ウインドウマネージャのユーザインターフェースに関する考察

平成 15 年 03 月 13 日

情報電子工学科
山岸 寛

目 次

1 はじめに **1**

2 ウインドウマネージャの紹介 **1**
2.1 基本のウインドウマネージャ 1
2.2 他の OS に似せたウインドウマネージャ 4
2.2.1 fvwm95 4
2.3 ウインドウをグループ化するウインドウマネージャ 7
2.3.1 fluxbox 7
2.3.2 pekwm 7
2.3.3 treewm 10
2.4 画面を分割するウインドウマネージャ 12
2.4.1 ion 12
2.5 CUI に近いウインドウマネージャ 13
2.5.1 evilwm 13

3 CUI プログラムの紹介 **13**
3.1 shell 13
3.2 screen 13
3.3 FD 14

4 比較と検討 **16**
4.1 ウインドウマネージャ 16
4.2 CUI ツール 17

5 マウス操作に関する実験 **17**
5.1 概要 17
5.2 機能 17
5.3 操作内容 18
5.4 アンケート内容 19
5.5 アンケート結果 20
5.6 考察 21

6 まとめ **21**

参考文献 **23**

概要

コンピュータを使う場合、キーボードやマウス等を使い画面を操作したり、文字を打ち込んだりと、様々な処理をさせる。この操作の過程で図や絵を含んだ場合を GUI での操作、文字のみの場合 CUI を での操作という。UNIX 環境下では、GUI を採用していても CUI での操作が多く、相互の連携がなされていないように感じた。このことから、 ウインドウマネージャと CUI ツール群を比較、考察し、それらの抱える問題を指摘し、その操作性を向上させること目的としてマウス動作のキーボードエミュレーションを行うプログラムをつくって実験し、それによって得られた問題点を挙げる。

1　はじめに

ユーザインターフェースには CUI と GUI がある。CUI とは Character User Interface 略であり文字入力 (コマンド) のみで処理を行うというものである。GUI は Grafical User Interface の略で図形や絵などを用いて画面を表す。特徴として、CUI はキーボードに慣れていないと扱いづらいこと、コマンドを知っていなければ扱えないこと、慣れてくると同じような作業をする場合威力を発揮することが挙げられる。一方 GUI の方は、慣れてなくともマウスの使い方を知っていれば何とかなること、慣れても CUI のようにはいかず、同じような作業を繰り返さなければならないことが挙げられる。最近の OS では大体が GUI を採用していて UNIX でも採用する人がほとんどであろう。この GUI 機能提供機構が X であり、その一部分に ウインドウマネージャは属している。ウインドウマネージャの提供する機能は、ディスプレイに表示されるウィンドウの挙動、大きさ、デザインの変更などを行う。現存する機能を紹介しようと思う。

2　ウインドウマネージャの紹介

2.1　基本のウインドウマネージャ

まず、これから議論していく上で、 X に標準で付いてくる、最も基本となる twm というウインドウマネージャの機能について紹介していく。まず、図 1 をご覧いただきたい。この図は、twm の全体を撮った Screenshot である。この Screenshot をみながら、twm の機能を紹介していく。

twm はウインドウを図 2 のように表示し、上部にタイトルバーを付属させる。このタイトルバーにはウインドウの名前を表示させている。また、左にアイコンか、右にサイズ変更の機能をもつボタンを配置している。

図 3 はアイコン化した状態を示している。この図ではネットスケープと mule (UNIX のエディタ、MS-Windows にも別名で移植されている) がアイコン化されている。図 3 をみて分かるように、ネットスケープのアイコンには図形を、 mule のアイコンには文字列を表示している。 twm がそのアプリケーション用のアイコンを持っていた場合にはネットスケープのように、持っていなかった場合 mule のように表示される。 mule がアイコン化された場合、表示される文字列は、タイトルバーに表示されていた文字列である。

ウインドウのサイズ変更に関しては、 MS-Windows のようにウインドウの境界線をドラッグして拡大、縮小ということはできない。 twm の場合は、タイトルバーの右のボタンをドラッグして拡大しなければならない。縮小に関しては、面倒だと思われるかも知れないが、一度、縮小したい方向を一度拡大してからでないと、縮小はできない。

最後に、ルートメニューという機能について紹介する。ルートメニューは、ルートウインドウ (背景に表示された侵食されないウインドウ) でマウスの左ボタンをドラッグすると現れるメニューのことをいう。図 4 はルートメニューを表したものである。図のようにルートメニューは様々な文字列を表示させている。この中のどれ化にマウスカーソルを会わせ、マウスのボタンを離すと、いろいろな機能が実行される。また、ルートメニューに

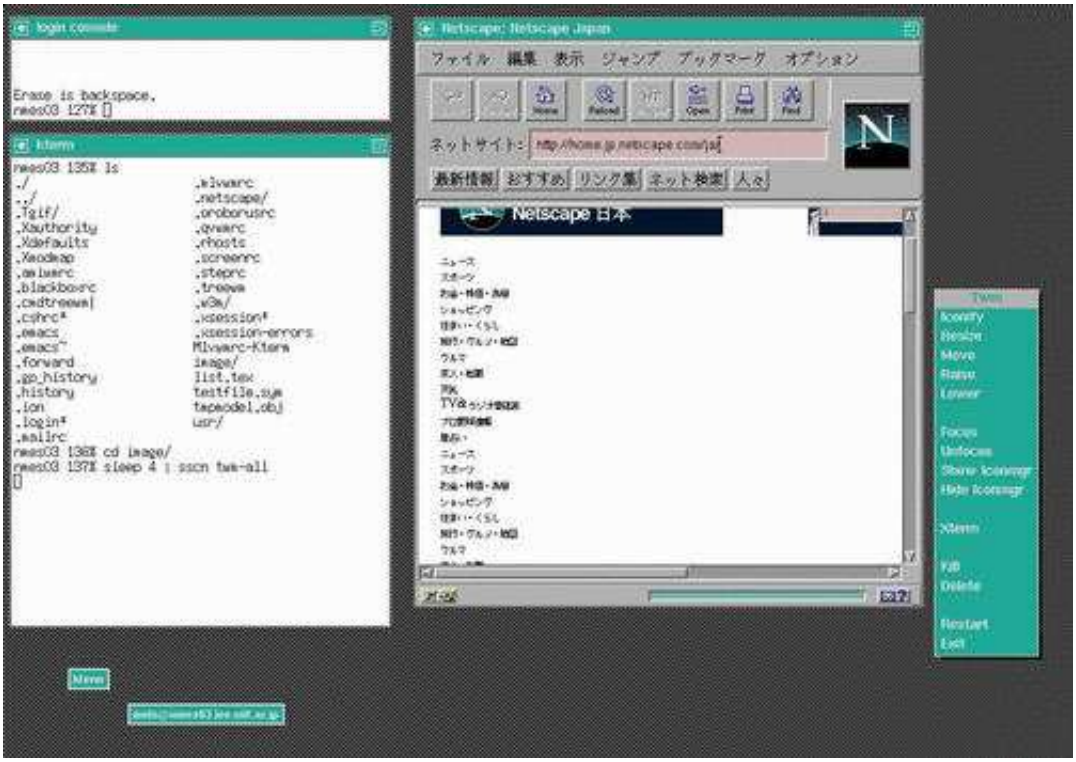
Erase is backspace.
Netscape: Netscape Japan
ファイル 編集 表示 ジャンプ ブックマーク オプション
Netscape 日本
Move

