なめらかなユーザインタフェース

増井 俊之, 水口 充, George Borden, 柏木 宏一
シャープ株式会社 ソフトウェア研究所
E-mail: {masui, mina, flash, kasiwagi}@slab.tnr.sharp.co.jp

概要

近年のワークステーションやパーソナルコンピュータ上では、画面上の表示物を実際の物体と同じようにマウスなどで操作する「直接操作」インタフェースや、検索条件を変更すると即座に検索結果が変化する「動的検索」インタフェースが注目を集めている。また、高速グラフィクスハードウェアにより 3 次元視覚化やズーミングを利用した各種のインタラクション手法が研究されている。これらに共通した特徴は、連続的な操作入力に対し連続的 / リアルタイムの直感的な反応が得られることであり、このようなインタフェースを「なめらかなユーザインタフェース」と定義する。本稿ではなめらかなインタフェースを実現するための各種の手法及び実際に構築されたシステムを紹介し、将来のユーザインタフェースについて議論する。

1 WIMP を越えて

Mac OS, Windows, X Window, NEXTSTEP 等に代表されるパーソナルコンピュータ (PC) やワークステーション (WS) のグラフィカルユーザインタフェース (GUI) のほとんどは、いわゆる WIMP (Window/Icon/Menu/Pointing device) インタフェースにもとづいている。 WIMP インタフェースの歴史は古く、初期の WS において採用されていたものとほぼ同じ手法や装置が現在の PC や WS でも使われている。

WIMP インタフェースはビットマップディスプレイを活用するために登場したものであるが、使いやすさのみを追及して発明されたものではなく、貧弱な計算 /描画速度と使いやすさとの折衷として開発されたものであるため、数々の制約を持っている。例えば、ウィンドウやその表示を任意に拡大 /縮小することができないため頻繁にアイコン化 / ウィンドウ化を繰り返さなければならなかったり、多くの項目からひとつを選択するのに深い階層メニューを使わなければならなかったり、以前の状態に戻ることが簡単にできなかったり、直感的に意味をつかみにくいアイコンを多用しなければならなかったりすることが多い。

プロセッサ速度や描画速度が向上して各種の新しい入出力装置が開発されている現在、 WIMP インタフェースの制約にとらわれない新しいインタフェース手法が可能になってきている。 WIMP インタフェースの限界については広く認識されつつあり、これを越える各種の新しいインタラクション手法が提案されているが、我々は WIMP を越えたインタラクション方式として「なめらかなインタフェース」という考え方を提唱している。

2 なめらかなインタフェース

2.1 なめらかなインタフェースの定義

「なめらかなインタフェース」とは、直接操作 /動的検索 /視覚化のような各種のインタラクション技術を統合したものである。我々は以下のような性質をもつインタラクション手法を「なめらかなインタフェース」と定義する。

連続性 ユーザの操作に対しシステムがリアルタイムに連続的に反応する。ユーザが小さな操作を行なったときはシステムの状態も大きく変化しない。アイコンをクリックすると突然ウィンドウが出現したり、メニュー操作により突然項目が出現したり消えたりするようなインタフェースは非連続的であるためなめらかではない。また処理の実行を指令するキーやボタンを持つインタフェースは、その前後の状態が連続的でないためなめらかではない。

可逆性 ユーザが逆の操作を行なったときシステムが前の状態に戻る。ウィンドウやアイコンなどのドラッグ操作は可逆的であることが多いが、ウィンドウを開く操作と閉じる操作が異なるインタフェースは非可逆的であるためなめらかではない。

直接性 ファイルを指定する場合やヘルプ機能を使う場合、キーボードから文字列を入力するのが一般的であるが、このような指定は間接的でありなめらかではない。

直感性 多数の機能のあるアプリケーションでは、何を意味するのか簡単には判別できないアイコンが多用されていることがあるが、このようなものは直感的でないためなめらかではない。

上記の各条件は独立したものではない。例えば連続的であれば可逆的にしやすいし、直接的であれば直感的であることが多いと考えられる。

連続性はなめらかなインタフェースにおいて最も重要な性質である。ユーザが操作を行なったときにシステムがす

"Smooth" User Interface.
Toshiyuki Masui, Mitsuru Minakuchi, George Borden, Kouichi Kashiwagi.
SHARP Corporation.

ぐに反応すればユーザは操作の効果について理解しやすいし、実生活上の作業においては人間の操作に対し対象が即座に反応することがほとんどなので直感的である。

システムが連続的に動作可能なときは可逆的にすることも容易であることが多い。WIMP インタフェースにおいても、スライダの操作やアイコンのドラッグ操作などは連続的かつ可逆的であるためわかりやすいといえる。

Shneiderman は、検索条件を変化させたときに実時間で検索結果も変化するような検索システムを「動的検索 (Dynamic Query) システム」と呼んでその有効性について述べたが [16]、動的検索は検索をなめらかに連続的に行なうための手法であると考えることができる。

直接性や直感性はいわゆる「アフォーダンス」[10] として重要さが知られている。Shneiderman は、GUI においてアフォーダンスを向上させるために「直接操作」(Dynamic Manipulation) が重要であることを述べている [15]。

日常的に使う機械はなめらかなインタフェースを持っているものが多い[1]。例えば水量を蛇口で調整するような場合、システム (水道) はユーザの操作 (ノブの回転) に対しリアルタイムに連続的に反応するし、逆の操作を行なえば簡単にもとの状態に戻る。ノブを回すという操作はネジと同様に一般的であるため比較的直感的である。

2.2 なめらかなインタフェースの効用

なめらかなインタフェースの採用により各種の操作を直感的に行なうことができるようになるだけでなく、WIMP インタフェースにおける無駄な機能の多くが不用になるという効用もある。

重なりあうウィンドウ　ウィンドウはもともと沢山の情報をひとつの画面に表示するための苦肉の策として考えられたものであるから、3 次元視覚化手法などにより沢山のデータをわかりやすく表示することができればウィンドウやその操作メニューは不用になる。

ファイルやディレクトリ　データを保存するための単位としてファイルやディレクトリが使われることが多いが、ディスク装置を使っていることを意識しなければならないことが多く面倒である。例えばファイルを修正するときは一般に「セーブ」操作が必要になるが、このような作業はデータの修正作業とは本質的に関係がない。また、多くのファイルを管理するために階層構造のディレクトリ構造が使われることが多いが、どこに格納したかわからなくなることが多いし、適切な名前付けが必要であったり、多くの面倒な操作がつきまとう。データをなめらかに格納 /分類 /検索することができれば、ユーザがこのような作業にわずらわされる必要はなくなる。

