コンピュータ画面情報の音化について

平成 13 年 2 月 5 日

情報電子工学科 竹野研究室
鈴木 義利

目 次

概要

コンピュータ画面に現われる情報を音化することで、視覚障害者もその画面情報を確認できるのではと考え考察してみることにした。いろいろな画面情報の中で、点字の読み上げ、コンピュータ上で文書を書くときに用いられるテキストカーソルの音化、マウスカーソルの音化の 3 つについて考えてみた。これらついて、その実現方法と可能性を考察したが、マウスカーソルの音化に関しては、その他に、実際にプログラムを用いてコンピュータ上で実験を行った。その結果、マウスカーソルの位置特定の他にも、単純な音で画面上に配置されている図形を特定することも可能になるかもしれないということがわかった。本稿では、ここに挙げた画面情報を音化するための方法について考察する。

1　はじめに

コンピュータのユーザインタフェースには大きく分けて 2 つある。それは、グラフィカル・ユーザ・インタフェース (GUI) と、キャラクター・ユーザ・インタフェース (CUI) である。グラフィカル・ユーザ・インタフェース (GUI) とは、いろいろな情報を簡単な絵や図形のように視覚的に表し、これを用いた OS として、 MS–Windows や MacOS などが挙げられる。いままでコンピュータを操作したことのない初心者でも、その視覚的な情報で簡単に操作が可能なため、近年コンピュータの OS には GUI が使われるようになった。一方、キャラクター・ユーザ・インタフェース (CUI) とは、テキストベースのインタフェースで、基本的にコマンドを入力することでいろいろな作業を行なう。これを用いた OS として、MS–DOS や UNIX などが挙げられる。ある程度のコンピュータの知識がないと使うことが容易でないため、 GUI に比べると使っている人は限られる。

しかし、 GUI を使いづらいと思っている人達もいるようだ。視覚障害者は、 GUI の特徴である視覚的な情報を理解することは難しいので、使いづらいと思っている方は多いようだし、晴眼者でも特にご年配の方は、マウスによる微妙な操作や、ダブルクリック等の特別な操作がうまくできず、使いづらいと感じている人は多いようだ。

CUI は、テキストベースのため、コンピュータから出力された情報を読み上げソフトを使って音声として出力することで、画面情報のほとんどを確認することができるため、視覚障害者はこれを用いることで、コンピュータの利用が可能となる。

読み上げソフトで出力される情報は文字情報のみで、その他の情報は出力されない。しかし、画面上の文字情報以外の情報を音で出力することによって、視覚障害者のコンピュータの使い道が広がる可能性がある。そこで、 GUI のみに存在するマウスで操作するときに用いるマウスカーソルと、 文字列を入力する所にしかなく、文字やコマンド等を我々が入力したときに、これらを入力する位置を示しているカーソル (このカーソルを以下ではテキストカーソルと呼ぶ) の 2 つについて考える。

また、視覚障害者が文字等の情報を得るには、耳でその情報を聞くという方法の他に、点字を用いて指の触覚でその情報を得るという方法もある。しかし、点字というのは全ての視覚障害者が理解できるものではなく、視覚障害者の中で点字が理解できる人は全体の 1 割にも満たない [2)]。コンピュータには、文章を音声に変換する読み上げソフトがあるが、それと同じように点字を音声に変換してくれる読み上げソフトがあれば、点字を理解できない視覚障害者が点字で書かれた文献を読むことが可能となることはもちろん、点字を知らない晴眼者にとっても、点字がより身近なものになる。

画面情報の音化に関しては、色々な企業等で研究がされており、すでに実用化されている物も存在するようだ。しかし、最終的にはフリーソフトとして一般に公開することを目標とするため、既存の方法は使わずに一から考えてみることにした。音化に関しては、本研究室のフリーソフトである日本語テキストファイル読み上げソフト yomi を用い、音声で情報を出力することとした。

2　点字を読み上げるソフトウエアの考察

2.1　点字の説明

点字の基本的な構造について説明する。点字とは、紙面に突起した点を一定の方式で組み合わせた文字である。点の組み合わせとしては、縦 4 点、横 2 行で一文字を構成する 8 点点字と、縦 3 点、横 2 行で一文字を構成する 6 点点字などがある。また点字ではこの一文字を「一マス」と呼ぶ。

今日では主に 6 点点字が使われている。

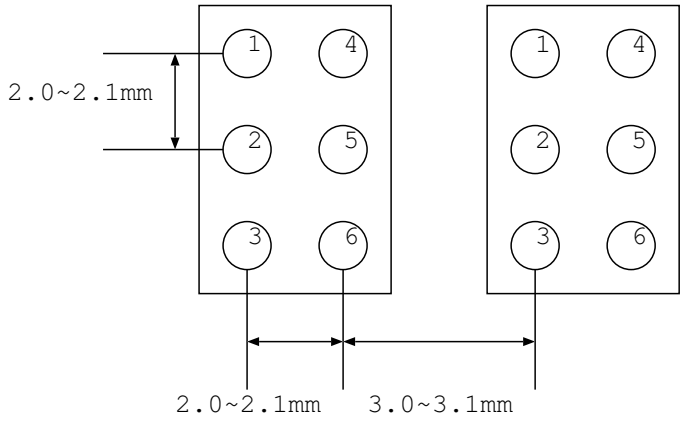

Fig. 1　6 点点字の配列と位置

Fig.1 のように、一マス内の各点は、左上最上段から下へ 1 の点、 2 の点、 3 の点、右上最上段より下へ 4 の点、 5 の点、 6 の点と呼ばれている。点字を読む場合は、普通左から右へ読んでいき、右端まできたら一段下の行へ移動してまた左から右へ読んでいく。また、日本の点字で表される文字は、漢字以外の文字 (ひらがな、アルファベット、数字、記号) となり、漢字を点字に訳したものも存在するようだが、これはほとんど使われていない。

日本の点字は、1825 年にフランスのブライユ (1809–1852) によって考案されたブライユの点字配列表に基づき、1890 年 (明治 23 年) に、当時東京盲学校の教員であった石川倉次 (1859–1944) がこれを翻案した。

2.1.1　日本の点字の基礎

以下では日本の点字について解説する。

- 五十音

  日本の点字はブライユの点字配列表を基にして作られている。(下表参照)

点字の母音対応表					
母音	あ	い	う	え	お
点字の母音	1 - - - - -	1 - 2 - - -	1 4 - - - -	1 4 2 - - -	- 4 2 - - -

そのほかの各行の構成は次のようになっている。

- 「か」行は、母音に 6 の点を加えて構成
- 「さ」行は、母音に 5,6 の点を加えて構成
- 「た」行は、母音に 3,5 の点を加えて構成
- 「な」行は、母音に 3 の点を加えて構成
- 「は」行は、母音に 3,6 の点を加えて構成
- 「ま」行は、母音に 3,5,6 の点を加えて構成
- 「ら」行は、母音に 5 の点を加えて構成

このように、母音を 1,2,4 の点で、子音を 3,5,6 の点で表し、この母音と子音の組み合わせが点字の基本になっている。例外として「わ」行は「あ」行の形を変えずに下へ落した形になっていて、「や」行は、その「わ」行に 4 の点を加えた形となっている。つまり、「あ」の文字を表す点字の点 (1 の点) を 3 の点へ移動させると「わ」になり、その「わ」の文字を表す点字の点 (3 の点) に 4 の点を加えると、「や」になる。

- 濁点・半濁点

  点字では濁点符や半濁点符という特別な文字を普通の文字 (「か」など) の前につけ、合計二マスで構成する。

- 拗音

  拗音とは、「きゃ」のような、小さな「ゃゅょ」を含んだ子音のことで、拗音には、例にあげた拗音以外に、「ぎゃ」のような拗濁音、「ぴゃ」のような拗半濁音がある。点字ではそれぞれに拗音符、拗濁音符、拗半濁符をつけたあと、主な母音と子音を組み合わせた点字を使い、二マスで構成する。

- 数字

  数字は、ブライユの点字配列表の点字をそのまま用いて構成されている。しかし、これでは五十音の「あ」「ら」の行と全く同じ記号を使うことになるので、区別するために数の最初に「数符」をつけ、次の文字は数字であることを示す。また、数字の桁数に関係なく、一つの数には数符を一つだけ用いる。

- アルファベット

  この点字も、かな点字と文字が重複している部分があるので、日本語の文中にアルファベットが使われるときは、「外字符」を前に置いて使う。またアルファベットの点字は基本的に小文字を表す。大文字を使うときは「大文字符」をアルファベットの点字の前に置く。二つ以上の大文字が連続する場合には、大文字符を二つ続けた「二重大文字符」を前に置く。

