を対応させて入力するため, 子音を 2 つ重ねることで促音や拗音を表現するローマ字入力方式とキーの入力数などが異なるが, ローマ字として入力とした場合のキー入力数に補正して記録している.

以下に M 文字の読みからなる単語を入力する場合の, 子音入力方式とローマ字入力方式の自動入力プログラムの処理の流れを示す.

自動入力プログラムの処理の流れ

子音入力方式

1. $if(c = 0 \wedge v = 0)then$ Find(N) を実行
2. $if(c < M \wedge v = 0)then$ 子音を 1 文字入力し Find(N) を実行する
3. $if(c = M \wedge v < M)then$ 母音を 1 文字選択し Find(N) を実行する
4. $if(c = M \wedge v = M)then$ 全ての候補から目的の単語を探す (スクロール)

ローマ字入力方式

1. $if(c = 0 \wedge v = 0)then$ Find(N) を実行
2. $if(c < M \wedge v < M)then$ 子音と母音を 1 文字入力し Find(N) を実行する
   （ただし, 促音や拗音が続く場合は同時に入力)
3. $if(c = M \wedge v = M)then$ 全ての候補から目的の単語を探す (スクロール)

ただし,

c : 子音の入力数
v : 母音の選択個数
Find(N) : 候補の上位 N 個から目的の単語を探し, ヒットした場合には入力して終了し, ヒットしない場合なにもしないという動作と定義する.

それぞれの手順に従って自動入力を行い, 候補から目的の単語を見つけられなかった場合は, その単語は計測の対象外とした.

Find(N) 関数の引数 N はプログラムの実行時に与えるものとし, Popie の現段階の実装では上位 3 つの候補をスクロールせずに選択できるため, 標準では N=3 としている. また, 子音入力方式について, 母音選択や補完・予測の機能のオン, オフもプログラムの実行時に与えられるようにした.

自動入力プログラムは, キーの入力回数等の情報を記録した. 記録する情報の詳細を表 4.1 に示す.

表 4.1: 自動入力プログラムが記録する情報

項目	説明
子音入力回数	入力に使われた回数. 子音入力方式では AKSTNHMYRW の 10 種類. ローマ字入力方式では KSTNHMYRWGZDBPVQFX の 18 種類.
母音入力回数	入力に使われた回数. 子音入力方式では AIUEO の母音選択の 5 種類. ローマ字入力方式も AIUEO の 5 種類
候補選択回数	候補を選択した回数. スクロールした場合としていない場合に分けた.
スクロール操作数	スクロール操作の回数. 操作 1 回につき候補は N 個ずつスクロールする. N=3 のとき 1〜3 位は 0 回, 4〜6 位が 1 回, 7〜9 位が 2 回となる.
選択時のキー入力数	各単語に対して候補を選択するまでにいくつのキーが入力されていたか.

4.2.2 シミュレーション 1. 子音入力方式とローマ字入力方式

子音入力方式とローマ字入力方式それぞれのシミュレーションに, 社会面の記事約 3ヶ月分 (約 145 万単語) を入力として用い, 候補選択は候補上位 3 つから行った (N=3).

表 4.2 にそれぞれの方式について, 1 単語あたりの子音・母音入力回数, 候補選択回数, スクロール操作数, および操作数の合計を示す. 子音入力方式では, ローマ字入力方式に比べて, 子音入力回数が約 24%多いが, 母音入力回数が約 75%も削減され, 全体として約 18%の操作数を減らす事ができている.

表 4.2: 社会面の記事入力における単語あたりの操作数

項目	子音入力方式	ローマ字入力方式
子音入力回数	2.054	1.650
母音入力回数	0.376	1.547
候補選択回数 (スクロール無)	0.926	0.933
候補選択回数 (スクロール有)	0.074	0.067
スクロール操作数	0.341	0.370
操作数合計	3.772	4.567

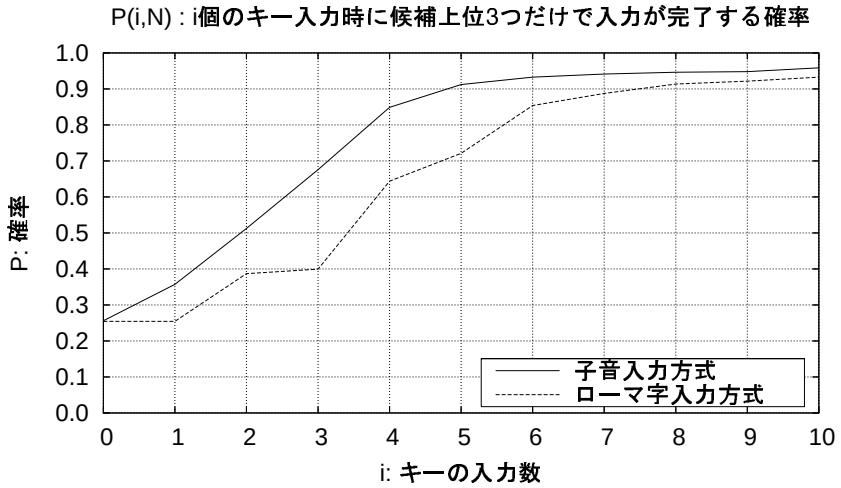


図 4.1: 入力の数に対して, 入力が完了する確率

図 4.1 にそれぞれの方式について, キーの入力数に対して, 候補の上位 3 つで入力が完了した確率を示す. 予測機能があるため, 入力がない状態でも約 4 分の 1 の確率で入力が完了し, キーの入力数が増えるにしたがって完了する確率が高くなっている. 候補の上位 3 つでは入力が完了できない場合はスクロールする必要がある. この図から, 子音入力方式の方が, より少ないキーの入力数で入力が完了していることが分かる.

4.2.3 シミュレーション 2. 子音入力方式の母音選択と予測・補完機能

Popie の子音入力方式において, 予測・補完と, 母音選択の 2 つの機能を有りと無しとした 4 通りをシミュレーションした. 入力には社会面の記事約 3ヶ月分 (約 145 万単語) を用い, 候補選択は候補上位 3 つから行った (N=3).

表 4.3 にそれぞれの機能が有り, 無しに対応する, 1 単語あたりの子音・母音入力回数, 候補選択回数, スクロール操作数, および操作数の合計を示した.

両機能とも有りの方式 (Popie) では, 両機能とも無しの方式に比べ, 操作数が約 25%削減さ

れている. また, 予測・補完の機能のみ有りの方式や, 母音選択の機能のみ有りの方式に比べてもそれぞれ, 操作数が約 27%, 約 11%削減されている.

しかし, 予測・補完の機能のみ有りの方式では, 両方無しの方式に比べて操作数が増えている. これは, 予測・補完の機能により候補の数が増大し, スクロール操作が余計に増えてしまったことが原因であると考えられる. 一方母音選択は, 濁点等を含め 5 つ以上ある母音の可能性を 1 つに絞り込むことができるので, スクロール操作よりも有効な手段であることが分かる. また, 母音選択と予測・補完の機能の組合せの相性が良く, 相乗効果があったことが分かった.

表 4.3: 社会面の記事入力における単語あたりの操作数

項目	予測補完有 母音選択有	予測補完無 母音選択有	予測補完有 母音選択無	予測補完無 母音選択無
子音入力回数	2.054	2.603	2.005	2.534
母音入力回数	0.376	0.378	0.000	0.000
候補選択回数 (スクロール無)	0.926	0.931	0.738	0.752
候補選択回数 (スクロール有)	0.074	0.069	0.262	0.248
スクロール操作数	0.341	0.266	2.154	1.519
操作数合計	3.772	4.247	5.159	5.054

4.2.4 シミュレーション 3. 候補生成パラメータ

Popie では, 3 つの辞書データから候補の検索をする際, 入力された子音の数と読みの長さが一致する単語 (“一致”) か, 入力された子音の数よりも読みの長さが長い単語 (“長い”) かで, それぞれ係数が設定されているが, パラメータには経験的な値を用いていた.

そこで, パラメータの設定に関して定量的な指標を得るため, 複数のパラメータによってシュミレーションを行った. パラメータは 10000 から 10000 刻みで 50000 までの 5 つの自由度で, 6 パラメータ, $5^6 = 15625$ 通りをシミュレーションした. 組合せが膨大なため, 入力には約 2000 単語だけを用いた.

各シミュレーション結果の評価値には, 操作数を基本とする関数を用いた. この関数は, より子音だけで入力できる単語が増え, スクロール操作が少なくなるようなパラメータの組み合わせに対して, 低い評価値を与えるように, 子音入力数に 1, 母音入力数に 2〜4 と, スクロール操作数に 10 の重みを付けた.

表 4.4 に評価値の低いパラメータの組合せ上位 5 組を示す. 上位 5 組のパラメータは, 単語辞書の “一致”, “長い” が 50000, 10000 で一定である. ユーザ辞書に関しては, “一致” の方が高いパラメータになる傾向があった. 単語間関係辞書については, この表からは傾向が分からない.

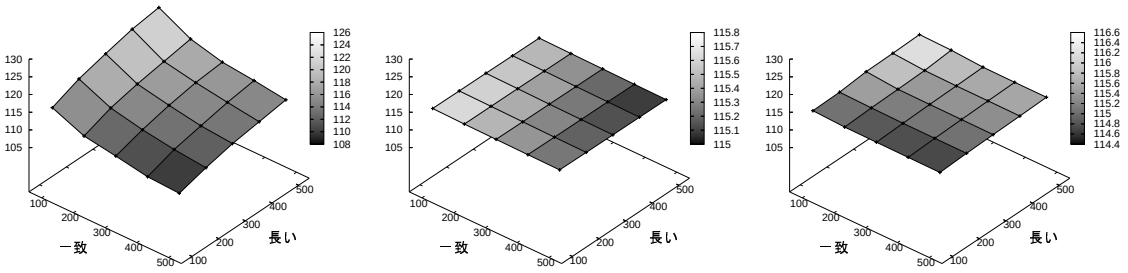
図 4.2 に各辞書の “一致” と “長い” のパラメータに対して, 他の 4 つのパラメータ 625 通りの評価値を平均した値をプロットしたグラフを示す. このグラフでは, 色が濃い方が評価値が低いことを示している. 単語辞書については, “一致” のパラメータが高く “長い” のパラメータが低いほど評価値が良くなる傾向が見られる. 単語間関係辞書については, “一致”, “長い” の

パラメータが共に高いほど評価値が良くなる傾向が見られる. ユーザ辞書については, “一致” のパラメータが高く “長い” のパラメータが低いほど評価値が良くなる傾向が見られる. しかし, 単語間関係辞書と, ユーザ辞書については変化が小さい.

以上のシュミレーションによる評価値の良いパラメータの設定をまとめると, 表 4.5 のようになる. 単語辞書, ユーザ辞書に関しては, 読みの長さに一致する単語に対してパラメータ値を上げ, 単語間関係辞書に対しては, 両方とも高くすると良い. ただし, パラメータどの程度高くするか, 低くするかについては更なる検討が必要である.

表 4.4: より適切なパラメータの組合せ上位 5 組

評価値	単語辞書		単語間関係辞書		ユーザ辞書	
	一致	長い	一致	長い	一致	長い
10777	50000	10000	40000	50000	50000	10000
10777	50000	10000	50000	50000	50000	10000
10779	50000	10000	40000	50000	40000	10000
10779	50000	10000	50000	50000	40000	10000
10781	50000	10000	20000	50000	50000	10000



単語辞書　　単語間関係辞書　　ユーザ辞書

図 4.2: 各辞書について “一致”, “長い” のパラメータに対する評価値 (単位 100)

表 4.5: パラメータ設定の指標

単語辞書		単語間関係辞書		ユーザ辞書	
一致	長い	一致	長い	一致	長い
高	低	高	高	高	低

4.3 日本語入力システム全体としての評価

日本語入力システム全体としての評価をするため, ユーザによる Popie の入力実験を行った.