アイコン　アイコンはファイルやアプリケーションを特定するために使用されることが多いが、項目を選択するための直感的な手段が他にあればその多くは不用になるはずである。

メニュー　メニューは項目を大きなリストから選択するのによく使われているが、リストから項目を選択する手法としてはもっと直感的手法が存在するため、メニューは必ずしも最適ではない。

2.3 なめらかなインタフェースの実現

2.1節で述べたようなインタフェースを実現するには、具体的には以下のような手法を採用すればよい。

処理の高速化と実時間実行　高速計算 /表示アルゴリズムを使用し、条件変化に対応してリアルタイムに処理実行や表示の更新が行なわれるようにする。

情報視覚化の工夫　表示の更新を高速に行なうため、3 次元表示を用いて自分に近いところのみ表示するようにしたり、Fisheye Views[4] や Fractal View[5][18] のような手法を用いて重要な部分のみ表示を行なうようにする。

入力手法 /操作手法の工夫　移動やズーミングのような連続的で直感的な操作方法を採用し、そのための入力装置を工夫する。

モードの除去　なるべく状態切換の必要がない仕様とする。

2.4 なめらかなインタフェースと情報検索

低価格な大容量記憶装置やインターネットなどの普及により大規模なマルチメディアデータを手軽に扱えるようになってきた現在、必要な情報を簡単に検索する手法の開発が大きな課題となっているが、なめらかなインタフェースは大量の情報検索において特に効果を発揮すると考えられる。

大規模データの検索にはキーワードを用いたテキスト検索手法が従来一般的であり、研究や実用システムの数も多いが、検索に使用されるキーワードがデータ中に含まれているとは限らないし、適切なキーワードを選びにくい場合も多いため、本質的に高い精度 (precision) と再現率 (recall) を得ることはむずかしい。また、文書以外のデータに対して検索キーワードを指定するのは直感的でないし、{ キーワード入力→検索実行指示→検索結果リスト表示 } というサイクルが実時間で行なわれないため連続的でなく、なめらかな操作感が得られない。このような従来のキーワード検索を越えてマルチメディアデータを直感的に効果的に検索するための各種の新しい情報検索手法が提案されている。

新しい検索手法として、情報の効果的視覚化を利用するものが近年特に注目を集めている。データを 3 次元的に表現して遠くの情報を小さく表示することにより多くの情報を一度に提示する手法 [7][12]、データ全体を縮小表示しながら重要な部分のみ局所的に大きく表示する手法 [6][13][14]、重要な情報を効果的に選択表示する手法 [4][5]、視覚化された情報を対話的に拡大表示することにより詳細な情報を表示する手法 [3][11] などが提案されている。

[1]電気機器を除くほとんどの装置はなめらかなインタフェースを持っているので、なめらかでないインタフェースの起源は電気スイッチなのかもしれない。

図 1: WING システム

情報の効果的な視覚化手法により、大量の情報を眺めるようにして必要な情報を探すことが可能になるが、人間は日常もっと効果的な方法で検索を行なっている。例えば人が図書館で本を探す場合、目的の本が含まれる分類の書架で探すこともできるし、本の題名や著者名から索引カードで探すこともできる。また正確な題名や著者を覚えていなくても関連する書架を見渡して思い出すこともあるし、目的の本が見つからなくても関連した本を探し出すことは難しくない。大きさや色を頼りに本を探すことも可能である。このように、日常の検索活動においては、場所の情報 /名前やキーワードの情報 /概念の情報 /形や色の情報といった各種の情報を組みあわせることにより効果的な検索が可能となっている。これを考慮すると、情報の視覚化に加え、キーワード検索 /ハイパーテキスト /情報の分類といった各種の異なる検索手法をうまく組み合わせることで、より強力な情報検索システムを構築することができると考えられる。

我々は以下のような手法を「なめらかなインタフェース」で統合することによりマルチメディア情報を効率的に検索することができると考えている。

大量の情報の視覚化　3 次元視覚化などの効果的な手法を用いて情報を視覚化することにより、直接的な検索や関連情報からの検索を可能にするとともに、情報空間全体の把握を容易にすることができる。

情報の表示を連続的に変化　検索条件を変更すると検索結果の表示が直ちに更新される動的検索の手法により、曖昧な手掛かりでの検索からより詳細な検索を試行錯誤的に行なうことが容易となるし、条件と情報の関係を直感的に理解しやすくなる。また、連続的に表示の拡大 /縮小を行なうことにより全体と詳細の関係を把握しやすくなる。

効果的なキーワード検索　検索キーワードを視覚化することによりキーボードを使わずにキーワード検索を行なうことができる。

効果的な分類検索　複雑な分類構造を効果的に視覚化することにより、必要とする分類を簡単に探すことができる。

曖昧検索　多少あやふやな条件を与えた場合でもある程度の検索を実行する。

なめらかインタフェースによる検索システムの実現例として、 3章において、地図の 3 次元表示を活用した奈良観光ガイドシステム **WING** (Whole Interactive Nara Guide)[2]、4章において、曖昧検索とズーミングを利用した “**ピテカン英和辞書**[3]” を紹介する。

3　奈良観光ガイド WING

地図情報検索システム WING[9][19] は、奈良近辺の名所 /地形 /道路 /建造物などの情報をなめらかな操作で検索するシステムである。図 1に WING の全体の画面を示す。 WING の画面は 4 つのウィンドウから構成されており、左上を地

[2]aka “WIMP Interface No Good” “Weird Idiosyncratic Neurotic Gizmo”