Fig. 1 twm

Fig. 2 twm のウインドウ

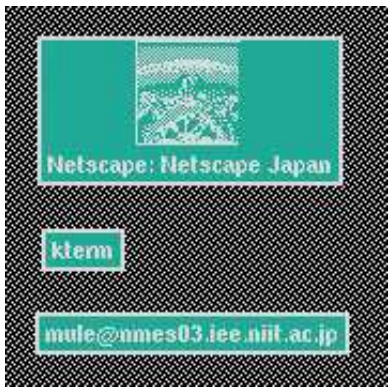

Fig. 3 twm のアイコン

Fig. 4 twm のメニュー

表示された機能を削除したり、追加したりを設定ファイルの変更により実現できるようになっている。

これらが、基本的な twm の機能である。現在、ウインドウマネーは多種多様に存在していが、これらの機能を改良したり、新たに追加したりして、様々な機能を提供している。次にそれら様々なウインドウマネージャの機能を紹介していこうと思う。

2.2 他の OS に似せたウインドウマネージャ

2.2.1 fvwm95

fvwm95 は MS-Windows に似せた作りになっているため、機能も MS-Windows に近い機能を持っている。 fvwm95 の様子は 図 5 のようになっている。

fvwm95 のウインドウは図 6 のようになっている。fvwm95 のウインドウに関して、twm のウインドウと違う機能は、上部のタイトルバーに付属されたボタンの種類が増えたこと、その機能はウインドウを閉じる機能とウインドウを最大化する機能である。また、サイズ変更に関するボタンはなくなり、ウインドウの境界線をドラッグすることによりウインドウのサイズを変更可能となった。