4.3.1 実験方法

実験にはタブレット PC, (LaVieTB TB700/5T, WindowsXP, Mobile Pentium3 933MHz, 512MB RAM) を使用した. 被験者は 18〜21 才の男子学生 6 名である. 文章の入力は 1 セッションを 15 分とし, 1 セッションの練習と 7 セッションの本番で計 8 セッション, 合計 2 時間行った. セッション中は文章の入力だけに専念してもらい, 入力は 1 日に 1 セッションないし 2 セッションとし, セッション間の休憩は十分に取った. 文章は辞書作成に使用したものとは別の新聞記事, 童話, 小説から抽出し, 1 文を 15 字〜45 字程度とした. ユーザの入力による学習機能は有効にして行ったが, 文章は各入力毎に必ず違うものを用意し, 同じ文章を入力することがないようにした.

4.3.2 実験結果

結果を図 4.3 に示す. 図 4.3 のグラフは Popie による入力速度を示している. 横軸にセッション, 縦軸に 1 分あたりの入力文字数（cpm）をプロットした. グラフには 1 セッション目の練習における入力速度は含めていない. 入力速度は 2 セッション目では 9cpm から 14cpm に分布し, 平均も 11.9cpm となっているが, 8 セッション目では 19cpm から 29cpm に分布しており, 平均も 24.7cpm まで上昇している. 2 時間のみの使用で, 十分な学習効果が得られていることが分かる.

予備実験として行った WindowsXP TabletPC Edition 付属の手書き文字認識による入力速度が 18cpm だったことから, Popie の入力速度は実用上, 十分であると考えられる.

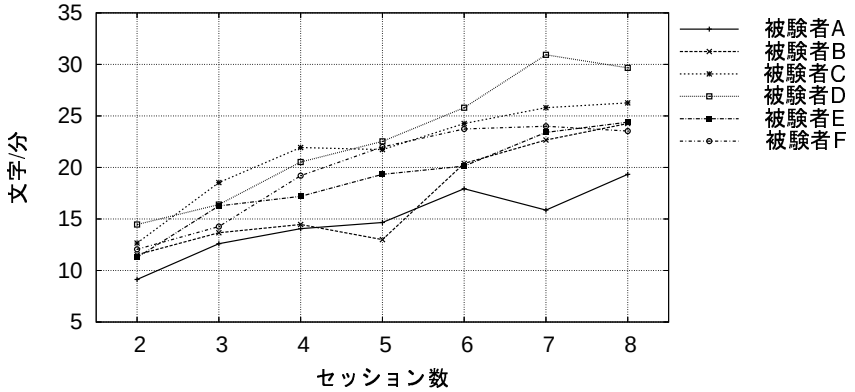
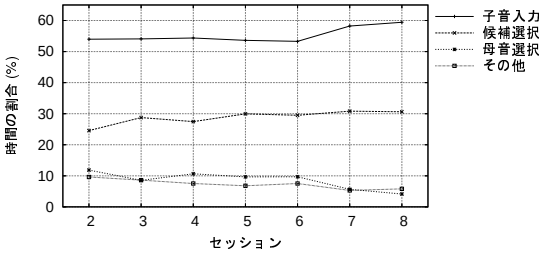
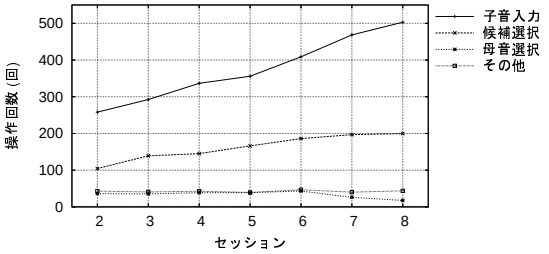


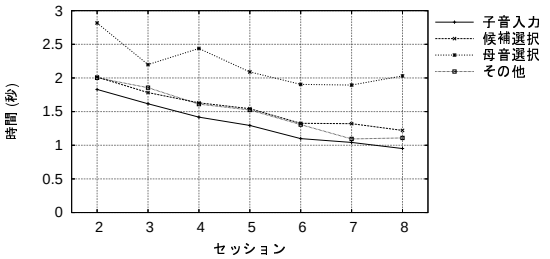
図 4.3: 入力速度の推移



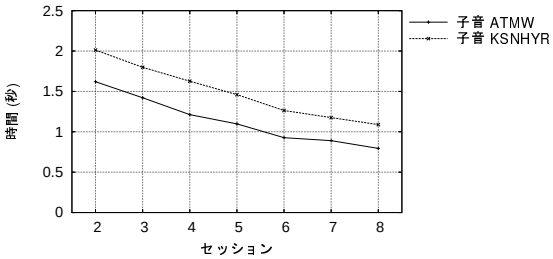
a) 各操作が入力時間全体に占める割合



b) セッション中の各操作の操作回数



c) 各操作に要した時間の平均



d) 子音入力に要した時間の平均

図 4.4: 入力時間と入力回数に関する詳細のグラフ

4.3.3 入力の詳細

実験のログから, 子音入力, 候補選択, 母音選択, その他 (バックスペースやアンドゥなど) の操作時間や操作回数を算出し, 入力についての詳細を調べた. 1 セッション目は練習を兼ねるためデータからは除外した.

図 4.4 a は, 第 2〜8 セッションまでの, 各操作の合計時間が, 入力時間全体に占める割合を示している. 第 8 セッションでは, 子音入力 59.4%, 候補選択 30.6%, 母音選択 4.2%, その他 5.8%である. システムに慣れるに従って, 子音入力と候補選択の割合が若干増え, 母音選択とその他の割合が若干減る傾向が見られた.

図 4.4 b は, セッション中の各操作の操作回数を示している. 慣れるに従い, 子音入力, 候補選択の回数が増加している. 一方, 母音選択の回数は増加すると予想されたが若干減少し, その他の回数は相対的に減っており, エラー率が下がったことを示している.

図 4.4 c は, 各操作を 1 回行うのに要した時間の平均値を示している. また図 4.4 d は, 子音入力を 1 回行うのに要した時間を, キーの位置で分類して平均値を算出したものである. キーの位置の分類は, ストロークが直線になる “ATMW” と, ストロークが三角形になる “KSNHYR” の 2 つである. “KSNHYR” では, “ATMW” に比べ軌跡が長くなるので, 操作にかかる時間も長くなっている.

母音選択はストロークは直線になるが, 他の操作に比べ, 入力に時間がかかっている. これは, オクタントの外側までペンを移動することによる, ペンの移動距離からくる時間の増加と, 子音入力から母音選択への切り替えにかかる時間を含むと考えられる.

どの操作も慣れるに従って, 操作時間が短くなっている. 候補選択では, 上位 3 候補を選択する場合, ストロークが直線になるが, 同じストロークの “ATMW” よりも, 0.4 秒ほど時間がかかっている. しかし, 候補を認識する時間や, スクロールの時間を含む割には, それほど遅くなっていないと言える.

一方で, 母音選択には子音入力の倍近い時間が掛かってしまっている. この原因として考えられるのは, 濁点や半濁点の選択には, 余計に時間がかかるため, 平均して遅くなったのではないかということと, 子音を入力した後, 候補を見て目的の単語がないことを確認した上で母音選択に移るため, 候補の認識に時間がかかっていることの 2 点である.

4.4 インタフェースの評価 1. 候補選択インタフェース

4.4.1 候補選択インタフェースの問題点

子音のみの入力では, 組み合わせによって爆発的に候補の数が増えるため, システムによる候補のソートが有効でなかった場合, 選択操作の負担が大きくなる. 我々は, 子音列の先頭から母音を決定できるインタフェースを導入し, ユーザが適宜候補を絞り込むことで, 選択操作の負担を軽減させてきた. しかし, 同音多義語の多い日本語では, 母音を確定しても候補数を減らせない場合も多く, さらに全体の入力時間の 3 割を占めるため, 候補選択のインタフェースは重要な役割を持つ.

しかし, 従来の候補選択のインタフェースは, 以下の 3 点の問題があった.

1. 候補の上位 3 つに候補がない場合スクロールが必要になるが, 候補のリストを 1 つずつしかスクロールできないため, 目的の単語が候補の下位になるほど, 選択にかかる時間がより多くなってしまう点

2. 候補がリストの下部に表示されていても, 中央の 3 つのオクタントに対応する位置まで, 間接的な操作によるスクロールを続けなければならない点

4.4.2 候補選択インタフェースの比較

我々は, 前述の候補選択インタフェースの問題を解決するため, いくつかのインタフェースの比較実験を行った [34]. 比較するインタフェースを設計するため, 以下の 3 つの方法についての検討を行い, 従来手法を含めた 4 つのインタフェースを用いて実験を行った.

スクロール量を変える

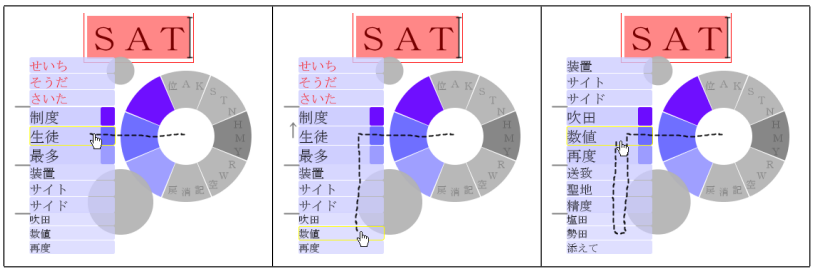
1 つずつスクロールさせるのではなく, 一度に複数個スクロールさせる方法を考える. 候補選択に使用できるオクタントは 3 つであり, 1 度に 3 つより多くスクロールさせると, 選択できない候補ができてしまうので, 従来手法で最大の 3 つずつスクロールする方法を最初の比較対象とした (Scroll 3).

ダイレクトにスクロール

従来は, フローメニューの枠組みで間接的にリストをスクロールしていたが, リストの上でペンを動かすことでリストをダイレクトにスクロールする方法を考える. リストの中央より下側の部分では上方向にペンを動かしたときだけスクロールし, リストの中央より上側の部分では下方向にペンを動かしたときだけスクロールする (Scroll Direct). 図 4.5 に, 「数値」をリストの中央までスクロールする例を示す. まず, リストの上にペンを移動し (a), リスト中の「数値」の部分までペンを移動する (b). このときは何も変化はない. 次に, リストの中央までペンを戻すと (c), 「数値」が中央までスクロールされる. スクロール後の選択方法は従来と同じように, 対応するオクタントからレストエリアにペンを移動することで行う.

ダイレクトに選択

中央 3 つのオクタントに対応する位置までスクロールさせて, レストエリアにペンを戻すという動作をしなくても, 目的の候補の上でペンを離すことで選択が行えるようにする方法を考える (Select Direct). スクロールは, リストの上端, 下端を横切ることで, それぞれ 3 つずつスクロールするようにした. この場合は一度に 12 個分までスクロールすることが可能であるが, 従来の方法の最大値に合わせた. この方法で, 図 4.5 において「数値」を選択する場合は, 最初に, リストの上にペンを移動し (a), リスト中の「数値」の部分までペンを移動する (b). そして, 決定を実行するためにペンを画面から離す.



(a) (b) (d)

図 4.5: “数値” を中央までスクロールする例

表 4.6 に以上 4 つのインタフェースの特徴を示す.