[3]サルでも「ピテカントロプス」をひける辞書

図 2: 上空から見下ろした地図

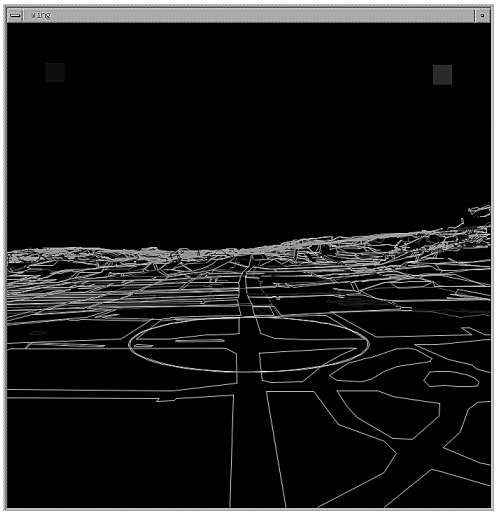

図 3: 地上から見渡した地図

図ウィンドウ、右上をガイドウィンドウ、左下を分類ウィンドウ、右下を索引ウィンドウと呼ぶ。以下にこれらについて詳細に説明する。

3.1 地図ウィンドウ

地図ウィンドウでは奈良近辺の地図が 3 次元的に表示される[4]。ユーザがマウス操作で視点を連続的に移動させることにより、地図の表示を連続的に変更することができる。上空から見下ろすように視点を操作すると普通の地図のような表示になり (図 2)、低い位置から見渡すように視点を操作すると地上を歩いているような表示になる (図 3)。このようにして地図全体の把握や特定の地点からの地形の確認を行なうことができる。

地図ウィンドウの中央部の地図上にはおおよその注目範囲を示す円が描かれている。円の中心に近い位置のデータはユーザの関心度が高いと判断され、関心度に比例した大きさの立方体で表示される。関心度は注視点からの距離の他、関連する項目データの表示状態から算出される。視点と地図の距離および注視点と対象との距離によって関心度を計算することにより、視点を低くすると狭い範囲の情報を、高くすると広い範囲の情報を見ることができる。また関連情報の関心度も考慮しているので、注視点の近くにある、現在着目している項目に関連する情報を探し出すことができる。

立方体で示された項目データに関連する情報は図 1右上のガイドウィンドウに表示される。立方体の色はガイドウィンドウでの表示項目の色に対応しており、ユーザはガイドウィンドウを見ながら視点を操作して必要な項目を探し出すこともできる。地図ウィンドウとガイドウィンドウは連動しているので、フライトシミュレータのように視点を移動させながらガイドウィンドウを見ることで奈良に関する知識を得ることもできる。

3.2 ガイドウィンドウ

右上のガイドウィンドウでは地図ウィンドウでの表示に対応する項目名 /解説文 /画像が表示される。これらの情報は関心度でソートされ上から順に表示される。ガイドウィンドウで表示されている項目をマウスでクリックすると、選択した項目の場所を注視するように地図ウィンドウでの視点が徐々に移動する。視点の移動の過程を表示するので、現在着目している場所と選択した項目との位置関係やそれらの間にある項目を知ることもできる。

3.3 分類ウィンドウ

WING で扱うデータは階層的に分類されている。例えば、「東大寺」は「寺院」クラスに属しており、「寺院」クラスは「旅行 /観光」クラスに属している。階層構造の最上位には「ルート」クラスがあり、その下位クラスとして 7 つの分類クラスがある。最初の状態ではこれらが図 4のように表示されている。

図 4: 分類ウィンドウ

分類ウィンドウは Pad システム [11] と同様にマウスを使って拡大 /縮小表示することにより大きな階層構造を簡単な操作で直感的に扱えるようになっている。ひとつの分類クラスを拡大表示していくとそれに属する下位分類クラスが表

[4]国土地理院発行の「数値地図」および「数値地形図」を使用

示され、上位分類クラスは徐々に表示が消えていく。例えば「旅行 /観光」クラスを拡大していくと表示は図 5のようになり下位分類クラスが表示される。この中には「寺院」クラスがあり (図 6)、「寺院」クラスを更に拡大していくと「寺院」クラスに属するデータ項目が表示され、この中から「東大寺」をみつけることができる (図 7)。

項目に対応する表示をマウスでクリックすると、ガイドウィンドウでの項目の選択と同様に、地図ウィンドウでの視点が移動していく。分類クラスに対応する表示をクリックするとその分類クラスが選択され、その分類に属する項目だけがフィルタリングされ地図ウィンドウやガイドウィンドウで表示される。選択されている分類クラスをもう一度クリックすると選択は解除される。

図 5: 「旅行 / 観光」クラスを拡大

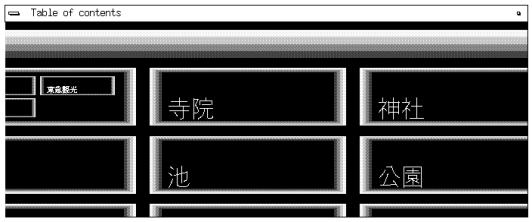

図 6: 「寺院」クラスが現れる

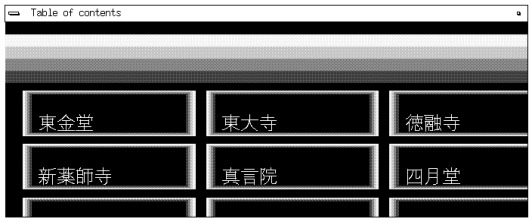

図 7: 「寺院」クラスを拡大

3.4 索引ウィンドウ

索引ウィンドウは項目のリストを辞書順に表示したものであり、ユーザはその中から名前を頼りに検索を行なうことができる。項目の数は多いので、リストから高速に目的の項目を探す手法が必要になる。

多数の項目のリスト中から一つの項目を選択するためのインタフェース手法は各種提案されている。項目数がそれほど多くないときはスクロールバーが使われることが多く、また何万個もの項目を含むリストから一つの項目を選択するのにスライダが使われることもある [1][8]。いずれにおいても、項目の数が非常に多いときは項目の選択に微妙な操作が要求されるため操作が難しいという欠点がある。

WING では索引の項目リストに対しても拡大 /縮小操作を可能とすることにより、大きなリスト中の項目を高速に選択できるようにしている。実際には、リストの拡大率を s、s の整数部分を n $(= \lfloor s \rfloor)$、小数部分を f $(= s - n)$ とすると、項目リストのうち 2^n 個目毎の項目を f 間隔で表示している。ユーザが s の値を連続的に変更することにより、s が大きい時はリスト全体が粗く、s を小さくするとリストの一部分が詳細に表示されるので、地図ウィンドウや分類ウィンドウと同じように、リストの全体から一部分までを連続的に拡大 /縮小する感覚で項目の検索を行なうことができる。