2.1.2 単語について

いろいろな単語を点字で表現すると、普段我々が使っている単語とは違った書き方をする場合がある。具体例をあげて説明する。

1. 「う」の長音

   「う」の長音は、点字では「う」と書かず、長音符を用いる。例えば、「がっこう」は、点字では「がっこー」となる。

   ただし、「言う、会う、買う」など長音でない「う」は長音符を用いず、そのまま「いう、あう、かう」と書き表す。

2. 数字と数助詞の複合語

   数字は、「あ」行と「ら」行と全く同じ 10 種類の点字を用いる。よって、文書中に数助詞が「あ」行、「ら」行で始まる時は、区別するために、その間につなぎ符という記号をつける。

   例えば「一塁手 (いちるいしゅ)」のように数字の後に「あ」行または「ら」行で始まる数助詞が来る場合は、数字と数助詞の間につなぎ符をつけて表す。

2.1.3 文について

点字では漢字を使わずにかなで文章を書くため、読みやすくする工夫がされている。

- 分かち書き

  文を理解しながら速く読むことができるようにするために、点字ではあらかじめ文節と文節の間を区切って表現することになっている。

- 助詞の「は」「へ」

  点字では助詞の「は」「へ」は、あらかじめ「わ」「え」に直されて表現されている。しかし、「を」はそのまま「を」で表現されている。

以上のことから、例えば、「明日は海へ行く」という文章は、点字では以下のように表現されている。

あすわ　うみえ　いく

2.2 点字の読み上げに関する考察

ここでは点字の文章を読み上げソフトで読み上げることが可能かどうか。また、読み上げに必要なものは何かなどを考えてみた。

2.2.1 点字を読み上げるメリット

印刷物の文章を読み上げることと比べると、点字を読み上げるということは、以下のメリットが考えられる。ここで、印刷物の文章は、点字に対して「墨字」と呼ぶ。以下では印刷物の文字を墨字と呼ぶことにする。

- 漢字の変換ミスがない

  墨字の文章を読み上げソフトで読ませる時に、漢字をいったんひらがなに訳してから読み上げを行なう。ひらがなに訳すときに例えば、「今月 (こんげつ)」を「いまつき」と変換する、というような読みがなの変換ミスが起こる可能性がある。しかし、点字ではもともとひらがなの文字が点字で書かれているので、漢字をひらがなに訳すときに発生するこのような変換ミスは起こらない。

- 分かち書きの操作が不要。 助詞「は」「へ」の変換の操作も不要

  墨字では漢字、かな混じり文からひらがなのみに変換する場合に、適当なところで文節と文節を区切る「分かち書き」という操作が必要になる。また、助詞の「は」「へ」を「わ」「え」に変換するという作業も必要とされ、コンピュータで変換する場合、どの場合の「は」「へ」を「わ」「え」に変換すれば良いかということをはっきりと区別することが必要で、コンピュータ上でこの変換をする場合、文書の文法構造を明かにしなければいけないが、この作業は面倒な作業となり、時間もかかってしまう。

  しかし点字の場合はすでに分かち書きや助詞の変換作業は終わっているため、上にあげた複雑な変換作業は必要無く、書いてあるままの文字を読み上げソフトに読んでもらえばいいというメリットがある。

  以上のようなことから、点字の文章は、 yomi には最も適した文章構造となっている。yomi 自体には文章を読み上げる機能しか付いていないため、ひらがなの文章の文字を何の操作もしないで、そのまま読み上げることができる。しかし、市販されている読み上げソフトウエアには、そのソフトの中にわかち書きの操作や、ひらがな変換のソフトが組み込まれているため、このような、すでにわかち書きや、ひらがな変換の済んでいる文章を読ませると、再びそのような操作が行なわれ、間違って出力される可能性がある。

2.2.2　点字の読み上げの問題点

点字を読み上げソフトで読ませるためには問題がある。それは、点字の情報をコンピュータに入力する方法である。

2.2.1 節で書いたメリットは、「点字の情報がコンピュータ上で文章として認識された後」のメリットであり、これよりも前の段階の、どのようにして点字をコンピュータに入力させるかという点が問題となる。一般的に点字には凹凸があっても色はついていないので、イメージスキャナでの読み取りは困難であると考えられる。また、人間が手作業で入力するという方法もあるが、この方法では点字を入力する手間がかかるので、あまり実用的ではない。このようなことを解決しなければいけない。

イメージスキャナで点字を読み取ることに関しては、色々と研究されていて [5)]、例えば、点字の凸面を斜めから照明をあてると発生する陰影像を用いて、市販のイメージスキャナで読み取るという方法があり、この方法では、点字タイプライタ等で印字された点字文章をほとんど間違いなく読み取ることが可能なようだ。しかし、手打ちで印字した点字の文章については、点の形や位置のばらつきが大きく、イメージスキャナで読み取ると、読み間違いが起こってしまうようである。

点字には色がついていないために、以上のような問題点が出てくる。よって、点字には色がついている方が良いように感じる。点に色がついていれば、以上のような問題点が解決できるだけでなく、晴眼者の人ならば離れていても点字を確認することができるので、点字を見る機会が増え、これにより点字に興味を持つ人達が出てくると思う。しかし、今現在点字に色がついていないということを考えると、点字に色をつけると点字の突起部の強度が低下するなどの問題点があるのかも知れない。

近年は、点字の文章を作成する作業はコンピュータ上で行なう場合が多くなっているらしい。これを作成した時のデータはコンピュータ内に残っていることになるので、そのような文章に関しては、このデータを用いることでコンピュータの入力の問題は解決できることになるだろう。

また、どれ位の精度かは不明であるが、点字を識別するソフトウエアも存在するらしい。

3　テキストカーソルの音化

1 節で説明したように、カーソルの種類の一つとして、文字列を入力する所のみに存在して、文字やコマンド等を我々が入力したときに、これらを入力する位置を示しているテキストカーソルがある。このテキストカーソルの位置を音で表現することで、画面を見ずにカーソルの位置を特定できるようにすることを目標とした。

3.1　カーソルの位置を示すメリット

テキストカーソルの位置がわかると次のようなメリットがある。今読んでいる文章を中断して、後でまたその続きから読もうとしたとき、その中断した所の位置、つまり、中断した時のテキストカーソルの位置を知っていれば、すぐに目的の位置を探すことが可能と

なる。よって、その位置情報が「しおり」の代わりとなる。また同様の方法は、文章を作成する場合などにも利用できる機能である。

3.2 出力する情報の考察

テキストカーソルを音で表現する場合に必要な情報として、文章の何行目にあって、その行の左から何文字目にあるかといった情報が必要になってくる。また、テキストカーソルがある行より下には何行あるか、テキストカーソルの右側には何文字あるかといった情報も付け加えることで文章全体のどこにテキストカーソルがあるかという情報もわかるため、これも必要になる。以上を音声で表現するための方法としては、今テキストカーソルがあるところよりも前の行数 (Fig. 2 の (1))、後ろの行数 (Fig.2 の (2)) を音声で表現して、続いて今度はテキストカーソルがあるところよりも左にある文字数 (Fig.2 の (3))、右にある文字数 (Fig.2 の (4)) の情報を (必要ならば音を変えて) 音声で表現するという方法を考えた。