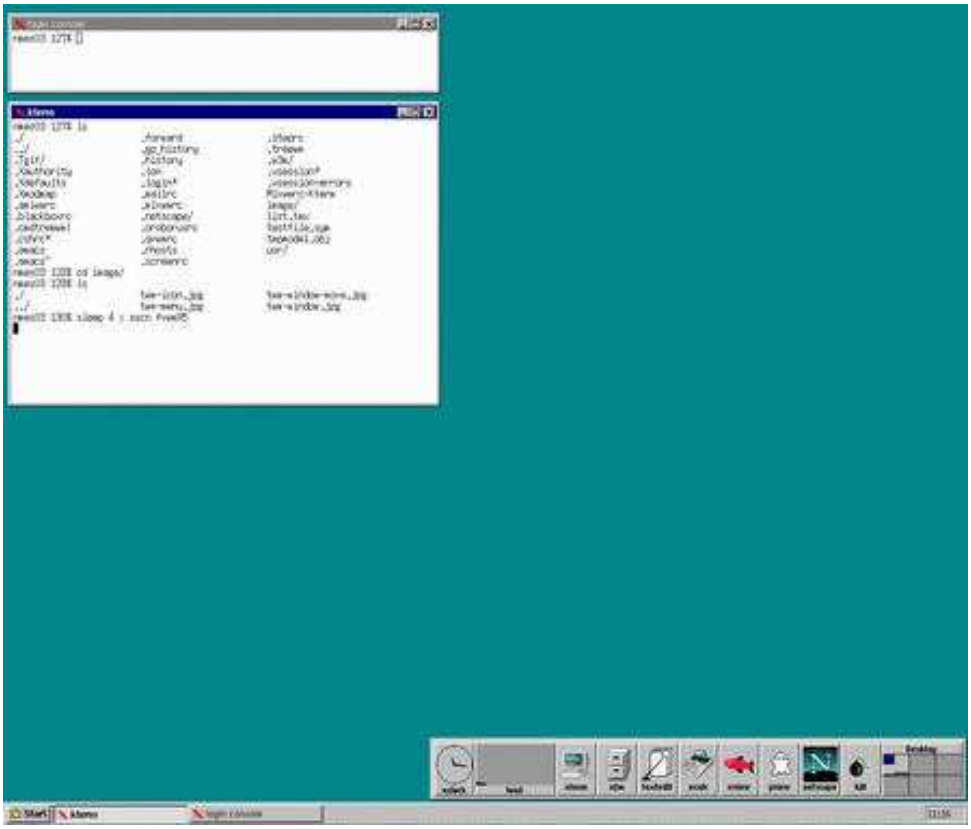

Fig. 5 fvwm95 の様子

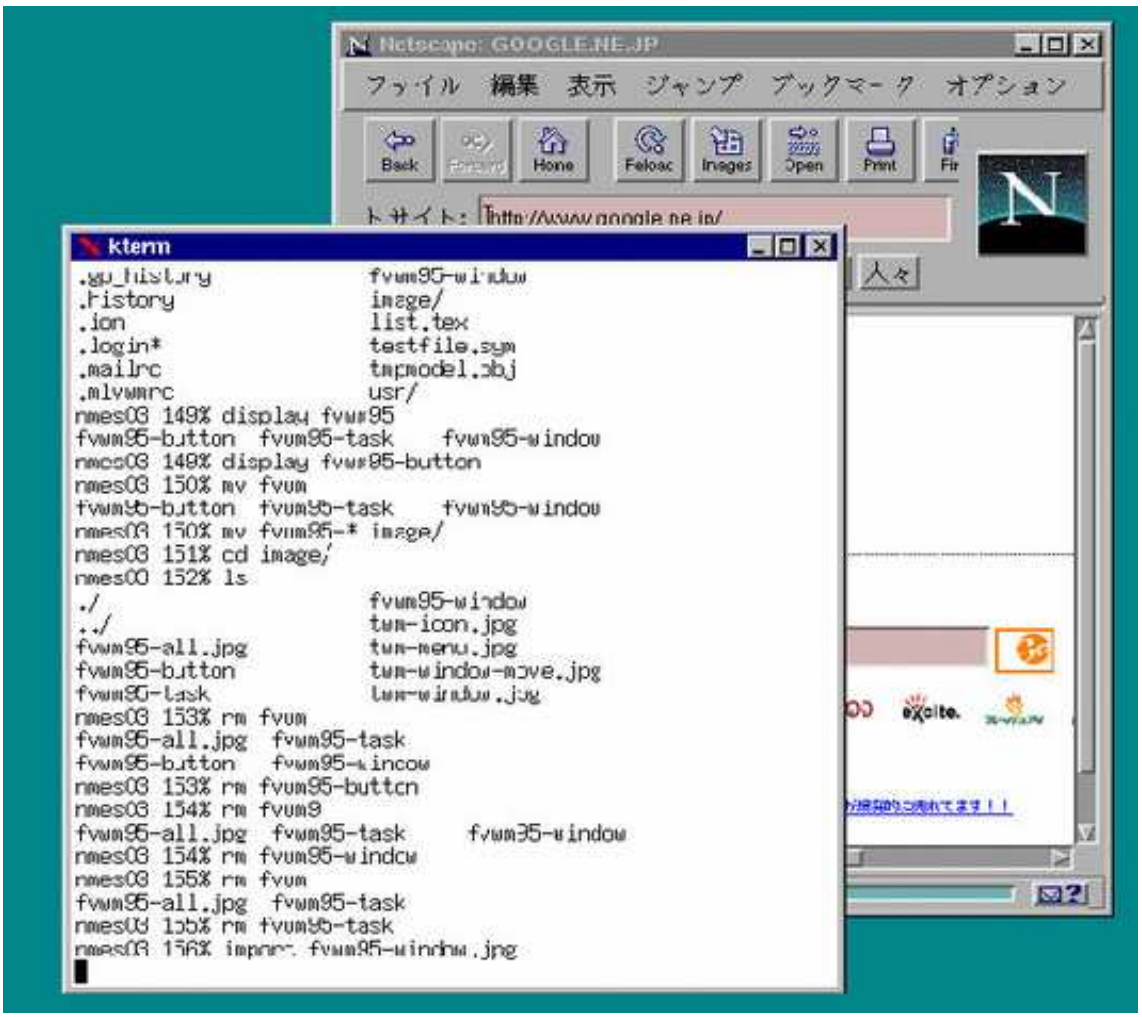

Fig. 7 fvwm95 のタスクバー

Fig. 8 fvwm95 のボタンバー

fvwm95 の機能と MS-Windows が提供する機能で最も良く似た機能がタスクバーでのウインドウの管理である。このタスクバーでは、ウインドウの位置ウインドウの表示、非表示、ウインドウがアクティブであるかどうかを管理している。タスクバーに登録されたウインドウは、それぞれの名前がつけられ左から順に区切られボタンのように配置される。このボタンをクリックすると割り当てられていたウインドウにフォーカスを移し、アクティブにする。アクティブになったウインドウのボタンは、へこんだように表示される。このアクティブウインドウの変更に関しては、MS-Windows のような `Alt-Tab` での変更が用意されている。しかし、 fvwm95 では、一番新しくアクティブになった順で二つまでしか記憶しておかず、その二つのみで変更可能となっている。

図 8 は fvwm95 のボタンバーの図である。このボタンはアプリケーションを実行する機能をもったり、アプリケーションの実行画面をうつしたり (xclock、xload) 、仮想ディスクトップを管理する機能を持っていたりする。

2.3 ウインドウをグループ化するウインドウマネージャ

2.3.1 fluxbox

fluxbox の全体図は図 9 の用になっている。普通のウインドウマネージャのウインドウとは異なり、ウインドウのタイトルバーの上部に長方形の物体が配置されている。個々では、仮にこれをタブと呼ぶ。このタブを、マウスの中ボタンでドラッグし、ウインドウのタイトルバーまで持っていき、ボタンを離すと、持っていったウインドウのタブが他ウインドウのタブの横に配置される。その状態を示した図が図 10 である。そして、もっていったウインドウが見えなくなり、これで、ウインドウがグループ化される。見えなくなったウインドウは、その情報を示したタブをクリックすることによって参照される。

2.3.2 pekwm

fluxbox とは違いタブを追加してグループ化を図るのではなく、タイトルバーを分割しながらグループ化を図るウインドウマネージャが pekwm である。pekwm の全体図を示す図は、図 11 で、グループ化を果たした様子を示す図は、図 12 である。ウインドウをグループ化する方法については、 fluxbox とさほど変わらない。