また、項目リストとしては項目名の部分文字列も整列に使用した「置換索引」を使用している。例えば「ホテルフジタ」はリスト中の「ほてる」の位置にも現れるし、「ふじた」の位置にも現れる。このような置換索引により項目名の一部分からでも検索することが可能となり、**grep** コマンドと似た感覚の検索を行なうことができる。

WING の索引ウィンドウでの検索の例を以下に示す。索引ウィンドウの表示は最初は図 8のようになっている。この状態では索引リスト全体の中から 256 個毎の項目が表示されている。それぞれの項目は途中の文字で五十音順に並べられている (図中の白い文字で示されている部分文字列で並べられている)。実際の WING システムの索引ウィンドウでは、拡大率や項目のリスト中での位置により背景色を変えることにより、索引リストのどの辺りをどの程度の間隔で表示されているかがある程度わかるようになっている。

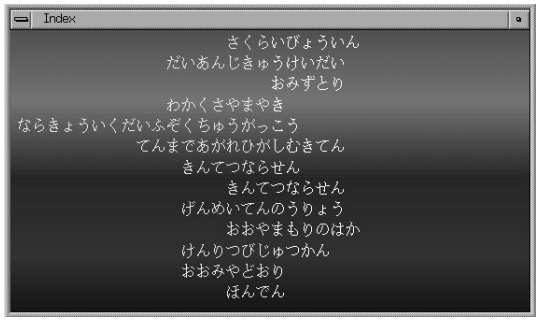

図 8: 索引ウィンドウ初期状態

図 8から「ホテルフジタ」を探す場合、「ほ」は「の」と「ま」の間にあるのでその間をマウスでポインティングし拡大表示の操作を行なう。その様子を図 9〜図 12に示す。

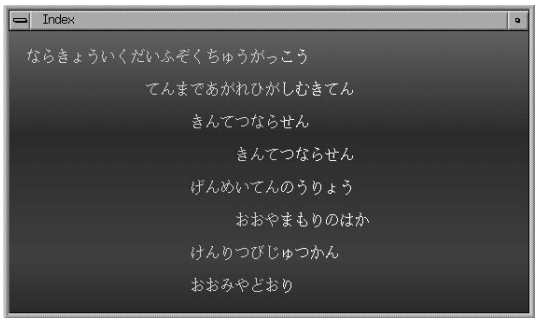

図 9: 索引ウィンドウの拡大

表示をわずかに拡大させた状態が図 9である。図 8に比べそれぞれの項目の表示間隔が広がっている。更に拡大を続け、それぞれの項目の表示間隔が十分広くなるとその間の項目が表示されるようになる (図 10)。この状態ではリスト全体の中から 128 個ごとの項目が表示されている。

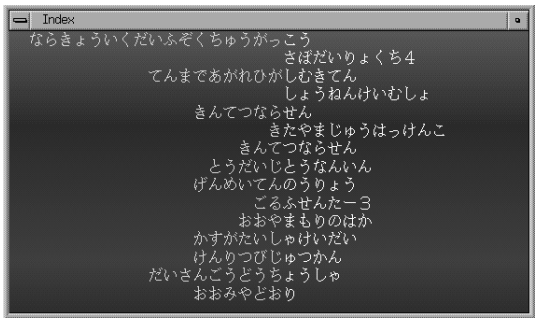

図 10: 索引ウィンドウの拡大

このような操作を続けて図 11、図 12のようにリストを拡大表示することにより目的とする項目を探し出すことができる。

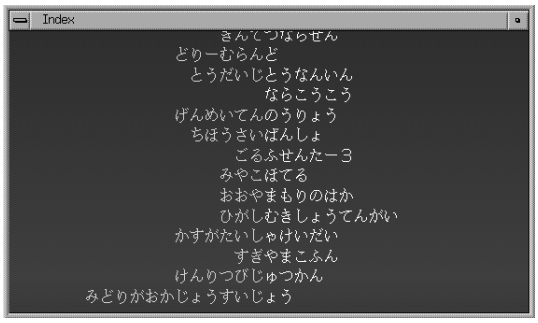

図 11: 索引ウィンドウの拡大

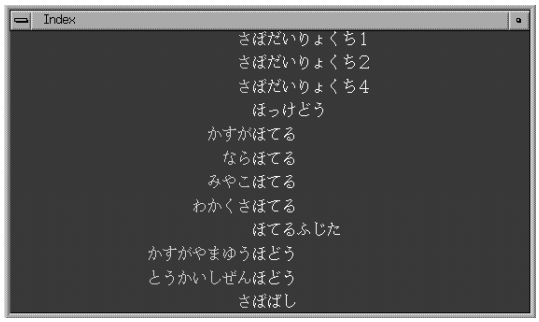

図 12: 索引ウィンドウの拡大

図 12で分かるように「ホテル」という文字列を含む項目は同じ位置に並んでいるので、例えば「ホテルフジタ」を「フジタホテル」と間違えて覚えている場合や「フジタ」という名前すら忘れてしまった場合でも、索引ウィンドウのリストを見ることで正確な名称を思い出して必要な項目を探し出すことができる可能性がある。

ガイドウィンドウや分類ウィンドウの場合と同様に、索引ウィンドウで表示されている項目をマウスでクリックすると地図ウィンドウの視点が移動して選択した項目が地図ウィンドウの中央に位置するようになる。

WING では約 600 個の項目データから生成された約 5700 個の項目の置換索引を使用しており、数秒程度で必要な項目を探し出すことができる。 1 万件の映画タイトルから検索を行なう実験を行なったところ、平均 10 秒程度で検索を行なうことができ、 AlphaSlider[1]、 FineSlider[8] でのスライダによる検索よりも高速であった。

3.5 WING の使用例

WING では従来のシステムでは難しかったような情報の検索を行なうことができる。奈良観光を例に WING での検索の方法を示す。例えば次のような状況で「正倉院展」を見に行こうとしたとする。

- 正倉院展が奈良の博物館で開催されているのは覚えているが博物館の正式な名称は知らない。
- 博物館の正確な場所は知らないが奈良公園の中にあることは知っている。
- 正倉院展を見た後で東大寺も参拝したいが、その他の近くにある寺院もできるだけ見て回りたい。
- 博物館の周辺で昼食をとる場所も探しておきたい。