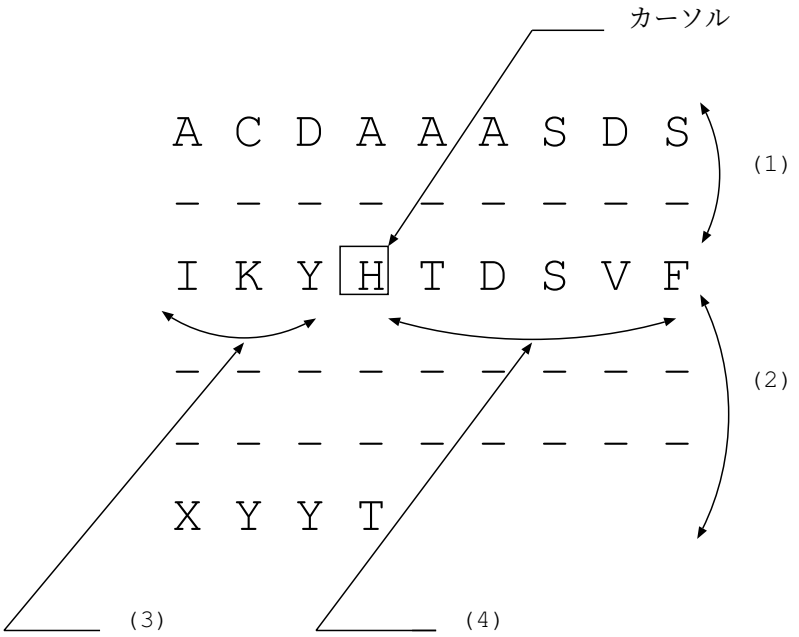

Fig. 2 テキストカーソルの音化に必要な情報

また、今回、出力情報には音声を使うことにしているが、これを「ピッ」という音で出力する方法について挙げてみた。

方法 1

例 ： 「行数 (文字数) が 5 の時」は、「ピッピッピッピッピッピッ」と鳴る。

これは、単純に行数 (文字数) 分だけ音が鳴るという方法で、行数が長くなると、数えるのに時間がかかり、また、人間が数えることになり、数え間違いも起こる可能性があるため、長い行を含んだ文章に使うことは難しい。しかし、この場合はただ鳴った回数を数えるだけで良いため、少ない数ならば簡単に数えることが可能となる。

この方法よりは音声で行数、文字数を音声で出力させた方が速いし、確実な情報を伝えることができる。

方法 2

音のみで情報のやりとりができるものの代表として「モールス信号」があげられる。このモールス信号を使って表現することを考えてみた。

モールス信号は短音と長音でアルファベット、数字、記号などを表現する。できるだけ信号の数を少なくするために、使う信号は数字のみとした。

なお、モールス信号は、次表のように表される。ここで、短い線は、短音、長い線は長音を表す。

モールス信号の数字配列表

数字	モールス信号
1	- —
2	- - —
3	- - - —
4	- - - - —
5	- - - - -
6	— - - - -
7	— - - -
8	— - -
9	— -
0	—

例えば、テキストカーソルより上に 4 行、下に 10 行 (つまり全体として 15 行) あって、テキストカーソルより左に 7 文字、右に 3 文字 (つまり全体として 11 文字) ある場合には「4 10 7 3」とモールス信号で音として出力されるようにし、また、テキストカーソルより上に 0 行、下に 11 行、左に 0 文字、右に 10 文字ある場合、つまりカーソルが一番先頭にある場合は、「0 11 0 10」というふうに、 0 も使ってモールス信号で音として出力する。

この方法では、モールス信号を覚えていなければ使えないという欠点があげられる。また、桁が多くなると、出力される音の情報も多くなってしまい、音声で位置情報を出力するよりも出力される時間が長くなってしまう。しかし、その音の情報に対して、しっかり

とした決定的な情報が与えられているので、テキストカーソルがいまどこにいるかを正確に表現することができる。

音声化と比べると、細かい情報はどちらの出力情報でも得られることができる。しかし、出力時間や、情報のわかりやすさよいう所から考えると、まだ音声化の方が良いと思う。

方法 3

音の長さを変えてテキストカーソルの位置を表現してみようと考えた。

最初に一行 (一文字) あたりの音の長さを決めておく。これと実際の行数 (文字数) の積が出力する音の長さとなる。

例 カーソルの前に 20 行、後ろに 40 行 (つまり全体として 61 行)、左に 5 文字、右に 10 文字 (つまり全体として 16 文字) ある場合で、一行、一文字あたりの音の長さを 0.5 秒と設定した場合、

カーソルより前の行：20×0.5 =10 秒
カーソルより後の行：40×0.5 =20 秒
カーソルより左の文字：5×0.5 =2.5 秒
カーソルより右の文字：10×0.5 =5 秒

最初の行 (または行の中の最初の文字) や最後の行 (または行の中の最後の文字) にカーソルがある場合、特別にそれらを示す音を予め設定しておく。またその場合、文章全体の行の長さや、行の中の文字列の数を表した音も出力するようにする。

行のみで具体例を示す。

- 最初の行にカーソルがある場合、最初に文全体の行の長さに合わせた音を出力した後、「ピッピッピッ」と短音を 3 回鳴らす。
- 最後の行にカーソルがある場合、「ピッピッピッ」と短音を 3 回鳴らした後、文全体の行の長さに合わせた音を出力する。

  この例の後に今度はテキストカーソルより左には何文字あり、右には何文字あるという情報が続く。

この方法の欠点としては、音の長さで表現しているので、出力される音の情報だけでは正確な位置は特定できないということがあげられる。また、今回例にあげたものでも、情報の出力に 40 秒近くかかってしまう。プログラムデータなど、膨大な行数の文章の場合では「ピー」という音がもっと長く続いてしまう。逆に短かすぎる文章の場合はこの機能はあまり役に立たない。

出力情報の細かさ、出力時間等を考えると、音声で出力した方が、良い場合が多いようだ。

音の長さでテキストカーソルの位置を表現することを例えばメールを読む前に実行すれば、文章の中身を読み上げる前に音の長さでメールの文章の長さが分かり、どれくらいの

文章量があるのかを知ることが可能となる。また、カーソルがそのテキスト全体のどこにあるのかもだいたいわかるようになる。この表現方法は、スクロールバーの利点も含まれていることになる。

方法 4

あいまいな情報を音声で出力するという方法を考えた。

テキストカーソルより上にある行数 (カーソルより左にある文字数) を文全体の行数 (文字数) で割って、その商が

$0 \leq$ 行数 (文字数) の商 $\leq 1/3$ ならば「上 (文字数の場合は左) 」

$1/3 <$ 行数 (文字数) の商 $\leq 2/3$ ならば「中央部 (文字数の場合は中央部)」

$2/3 <$ 行数 (文字数) の商 ≤ 1 ならば「下 (文字数の場合は右)」

という情報を音声で出力するというものである。

例えばテキストカーソルより上に (カーソルがある行も含めて) 4 行あり、全体として 10 行あり、そしてカーソルより左に (カーソルも含めて) 7 文字あり、全体として 10 文字ある場合、

行数 $= 4 \div 10 = 0.4 =$「中央部」

文字数 $= 7 \div 10 = 0.7 =$ 「右」

となり、「中央部、右」という情報が出力される。しかし、あいまいな表現のため正確な位置を知ることは困難となるが、この場合は数秒で表現することができる。

3.3 カーソルの位置情報

これまでは何行目の何文字目にあるかといった 2 次元的な情報で位置の情報を表すことを考えてきた。この他に、一番先頭から何文字目にあるかといった 1 次元的な方法があげられる。

この場合は、「何文字目」という情報だけを出力すればいいというメリットがあるだろう。「行」というものは、あくまでも視覚的な情報のため、文字数の情報だけでも「文章全体のどこにカーソルがあるか」ということがわかると思われる。

3.4 テキストカーソルのまとめ

これまでは音声のみでテキストカーソルを表現することを考えてきたが、音声出力だけでなく、「ピッ」という「音」でテキストカーソルを表現した方が良い場合も考えられる。

「ピッ」という音は、音声に比べると情報量が少ない。よって、音を用いて位置情報を瞬時に出力した場合、その情報自体はあいまいな情報となる。

マウスカーソルの場合は、画面上のグラフィカルな情報に対して、このカーソルの操作で指示を与えるもので、そのある一つのグラフィカルな情報のどこを操作しても同じ結果となる。つまり、位置情報があいまいでも、それなりの操作は可能となる。音自体の情報

量が少ないために、出力にこれを用いると、方法によっては出力時間が短縮される可能性がある。

しかし、正確な情報が要求されるものは、「ピッ」という音によるあいまいな表現では使いづらいものとなる。カーソルがいる場所を正確に知るには情報量の少ない「音」よりも、「音声」のほうが正確な位置を伝えることが容易になる。このように、時と場合によって、テキストカーソルの位置情報の出力方法を変えていくことで、より使いやすくなる。

4　マウスカーソルの音化

4.1　マウスのメリット

ここで、一般的なマウスのメリットを考えてみた。

1. ランダムアクセス可能
2. 容易に絵を書くことができる
3. 片手での操作が可能
4. 簡単にソフトを起動できる

1 は、いま作業をしている場所から他の作業場へ移って作業する場合、その作業場へマウスカーソルを移動させて、ボタンを押すことで、すぐに移った作業場で作業でき、同じ作業場内でも、目的の場所へマウスカーソルを移動してボタンを押すことで、その場所へジャンプできるということを意味している。 2 は、マウスカーソルを動かすことによって、特殊な装置を必要とせずに自由に絵が書けるということを表す。3 は、マウスの作業は片手で操作可能なので、例えば、本を持ちながらコンピュータを操作することが可能になるということを意味する。4 は、画面上の特定の場所にマウスカーソルを移動して、ボタンを押すことで、キーボードで特別なコマンド入力をすることなしに簡単にソフトを起動することができるということを意味している。このようにマウスには、実用的な色々なメリットがある。