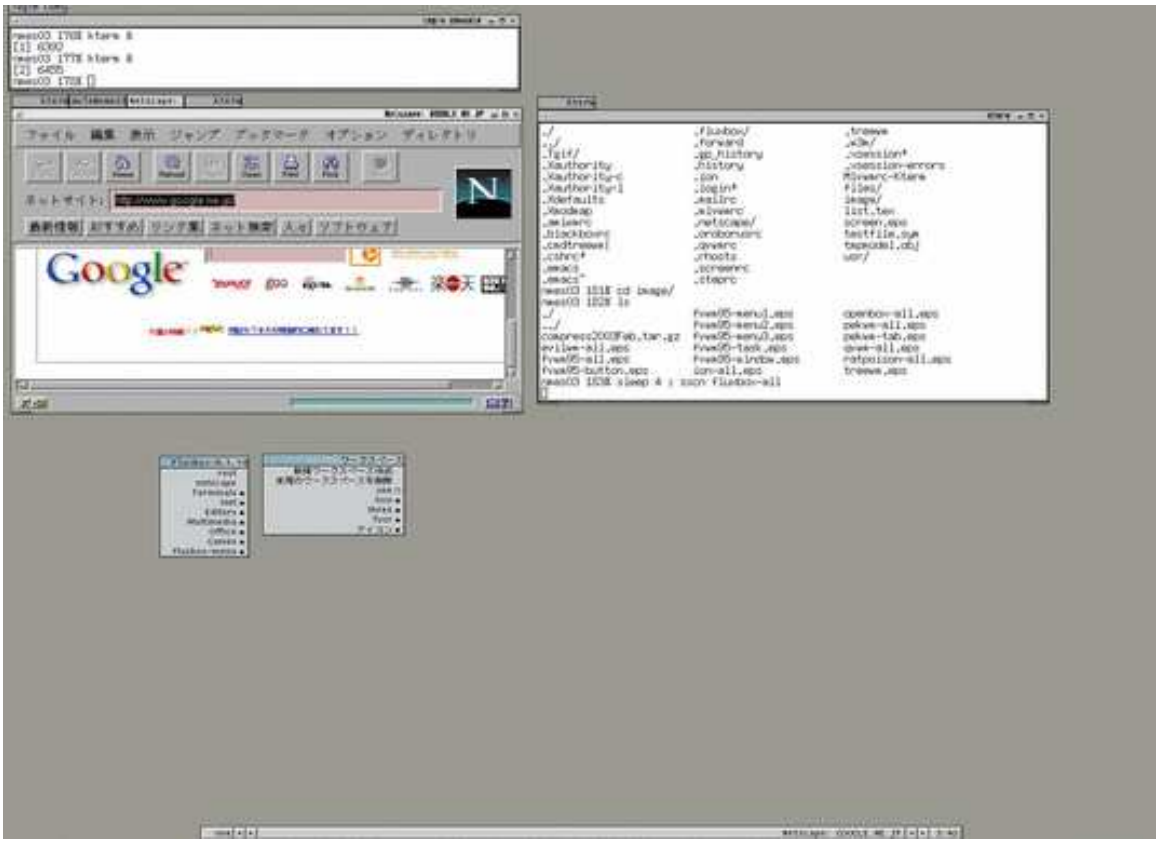

Fig. 9 fluxbox

Fig. 10 fluxbox のグループ変

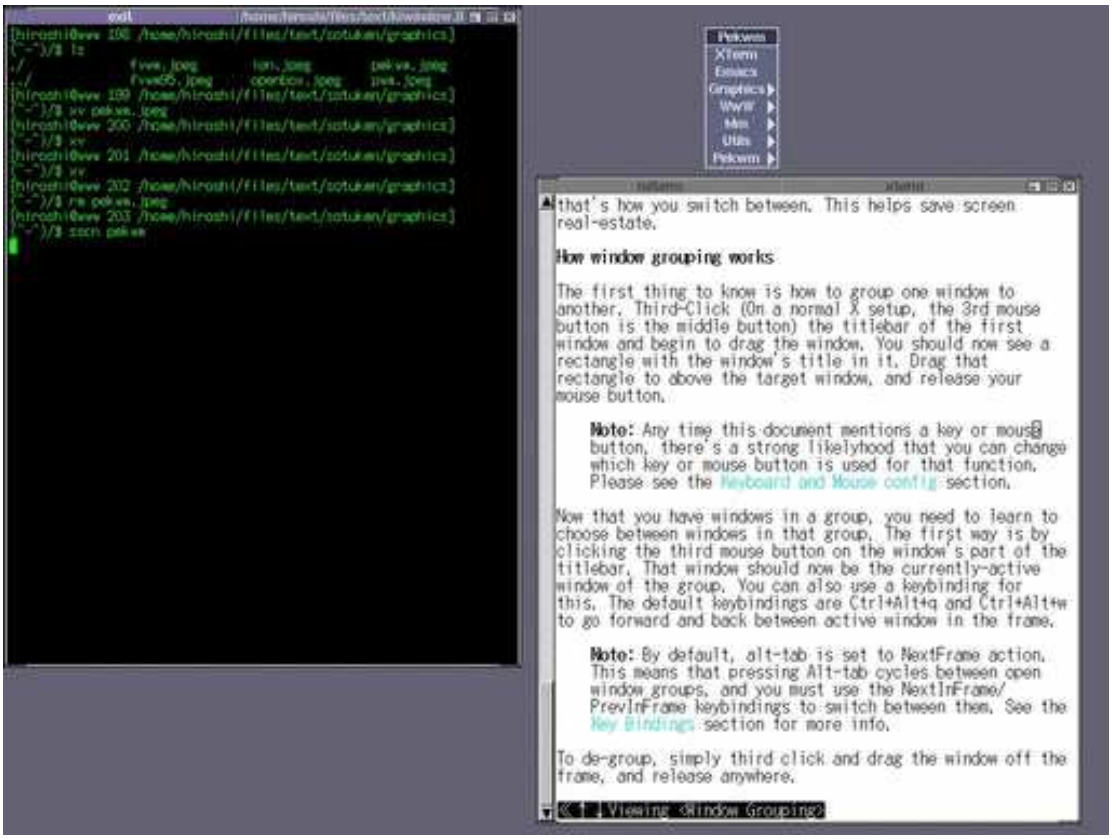

Fig. 11 pekwm の様子

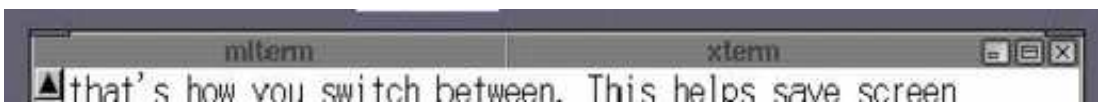

Fig. 12 pwkwm のグループ化

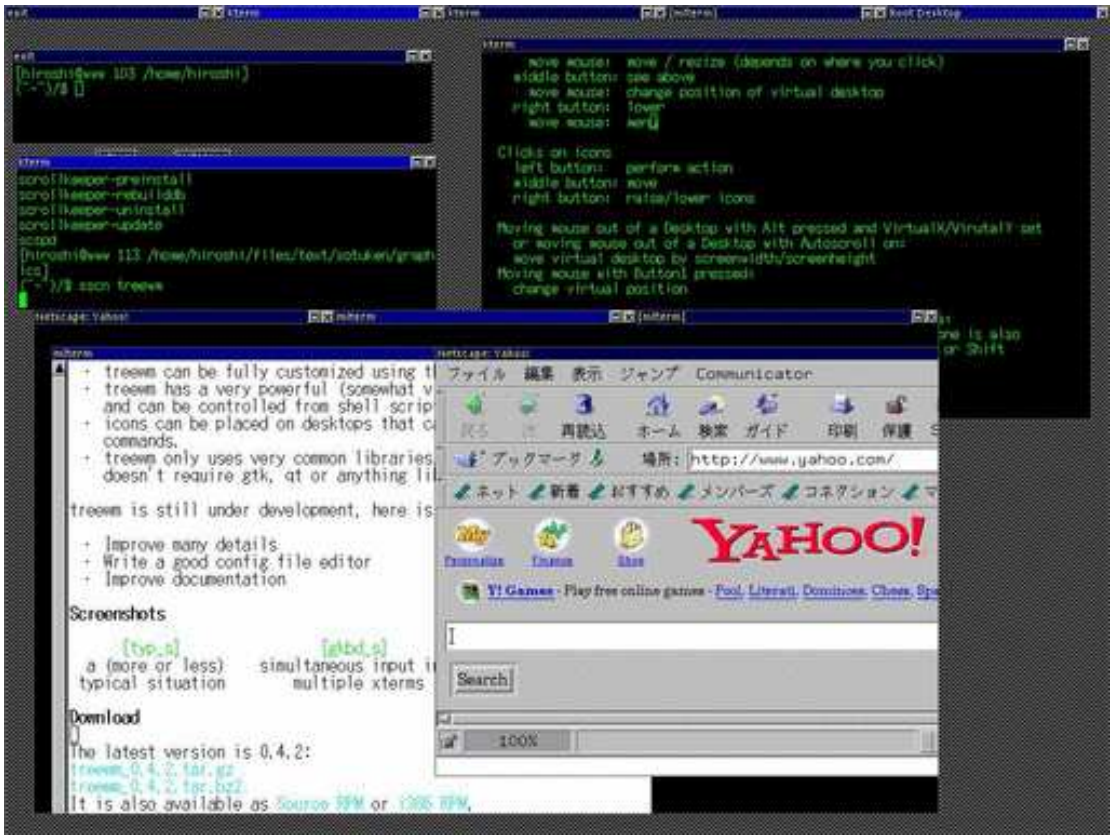

Fig. 13 treewm の様子

2.3.3 treewm

図 13 は、 treewm の全体の様子を示したものである。このウインドウマネージャは、ルートウインドウの中にルートウインドウを配置し、お互いの情報を干渉しあわないようにする機能をもっている。このルートウインドウを作成するためには、本物のルートウインドウの中でダブルクリックすると作成できる。そして、その中でアプリケーションを起動したい場合、その疑似ルートウインドウをアクティブにし `Alt+Spece` でててくるアプリケーションランチャで起動する。図 13 の中央下に表示されたウインドウがその状態を示している。この隔離されたウインドウの中にはネットスケープと kterm が立ち上がっているのがわかるだろう。その二つの中で、ネットスケープの方はウインドウが途中までしか表示されていない。これが、干渉しあわない機能である。