このような漠然とした知識しかない状況でも以下のようにして WING を使って観光計画を立てることができる。

博物館を探す　博物館が奈良公園の中にあることを知っており、奈良公園の大体の位置も知っていれば、地図ウィンドウを操作して奈良公園近辺を見て探すことができる。更に、博物館は「文化」の分類に含まれていると予想されるので分類ウィンドウで「文化」の分類を選択しておけば地

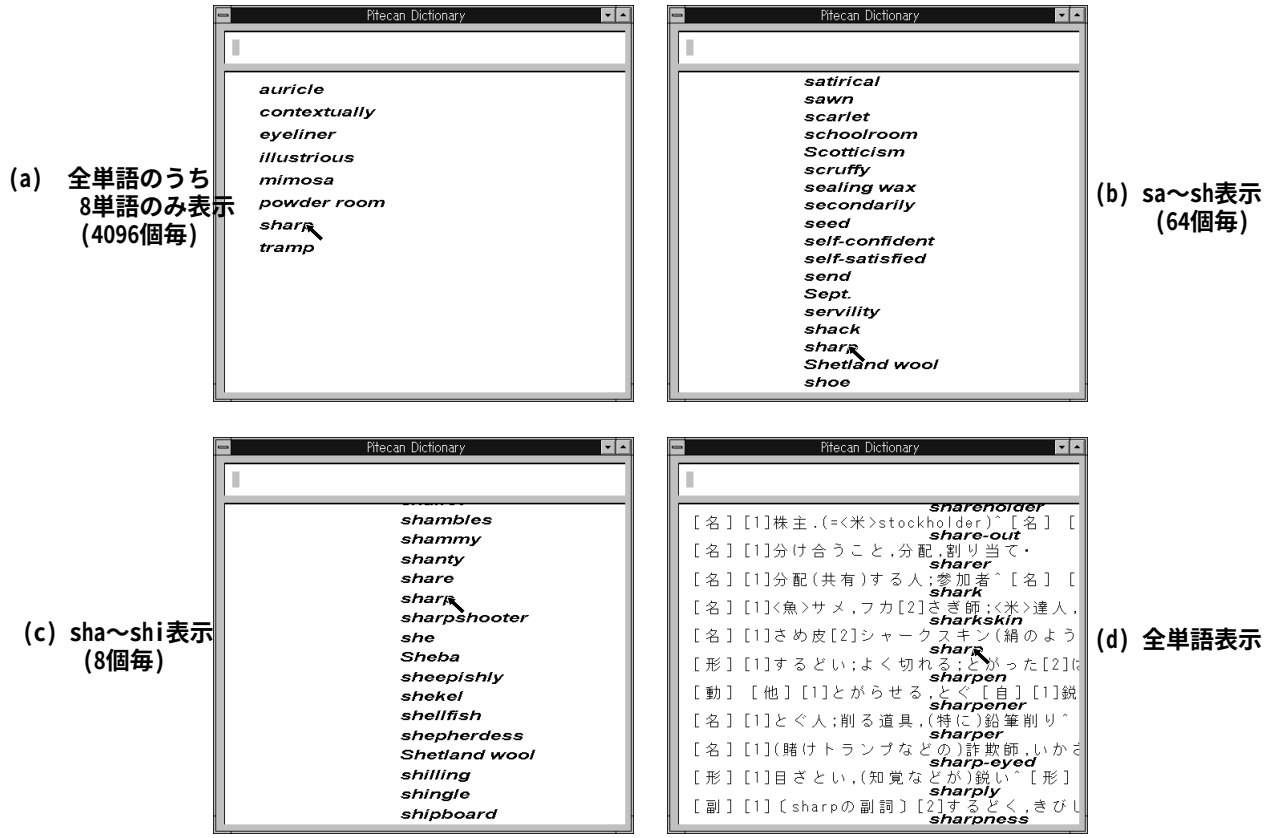

図 13: ズーミングによるピテカン辞書検索

図ウィンドウやガイドウィンドウに表示される項目をフィルタリングできるので更に容易に博物館を見つけることができるし、「文化」の分類をさらに拡大して探すこともできる。また、索引ウィンドウで「はくぶつかん」で探せば正式名称が「奈良国立博物館」であることが分かり、それをマウスでクリックして地図上の位置を知ることもできる。

博物館の近くの寺や神社を探す　次に東大寺を分類ウィンドウや索引ウィンドウを使って探してマウスでクリックすると地図ウィンドウでの視点が現在着目している博物館から東大寺へ移動していくので、博物館と東大寺の位置関係を把握することができる。また、地図ウィンドウで視点を遠ざけたり動き回るように操作することで博物館の近くに春日大社や興福寺などがあることも分かる。

昼食をとる場所を探す　分類ウィンドウで「食事」を選択すると食事のできる場所のみが表示されるようになる。地図ウィンドウで博物館を注視したまま、視点を遠ざけるように操作すると博物館を中心として近い順に食事のできる場所が表示されてくる。

このように、WING を使用すると曖昧な手掛かりからでも必要な情報を得ることができる。また、検索の過程が見えるので探している場所の近辺にある項目を知ることもできるし、地図ウィンドウを操作することでガイドブックを見るようにして奈良の情報を得ることもできる。

4　ピテカン辞書

WING の索引ウィンドウで使われているズーミングを用いた辞書検索手法は自然言語辞書の検索に応用することができる。3 万語程度の辞書に適用した場合、平均 8 秒程度で単語の検索を行なうことができるが、キーボードが使える環境ではキーボードを用いた方が高速に検索を行なえる場合が多い。「ピテカン辞書」は、キーボードによる検索とズーミングによる検索を融合したなめらかな辞書検索システムである。

4.1　ピテカン辞書使用例

ピテカン辞書とその検索の様子を図 13に示す。画面は単語表示領域と検索パタン指定領域に分かれている。図 13では検索パタンが何も指定されておらず、 WING の検索ウィンドウと同じ手法により、全単語のリストから目的の単語を検索することができる。図 13(a) のマウスカーソル位置 (文字列 “sharp” のすぐ右側) でマウスボタンを押し始めてマウスを右に動かしていくことにより、 WING の索引ウィンドウと同様に (b), (c) のように表示が拡大されて最後に (d) のように和訳が表示される。表示状態は現在のマウスの位置と操作開始時 (マウスボタンを押しはじめた時) のマウスの位置との差により決定されるため、 (b), (c), (d) のように表示を変化させた後でマウスをまた (a) の位置に戻した場合、表示も全く同じ状態に回復する。 AlphaSlider[1] などの場合