4.2　マウスカーソルの表現方法の考察

マウスカーソルの位置情報を教えるためにはどのような情報を音で出力すればよいかを考察してみた。

方法 1

これは、画面を格子状に区切り、区切った一マス一マスを「座標の点」と考えてその点の x 軸、 y 軸のマスの値を読み上げる。という方法である。

ここで、座標の値を読み上げさせずに、大まかに格子状に区切った理由として、次のようなことを実現させたかったからである。

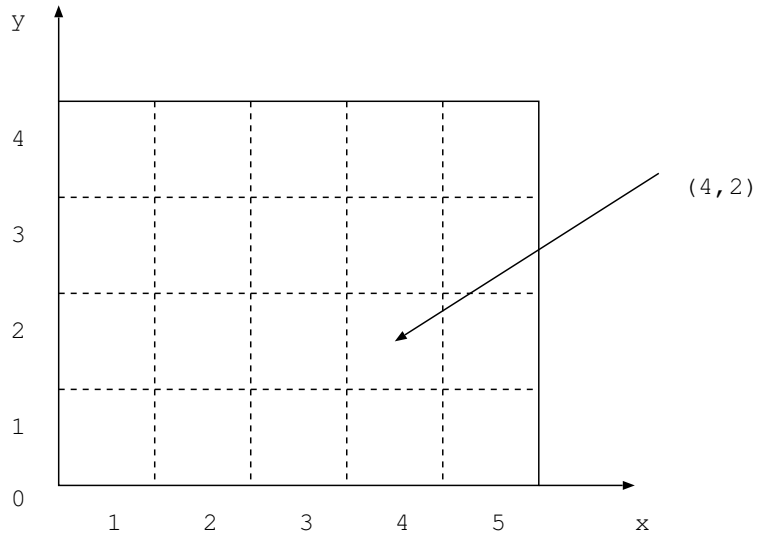

Fig. 3 画面を格子状に区切る概念図

1. 迅速な位置特定

2. 出力時間の短縮

1 を目指すためには、あまり細かくない情報を出力させる必要がある。出力される情報が細かすぎると、どこにカーソルがあるのか理解することが難しいからである。大まかに区切ることで、どこにカーソルがあるか素早く確認することが可能となる。

2 を目指すためには、音声出力は短いほうが良い。しかし座標で出力させた場合、出力される値は、どうしても 2 桁や 3 桁となる場合が多くなってしまう。基本的に、2 桁や 3 桁の数字を音声で出力させるよりも、1 桁の値の出力の方が時間の短縮が可能となる。よって、 1 桁の数で表示させるために x, y の座標をそれぞれ 9 個以内のマスで分割し、格子状に区切ることにした。

方法としては、Fig.3 のように、まず画面を縦、横, 等間隔に区切る。その後画面の左下を原点として座標でいう x 方向、 y 方向のそれぞれのマスに 1,2,3 と数字をあてていく。そのようにして画面を仮想的に区切り、それぞれのマスに数字を割り振った状態で、たとえば Fig.3 で考える場合、図の位置にマウスカーソルが止まった場合には、「 x 5 y 4 のうち、x 4 y 2 」 というような情報を知らせれば、視覚障害者にもマウスカーソルのだいたいの位置が確認できると思う。ここで、最初の x, y の値は x 軸、 y 軸の最大の値、後ろの x, y は現在のマウスカーソルの位置を表す。また、「 x 5 y 4 のうち、x 4 y 2」という所は画面を区切る設定を変えずに何度も聞いていると、最初の最大値の情報や x, y の部分をとって、マウスカーソルが今いる所の x 軸 y 軸のそれぞれのマスの数字 (Fig.3 でいうと「4 2」) だけの情報を出力すればカーソルの位置がわかると思う。よって、x と y の最大値の情報や、x,y は始めだけ出力することにして、 2 回目以降は出力しないことにする。

この方法では、詳しい位置特定は無理だが、迅速に位置情報を知ることは可能となる。

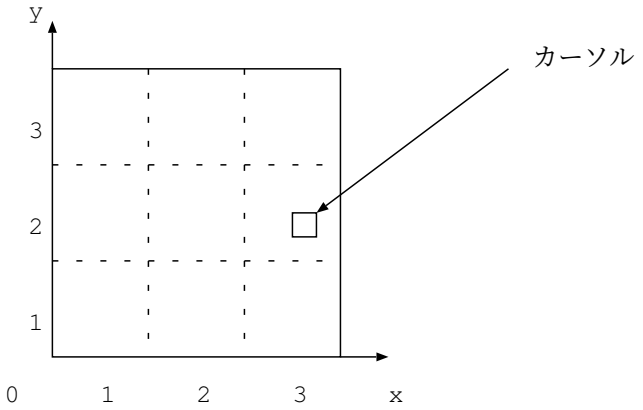

Fig. 4 マウスカーソルのあいまいな表現

また、数字の情報ではわかりにくい場合、もう少しあいまいな情報を「単語」で出力することも可能である。画面の左上にある、下にあるなどを表す単語で表現することで、だいたいの位置がつかめるのではないかと考えた。Fig.4 を用いて説明する。

Fig.4 の 1 つ 1 つのマスにあいまいな単語の情報を入れていく。(例えば、x 1 y 1 のマスには「左下」、x 2 y 2 のマスには「中央」など) そのとき Fig.4 の矢印の位置にマウスカーソルが来た時には、「右」という情報を出力することでカーソルのだいたいの位置がわかるのではないかと思う。

画面を格子状に区切り、単語で出力するという考え方は上に挙げた通り、瞬時に今の位置の情報がわかるというメリットがある。しかし、一番大きな問題点は本当にだいたいの位置しかつかむことができないということである。もっと細かく位置情報を伝えることができなければ逆に使いにくいものになってしまう。

方法 2

Fig.5 のように画面の左下を原点とおいて、今カーソルがある所と原点との直線の長さ (Fig.5 の (1) の部分) と画面下の直線部分 (Fig.5 の (3) の部分) と (1) とのなす角 (Fig.5 の (2) の部分) で表す。つまり、極座標の形式で表現するということである。この場合の問題点としては (1) の長さの単位は何を使えばいいのかという問題点が挙げられる。例えば原点からカーソルまでの長さが 6 で、なす角が 45 度の場合は、「長さ 6 角度 45」という情報だけではカーソルの位置をすぐに特定するのは難しいと思われるが、しかし使い方次第ではメリットがある方法だと思う。

長さに関しては、止まっている時にはあまり意味のない情報となるかもしれないが、マウスカーソルを動かしている時には、原点からの相対的な距離の変化を知る目安となるため、長さの情報も意味のある情報となる。

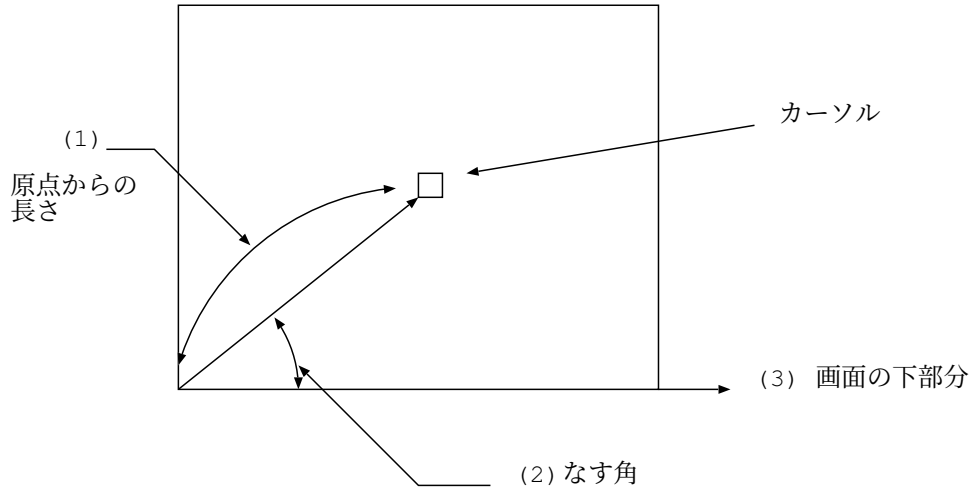

Fig. 5 マウスカーソルの位置の表現方法

4.3 マウスカーソルの軌跡の表現方法

動いている時のマウスカーソルの情報も表現できなければ絵を書くという作業等は行うことが難しくなる。今まではマウスカーソルの止まっているときの位置情報の表現方法を考察してきたが、今度はマウスカーソルがどのルートを通ったかという跡のようなものを音声で表現する方法を挙げてみた。