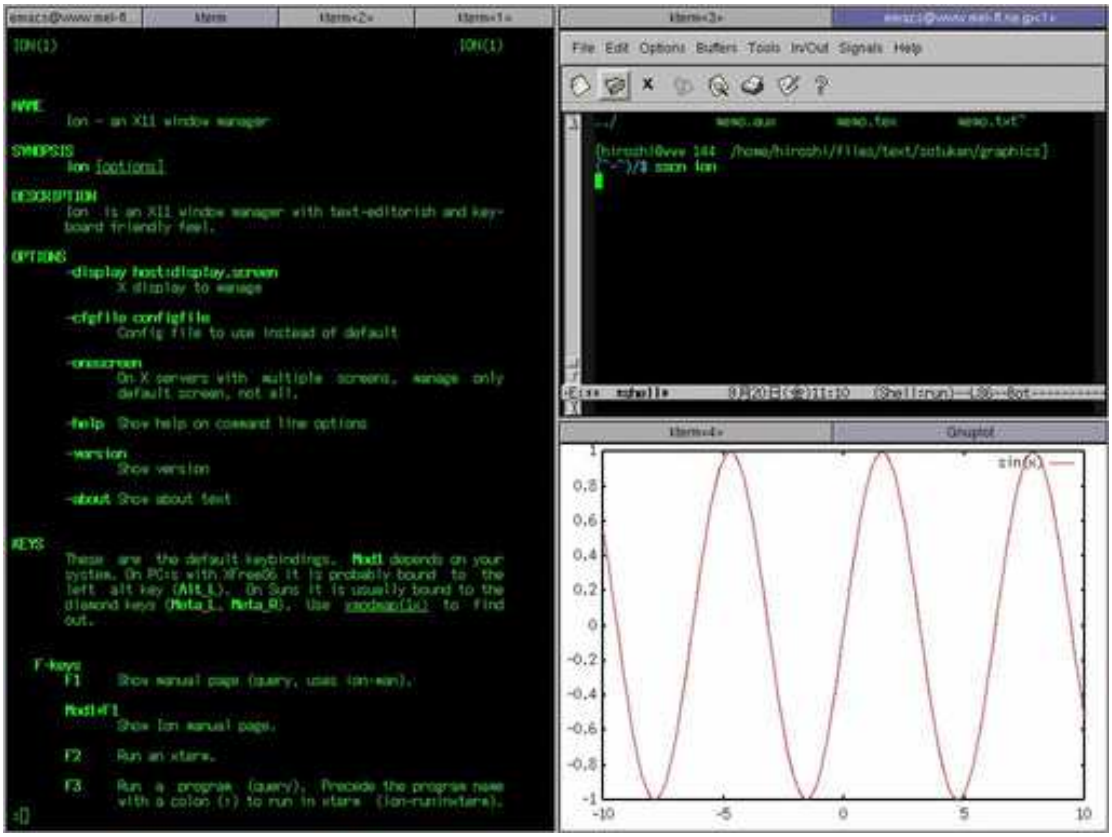

Fig. 14 ion の様子

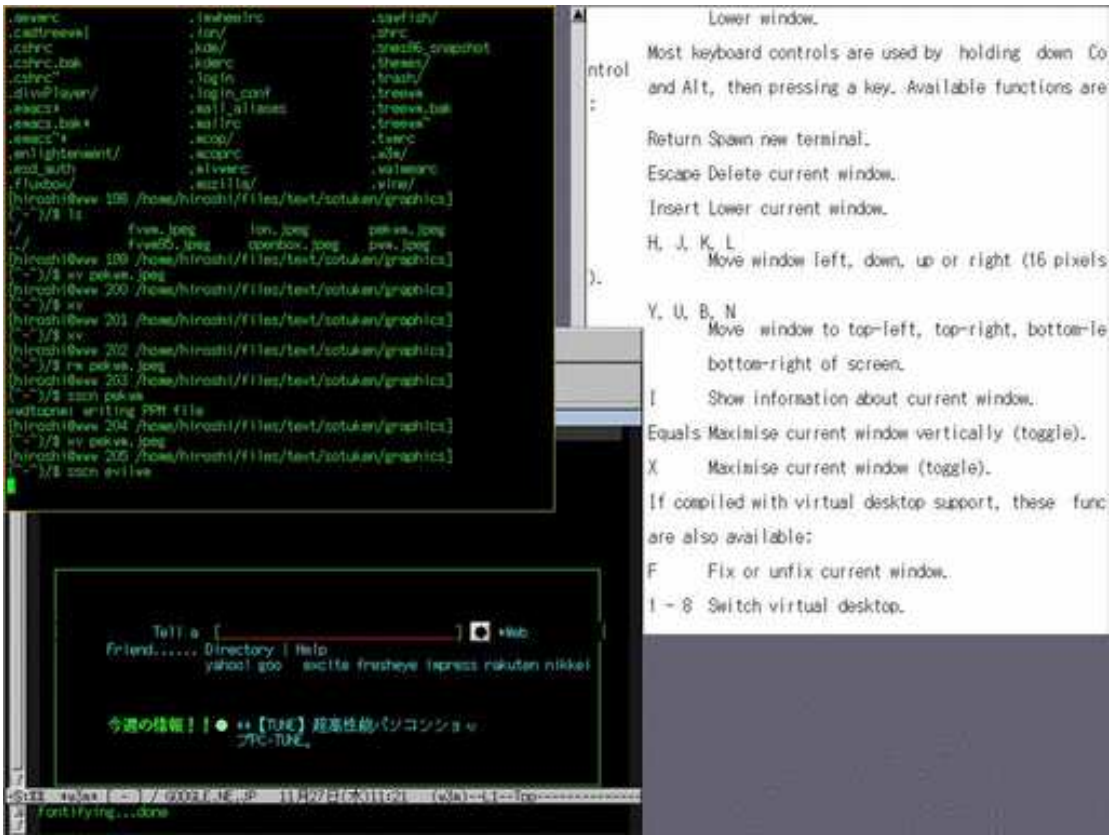

Fig. 15 evilwm の様子

2.4 画面を分割するウインドウマネージャ

2.4.1 ion

図 14 は ion を示す図である。このウインドウマネージャは画面を分割しながら、新たな作業領域を確保することができる。また、その作業領域に複数のアプリケーションを納めることができる。図 14 は、三個の作業領域に分割されていて、それぞれ、複数のアプリケーションが納められている。どのアプリケーションが納められているかは、それぞれの作業領域の上部に配置されたタイトルバーの内容によって知ることとができる。この分割する作業はすべてキーボードにより操作されマウスを使用せず、この作業領域のサイズ変更のみマウスを使用する。

2.5 CUI に近いウインドウマネージャ

2.5.1 evilwm

このウインドウマネージャには、タイトルバーなどが見当たらない。ウインドウの操作は、キーボードによって行う。

3 CUI プログラムの紹介

3.1 shell

シェルは入力されたコマンドを読み込んで OS に渡すという仕事をしている。そのコマンドを渡す際に、いろいろな仕事をしてくれる。以下にその機能を述べる。

- 補完機能

  この機能は、途中まで打ったコマンドやファイル名を該当するコマンドやファイル名に変換してくれる機能である。例えば `note-20021002.txt` と言うファイル名があるとすると `note-` と打った後 <tab> を打てば `note-20021002.txt` と変換してくれる。ただし、当該ファイル名が複数あると変換してくれない。設定さえすれば該当ファイル名が複数ある場合表示してくれる機能もある。

- 置換機能

  この機能は、ある性質をもったファイル群を一括にして扱う機能である。例えば、`rm abc*` として改行を行えば、ファイル名が `abc` のファイルを含む頭文字 `abc` のファイル群が全て削除される。

- パイプ、リダイレクション

  通常、コマンドを入力して実行すると、画面に表示されるが、このデータを画面に表示せずに別のコマンド渡したり、ファイルに保存又は追加する機能がある。これをパイプ、リダイレクションという。

- ヒストリ機能

  シェルは過去に実行したコマンドを保存している。これをヒストリ機能と言い、過去に入力したコマンドを楽に再度実行できる。

3.2 screen

Screen (図 16) とは、一つの端末の中に仮想的な端末を作り、操作することができるアプリケーションである。 Screen の持つ機能には以下のようなものがある。

- 複数の仮想端末を 1 つの端末で管理することができる。

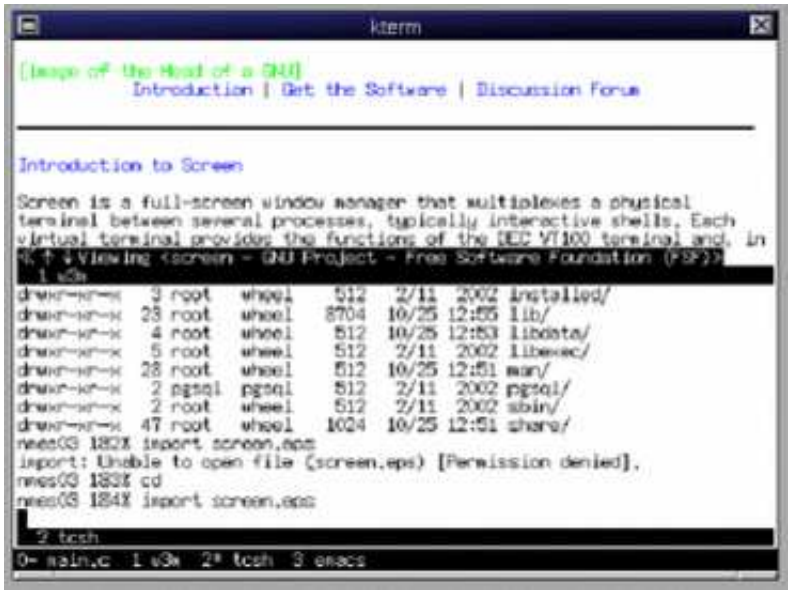

Fig. 16 screen

これは、§2 で紹介したウインドウマネージャの機能にグループ化する機能があったが、それと似たことを実現する。

- 仮想端末間でコピーやペーストができる。

  Screen はマウスを必要とせずに、端末間での文字列のコピーやペーストとを実現している。Screen ではこの作業を行う場合、コピーモードというモードに移行して行う。このコピーモードには、コピーやペーストの他に、流れてしまった情報をみたり、文字列を探索したり、画面のログを取ったりと、 様々な機能を持ち合わせている。