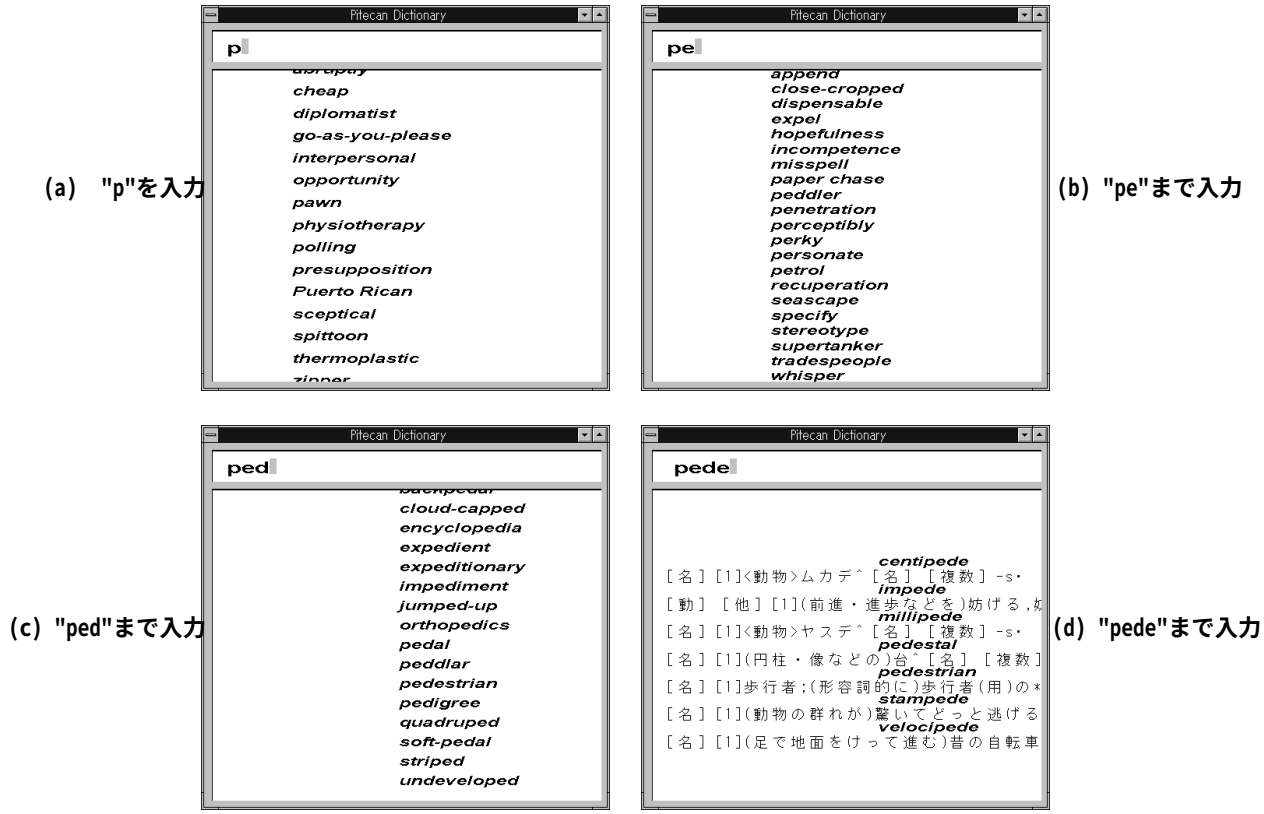

図 14: ピテカン辞書のパタン検索

と異なり検索操作が可逆的であるため、間違った操作をしてもその前の状態に復帰して検索をやり直すことが簡単にできるし、マウスボタンを押しながら移動させるとき常にカーソルの下には同じ単語が表示されているため、直接操作感が得られる。

パタンをユーザが指定して検索を行なう場合を図 14に示す。画面上側のテキスト入力枠でパタンを指定する。図 14(a) は、キーボードからパタンとして "␣**p**" を入力したところである。ピテカン辞書では空白文字は任意の文字列 (**grep** 等の正規表現における "**.***" に相当) として扱われるので、この場合 "**p**" を含む全ての単語のリストの一部が表示されている。図 14(b) は、キーボードからパタンとして "␣**pe**" まで指定したところである。ここでは "**pe**" を含む全ての単語のリストの一部が表示されている。ピテカン辞書でマウスを使って検索を行なう場合、単語リストが右に移動した場合ほど単語リストの拡大率が大きくなるので、パタン検索の場合も同様に、表示に必要な拡大率に応じて単語リストを左右に移動して表示するようにしている。図 14(a) の状態では拡大率が小さいので単語リストは画面の左の方に表示されている。

ピテカン辞書は動的検索を採用しており、パタンを指定したとき (キーボードで文字をタイプしたとき) パタンマッチはリアルタイムに行なわれ、即座に表示が更新される。例えば (a) の状態において "**e**" をタイプした瞬間に表示は (b) のように変化する。

"␣**pede**" まで入力した時点で表示は図 14(d) のようになる。 "**pede**" を含む単語は辞書中に 7 個しか存在しないのでそれら全てが和訳とともに表示されている[5]。

"**p**" を含む単語は沢山あるので図 14(a) の状態ではマッチする単語リストの一部のみが表示されているが、この状態からマウスを使用してズーミングによりさらに検索を行なうことも可能である。

4.2 ピテカン辞書による曖昧検索

前述のように、ピテカン辞書の検索パタン指定においては空白文字は任意の文字列として扱われるので、単語中の文字の一部だけ指定しても検索を行なうことができる。この様子を図 15に示す。図 15は、「ピテカントロプス」を検索しようと考えたユーザが、正しい綴りがわからないので、確かに含まれていると思われる文字のみ (ここでは "**p**", "**t**", "**n**", "**t**", "**s**") 入力した状態を示している。このように、ピテカン辞書では、検索したい単語の一部のみ指定しても目的の単語を見つけることが可能である。

また、ピテカン辞書ではパタンが間違っている場合でも最もパタンに近い単語を検索し表示する。この例を図 16に示す。図 16は「ピテカントロプス」を検索しようと考えたユーザが、誤ったパタン "**pitekan**" を指定したところである。このようなパタンを含む単語は辞書に無いため、ピテカン辞書は新たに 1 文字のミスマッチを許す検索を実行する。それでもマッチする単語が存在しないため、さらに 2 文