表 4.6: 比較するインタフェースの特徴

識別子	スクロールの仕方	選択方法
Scroll 1	メニュー上で間接的に 1 つずつスクロール	メニューの中心に戻って選択
Scroll 3	メニュー上で間接的に 3 つずつスクロール	メニューの中心に戻って選択
Scroll Direct	リストをなでるようにスクロール	メニューの中心に戻って選択
Select Direct	リストの上端, 下端で 3 つずつスクロール	候補の上でペンを離して選択

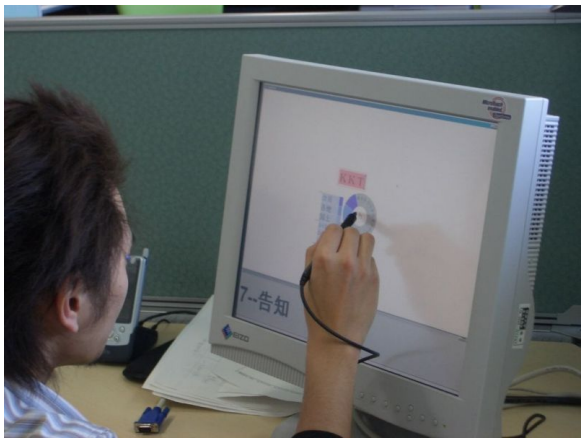


図 4.6: 実験の様子, タッチパネル付液晶ディスプレイ

4.4.3 実験方法

これらのインタフェースを用い, 候補リストから指定された単語を選択する実験を行った. 被験者は男性 7 名, 女性 1 名の計 8 名, 年齢は 20〜32 才であった. 被験者の中で Popie を使用したことがある人は 4 名いた. 被験者には, 10 分程度のインタフェースの説明と練習の後, 各インタフェースで単語を 30 回ずつ選択してもらい, 最後にアンケートに答えてもらった. 実験に使うインタフェースの順番は, 順序による影響を考慮し 8 人とも異なるようにした.

各試行では, 被験者が開始の意思を示した時点で, 選択すべき単語と候補のリストを表示させた. そして, 候補の表示から候補を選択し終えるまでの時間を計測した. ただし, DirectSelect では, ペンを離して候補を選択した後に, フローメニューの中心までペンを戻すまでを計測の対象とした. これは, 実際に文章を入力する際は, 選択に続いて次の子音の入力を行うからである.

選択すべき単語の候補リスト中での順位は, 4〜9 位, 10〜15 位, 25〜34 位, 45〜54 位, 75〜84 位の 5 つのグループから, それぞれのグループで同じ回数選ばれるようにランダムに選択した. 1〜3 位の候補は対応するオクタントを用いてすぐに選択できるので, 今回の実験では対象外とした. 4〜9 位は候補表示の最初の段階で, スクロールせずに視認できる範囲の候補群であり, 10〜15 位は視認するのに若干スクロールが必要な候補群である. 残りは, 30 位, 50 位, 80 位前後の候補群である.

実験にはタッチパネル付液晶ディスプレイ EIZO FlexScapL661P を使用した (図 4.6 参照). 使用した計算機は, WindowsXP Professional, Pentium4 3.0GHz, 1.5GBRAM である.

4.4.4 実験結果

図 4.7 は, 候補の順位グループごとに選択するのに要した時間の平均値をグラフにしたものである. 平均値を計算するにあたり, 間違った候補を選択したり, 選択すべき候補を見逃したた

めに著しく時間がかかったと思われるデータは, ミスをしたと判断してデータから除外した. ここで, 著しく時間がかかったと判断するのに用いた基準は, 間違った候補を選択した場合のデータを除いたものの平均値に, その標準偏差の 3 倍を加えた時間より遅かったものとした. また, 図 4.8 は, ミス率を示したグラフである. ミス率は間違った候補を選択したものと, ミスと判断したものを合わせた割合として計算した.

また, 実験後に行ったアンケートの結果を表 4.7 に示す.

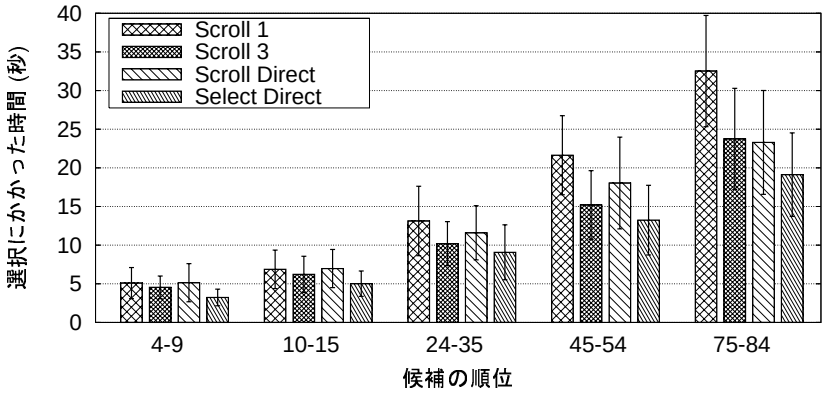


図 4.7: 選択に要した平均時間

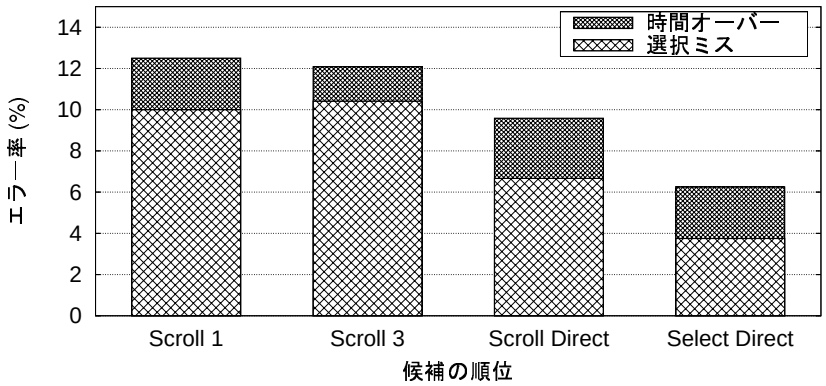


図 4.8: 各インタフェースでのミス率

4.4.5 実験の考察

実験から, Select Direct が最も選択に要した時間が他のインタフェースに比べて短いことが分かる. またエラー率も最も少ない. アンケートにおいても, 使いやすい, 疲れないと答えたのが 8 人中 5 人であり, 10 点満点の主観評価でも平均 8.5 点と高い値を示した. この結果は, 従来

表 4.7: 実験後のアンケートの結果

	Indirect1	Indirect3	DirectScroll	DirectSelect
最も使いやすいと思ったインタフェース			3 人	5 人
最も使いにくと思ったインタフェース	4 人	1 人	3 人	
最も疲れなかったインタフェース	1 人		2 人	5 人
最も疲れたインタフェース	5 人	3 人		
各インタフェースの評価平均 (10 点満点)	3.38	5.50	6.63	8.50

のフローメニューのパラダイムによる候補選択インタフェースよりも, ダイレクト操作を用いた候補選択インタフェースの方が効果的な候補選択ができることを示している.

Scroll 3 と Scroll Direct では若干 Scroll 3 が良い結果になっており, 30 位前後と, 50 位前後では優位差が認められているが, アンケートの評価やエラー率の点から言うと, Scroll Direct の方が良い結果となっている.

Scroll 1 は候補の順位が低くなるほどに, 他のインタフェースに比べて明らかに遅く, エラー率も高くなっている事が分かる. アンケートでも, 「使いにくい」, 「疲れる」と半分以上の人が回答しており, 主観評価も平均 3.38 点と最も低くなっている. これが, 従来の Popie の候補インタフェースが使いづらいとされる原因である.

その他, 3 つずつスクロールする際に, 一度にいっぺんに動いてしまっているので, アニメーションで中間を補完すれば, 表示されている候補を把握しやすく, 使いやすいのではないかという意見や, Scroll Direct や Scroll 1 の方が目が疲れないという意見があった. このことから, スクロールさせる際のアニメーション等の表示に関しても, 操作のしやすさに影響を与えていると考えられる.

4.5 インタフェースの評価 2. 子音キーの配置

Popie のインタフェースにおいて, 子音キーの配置は “AKSTNHMYRW” である. この配置は仮名順であり, ユーザにとって覚えやすいと言えるが, 入力速度の観点から見ると最適な配置であるとは言えない. 本節では, 現在の子音キーの配置 “AKSTNHMYRW” を評価するため, 子音キーの最適な配置を計算した [35].

4.5.1 配置の最適化問題

子音キーは図 4.9 中の番号 1〜10 の場所に並べるものとする. 配置の組み合わせは $10! = 3628800$ 通りあり, 全ての組み合わせを計算した. ある配置についての評価値は, 各キーの配置による入力のコストと, キーの使用頻度の積和で表現できる. しかし, 各キー 1 つ 1 つに対して入力コストが独立ではなく, 直前に入力した n 個のキーの位置に入力コストが依存する可能性があるため, n = 0, n = 1 の場合について計算を行った. $n > 1$ の場合も考えられるが, Popie で

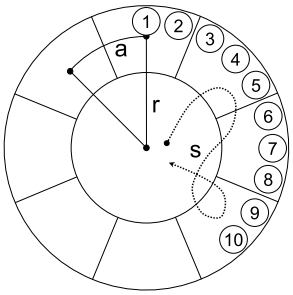


図 4.9: Popie の子音キーの配置最適化に使用するレイアウト

描かれるストロークの特性上, 2 つ以上前のストロークが現在のストロークに影響を与える影響は, 非常に少ないと考えられるため除外した.

ある配置における評価値 E_n は以下のように定義する. 評価値が低いほど, より良い配置であると言える.

$$E_0 = \sum_{i=1}^{10} Cost(i) \times Freq(Key(i))$$

$$E_1 = \sum_{i=1}^{10}\sum_{j=0}^{10} Cost(i,j) \times Freq(Key(i), key(j))$$

ただし

$Key(pos_i)$	:	番号 pos_iに配置されたキー
$Cost(pos_i)$	:	番号 pos_iの位置での入力コスト
$Cost(pos_i, pos_j)$	:	番号 $posi$ の入力に続けて番号 $posj$ を入力するコスト
$Freq(key_i)$	:	key_iの使用頻度
$Freq(key_i, key_j)$	:	key_iに続けて key_jを使用する頻度

4.5.2 配置によるコスト

配置場所におけるコストには, 1) メニューの中心からオクタントの中心までの距離を r, オクタントの中心間の距離を a とした場合に (図 4.9), 番号が 1, 4, 7, 10 の場所はコストが 2r であり, 番号が 2, 3, 5, 6, 8, 9 の場所はコストが 2r+a とするもの, 2) ユーザ実験のログから算出した, 各場所で選択するのにかかった時間, 3) ユーザ実験のログから算出した, 各場所で選択するのに移動したペンの距離, の 3 種類を用意した.

1) のコストでは n = 0, n = 1 のいずれでもコストの値は同じである. 2), 3) のコストには, 慣れに伴って入力速度が速くなり, ペンを動かす軌跡が最適化することを考慮し, 各ユーザ各セッションごとに最大値で正規化したものを, 全ユーザ全セッションで平均し, 再度最大値で正規化したものを用いた. n = 0 の場合のコストを表 4.8 に, n = 1 の場合のコストを表 4.9 に示す.

1, 4, 7, 10 番の配置ではペンの移動が直線的になり, 2, 3, 5, 6, 8, 9 の配置ではペンの移動が三角形を描くため, 前者の方がコストが低くなると考えられる.