また、この Screen の機能を真似たウインドウマネージャも存在する。それが、図 17 に示された ratpoison というウインドウマネージャで、このウインドウマネージャを使うことにより、 Screen を複数のウインドウで使う感覚をもつことができる。

3.3 FD

FD は、ファイルやディレクトリを管理するツールである。 図 18 が FD を示す図である。FD は起動したとき、現在のディレクトリのファイルまたはディレクトリの一覧を表示する。図のなかで文字列が反転表示されているのがカーソルである。このカーソルの位置にあるファイルまたはディレクトリを対象として、入力されたキーによって様々なコマンドが実行される。そのコマンドは、ファイルのコピーや移動、閲覧、実行といったファイル操作を実行する。また、ディレクトリをツリー化して視覚的に見せることや、アーカイブ内のファイル名探索などの機能も持っている。

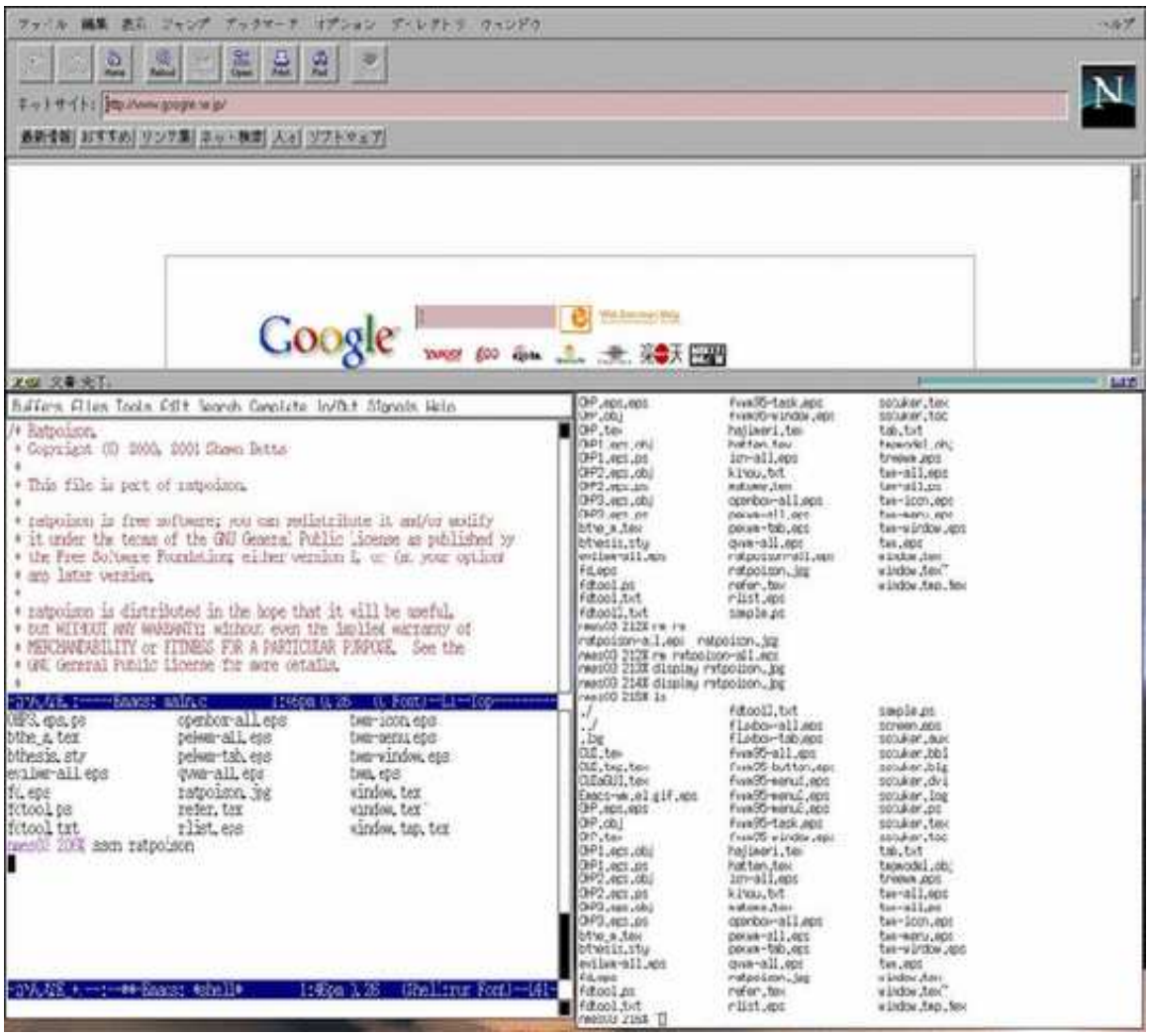

Fig. 17 ratpoison

Fig. 18 fd

4 比較と検討

4.1 ウインドウマネージャ

- 他 OS に似せるウインドウマネージャ

  他 OS に似せてあるため、その OS を使用していたのならば、操作はとても楽になる。そして、 UNIX 特有のカスタマイズできるということも相まって、快適性を味わえるだろう。しかし、その操作法のみにとどまり、他の操作を覚えようとはしないかもしれない。

- グループ化をするウインドウマネージャ

  同種の内容のウインドウをグループ化することによって画面内をすっきりさせることができる。しかし、スクリーンショットをみてもらえれば分かると思うが、タイトルバーやタブにそれらウインドウの情報が名前でしか読み取れないから、それらのウインドウの情報を細かく知ることはできない。ただし、同種のウインドウをグループ化すれば、多少は改善可能である。

- 画面分割をするウインドウマネージャ

  画面を分割していくから、だんだんと端末の情報が削られていくので、画面を分割できるのはせいぜい 4 から 5 くらいまでだろう。ion は、それぞれのフレームにタブを付け、ratpoison は screen の機能をそのまま付けることで、その問題を回避している。

4.2 CUI ツール

CUI ツールは、端末で全てのことを行うようつくられている。そのため、無駄な移動が少なくすばやい操作が出来ている用に感じられた。evilwm はウインドウ操作をキーボードで行う CUI に近いウインドウマネージャと紹介したが、使いやすいと思えるようなものではなかった。