[5] "**ped**" は「足」を意味する。 *ped*al(ペダル), tri*pod*(三脚) 等も同語源。

図 15: ピテカン辞書の曖昧検索

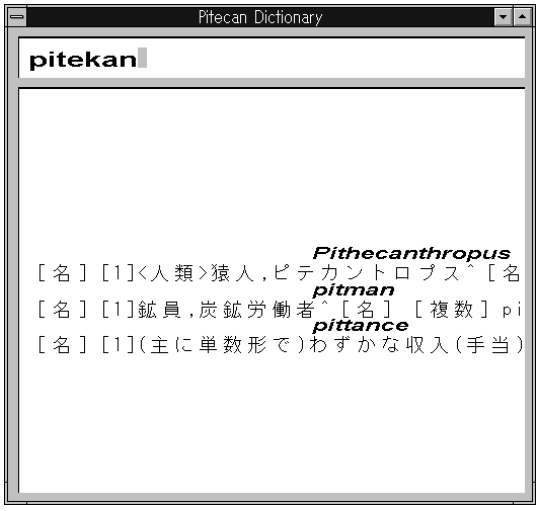

図 16: 誤りを含むパタンの検索

字のミスマッチを許す検索を実行した結果、図 16のような 3 個の単語が候補として表示される。以上の処理は自動的に瞬時に実行されるため、ユーザは多少の綴りの間違いを気にすることなくパタンを入力することにより辞書検索を行なうことができる。

4.3 ピテカン辞書の実装

前述のような曖昧検索を高速に行なうため、ピテカン辞書では 2 種類の正規表現認識アルゴリズムを併用している。

正規表現 “**ab.*ca**” (ピテカン辞書では “**ab␣ca**” と表現) を認識する状態遷移機械は図 17のように表現できる。

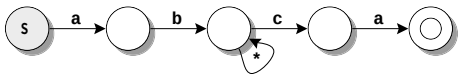

図 17: パタン “**ab.*ca**” を受理する状態遷移機械

図 17の状態遷移機械は、図 18のように状態数を増やすことにより、ミスマッチ (誤字 /脱字 /誤挿入) を許す機械に拡張することができる。 A0 はミスマッチを許さない受理状態で、A1/A2 は 1 文字 /2 文字誤りを許す受理状態である。

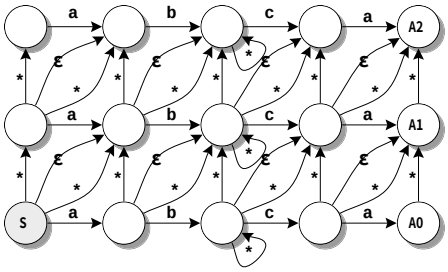

図 18: 誤字 /脱字を許容する状態遷移機械

図 18の状態遷移機械が “**abracadabra**” を認識する様子を図 19に示す。 “**ab**” まで読み込んだ状態で 2 文字誤りを検出し、 “**abra**” まで読み込んだ状態で 1 文字誤りを検出していることがわかる。

図 18のような非決定性状態遷移機械を計算機上に実現するために、ピテカン辞書では以下の 2 種類のアルゴリズムを併用している。

1. 決定性状態遷移機械に変換して状態遷移表を作成

   これは UNIX の**egrep** コマンドで用いられている手法であり、 (状態数 × 入力文字種) の大きさの状態遷移表を用いて現在の状態と入力の組から次の状態を計算する。遷移計算は高速であるが、非決定性機械の状態数が多くなると遷移表の作成に大きな時間がかかり、遷移表も大きくなる。
2. 非決定性状態遷移機械の状態をビットマップ表現し、シフト演算により状態遷移を計算

   例えば図 19の上からふたつめの状態を 3 個のレジスタによって**01110**, **11100**, **11000** と表現する。次の状態は右シフト演算及びマスク演算により計算する。パタンマッチのための前処理計算は少なくてすむが、遷移計算はアルゴリズム 1 より低速である。

   シフト演算を用いてパタンマッチを行なう手法は Baeza-Yates の “Shifter Algorithm” として知られており [2]、曖昧**grep** プログラム**agrep**[17] において一部使用されており、またハードウェアによる実現も存在する [20]。

ピテカン辞書では、曖昧度が低い場合はアルゴリズム 1, 高い場合 (3 個以上の誤りを許す場合) はアルゴリズム 2 を使用している。

4.4 ピテカン辞書の評価

辞書のような離散的なデータの検索を連続的に実行することはむずかしいが、ピテカン辞書ではズーミングと動的曖昧検索を組みあわせることにより連続性 /可逆性を実現している。また、単語をドラッグしてリストのズーミングと移動を行なうことにより直接性を実現している。左右の移動によりズーミングを行なうという方式はあまり直感的とはいえないが、おおむねなめらかな操作を実現することができたと考えている。なめらかなインタフェースの採用により検索実行ボタンなどが不要になっていることも重要である。

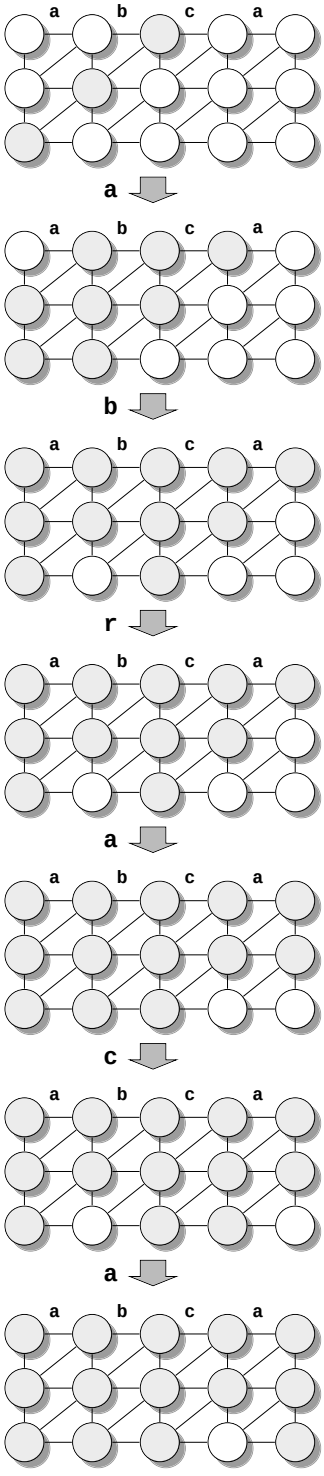

図 19: “**abracadabra**” に対する状態遷移

5 なめらかなインタフェースの将来

本稿ではなめらかなインタフェースの実例をふたつ紹介したが、今後各種のなめらかなインタフェースを持つシステムがますます増えてくると考えられる。実生活では現実の物体を対象とするほとんどの作業がなめらかに行なわれているにもかかわらず、計算機上の操作では非連続的 /非可逆的ななめらかでない操作が何故か許容されており、そのような使いにくさに鈍感な人間だけが機械を活用できるという不思議な状況になっている。このような状況はどんどん変化していくと考えられる。