表 4.8: 実験のログから算出した入力コスト (n=0)

	1	2	3	4	5	6	7	8	9	10
時間	0.555	0.759	0.805	0.682	1.000	0.995	0.952	0.936	0.812	0.596
距離	0.729	0.964	0.980	0.754	1.000	0.990	0.773	0.957	0.969	0.705

表 4.9: 実験のログから算出した入力コスト (n=1)

時間		1	2	3	4	5	6	7	8	9	10
直前に入力を行う場所	1	0.472	0.528	0.529	0.552	0.620	0.584	0.634	0.709	0.668	0.643
	2	0.560	0.656	0.630	0.613	0.793	0.862	0.753	0.801	0.661	0.578
	3	0.531	0.683	0.721	0.585	0.858	0.746	0.678	0.816	0.719	0.657
	4	0.572	0.585	0.723	0.482	0.755	0.679	0.677	0.781	0.664	0.689
	5	0.613	0.798	0.776	0.711	0.646	0.872	0.805	0.882	0.827	0.744
	6	0.669	0.806	0.763	0.737	0.890	0.715	0.865	0.815	0.796	0.685
	7	0.635	0.897	0.741	0.794	0.785	0.700	0.623	0.719	0.714	0.792
	8	0.641	0.763	0.816	0.824	1.000	0.932	0.783	0.866	0.746	0.748
	9	0.658	0.705	0.677	0.655	0.746	0.761	0.747	0.954	0.815	0.700
	10	0.617	0.587	0.650	0.591	0.727	0.627	0.719	0.572	0.685	0.674
距離		1	2	3	4	5	6	7	8	9	10
直前に入力を行う場所	1	0.705	0.811	0.816	0.726	0.818	0.830	0.719	0.864	0.827	0.726
	2	0.804	0.933	0.903	0.823	0.979	0.977	0.799	0.932	0.926	0.795
	3	0.807	0.923	0.978	0.833	0.960	0.959	0.825	0.928	0.931	0.822
	4	0.717	0.828	0.834	0.696	0.891	0.826	0.736	0.839	0.819	0.686
	5	0.788	0.940	0.981	0.829	0.919	0.934	0.841	0.936	0.903	0.825
	6	0.824	0.933	0.957	0.837	0.882	0.958	0.842	0.912	0.948	0.803
	7	0.726	0.853	0.811	0.759	0.818	0.692	0.667	0.694	0.810	0.736
	8	0.789	0.940	0.930	0.821	0.915	1.000	0.832	0.945	0.914	0.815
	9	0.795	0.922	0.899	0.800	0.924	0.911	0.816	0.978	0.923	0.819
	10	0.693	0.805	0.815	0.692	0.824	0.825	0.627	0.760	0.802	0.601

表 4.10: 子音キーの使用頻度

A	K	S	T	N	H	M	Y	R	W
16.59%	13.57%	15.66%	16.68%	6.49%	6.11%	3.80%	5.35%	7.34%	8.42%

表 4.11: 連続する 2 つの子音キーの使用頻度

		A	K	S	T	N	H	M	Y	R	W
直前に入力されるキー	A	1.40%	2.53%	5.18%	2.37%	1.28%	1.03%	0.52%	0.28%	0.96%	1.17%
	K	2.31%	1.54%	1.69%	1.95%	1.11%	0.64%	0.50%	0.93%	1.12%	1.80%
	S	3.85%	1.35%	0.62%	1.80%	0.49%	0.38%	0.45%	2.28%	3.04%	1.76%
	T	3.08%	2.72%	2.34%	2.56%	1.04%	1.43%	0.90%	0.78%	0.58%	0.88%
	N	0.90%	1.14%	1.05%	1.21%	0.35%	0.66%	0.33%	0.26%	0.28%	0.45%
	H	0.85%	0.67%	0.77%	0.85%	0.39%	0.23%	0.16%	0.31%	0.52%	1.12%
	N	0.49%	0.39%	0.57%	0.88%	0.36%	0.14%	0.12%	0.09%	0.31%	0.43%
	Y	2.92%	0.78%	0.44%	0.34%	0.21%	0.18%	0.17%	0.03%	0.16%	0.29%
	R	0.53%	0.71%	0.64%	3.20%	0.43%	0.38%	0.22%	0.37%	0.24%	0.51%
	W	0.55%	1.59%	1.77%	1.52%	0.82%	0.84%	0.39%	0.14%	0.40%	0.33%

4.5.3 子音キーの使用頻度

CD-毎日新聞 2001 年度版 [31] のテキストデータをもとに, 子音キーの使用頻度を計算した. 各子音キーの単独での使用頻度を表 4.10 に, 直前の入力による各子音キーの使用頻度を表 4.11 に示す. 表 4.10 から, “AKST” の 4 つキーの使用頻度が他に比べて高いことが分かる. また表 4.11 から, ‘A’‘S’ と続けて入力する頻度が 5.18%と最も高く, 続けて ‘S’‘A’, ‘R’‘T’, ‘T’‘A’, ‘S’‘R’, の順でキー連続して使う頻度が高いことが分かる.

4.5.4 最適化の解とその考察

表 4.12 に, 直前の n 個の入力に依存することを過程した場合の, コスト別の最適化の解と, 仮名順と比較した割合を示す. また, 評価値が最も高くなる最悪解と仮名順を比較した割合も併せて示す.

最もよい結果は, n = 1 におけるペンの移動距離を用いたものであり, 子音キーが 1 番から順に “TNHKMYSRWA” の配置, 仮名順の 94.04%のコストであった. 一方, 最悪のケースを見た場合, 7%〜18%の評価値の上昇が見られた. また, 各 n, 各コストにおけるキー配置の最適解はバラバラになっているが, 1, 4, 7, 10 のコストの低い場所に, 使用頻度の高い “AKST” のキーが配置される傾向が分かる.

キーの配置が変わると, ユーザは時間をかけて配置を学習しなければならない. ここで示した最適化の結果から, パフォーマンスの向上は, 最高でも 6%程度しか期待することができない. さらに, 時間をコストとする最適解では, わずか 1, 2%の改善しか期待できない. これでは, 覚え易い仮名順の配置ではなく, バラバラになったキーの配置を時間をかけて学習するだけのメリットがないと言える.

表 4.12: 最適化された配置と, 最適解・最悪解での仮名順と比較した評価値の割合

n	コストの種類	最適な配置										最適解	最悪解
		1	2	3	4	5	6	7	8	9	10	割合 (%)	割合 (%)
0	幾何的な距離	T	W	R	A	N	H	S	Y	M	K	94.50	107.68
	ペンの移動時間	T	K	W	S	M	Y	H	N	R	A	97.63	116.24
	ペンの移動距離	A	R	H	S	M	Y	K	W	N	T	95.28	107.15
1	幾何的な距離	A	N	M	S	H	R	T	Y	W	K	94.59	107.74
	ペンの移動時間	T	A	W	K	M	R	H	Y	N	S	99.48	118.00
	ペンの移動距離	T	N	H	K	M	Y	S	R	W	A	94.04	107.54

第5章　Popieの汎用化

本章では, 日本語入力 Popie, およびフローメニューの汎用化について, 機能の検討と, Windows 上で動作する汎用的なプログラムの実装の紹介をする.

5.1　汎用化の必要性

FlowMenu はメニュー操作やアルファベット入力を, ペンの周囲で行えるインタフェースである. しかし, 専用のアプリケーションに実装されているため汎用的ではない. 日本語入力システム Popie を実装した PopiePoint もプロトタイプの実装のため, PopiePoint 上でしか入力を行う事ができない.

より多くのアプリケーションに対して, ペンの周囲で操作が可能なインタフェースを利用できるようにすることで, ペンで操作するコンピュータをより使いやすくことができる. よって, これらのシステムの汎用化が必要である.

しかし, 現在のコンピュータ社会においてこのシステムの汎用化を実現するのは, 簡単なことではない. そこで, 汎用化のステップとして Windows 上で動作し, 各種アプリケーションに対してメニュー操作や文字入力を実行することができるアプリケーションを実装した [36].

5.2　機能の検討

まず, マウスとキーボードによる操作を分類し, どのような機能が必要であるかを検討する.

- **メニュー操作** マウスで選択
- **ボタン操作** ボタンをマウスでクリック
- **ドラッグ操作** ある特定の部分をマウスでドラッグ
- **文字入力** キーボードで入力

メニュー操作は FlowMenu の機能そのものではあるが, 対象となるアプリケーションや, 使用する状況によって表示するメニュー項目が異なる. そのため, アプリケーションで使われているメニューを取得したり, プロファイルを用いてメニューのカスタマイズができるような実装が必要である.

ボタン操作とドラッグ操作は, メニュー操作を組み合わせることで, より使い勝手が向上すると考えられる. 例えば, ボタン操作はペンでタップするという動作になるが, ウィンドウのタ

イトルバーにあるような小さなボタンは押しにくい. そこで, 一般的な “閉じる” やブラウザの “戻る”, “進む” などのボタンについては, メニュー項目に組み入れる. また, ドラッグ操作によるウィンドウのサイズ変更でも, ウィンドウの端をタップするのは難しいので, 同様にメニュー項目からサイズ変更できるようにする. これらは, 一種のショートカットのような機能である.

文字入力については日本語入力システム Popie を用いることで対応できる.

5.3 PopieWin のインタフェース

本節では Windows 上で, Popie による日本語入力が行えるフローメニューのシステム PopieWin について示す. PopieWin も Popie と同様, ペンを画面にタップ&ホールドすることで, ペンの位置にあるアイコンや入力フォームなどのコンポーネントに対応するメニューを表示する. またペンをタップ&ホールドして一定時間経過後, ペン (カーソル) 位置のを中心として, 小さい円を円形に順に表示することで, ユーザにメニューが表示されるまでの時間をフィードバックするようにした (図 5.1). 円が表示されている間にペンを画面から離すと, メニューの出現をキャンセルすることができる. 図 5.2 はデスクトップ上のディレクトリに対して, フローメニュー

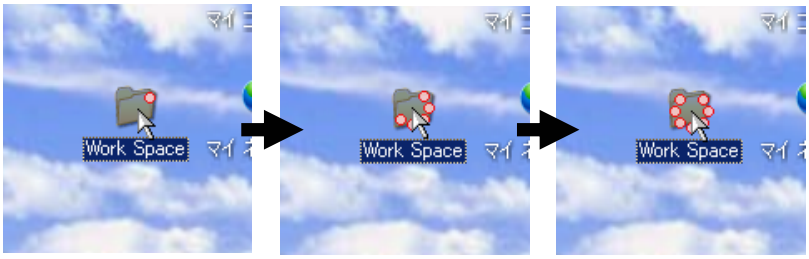


図 5.1: タップ&ホールド中にメニュー表示までの時間をフィードバックする.

を表示した時のスナップショットである. 基本的には, マウスで右クリックした場合とほぼ同じメニューを実行することが可能だが, いくつかのメニュー項目をまとめ, トップレベルのメニュー項目を少なくしている. 図 5.3 は Popie を用いて, Web ブラウザ上のフォームに文字を入力した場合のスナップショットである. 本システムでは, キーボードで文字入力できるほぼ全てのコンポーネントに対して文字を入力することが可能である. ユーザは文字を入力したいフォーム, またはテキストボックスの上でフローメニューを表示する.

図 5.4 はテキストエディタのメニューを実行する様子を示したものである. PopieWin は操作対象になるアプリケーションのメニュー項目を取得し, 表示・実行することが可能になっている. ここでは “右端で折り返す” というメニューを実行し, 2 行で横長に表示されていたテキストを, 右端で折り返して全体を見れるようにしている.

メニュー項目の取得は, アプリケーション個別に実装しているのではなく, アプリケーションに対して一般化した形式で行っているので, 様々なアプリケーションのメニュー項目を取得可能である.

図 5.2: ファイル操作のメニュー表示

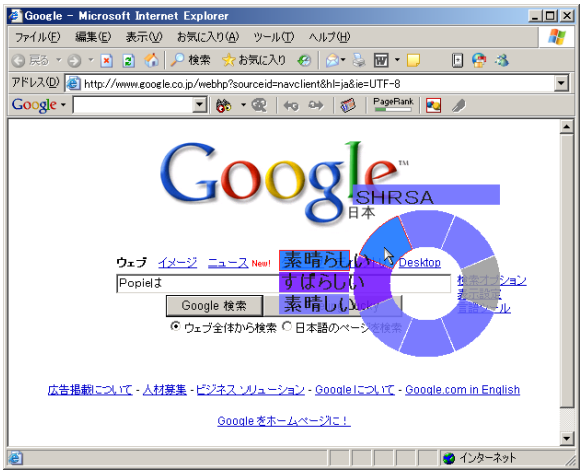


図 5.3: ブラウザのフォームに Popie で文字入力

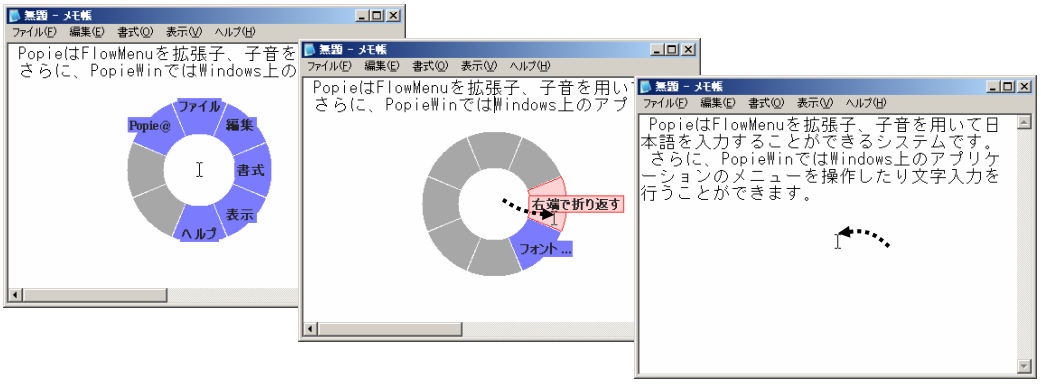


図 5.4: アプリケーションのメニュー項目を表示・実行する

5.4 PopieWin の実装

PopieWin は Microsoft Visual Studio .NET により開発を行った. プログラム言語は C と C++ である. 本システムは Windows のマウスイベントを捕捉することで, どのアプリケーション上でも動作可能にしている. 表示は, 常に最上位に表示される半透明なウィンドウを作成し, そこに描画を行うことで実現した.