方法 1

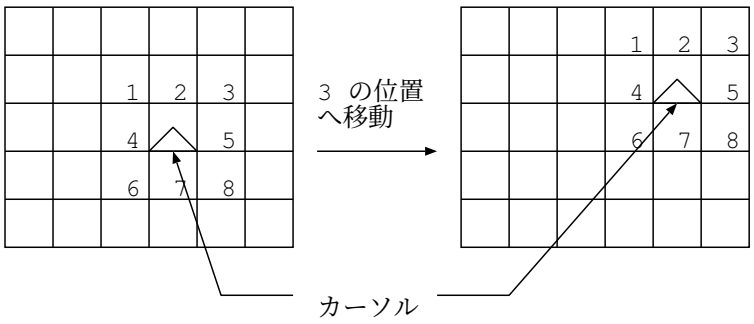

Fig. 6 カーソルの移動に伴う数字の移り変わり

Fig.6 の左図のように、マウスカーソルの周りのマスにマウスとの位置関係に応じた番号を振り、カーソルが Fig.6 の右図に示す位置へ移動した場合は、音声で 3 が出力され、これに伴い、移動したマウスカーソルの周りの各マスに同じように番号が振られて行く。

このようにすると、番号の出力でどちらの方向へ進んだか確認することができるようになるかもしれない。

方法 2

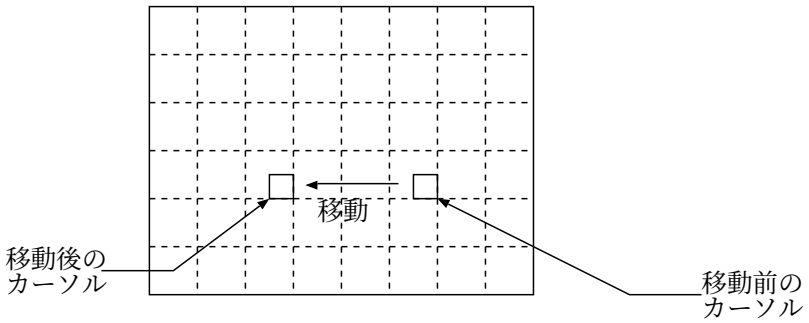

Fig. 7 概念図

他のマスへ移動したことを「ピッ」という音で表現するという方法である。

Fig.7 を用いて説明する。この図の各マスに「ピッ」と鳴る情報を与えておく。それで、マウスカーソルが各マスに来たときに「ピッ」と鳴るようにする。つまり、この図のように移動した場合は、「ピッピッピッ」という音が出力されることになる。この場合は、マウスカーソルの動きにあわせてリアルタイムに情報が出力されることから、マウスを速く動かせば「ピッピッピッピッピッ」と素早く音が出力されるし、逆にゆっくり動かすと「ピッ…ピッ」というふうに間を置いて音が出力されることから、マウスの動きを簡単に確認することができる。しかし、これだけではマウスの位置を知ることは困難となる。

4.4 プログラムによる実験と考察

前節の提案のうち、まずマウスカーソルを極座標のように原点からの長さと、 x 軸とのなす角で表すことを実際にプログラムを作ってコンピュータ上で実験してみた。なお、カーソルの位置を読み上げる範囲は、プログラム実行後に表示されるウインドウ内のみとなる。また、長さの単位はピクセル値を用いている。

基本となるプログラムをそのまま実行すると、新しいウインドウが現われる。例えば Fig.8 のカーソルの位置でマウスの左ボタンをクリックすると、位置情報 (この図の場合は a, b) が数値で出力される。さらに、音声出力を加えた、プログラムを実行すると位置情報の数値を音声で (この図の場合は「a, b」) 読み上げてくれる。そして、出力される a, b のデータを、 Fig.9 のように、長さ r と角度 θ に変換するために、Fig.10 より、

$$r = \sqrt{a^2 + (ymax - b)^2}$$

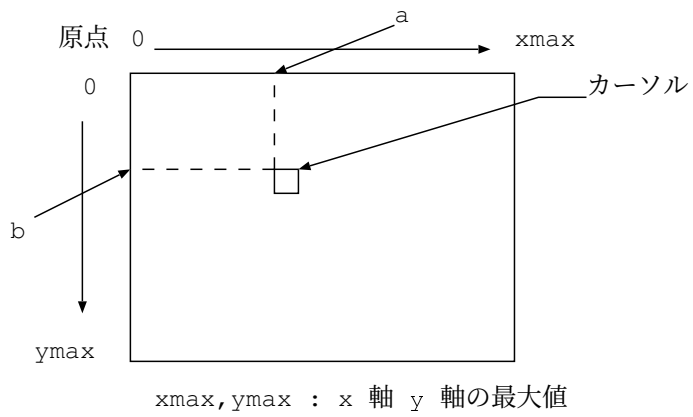

Fig. 8 そのまま実行した場合

$$\theta = \arctan\left(\frac{ymax - b}{a}\right)$$

の 2 本の式を用いて、データを変換した。Fig.9 のようにするために、Fig.10 のような考

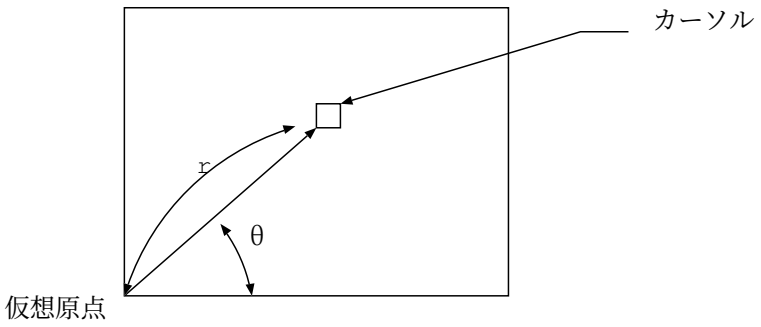

Fig. 9 極座標

え方をした。ここで $xmax, ymax$ は、本当の原点からの x 軸、 y 軸の最大値となる。

上記の実験を行った結果、この方法の改善すべき点として、以下の 2 つが挙げられる。

1. 長さの出力の問題
2. 使用する角度の拡大

1 は、この実験で長さの出力情報として、原点からカーソルまでの長さしか出力されず、比較の対象が存在しないことから、長さの把握が難しいという点が問題となる。これを改善するためには、スケール変換を行なうなどして、比較するに相応しい対象を適切に選ばなければならない。 2 は、90° までしか使わなかったという点が改善点となる。その理由

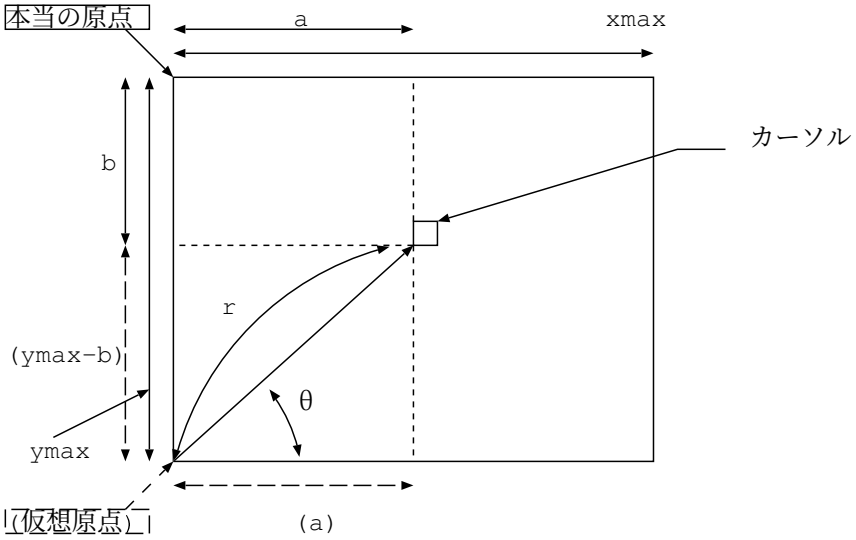

Fig. 10 極座標を表現するための考え方

として、この実験で、角度の情報のみでマウスカーソルは原点からどの方向に存在するのかという情報が比較的容易に理解できることがわかったという点が挙げられる。角度は基本的に 360° まで存在するため、360° 全てを用いればもっと詳しく原点との方向がつかめる可能性が考えられる。改善策としては、Fig.11 のように、原点を中心におき、角 AOB の角度を出力させれば 360° の表現は可能となる。ただ、arctan は、 $-90°$ から 90° しか表現できないので、360° 表現するには改善が必要となる。