なめらかな操作が阻害されている最も大きな原因は計算速度や記憶容量などの資源不足であるが、資源の問題はいずれ解決する可能性があるので、インタフェース設計者の考え方をまず変革しておく必要があると考える。つまり現在のインタフェースの中でなめらかでない部分に着目し、将来の対策について考えておくことが重要であろう。例えば、現在最も問題のある非可逆的操作は文書やファイルの修正 /削除ではないかと考えられるが、一度作成した文書やファイルは絶対に消さないようなファイルシステムやエディタを使うことにより、連続的に過去の状態に復帰することが可能になるであろう。今後さらに大容量の記憶装置が普及すると考えられるし、人間の入力速度はたかが知れており 1 年間キーボードを 1 秒に 10 回タイプし続けたとしても入力できるのは高々 300MB であるから全操作履歴を保存しておくことにしたり、 SCCS や RCS のように差分を使ったりすることにより「なめらか undo」が可能となるであろう。このように、現在の計算資源ではむずかしくても将来解決するであろう問題に対してなめらかさへの対応の検討をしておくことが重要であると考えられる。また計算資源の問題がなかなか解決しない場合のために各種の実装の工夫が有用であろう。

2節で述べたように、新しいハードウェアやソフトウェアを使った各種の新しいインタラクション手法が続々と提案されているが、新しい技術を開発しつつ既存の技法をうまく組み合わせることにより、よりなめらかなインタフェースを実現することができるであろう。

6 結論

本稿では、将来のインタフェース手法の考え方として有望な「なめらかなインタフェース」という統一的な概念を提唱し、その実現例をふたつ紹介した。なめらかなインタフェースを実現するための新しい技術を順次開発し各種の機器に適用していく予定である。

参考文献

[1] Christopher Ahlberg and Ben Shneiderman. AlphaSlider: A compact and rapid selector. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, pp. 365–371. Addison-Wesley, April 1994.

[2] Ricardo A. Baeza-Yates and Gaston H. Gonnet. A new approach to text searching. *Communications of the ACM*, Vol. 35, No. 10, pp. 74–82, October 1992.

[3] Benjamin B. Bederson and James D. Hollan. Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*, pp. 17–26. ACM Press, November 1994. ftp://ftp.cs.unm.edu/pub/pad++/pad-uist94.ps.gz.

[4] G. W. Furnas. Generalized fisheye views. In *Proceedings of the CHI'86 Conference on Human Factors in Computing Systems and Graphic Interfaces*, pp. 16–23, Boston, May 1986. Addison-Wesley.

[5] Hideki Koike. Fractal Views: A fractal-based method for controlling information display. *ACM Transactions on Office Information Systems*, Vol. 13, No. 3, pp. 305–323, 1995. http://www.vogue.is.uec.ac.jp/papers/fv.ps.

[6] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 2, pp. 126–160, June 1994.

[7] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp. 173–179. Addison-Wesley, April 1991.

[8] Toshiyuki Masui, Kouichi Kashiwagi, and George R. Borden. Elastic graphical interfaces for precise data manipulation. In *CHI'95 Conference Companion*, pp. 143–144. Addison-Wesley, May 1995. http://www.acm.org/sigchi/chi95/Electronic/documnts/intpost/tm_bdy.htm.

[9] Toshiyuki Masui, Mitsuru Minakuchi, George R. Borden IV, and Kouichi Kashiwagi. Multiple-view approach for smooth information retrieval. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'95)*, pp. 199–206. ACM Press, November 1995.

[10] Donald A. Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988. 邦訳: 「誰のためのデザイン?」 (新曜社).

[11] Ken Perlin and David Fox. Pad: An alternative approach to the computer interface. In *ACM SIGGRAPH'93 Conference Proceedings*, pp. 57–64, August 1993.

[12] George Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp. 189–194. Addison-Wesley, April 1991.

[13] Manojit Sarkar and Mark H. Brown. Graphical fisheye views. *Communications of the ACM*, Vol. 37, No. 12, pp. 73–83, December 1994. http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-084a.html.

[14] Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill Dill, Shelli Dubs, and Mark Roseman. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Information Systems*, 1995. http://www.cpsc.ucalgary.ca/projects/grouplab/papers/Visualization.TOIS.ps.

[15] Ben Shneiderman. Direct manipulation: A step beyound programming languages. *IEEE Computer*, Vol. 16, No. 8, pp. 57–69, 1983.

[16] Ben Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, Vol. 11, No. 6, pp. 70–77, November 1994.

[17] Sun Wu and Udi Manber. Agrep - a fast approximate pattern-matching tool. In *Proceedings of USENIX Technical Conference*, pp. 153–162, San Francisco, CA, January 1992. ftp://cs.arizona.edu/agrep.

[18] 小池英樹, 石井威望. フラクタルの概念に基づく提示情報量制御手法. 情報処理学会論文誌, Vol. 33, No. 2, pp. 101–109, 1992.

[19] 水口充, 増井俊之, ボーデンジョージ, 柏木宏一. なめらかなユーザインタフェースによる地図情報検索システム. 田中二郎（編）, インタラクティブシステムとソフトウェア III: 日本ソフトウェア科学会 WISS'95, pp. 231–240. 近代科学社, 1995.

[20] 山田八郎, 高橋恒介, 平田雅規, 永井肇. あいまい検索が可能な文字列検索 LSI. 日経エレクトロニクス, No. 422, pp. 165–181, 1987.6.1.