5.4.1 システムの状態遷移

システムの状態遷移図 (図 5.5) とその説明を示す.

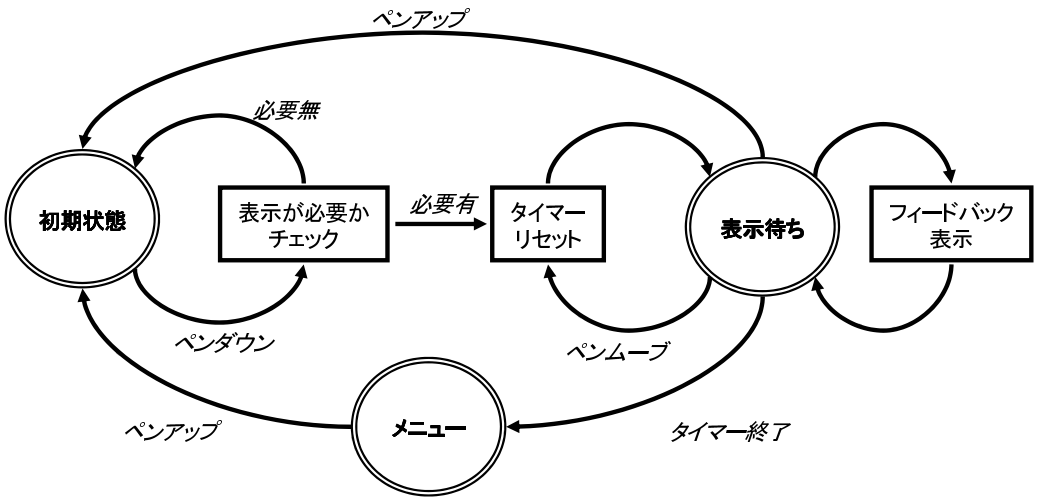


図 5.5: PopieWin の状態遷移図

- 初期状態においてペンを画面につけると, システムがペンダウンを捕捉し, その位置のアプリケーションやコンポーネントが何であるかを取得し, メニューの表示が必要かどうかをチェックする.
  - メニューの表示が必要無い場合は初期状態に戻る.
  - メニューの表示が必要な場合は, タイマーをリセットし表示待ち状態に移行する.
- 表示待ち状態では一定時間間隔で, フィードバックのための表示を行う.
- 表示待ち状態でにペンが動かされるとタイマーをリセットする.
- 表示待ち状態でペンが画面から離されると, 初期状態に戻る.
- 表示待ち状態でタイマーが終了すると, 操作対象のアプリケーションのメニュー項目を持つフローメニューを表示する.

  - このとき, 操作対象のアプリケーションに対してマウスアップのイベントを発行し, メニュー操作に併せてペンを動かしたときに, アプリケーションに対してドラッグ操作が発生しないようにする.

- メニュー状態ではペンの動きに合わせ, メニュー項目の更新やメニューの実行を行う.
- メニュー状態でペンを離したら初期状態に戻る.

5.4.2 メニューの取得

Windows におけるメニューの取得方法は大きく 3 通りある. 1 つはウィンドウハンドルに対して, Win32API の GetMenu 関数を実行することでメニューハンドルを取得する方法である. この方法でメニューが取得できると, メニュー項目の取得やメニューの実行は容易に実現できる.

しかし, 多くのアプリケーションはこの方法ではメニューを取得することができないため, 2 つ目の方法として IAccessible インタフェースを用いる. このインタフェースはアクセシビリティ向上のために作られたもので, これを用いることでメニュー項目の取得, 実行が可能になる. ただし, このインタフェースに対するアプリケーション側の実装は, 開発者に対してガイドラインが示されているだけで, インタフェースの中身が実装されていないアプリケーションもある.

現在のシステムでは, 上記 2 つの方法で取得できないメニューを使うことはできない. 将来的に, 各アプリケーションごとにプロファイルを作成し, どのメニュー項目にどの機能を割り当てるか等の情報をあらかじめ用意することで, ショートカットキーの設定されている機能には比較的容易に対応可能である. しかし, それ以外のメニューを実行するには, マウスイベントやキーイベントをエミュレーションする必要があり, 現実的な実装ではない.

5.5 汎用化の今後の課題

本章では, フローメニューおよび, Popie の汎用化について検討し, Windows 上で動作するシステム PopieWin を実装した. しかし, PopieWin はアプリケーションの外部から操作を実行するため, 対応できないアプリケーションがあったり, 動作速度などの面で十分とは言えない.

今後の課題としては, 開発したシステムを公開することや, Java や Windows のライブラリとしてフローメニューや Popie の機能を整備することが考えられる. また, プロファイルを用いたメニューのカスタマイズを必要な機能として示したが, 今後実装する必要がある.

第6章　考察

本章では, 子音入力手法, 予測・補完と母音選択機能の効果, Popie のインタフェース, システムの汎用化についての考察を述べる.

6.1　子音入力手法について

本研究で用いた子音入力手法について, 自動入力プログラムにより, 入力のシミュレーションを行った結果, ローマ字入力手法に比べて, 約 18%の操作数を削減できることが分かった. これは, 子音 10 個に対し, 母音が 5 個であり, 子音・母音と交互にキーを入力した場合は, 組合せの可能性が 50 通り, 子音・子音と続けてキー入力した場合は, 組合せの可能性が 100 通りになるため, 対象となる単語の中から, 候補を絞り込み易いからだと考えられる.

子音による入力では覚えるキーの数も少なくなり, ローマ字入力よりも操作数が削減できるため, ユーザへの負荷の少ない入力手法であると言えるだろう.

今回の評価では, 単純に操作数による比較を行ったが, さらに正確な評価をするとならば, 各キーごとに重みを定義し, 重みに従った評価値を用いるべきだろう. この場合, 各キーの重みには, 選択にかかる単純な時間に加え,「子音入力」から「候補選択」,「子音入力」から「母音選択」などの, 機能に対する切り替えの時間等を含める事が望ましいと考えられる.

また, Popie では上位 3 つの候補をスクロール無しで選択することが可能であるが, 同時に幾つ選択できると良いのかといった点についても検討が必要だと思われる.

6.2　予測・補完と母音選択機能の効果ついて

ローマ字入力や, 携帯電話による入力において, 予測・補完機能を持った入力手法は効果を挙げていた.

本研究では, より候補が多い子音入力手法においても, 予測・補完機能がその効果を発揮することが分かった. また, 母音選択の機能も, 操作数を削減することに効果を示していた. ただし, 予測・補完機能だけを追加しただけでは, 候補数を増やすだけで逆効果となるため, 母音選択と一緒に利用することが必要である.

6.3 Popie のインタフェースについて

Popie のインタフェースがフローメニューと大きく異なる点は, 候補のリストを表示するところにある. Popie では, 全ての操作をフローメニュー上で行うというポリシーで開発し, リストの操作も, フローメニュー上で行うという方法であった. このインタフェースはユーザによる実験から, 直感的でなくあまり好まれないという評価を受け, さらに, 選択だけのタスクにおいては, リストの上にペンをスライドさせ, 選択したい候補の上でペンを離すという, 直接的な操作の方が速い結果になった.

しかし, 連続的な操作と, 非連続的な操作 (ペンを一度上げる) が交互に起きた場合, 連続的な操作を続ける場合と比べて, ユーザが頭の中で操作のモードを切り替えるのに時間が掛かることが考えられるので, 一概にどちらが優れているとは言えないが, 選択操作に関しては, 更なる研究が必要であるといえる.

6.4 システムの汎用化について

本研究では, 汎用化のステップという位置づけで, Windows 上のアプリケーションを操作することができるシステム PopieWin を開発した.

実験による評価は行っていないが, 様々なアプリケーションに対して入力できるようになったことで, 使い勝手が向上したのは確かである. 例えば, Popie を実装した Java のアプリケーションでファイルを保存するために, 標準のファイルビューアを開くと, そこでは Popie は使えず, 結局ソフトキーボードを開くことになり, かえって面倒になってしまっていた.

PopieWin によって, 少なくともソフトキーボードは, ほとんどのアプリケーションに対して必要なくなり, 先ほどの例のようなことは起こらなくなったことは, ペンコンピュータのためのインタフェースとして進歩であると言える. 今後さらに汎用化を進め, このインタフェースを, ペン・コンピュータのインタフェースとして広めていきたい.

第7章　結論

本研究では, ペンのための日本語入力システム Popie を開発し, その評価を行った. また, 汎用化についても検討を行い, Windows 上で動作するシステムを実装した.

Popie は子音入力と母音選択, 予測・補完による入力をフローメニュー上で行うというシステムである. 子音キーの位置を覚えることにより素早い入力が行え, また母音選択と予測・補完の機能により操作数を削減し, ユーザへの負担を減らすことが可能になった. そして各種評価から, 日本語入力手法 Popie が有効な手法であることが示され, Popie の改善すべき点についても明らかになった.

さらに Windows 上で動作するアプリケーション PopieWin の実装により, これまで以上にフローメニュー, そして Popie を利用し易くなり, これによって, ペンコンピュータの操作性が向上したと言える.

謝辞

本論文の執筆にあたり, 指導教官である田中二郎教授ならびに, 三末和男助教授, 講師の志築文太郎先生と高橋伸先生, 現在は北陸先端科学技術大学院大学助手の三浦元喜先生には, 大変貴重な意見やご指導、激励を頂きました. 本論文の執筆に至るまでにも色々とお世話になったことを含め, この場を借りて深く御礼を申し上げたいと思います.

また, 田中研究室の皆様にも, ゼミ等を通じて多くの意見, アドバイスを頂きました. 実験なども快く引き受けて下さり, 研究に協力して頂いたこと改めてお礼申し上げます. それから忙しい中, 実験に協力してくれた陸上競技部の後輩にも感謝したいと思います.

そして最後に, 自分を支えてくれた両親や, 全ての友人にも感謝申し上げたいと思います. 本当にありがとうございました.

参考文献

[1] Ivan Sutherland. Sketchpad - a man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference*, pp. 507–524, May 1963.

[2] François Guimbretière and Terry Winograd. FlowMenu: Combining command, text, and data entry. In *Proceedings of ACM User Interface Software and Technology 2000 (UIST 2000)*, pp. 213–216, May 2000.

[3] Ben Shneiderman. *Designing the User Interface*. Addison Wesley Longman, Inc., third edition, March 1998.

[4] Don Hopkins. The design and implementation of pie menus. *Dr. Dobb's Journal.*, Vol. 16, No. 12, pp. 16–26, December 1991.

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

[6] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 95–100. ACM Press, 1988.

[7] Gordon Kurtenbach and William Buxton. The limits of expert performance using hierarchic marking menus. In *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 482–487. ACM Press, 1993.

[8] Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Control menus: excecution and control in a single interactor. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pp. 263–264. ACM Press, 2000.

[9] I. Scott MacKenzie and Shawn X Zhang. The design and evaluation of a high performance soft keyboard. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1999 (CHI 1999)*, pp. 25–31, May 1999.

[10] Textware Solutions. Fitaly. http://www.fitaly.com, 1998.