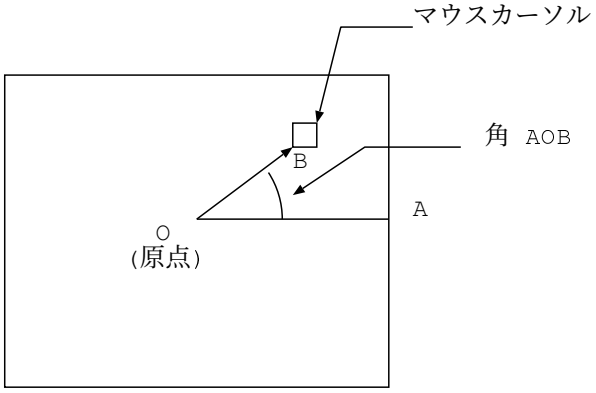

Fig. 11 360° 表現可能

4.4.1 情報の出力の問題

出力は音声で行うことを目的としている。ここでの音声とは、人の声を表す。しかし今回実行してみると、音声で出力した場合は問題があることがわかった。問題点としては、

- マウスをクリックしてから音声が出力されるまでの間が長い
- 音声が出力されてから終了までに時間がかかる

ということが挙げられる。時間の短縮を目標とすれば、これを解決するために、出力はやはり「音声」化よりも、すばやく出力できる「ピッ」というような「音」化の方向へ進んだほうがいいのかもしれない。

その音化にも問題がある。もっとも大きな問題は、何の決まりもない状態で、「ピッ」という 1 音のみを出力しても、この音は何を表しているのか想像することはできないことからもわかるように、「音」自身の情報量の少なさがあげられる。情報が少ないと、出力という面では出力までの時間がかからないというメリットとなるが、その分情報が少ないために、その出力方法を適切に選ばなければ情報を的確に知ることはできない。音での表現方法としては以下にあげた表現方法が考えられる。

1. 音の長さを変える
2. 音の高さを変える
3. 音の大きさを変える
4. 音の鳴る回数を変える
5. 音の種類を変える

この表現方法を適切に組み合わせて表現しなければ使いづらいものとなる。

4.5 マウスカーソルの位置特定の考察

ここまでの考察を基にして、音でマウスカーソルを伝えるための方法を考えてみた。

方法 1

Fig.12 のように左下を原点として、そこからの x, y それぞれの座標を音の高さで表し、今いる位置をその x, y の音の高さで出力するという方法を考えた。この方法では原点の位置にマウスカーソルがある場合、 x, y 共に一番低い音を出力し、x, y 共に原点からの距離が最も遠くにある (Fig.12 の B) の場合、x, y 共に一番高い音が出力されることになる。

x, y それぞれにおける現在位置を「ピー、ピー」という 2 音で出力することから、出力される情報は音の高さとなるので、絶対音感のある人でない限りマウスカーソルの現在位置を特定することは困難である。しかし、出力にかかる時間は短くなるという面もある。

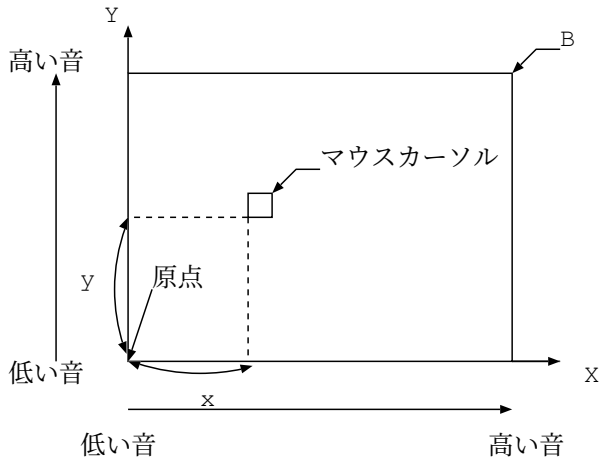

Fig. 12 音の高さでカーソルの位置を表現

方法 2

方法 1 の出力情報だけでは、絶対的な音の高さを理解できる人のみが使えるということになってしまう。そこで方法 1 の出力情報が多くの人に理解できるように工夫をしてみた。

方法としては Fig.13 のように x, y の値の他に中心 a の位置情報を出力することを考えた。中心 a の音は、音の高さでいうと、ちょうど真ん中の音ということになる。この音を、 x と y の音の間に入れて出力することを考えた。

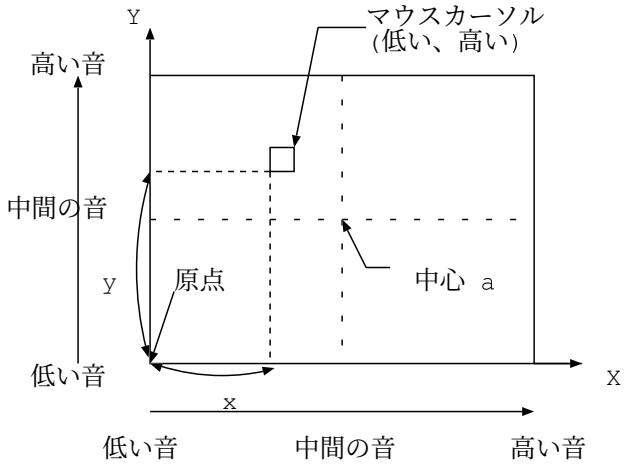

Fig. 13 中心を取り入れた例

こうすると、Fig.13 を用いて、音の出力を説明すると、

- 中心 a よりも、右上にマウスカーソルがある場合は、以下の 3 つの音が出力される。a よりも高い音、a の音、a よりも高い音
- 中心 a よりも、右下にマウスカーソルがある場合は、以下の 3 つの音が出力される。a よりも高い音、a の音、a よりも低い音
- 中心 a よりも、左上にマウスカーソルがある場合は、以下の 3 つの音が出力される。a よりも低い音、a の音、a よりも高い音
- 中心 a よりも、左下にマウスカーソルがある場合は、以下の 3 つの音が出力される。a よりも低い音、a の音、a よりも低い音

となり、x と y の間に中心の音を入れたことで、真ん中の音よりも高い音か低い音かという比較をすればいいということになるため、方法 1 の出力情報よりは位置の特定がしやすいものとなる。

少なくとも画面上の 4 つの位置は簡単に特定することができる。よって、この方法を応用すると、この 4 つのそれぞれにソフトウエアを起動するためのアイコンや、コンピュータを終了するためのアイコンなどを当てはめておくことで、キーボードからのコマンド入力をせずに、マウスのボタンをクリックすることで、簡単にソフトウエアが起動できる。

ワープロ	メール
コンピュータの終了	インターネット

Fig. 14 アイコン化の例

それによって、視覚障害者でこれからパソコンをはじめようと思っている人や、パソコンを使いはじめたばかりの初心者にとってメリットがあるのではと考える。普通、いちいちコマンドを入力して実行する CUI よりも、マウスを用いてソフトウエアの起動や終了などの操作が簡単に行える GUI の方が操作しやすいと感じる。視覚障害者も、初心者は、マウスでの操作のほうが気軽にコンピュータが使えるようになるのではないかと思われる。

しかし、この方法の問題点としては、視覚障害者が、ワンタッチで起動するだけにマウスを使うのならば、キーボードのキーの一つ一つに起動するソフトウエアを当てはめ、キーボードで操作した方がよいと思う。それは、視覚障害者は主に CUI を用いるため、マウスはほとんど使用しないでキーボードのみを使う機会が多く、この機能のためだけ

に、いちいちキーボードからマウスに手を移動して操作するというのはあまり合理的ではないからである。また、初心者をターゲットとするならば、情報の出力は、「ピッ」という音による出力よりも、「ここを押すと終了します」というように、音声で出力したほうがわかりやすい。したがって、この場合は音のみでは情報不足ということになるだろう。