[11] Shumin Zhai, Michael Hunter, and Barton A. Smith. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the ACM Symposium on User Interface Software and Technology 2000 (UIST 2000)*, pp. 119–128, November 2000.

[12] Conrad H. Blickenstorfer. Graffiti: Wow! *Pen Computing Magazine*, pp. 30–31, January 1995.

[13] David Goldberg and Cate Richardson. Touch-typing with a stylus. In *Proceedings of ACM INTERCHI 1993 Conference on Human Factors in Computing Systems (CHI 1993)*, pp. 80–87, April 1993.

[14] Per-Ola Kristensson and Shumin Zhai. Shark^2: a large vocabulary shorthand writing system for pen-based computers. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 43–52. ACM Press, 2004.

[15] 堀井真吾, 菊池猛, 千葉玉三, 赤池英夫, 角田博保. 速記型ペン入力方式の検討. 情報処理学会ヒューマンインタフェース研究会研究報告 95-HI-59, pp. 1–8, March 1995.

[16] Dan Venolia and Forrest Neiberg. T-cube: a fast, self-disclosing pen-based alphabet. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 265–270. ACM Press, 1994.

[17] Jennifer Mankoff and Gregory D. Abowd. Cirrin: A word-level unistroke keyboard for pen input. In *Proceedings of User Interface Software and Technology 1998 (UIST 1998)*, pp. 213–214, November 1998.

[18] Ken Perlin. Quikwriting: Continuous stylus-based text entry. In *Technical Note of ACM User Interface Software and Technology 1998 (UIST 1998)*, November 1998.

[19] Michael D. Fleetwood, Michael D. Byrne, Peter Centgraf, Karin Q. Dudziak, Brian Lin, and Dmitryi Mogilev.

[20] Poika Isokoski and Roope Raisamo. Quikwriting as a multi-device text entry method. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pp. 105–108. ACM Press, 2004.

[21] 森田祐二. TUT-Code. http://www2b.biglobe.ne.jp/˜g_morita/tut-code, 1997.

[22] 増村成吾. G-Code. http://www.asahi-net.or.jp/˜QX5S-MSMR/index.html, 1997.

[23] 森田正典. 日本文入力方式と鍵盤方式の最適化. 電子情報通信学会論文誌 D, Vol. J70-D, No. 11, pp. 2047–2057, 1987.

[24] 増井俊之. ペンを用いた高速文章入力手法. In *Proceedings of Workshop on Interactive Systems and software 1997 (WISS 1997)*, pp. 51–60. 日本ソフトウェア科学会, 近代科学社, December 1997.

[25] 小松弘幸, 高林哲, 増井俊之. 文書蓄積システム kukura を用いた予測入力. In *Proceedings Workshop on Interactive Systems and software 2002 (WISS 2002)*, pp. 43–47. 日本ソフトウェア科学会, December 2002.

[26] T9. T9 text input home page. http://www.t9.com.

[27] Kumiki Tanaka-Ishii, Yusuke Inutsuka, and Masato Takeichi. Japanese text input system with digits –can japanese text be estimated only from consonants?-. In *Proceedings of Human Language Technology Conference 2001 (HLT 2001)*, March 2001.

[28] 田中久美子, 犬塚祐介, 武市正人. 携帯電話における日本語入力-子音だけで日本語が入力できるか-. 情報処理学会論文誌, Vol. 43, No. 10, pp. 3087–3096, October 2001.

[29] 森稔, 澤木美奈子. [prpr サーベイシリーズ] 低品質文字の認識手法とその応用に関するサーベイ. 信学技報, March 2002.

[30] 佐藤大介, 三浦元喜, 志築文太郎, 田中二郎. ペンによるメニュー選択に基づく日本語入力手法. 日本ソフトウェア科学会第 20 回大会論文集, September 2003.

[31] 毎日新聞社. CD-毎日新聞 2001 年度版, 2001.

[32] 松本裕治, 北内啓, 山下達雄, 平野善隆, 松田寛, 高岡一馬, 浅原正幸. 日本語形態素解析システム『茶筌』 version 2.2.1 使用説明書. 奈良先端科学技術大学院大学, December 2000.

[33] SKK Openlab. SKK. http://openlab.ring.gr.jp/skk/index-j.html.

[34] 佐藤大介, 志築文太郎, 田中二郎. メニュー選択に基づく子音による日本語入力手法 popie の候補選択インタフェースの検討. 情報処理学会研究報告 2004-HI-108, pp. 39–46, May 2004.

[35] Daisuke Sato, Motoki Miura, Buntarou Shizuki, and Jiro Tanaka. Menu-selection-based japanese input method with consonants for pen-based computers. In *Proceedings of 6th Asia-Pacific Conference on Computer-Human Interaction (APCHI2004)*, pp. 399–408. LNCS3101, July 2004.

[36] 佐藤大介, 志築文太郎, 田中二郎. ペンの周囲で操作することを可能にするインタフェース. In *Proceedings of Workshop on Interactive Systems and software 2004 (WISS 2004)*, pp. 71–76, December 2004.

付 録A　子音入力エンジンの仕様

日本語入力システム Popie の子音入力エンジンの仕様について示す. 子音入力エンジンは, Popie のインタフェースと独立した設計になっており, 別のインタフェースから子音入力エンジンを利用することが容易にしている.

子音入力エンジンを利用する場合, 子音キーや母音選択のためのキーを入力し, 候補数や候補などを読み出すという形になる. 子音入力エンジンは, 入力された文字列を全て記録し, アンドゥ・リドゥの機能も提供する. 開発者は現在のテキストの状態を読み取ることで, 表示などに反映する.

A.1　子音入力エンジンの概要

以下に子音入力エンジンに使用しているクラスについて示す.

Word クラス

辞書を構成する最小単位である. 単語辞書, 単語間関連辞書, 学習辞書で共有して使う. 保持するデータは, 「単語」「その読み」「スコア」「直前の単語」の 4 つである. 最後の「直前の単語」に関しては, 単語間の関連データを扱う場合のみ利用される.

Dictionary クラス

名前の通り辞書クラスである. 主な機能は, ファイルから辞書を読み込みメモリ上に辞書データを構築すること, ユーザの入力に対して単語を学習しファイルに出力することである. 辞書は先頭の文字ごとに分類され, 先頭の文字をキーとしてハッシュされている.

Search クラス

入力されたクエリ (ここでは仮名と子音の混ざった文字列) によって, 辞書を検索するためのクラスである. CandidateManager クラスによって使われ, 辞書データからクエリの先頭文字によって取得された単語の列と, クエリの長さと一致する単語を探すか, クエリの長さよりも長い単語を探すかの指定を受け検索を行う.

CandidateManager クラス

Search クラスを使って候補を生成するための部分である. このクラスで, 2 つの検索方法と 3 つの辞書を使った検索を実行する. Search クラスにはあらかじめクエリの先頭文字によって Dictionary クラスから取得した単語列と, 検索方法を指定して検索を実行する.

検索結果は, それぞれの検索方法ごとの係数に従いスコアを調整し, スコアの高い順に並び替える.

Candidate クラス

CandidateManager クラスでの検索結果を構成する最小単位で, データとして「確文字列」「確定文字列の読み」「未確定文字列」「未確定文字列の読み」「スコア」を保持する.

StringState クラス

ユーザの入力によって得られた文字列の状態を表すクラスである. すでに確定された文字列の途中にも文字列を挿入できるように, 文の先頭から見て, 確定文字列, 未変換文字列, 変換対象外の未変換文字列, 確定文字列の順でデータを保持している. また, 学習を行うために前回入力した単語等の情報を保持している.

StringStateManager クラス

子音入力エンジンのインタフェースとも言えるクラスである. Keybind クラスによって定義された子音キーをこのクラスに対して送ることで, 子音の入力, 母音の選択, 候補の選択, 候補のスクロール, 記号の入力, 削除, キャレットの移動, アンドゥ・リドゥなど全ての入力を行う.

Keybind クラス

StringStateManager クラスに送るためのキーを定義したクラス. 各キーと整数との対応づけと, 利用者が表示するべき文字列, 例えば KYE_A に対して 'A' といった対応づけがされている.

UndoRedoManager クラス

アンドゥ・リドゥの機能を管理するクラス. ユーザの入力ごとに, どのような操作が行われたかを記録する. 操作は Command 抽象クラスの各実装で定義されている.

Command 抽象クラス

操作の内容と, アンドゥに対して行う逆操作とリドゥに対して行う順操作が定義される. 操作の種類ごとに逆操作と順操作を定義する必要がある.

図 A.1 にクラス間の関係を現したものを示す. Keybind クラスの定義を使い StringStateManager クラスに入力を行うと, StringStateManager クラスは UndoRedoManager クラスに現在の状態を登録し, CandidateManager クラスを使って候補を取得する. CandidateManager クラスは Search クラス, Dictionary クラスを用いて単語の検索を行い候補を生成する. 候補の情報は StringStateManager クラスを通して取得することができる.

またアンドゥ・リドゥ操作に対応する入力があると, StringStateManager クラスが UndoRedoManager クラスに通知し, Command 抽象クラスの実装が StringState クラスを適切な状態に変化させる.

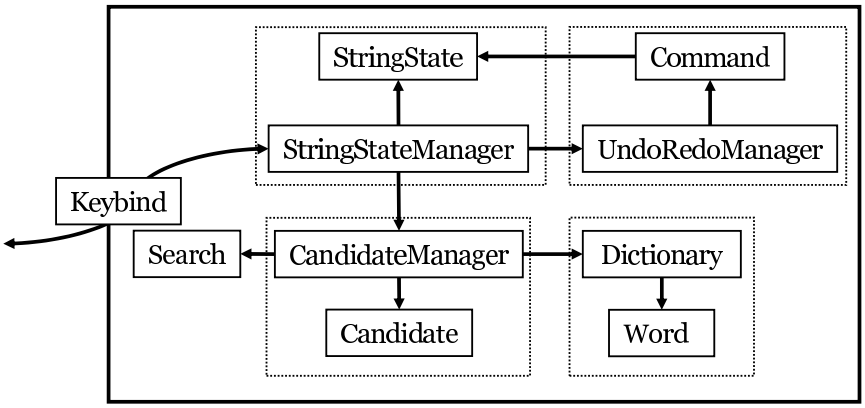


図 A.1: 子音入力エンジンのクラス構成

A.2　子音入力エンジンのインタフェース

ここでは, 子音入力エンジンのインタフェース部分と言える, StringStateManager クラスと Keybind クラスの詳細について示す.

StringStateManager クラス

StringStateManager クラスの主なメソッドを以下に示す.

void startEditing(String initStr, int position, UndoRedo ur)
新しい文章の編集を始めるための設定を行う. 文字列 initStr の position 文字目にキャレットを配置し, アンドゥ・リドゥの記録に ur を用いる. 文字列の編集を止めても, 引数を設定することで編集の続きを行うことができる.

void press(int key)
Keyboard クラスのキーを入力するためのメソッド.

UndoRedo getUndoRedo(), StringState getState()
現在の状態の UndoRedo や StringState を取得するための関数.

boolean isCandEnable(int n)
現在のスクロール位置から n 番目の候補が存在して使用可能かどうかをチェックする.

CandKey getCandKeyAt(int n)
現在のスクロール位置から n 番目の候補を取得する. isCandEnable(n) が true の場合だけ実行するようにする.

int getNCands(), int getPCands()
現在のスクロール位置で, 下側と上側にどれだけの数の候補があるか.

void getVowels(String[] vowel)

現在の状態で子音キーを入力すると確定される文字を取得する. vowel には長さ 5 以上の配列を渡す.

void endEditing()

文章の編集の終了処理を行う. 未確定文字列があった場合, それが子音を含んでいれば削除し, 含んでいなければ仮名のまま確定する.

Keybind クラス

表 A.1 に Keybind クラスで定義しているキーと, その説明を示す. また, Keybind クラスではキーに対応した表示すべき文字列を, 「表示有」と記したキーに対して定義している.

表 A.1: Keybind クラスのキーの定義

キー	説明
KEY_A, KEY_K, KEY_S, KEY_T, KEY_N, KEY_H, KEY_M, KEY_Y, KEY_R, KEY_W (子音キー, 表示有)	子音を入力するためのキー. 順にあ行, か行, さ行, た行, な行, は行, ま行, や行, ら行, わ行.
KEY_a, KEY_i, KEY_u, KEY_e, KEY_o (母音キー, 表示無)	母音を選択するためのキー. 順にあ段, い段, う段, え段, お段.
KEY_det (母音選択終了キー, 表示無)	母音選択の操作において, 1 文字分の母音選択が終了したらこのキーを入力する必要がある.
KEY_CAND1, KEY_CAND2, KEY_CAND3, KEY_CAND4, KEY_CAND5 (候補選択キー, 表示無)	n 番目の候補を選択するためのキー. Popie では右利き用のインタフェースで, 左上が CAND1, 左が CAND2, 左下が CAND3 を割り当てており, CAND4, CAND5 は未使用.
KEY_UP, KEY_DOWN, KEY_UP3, KEY_DOWN3 (スクロールキー, 表示有)	候補を上下にスクロールするためのキー. UP, DOWN はスクロール幅が 1, UP3, DOWN3 はスクロール幅が 3.
KEY_BACKSPACE, KEY_NEWLINE, KEY_SPACE, KEY_TAB (表示有)	バックスペース, 改行, スペース, タブ
KEY_UNDO, KEY_REDO, KEY_NEXT, KEY_PREV (表示有)	アンドゥ, リドゥ, キャレットを後ろに動かす, キャレットを前に動かす.
KEY_SYMBOL (記号キー, 表示有)	未確定文字列 “きごう” を挿入するためのキー.
KEY_UNDEF, KEY_NOOP (表示無)	未定義の状態のキーと, 何も操作を行わないキー.

付 録B　PopiePointのマニュアル

B.1　はじめに

B.1.1　PopiePoint とは

PopiePoint は日本語入力システム Popie を実装した, ペン[1]によって操作することを前提としたアプリケーションであり, Popie による文字列の入力の他に, 図形の描画や画像の貼り付けが行える. また, スライド形式でページを増やす事が可能で, プレゼンテーションにも利用することができる.

PopiePoint では Popie による文字列の入力, および他の全ての操作はフローメニューと呼ばれる円形のメニューにより行われる.

B.1.2　インストールと起動

PopiePoint は Java で実装されている. Java™2 SDK, Standard Edition バージョン 1.4.0〜1.4.2 での動作を確認している. ただし, 日本語を扱うため, UNIX 環境等では日本語フォントの設定が必要である. Java のインストールやパスの設定等についてはここでは触れない.

起動方法

PopiePoint は Jar ファイルにまとめてあり, コマンドラインから以下のように起動する.

```
# java -Xmx128m -jar PopiePoint.jar
```

ここで “-Xmx128m” は Java 仮想マシンに与えるオプションで, 最大のヒープ領域を指定するものである. PopiePoint では, Popie で使用する辞書の単語数によって必要なメモリ領域が変わるが, Java 仮想マシンの初期値である 64M では実行できない. 辞書サイズの小さいものは, `PopiePoint_M.jar, PopiePoint_S.jar` である.

また, 必要に応じて以下の起動オプションを利用することができる.

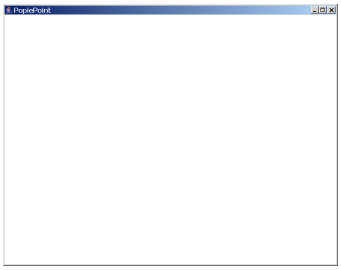
起動オプション

-file *filename*　　PopiePoint のファイルを指定.

[1] このマニュアルは, ペンで利用することを前提として書かれているが, マウスでも操作することは可能である.

-size *width height*	PopiePoint の起動時のウィンドウサイズを指定.
-max	画面領域いっぱいのウィンドウサイズで起動.
-demo	デモモードで起動. ペンの軌跡が表示される. (起動中のオン・オフも可)
-fontsize	Popie で入力する文字列のデフォルトのフォントサイズ.

PopiePoint を起動すると, 辞書読み込み中のメッセージと共にウィンドウが開く. PopiePoint には, メニューバーのようなインタフェースは全くないので, 初期状態では右のような真っ白なウィンドウが開く.



B.1.3 ディレクトリ構成

PopiePoint のディレクトリは以下のように構成される.

Makefile	make コマンドで使用.
image/	PopiePoint が使用する画像.
jar/	PopiePoint.jar などが保存される.
src/	PopiePoint のソースコード.
dictionary/	Popie の辞書.

コマンドラインから以下のように make を実行すると, jar ディレクトリに PopiePoint の jar ファイルが作成される.

```
# make jar
```

B.2 PopiePoint の基礎

B.2.1 ページ

PopiePoint は紙芝居のように, ページを複数枚作ることができる. また, 背景用に特別なページがあり, 普通のページを切り替えても背景が残るようになっている (図 B.1). 同様の構成を持つアプリケーションとして, Macintosh の HyperCard が挙げられる.

背景のページも普通のページと同じように, 複数枚作ることが可能になっており, タイトルページと 2 ページ目以降の背景を変えるというようなことができる. 通常は普通のページを編集するモードになっており, 背景にあるオブジェクトを操作することはできない. 背景編集モードに変更することで背景のオブジェクトを操作することができるようになる. 背景編集モードにおいてページを追加すると, 背景が追加される. このとき, 新しい背景には新しい普通

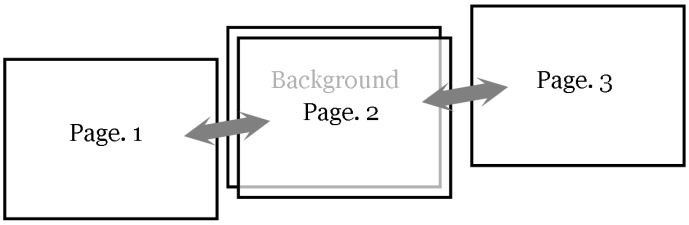


図 B.1: PopiePoint のページのイメージ

のページが用意される. 普通のページにおいてページを追加すると, ページ追加の操作を行ったページの背景が, 新しく作られたページの背景として登録される.

各ページには, 図形や文字列, 画像等のオブジェクトを配置することができる. 各オブジェクトはページごとに管理されている. 背景のページは, 実装上では普通のページを継承する形になっているので, 編集操作は基本的に背景ページでも普通のページでも同じである.

B.2.2 描画モードとメニューの表示

PopiePoint では, 初期状態は常に自由曲線を描画するモードである. ペンで描いたストロークが PopiePoint に追加される. このとき, 描かれた線はペンのイベントから取得された点列データではなく, 3 次ベジエ曲線で近似された曲線になる. そのため, あまり細かい線は描くことができない.

描画される自由曲線は, それを含む破線の矩形の領域を持っている. 次のストロークをその矩形の中から始めると, そのストロークは矩形を持つストロークと統合され, 1 つのオブジェクトとして扱えるようになる. 矩形以外の場所からストロークを描くと, 別のオブジェクトとして扱われる. また, ストロークを描いた後 1 秒間は, 一時的に矩形の領域が広く取られるが, 1 秒経つと, ストロークを囲むように矩形の領域が小さくなる.

ストロークを描かずに, ペンを画面に置いて 300ms 待つとフローメニューが表示される. フローメニューは基本的にペンを離すまでメニューのモードである. タブレット PC 等を用いると, ペンが動いていなくてもカーソルが微妙に振動することがあるため, 3 ピクセル以下の動きは無視するようになっている.

また, ペンである対象を選択する際に直接ペンでタッチするのではなく, ペンを画面に付けた状態でスライドさせて対象の領域に移動する方が, 選択時間も早く, エラーが少ないとされているため, [2] ある程度までの長さのストロークを描いたあと, ペンを離さずに停止させておくと, そこまで描いたストロークをキャンセルしフローメニューを表示するようにしている. (ただし, 線だけを描く場合には注意が必要である.)

[2] Xiangshi Ren, Kinya Tamura, Jing Kong, and Shumin Zhai. Candidate display styles in japanese input. In *INTERACT 2003, IFIP Conference on Human-Computer Interaction*, pp. 868–871, September 2003.

B.3　フローメニュー

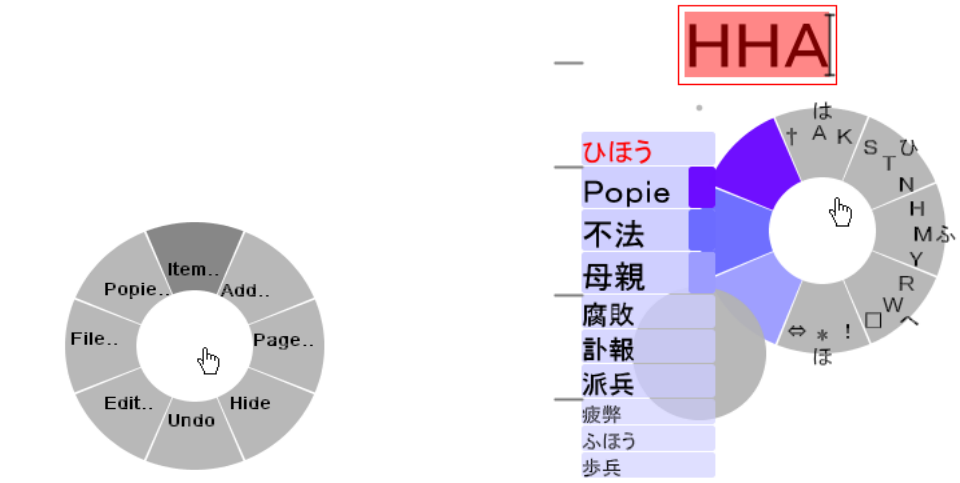
PopiePoint で使うフローメニューには, Popie を含め, いくつか通常のフローメニューからアレンジしたものがある.

B.3.1　通常のフローメニュー

フローメニューを表示すると, 図 B.2 a のような通常のフローメニューが出現する. フローメニューはドーナツ型であり, 中心部をレストエリア, 外側の 8 等分されたそれぞれをオクタントと呼ぶ. トップメニューには, 8 つのサブメニュー (オクタント) がある. グレーになっている部分は使用できないメニューであることを表している.

B.3.2　Popie のフローメニュー

図 B.2 b は日本語入力をするための Popie のフローメニューである. フローメニューの左側に, 候補のリストが表示されている. 上部に表示されている “HHA” の文字列は, PopiePoint の文字列オブジェクトである.



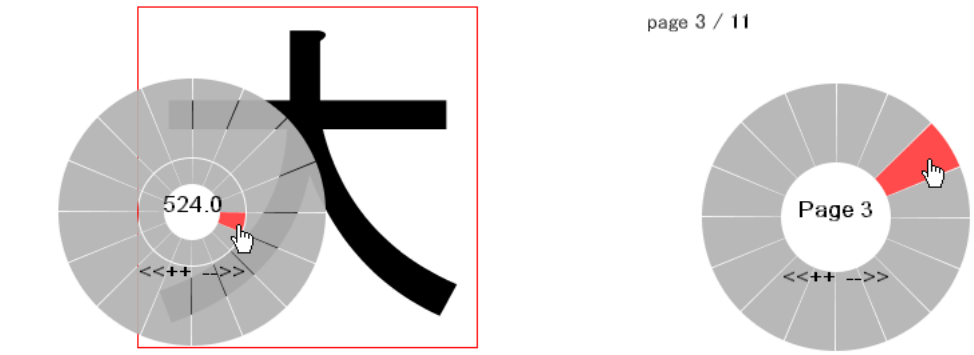
a)　　b)

図 B.2: フローメニューの初期状態と Popie のフローメニュー

B.3.3　連続値選択のためのフローメニュー

図 B.3 はフローメニューを使って, 連続値を選択するためにアレンジしたもので, 16 等分されている. 図 B.3 a はオブジェクトのズーム率を変更するためのものであり, 内側にある小さな部分でペンを動かすと, ズーム率が 1%ずつ変動し, 外側の部分でペンを動かすとズーム率が

10%ずつ変動する. また, 図 B.3 b は PopiePoint のページを選択するためのものである.



a) b)

図 B.3: オブジェクトのズームとページ選択のためのフローメニュー

B.3.4 カラー選択のためのフローメニュー

図 B.4 はオブジェクトの色を選択するためのフローメニューである. フローメニュー内側の色は透明度 0 の色であり, 外側の色はペンをフローメニューの外側に移動するほど, 透明度が高くなるようになっている.