上に挙げた例のように、単に 4 つの位置の特定したいだけならば、計 4 つの音のみを使って表現すれば良いことになる。しかし、 x, y の位置を音の高さで表し、その間に基準となる音を入れて比較することで、基準となる点よりもマウスカーソルは離れたか近付いたかという距離感や、どこに向かっているかという方向性もだいたいわかるというメリットが出てくる。その意味では、比較という考えには、まだ色々な使い道があるのかもしれない。

また、この音の高さを比較するという方法を方法 1 に当てはめると、方法 1 では、Fig.15 のように、x と y の音の比較により、 $y = x$ との 2 つの相対関係によって、 2 つの相対関係や、直線 $y = x$ に対して、近くにあるか離れているかというだいたいの距離感がわかることになる。

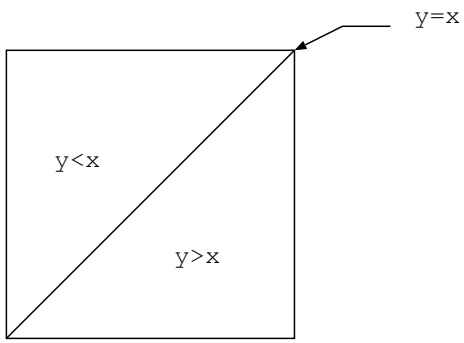

Fig. 15 識別可能な 2 つの領域

4.6 新たな考察

前節 4.5 の方法 2 で、出力される音同士を比べるという考えは、4.5 節の方法 1 のように、ただ単に今いる位置に対応した音を出力するよりは位置が特定しやすく、音同士を比較することで、基準となる点に対し、近付いたか離れたかというだいたいの距離感もつかみ取ることが可能であるということがわかった。これは、絶対的な音の高さを特定するよりも、音と音とを聞き分けることで、どちらの音の方が高いか低いかという相対的な音の高さを特定する方が容易であることによるためだと思う。そこで、4.5 節の方法 2 をもう少し考えてみようと思う。

4.6.1 基準点の考え方の見直し

4.5 節の方法 2 のように、 x と y の間に入れる情報として、画面上の中心という固定された位置の情報を入れた。これは、画面の中心という決まった位置の情報か出力されるのみなので、この中心に入れる情報を考え直してみることにした。そして、間に入れる情報を図形やアイコンといった、ある目標物の位置にするのはどうかと考えた。

また、これまで、 x と y の間に入れる情報を「中心」として考えてきたが、これだと、中心を移動させるときに、この情報自身の位置も示さなければならなくなる。位置情報は、2 次元的な情報なので、一つの音だけでは表すことは困難となり、音の出力情報を増やさなければいけなくなる。しかし、間に入れる情報を「原点」とすれば、一定の一つの音のみで表現することが可能となる。

画面を 4 つに区切った場合について Fig.16 を用いて説明する。目標物が原点となる

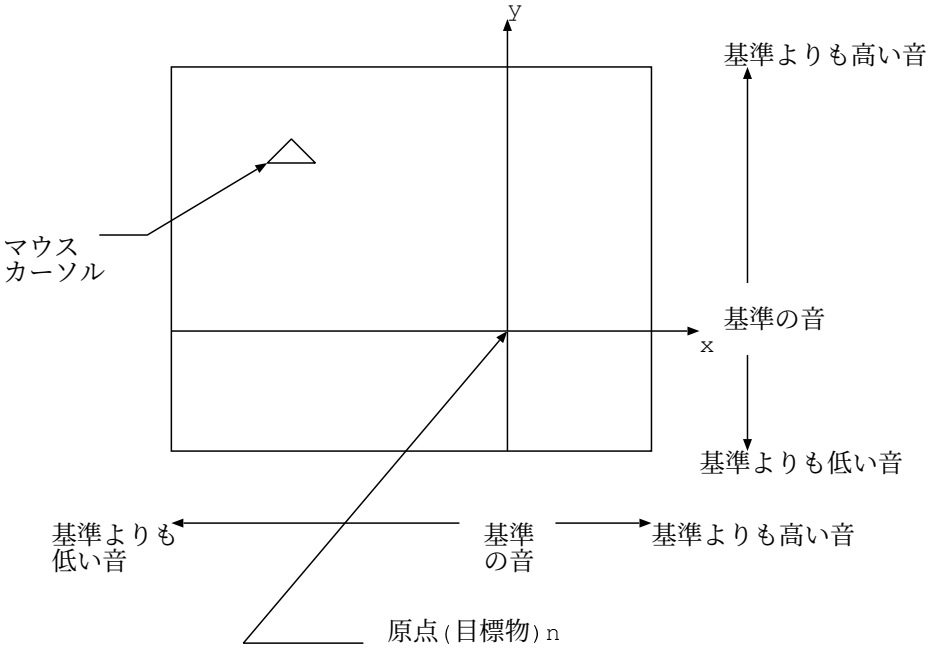

Fig. 16 位置情報の考え方

ため、この場合、考える範囲は Fig.17 のように、目標物より左側を調べるということになる。

Fig.16 の所にマウスカーソルがある場合は、 x 軸方向は負の値、y 軸方向は正の値になっているため、音としては、

「基準 (n) よりも低い音、基準 (n) の音、基準 (n) よりも高い音」

が出力される。音の高さは、x, y ともに原点 (目標物) n に近付くにつれて基準の音とだんだん近い音になってくる。これでカーソルが目標物に近付いていることが判断できる。

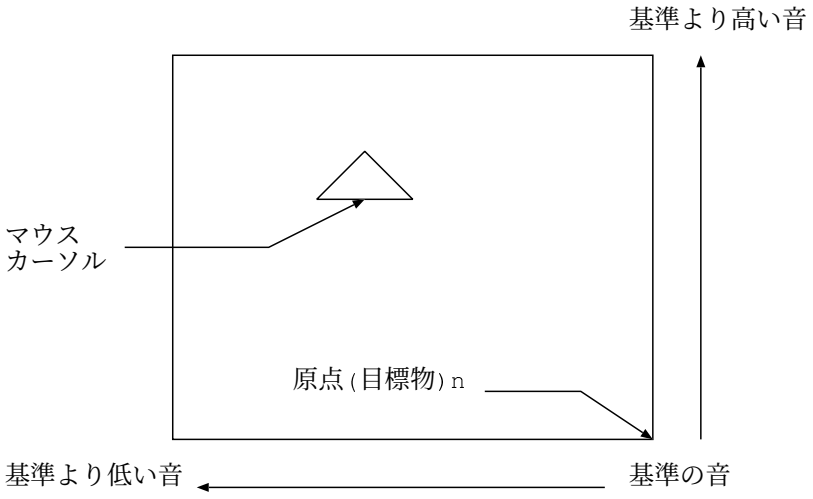

Fig. 17 考える範囲

今度は画面を 9 つに区切った場合について考える。

Fig.18 のように、 9 つに区切った中心に目標物があり、目標物の位置や大きさに合わせて区切るマスの大きさも変化する。Fig.18 の 1,3,6,8 のマスにカーソルがある場合、目標物のある中心のマスの 4 つの頂点のうち、カーソルに一番近い頂点が原点となり、そこからの音の高さの違いでマウスカーソルとの位置関係をつかむ。

Fig.19 の位置にマウスカーソルがある場合には、

「基準よりも高い音、基準の音、基準よりも低い音」

が出力される。

Fig.18 の 2,4,5,7 のマスにカーソルがある場合は、 2,7 のマスについては x の音の高さを、4,5 については y の音の高さを基準の音の高さと同じにしてしまう。(Fig.20)

この方法の問題点としては、マウスカーソルの位置が原点に限りなく近くなった場合に、音の高さも見分けがつかなくなってしまうため、目標物と重なっているのかいないのかがはっきりしないという点が挙げられる。解決策としては、カーソルが目標物と重なっているかどうかわからない場合は、ある任意の操作でもう一つの音を出力し、その音の種類の違いで特定するという方法が挙げられる (Fig. 21)。また、新たな音を出力するのではなく、もともと出力される 3 つの音の中の 2 番目に出力される音、つまり基準となる音に「音の長さ」という情報も付け加えて、出力される音の長さが、他の音と同じ長さか、他よりも長い長さかということで、目標物と重なっているかどうかを調べるという方法も考えられる。

また、いくつかある目標物の候補のうち、ある特定の目標物との位置関係を知りたいときは、どのようにしてその目標物を捜し出すかという問題もある。例えば、キーボードを使って捜し出そうとした場合、いままで考えて来たこと自体が無駄になりかねない。よって、このある特定の目標物を見つけることが大きな問題点となる。