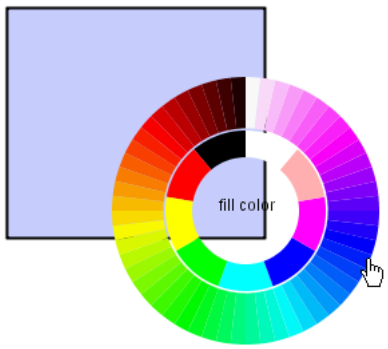


図 B.4: 色を選択するためのフローメニュー

B.3.5 フローメニューにおけるメニュー選択

フローメニューは, 一般的に良く使われるポップアップメニューとは異なり, ペンのストロークによってメニュー選択が行われる. フローメニューでメニューを選択する例を図 B.5 に示す. この例では, 矩形のオブジェクトを選択し複製している. まず, 矩形のオブジェクトの上でフ

ローメニューを表示し, 左下側のオクタントにペンを移動させて, サブメニューを表示する. さらに, そのままペンをレストエリアに戻すことで, 複製が実行される.

このように, フローメニューではメニューを選択後に, 連続したストロークによってオブジェクトの位置や, プロパティなどを変更することができる.

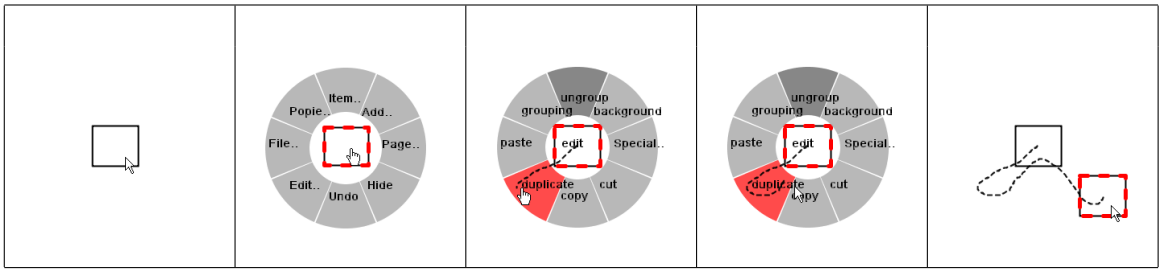


図 B.5: フローメニューでオブジェクトを複製する例

B.4 メニュー項目

ここでは, 各メニューアイテムの機能について説明する. フローメニューに表示されるメニュー項目のうち, “〜..” と表示されるメニュー項目は, サブメニューを持っていることを示している.

B.4.1 Item.. メニュー

オブジェクトのプロパティ等の設定をするメニューを配置している.

move メニュー
: オブジェクトを移動する. メニューを選択後, 連続したストロークでオブジェクトの位置を指定する.

zoom, rotate メニュー
: オブジェクトのズーム率, 回転角度を変更する. 連続値選択のフローメニューよってズーム率や回転角度を選択する.

fill, line メニュー
: オブジェクトの塗り色, 線の色を変更する. カラーのフローメニューよって色を選択する.

save メニュー
: 選択されたオブジェクトだけをファイルに保存する. “File..” メニューの “add obj” メニューにより, 保存したオブジェクトを読み込むことができる. ファイルの拡張子は “ppo (PopiePointObject)” である.

regist メニュー

文字列オブジェクトにのみ有効なメニュー. Popie において読みが登録されておらず, 普通の入力では読みが一致しない単語を登録するためのメニュー. まず, 登録する単語の文字列オブジェクトを作成し, その読みのひらがなだけの文字列オブジェクトを別に用意する. 登録する単語の文字列オブジェクトを選択し, このメニューを実行したあと, そのまま登録したい読みの文字列を持つオブジェクトに重ねる. 読みの文字列のオブジェクトが赤色に変化したらペンを離し, 登録を完了する.

image メニュー

文字列オブジェクトにのみ有効なメニュー. 事前に登録をする必要があるが, このメニューを選択することで文字列に関連する画像を取り出す.

B.4.2 Add.. メニュー

図形オブジェクト, 画像等を挿入するためのメニューである.

line, ■, ▲, ●, Person メニュー

順に, 直線, 矩形, 三角形, 楕円, 人型オブジェクトを挿入する. フローメニューを表示させた位置を基点とした図形オブジェクトを作成する. メニューの選択後, 連続したストロークでオブジェクトの大きさを指定する.

image メニュー

画像ファイルを PopiePoint に貼り付ける. 画像オブジェクトとして貼り付けられる画像ファイルの形式は, JPEG, GIF, PNG の 3 種類である. この実装には Java の javax.imageIO を利用している.

menu icon メニュー

このオブジェクトは, 見た目は楕円オブジェクトと同じであるが, 特殊なオブジェクトである. このオブジェクトを配置し, その上にペンを置くと, 一定時間を待たずにフローメニューが表示されるようになる. プレゼンテーションでページを移動する時など, フローメニューの出現を待てない場合などに利用する. このオブジェクトはファイルには書き込まれない.

B.4.3 Page.. メニュー

ページの追加や削除, ページの移動等を行うメニューである.

next, prev メニュー

ページを次のページ, 前のページに変更する. 最後のページで next を選択した場合は最初のページへ, 最初のページで prev を選択した場合は最魚のページへ移動する.

select メニュー

ページを連続的に変更するメニュー. メニューの選択後, 連続値選択のフローメニューによって, ページを変更する.

new メニュー

新しいページを, 現在表示されているページの次のページに挿入する. 背景は現在表示されている背景が登録される.

move メニュー

ページ全体のオブジェクトを一度に動かす. メニューの選択後, ページ全体の位置を決定する.

zoom メニュー

ページ全体のズーム率を変更する. メニューの選択後, 連続値選択のフローメニューによりズーム率を選択する.

fullscreen メニュー

フルスクリーンモードと通常モードに切り替えを行う. デュアルディスプレイの場合は, プライマリのディスプレイに対してフルスクリーンになる. フルスクリーンモード中に実行すると, 普通のモードに戻る. また, エスケープキー[3]を押す事でもフルスクリーンモードから抜けることができる.

edit.. メニュー

ページ全体をコピーしたり, 切り取ったり, 貼り付けたりするためのメニュー. 貼り付けを行うと, 現在表示されているページの次のページに挿入される. 切り取ったページを別の背景のページで貼り付けを行ってもの背景は変更されない.

B.4.4 Hide メニュー

オブジェクトを隠したり表示したりする. プレゼンテーションの機能として, アニメーションなどの効果を持っていないため, 間に合わせで作った機能である. オブジェクトを隠した場所を覚えていないと表示することができない.

B.4.5 Undo メニュー

オブジェクトの追加や削除, プロパティの変更などのアンドゥ・リドゥをする. メニューの選択後, 連続値選択のフローメニューにより, 操作を戻したり進めたりすることができる. アン

[3]通常, タブレット PC などではキーボードがついていないので, キーの入力はできないが, タブレットボタンなどに個別に設定するキーに対して動作を行えるように, いくつかキーによる操作が行えるようになっている. 例えば, タブレットボタンには, Page Up と Page Down のボタンがついているが, これによってページの切り替えを行うこともできる.

ドゥリドゥの記録はページごとに管理されており, 記録数はメモリの許す限り全てを記録している.

一度戻って新しい操作を行った場合は, 古い操作を再生しないが, データの実装としては, アンドゥ・リドゥ操作そのものも記録しているため, すべての操作を順に再生するような実装に変更するのは容易である.

ただし, 論文執筆時点では若干のバグが残っている機能である.

B.4.6 Edit.. メニュー

オブジェクトの複製や, 背景の編集モードの切り替え, オブジェクトのグループ化等のメニューを配置している.

duplicate メニュー
: オブジェクトを複製する. メニューを選択後, 連続したストロークで, 複製したオブジェクトの位置を決定する.

copy, cut, paste メニュー
: オブジェクトをコピー, カット, ペーストする. ペーストに関しては, 複製と同様, 連続したストロークで, 貼り付けたオブジェクトの位置を決定する.

background/foreground メニュー
: 背景ページの編集モードと, 通常ページの編集モードを切り替えるメニュー. 背景ページの編集モードでは, ウィンドウの端に破線を描画し, 背景ページの編集ページであることを示している.

grouping メニュー
: 複数のオブジェクトをグループ化する. オブジェクトは階層的にグループになっていても構わない. メニューの選択後, ペンのストロークによって選択範囲を決定する. 選択範囲は青色で塗りつぶされ, オブジェクトと少しでも重なっていた場合には選択対象になる.

ungroup メニュー
: グループ化されたオブジェクトの, グループ化を解除する. グループ化されたオブジェクトを選択した時だけ使うことができる.

Special.. メニュー

clear all メニュー
: 現在のページにあるオブジェクトを全て削除する.

flushleft メニュー
: グループ化されているオブジェクトの左位置をそろえる.

draw color メニュー

描画モードでのデフォルトの線の色を変更する. メニュー選択後, カラーのフローメニューにより色を選択する.

menu.. メニュー

zoom メニュー

メニューの大きさを変更する.

demo mode メニュー

メニューの選択中にもペンの軌跡を表示するモードの切り替えを行う. 起動オプション -demo と同等の機能

captuer メニュー

このメニューを選択すると, フローメニューが初期状態に戻る. そこで, 自由にメニューを選択した後ペンを離すと, ペンを離す瞬間のフローメニューの状態をキャプチャすることができる. キャプチャされた画像その場で PopiePoint に貼り付けられ, ファイルとして screenshot フォルダに保存される. screenshot フォルダがない場合は作成される.

pict story メニュー

capture メニューの拡張機能であり, フローメニューの状態が切り替わるごとにフローメニューの状態をキャプチャしていく. キャプチャされた画像は, 横並びに PopiePoint に貼り付けられ, ファイルは screenshot フォルダに保存される.

B.4.7 File.. メニュー

save メニュー

PopiePoint で編集しているページ群全体をファイルに書き出す. 書き出すファイルの拡張子は “pop (PopiePoint Project)” である.

open メニュー

書き出されたファイルを読み込む.

save image メニュー

現在のページの画像ファイルを PNG 形式で出力する. 出力されるファイルは, YYYY-MM-dd-HH-mm-ss.PNG のように, 日付と時刻をファイル名としたファイルに書き出される. 書き出される画像はウィンドウのクライアント領域の大きさに等しい.

add obj メニュー

拡張子 “ppo (PopiePoint Object)” のファイルを読み込み, 現在のページに読み込まれたオブジェクトを追加する.

about, quit メニュー

PopiePoint に関する情報を表示する. PopiePoint を終了する.

print メニュー

ページを印刷する. 細かい設定をすることはできず, デフォルトで紙 1 ページに対して 6 枚分のページを出力するモードである.

B.4.8 Popie.. メニュー

right, left

Popie の右利き用, 左利き用のインタフェースを表示する. 通常は, 右利き用のメニューが Popie メニューのサブメニューの左上側のオクタントに配置されている. しかし, 一度左利き用のインタフェースを選択すると, 右利きと左利きのメニューの位置が逆転し, 次から Popie モードに素早く移行することができる.

R extend, L extend

通常, フローメニューや Popie では, ペンを離すと操作終了, もしくはキャンセルを意味するが, これらのメニューを選択することで, Popie での入力中にペンを離しても, メニューが消えないようになる. Popie を終了する場合は, 下側オクタントのサブメニューの, 上側オクタントの “終了” メニューを選択する (表 3.4 参照).

optimized

キーの配置を最適化したものが表示される, Popie メニューの左上側オクタントが “right” の場合は右利き用, “left” の場合は左利き用のインタフェースが表示される. 配置は “TNHK-MYSRWA” を採用した (表 4.12 参照).