しかし、この位置特定の方法により、点と点との位置確認だけではなく、 2 次元物体

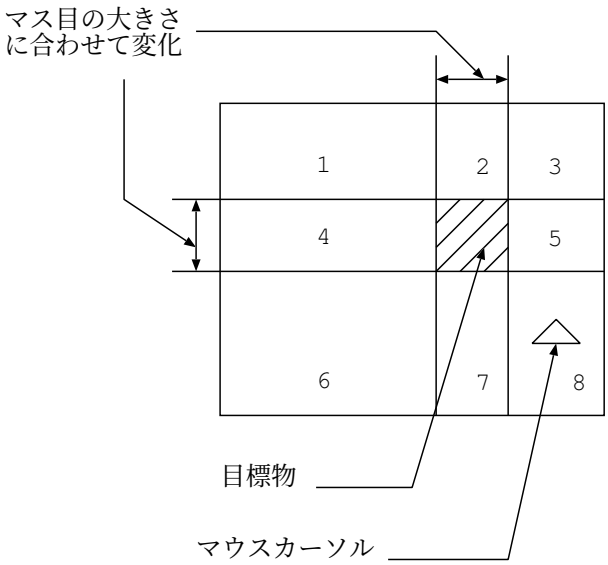

Fig. 18 9 つに区切った場合

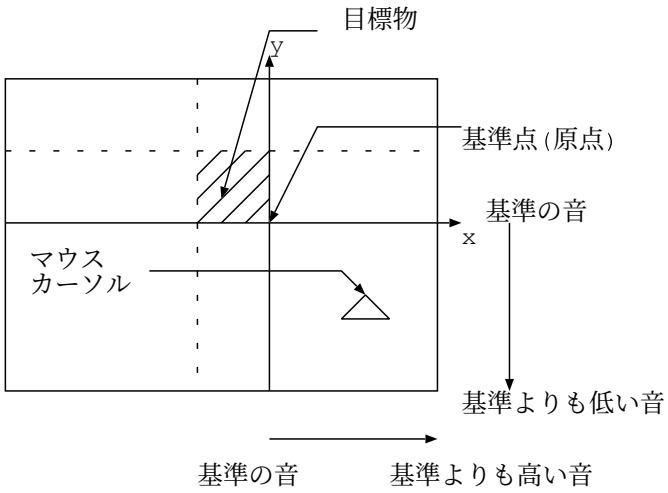

Fig. 19 出力する音の高さの例

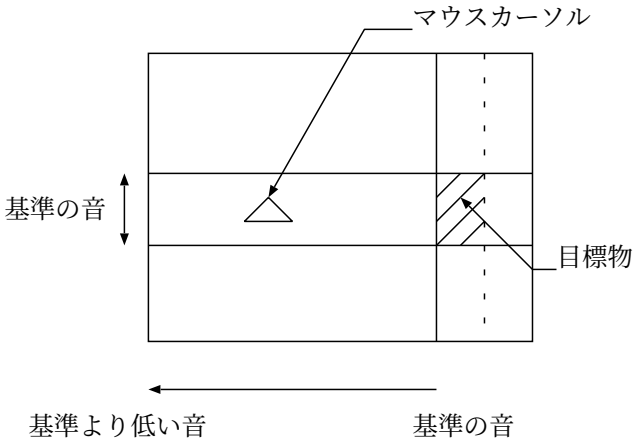

Fig. 20 2,4,5,7 のマスの音の高さの例

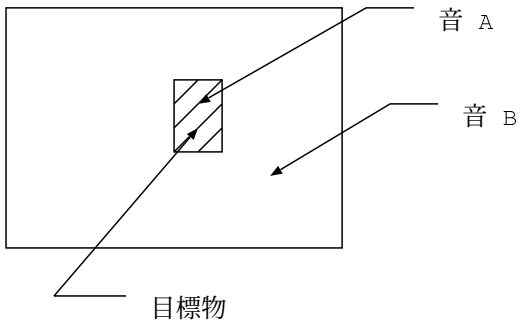

Fig. 21 2 種類の音で目標物との状態を判定

との位置確認が可能となる。ある位置から、目標物までの相対的な位置関係が音で表現できれば、マウスカーソルの特定という、これまで考えてきたこととは別に、画面上に描かれている図形の位置関係の特定が可能となるかもしれない。また、さらに進んで、音の情報だけで、与えられた図形を音のみで思い通りの位置に移動することが可能になるかもしれない。

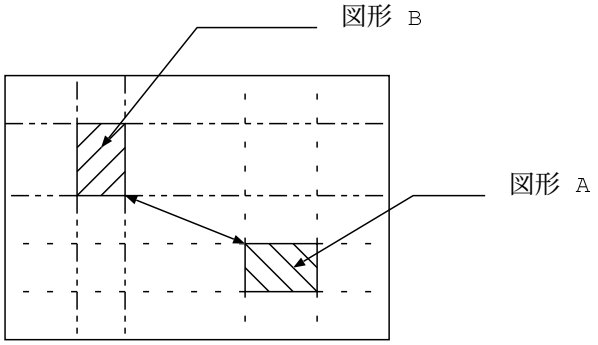

Fig. 22 図形同士の位置特定への応用例

5 まとめ

点字の読み上げ、マウスカーソルとテキストカーソルそれぞれの音化という 3 つのことについて考えた。

点字の読み上げについては、点字そのものを考察することで、読み上げさせることの可能性を探ってみた。その結果、点字を書くときの決まりを見てみると、yomi で読み上げさせるには最も適した文章方式であることがわかった。しかし、その点字の情報をコンピュータ内に入力するときに問題があることがわかった。

テキストカーソルの音化については、いろいろな方法を考察して、その実現の可能性を考えた。テキストカーソルの位置特定には、時と場合において、単純な音で位置情報を伝えた方が良い場合と、音声で位置情報を伝えた方が良い場合とに分けることができる。どの時にどちらの出力情報が良いのかもう少し詳しく探ってみる必要がある。

マウスカーソルの音化については、いろいろな方法を考察して、その中から実際にプログラムを用いて試してみた。この結果、音声で出力させる方法には出力時に時間がかかるという問題があることがわかった。最初、 情報の出力には、 yomi を使って音声で位置情報を伝えようと考えたが、どうしても出力まで間があることや、出力時間もかかってしまうという問題があり、この方法で伝えるよりも単純な音で伝える方がいろいろな面で可能性があるという結論となり、後半では音で位置情報を伝える方法について考察してみた。一つの音のみで情報を伝えるよりも、音と音との比較で情報を情報を伝えた方がわかりやすいということがわかった。また、実際に試してみたものは、マウスカーソルの位置情報の音化についてで、マウスカーソルの動きの音化については試していない。位置情報と動

きの情報を組み合わせて音として簡潔に出力することができれば、さらに進んだ使い道が提案できる。

今後の課題として、点字に関しては点字の文章をコンピュータに入力する方法と、点字読み上げソフトウエアの開発等が挙げられる。テキストカーソルについては、どのような場合に、音と音声、どちらで出力すれば良いかという使い分けの基準を調べること、実際にプログラムを用いて音化を試し、最も良い出力情報を探るということ等が挙げられる。マウスカーソルに関しては、マウスカーソルの動きの音化を実際に試すこと、目標物の特定方法を明確にすること、位置情報と動きの情報を組み合わせて表現する方法について考えること等が挙げられる。また、本稿では画面情報を音化することによる使い道の具体例を示すことができなかった。具体例を示すことができれば、今回報告した考えを発展させて、より簡潔に画面情報を音化できるかもしれない。

参考文献

[1] 村上 真雄:“視覚障害者がパソコンを使えるために”、
http://www.asahi-net.or.jp/~fe3s-mrkm/blind/ (1999)

[2] 厚生労働省:“平成８年身体障害者実態調査及び身体障害児実態調査の概要について”、
http://www1.mhlw.go.jp/toukei/h8sinsyou_9/ (2000)

[3] 阿佐博、遠藤謙一: 点字・点訳入門 (廣済堂出版、1993)

[4] 吉川忠久: アマチュア無線技士国家試験 第 3 級ハム教室、pp172–175 (東京電気大学出版局、1992)

[5] 東京工芸大学工学部光工学科 光情報処理研究室: “簡易点字認識システムの開発”、
http://laplace.photo.t-kougei.ac.jp/research/braille/index.html (2000)

[6] 安居院 猛、永江孝規: X アプリケーションプログラミング 1 X lib 編 (新紀元社、1992)