5 マウス操作に関する実験

5.1 概要

CUI ツールではマウスを使用してなくても操作性は悪く感じられなかった。それは、CUI ツールがマウスを使用しない環境のもとでつくられたからで、このことを現在のウインドウマネージャに適用しマウスをキーボードで動かしたら操作性はどうなるだろうか。テスト用プログラムを作成して実マウスとの比較する。また、 MS-Window98 はこのプログラムと同等なことを行える機能を有している。この MS-Windows98 の機能とも比較してみる。

5.2 機能

X 上で起動し、全画面を使ったウインドウの中でマウスをエミュレートするプログラムを作成し実行を行う。以下にその機能を記す。

- 方向キー
  - j:カーソルを下に移動
  - k:カーソルを上に移動
  - l:カーソルを右に移動
  - h:カーソルを左に移動
  - u:カーソルを左上に移動
  - i:カーソルを右上に移動
  - n:カーソルを左下に移動
  - m:カーソルを右下に移動
- Control キー + 方向キーでカーソルの大幅移動
- +,-でカーソルの加速度変更
- s キーでシングルクリック
- w キーでダブルクリック

- Control + Alt + 方向キーでドラッグ

MS-Windows98 の機能を以下に記す。

- 方向キー
  - 2:カーソルを下に移動
  - 8:カーソルを上に移動
  - 4:カーソルを右に移動
  - 6:カーソルを左に移動
  - 7:カーソルを左上に移動
  - 9:カーソルを右上に移動
  - 1:カーソルを左下に移動
  - 3:カーソルを右したに移動
- Control キー + 方向キーでカーソルの大幅移動
- Shift,Control でカーソルの加速度を変更
- 5 キーでクリック
- 5 キー 2 回でダブルクリック
- 0 キーでドラッグ開始、dot キーでドラッグを終わる

5.3 操作内容

以下の操作を、それぞれの環境で実行し、マウスの操作感、クリックの感触、キー配列の使いやすさを比較してもらう。

- 作成したプログラム
  1. カーソルを左上まで移動させる
  2. ダブルクリックをする
  3. カーソルを右上まで移動させる
  4. ダブルクリックをする
  5. カーソルを右下まで移動させる
  6. ダブルクリックをする
  7. 左上にカーソルを移動させる
  8. シングルクリックをしてウインドウを表示させる

9. カーソルをタイトルバーに移動させる
10. ドラッグしてウインドウを右回りで一周させる
11. タイトルバーのボタンでウインドウを終了させる

- MS-Windows の同機能

1. カーソルを左上まで移動させる
2. ダブルクリックをする
3. カーソルを右上まで移動させる
4. ダブルクリックをする
5. カーソルを右下まで移動させる
6. ダブルクリックをする
7. 左上にカーソルを移動させる
8. マイコンピュータをダブルクリックしウインドウを表示させる
9. タイトルバーまでカーソルを移動させる
10. ドラッグしてウインドウを右回りで一周させる
11. タイトルバーのボタンでウインドウを終了させる

5.4 アンケート内容

アンケート内容は、

1. 実マウスとの比較
   (a) マウス移動
   (b) クリック
   (c) ダブルクリック
   (d) ドラッグ
2. MS-Windows の同機能と比較
   (a) マウス移動
   (b) クリック
   (c) ダブルクリック
   (d) ドラッグ
   (e) キー配列

の各項目に対する最良 (2)、良 (1)、普通 (0)、悪 (-1)、最悪 (-2) の 5 段階評定とその理由を書いてもらった。

5.5 アンケート結果

アンケートを行い、得られた 5 段階評定の結果を以下に記す。

-	1				2				
被験者	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)	(e)
A	-2	0	1	-2	-1	0	0	-1	0
B	0	0	1	-1	0	1	2	0	-1
C	-2	0	1	-1	1	0	0	-2	-1
D	-2	2	2	0	-2	0	1	-1	-2
E	-2	1	1	0	0	1	1	1	0
平均値	-1.6	0.6	1.2	-0.6	-0.4	0.4	0.8	-0.6	-0.8

理由として挙げられた以下に記す。

1. 実マウスとの比較

- 操作性に関して
  - マウスの方が早い移動が可能だが、細かい移動はマウスより良い (0)
  - 実マウスで描く滑らかな曲線にはかなわない。(-2)
  - マウスになれたせいかも知れないが特に移動距離が多ければ多いほど実マウスの移動がすごく楽に感じる。(-2)
  - 移動方向に関しては自由度が少なすぎるため、気にしながら動かさなければならない。(-2)
  - 場所を確定する場合が難しい。(-2)
- クリックに関して
  - マウスでもキーボードでも特に変わらない。(0)
  - s ボタンはすぐに連想できて良い。(2)
- ダブルクリックに関して
  - マウスでダブルクリックした時のカーソル位置がずれることがないので良い。(1)
  - 正確さなどから多少の利点はあるがそれほどでもない。(1)
  - ボタン 1 つでのダブルクリックは良い。(2)
  - クリック回数が減ったことにより良。(1)
- ドラッグに関して
  - 実マウスの方が使いやすい。(-1)
  - 3 つのボタンより 2 つの方が良い。(0)
  - マウス移動で使い勝手が悪い。(-2)

2. MS-Windows の同機能と比較

   - 操作に関して
     - 大きな差は感じられない。(0)
     - どちらも使い勝手が悪い。(-2)
   - クリックに関して
     - 比較してみて違いが分からなかった。(0)
   - ダブルクリックに関して
     - 2 回押すよりは良い。(1)
   - ドラッグに関して
     - ドラッグが押しつづけてなければならないのが面倒だった。その点は MS-Windows の最初と最後だけ押す機能のほうが使いやすい。(-1)
   - キー配列に関して
     - MS-Windows のテンキーでの操作方法のほうがわかりやすい。(-1)
     - hjkl はわかり難い。(-1)

5.6 考察

人がマウスを動かす場合、上だったら上部へマウスを動かす。このとき腕を動かす方向とマウスカーソルが一緒に同じ方向へ動くためマウスの方が使いやすいと感じたのではないか、キーボードだと、例えマウスカーソルの移動する方向をちなんだキーに変更しても、キーを押すという操作が絡んでくる。このため違和感が生じたのではないだろうか。プログラムのマウスをクリックする機能が好評であったのもこういった理由からなのかもしれない。そういった理由であれば、ウスカーソルと同等な機能をもつ方法を実現するためには、マウスボタンのエミュレーション以外のマウスカーソルと同等な機能ともつ別な方法が必要になってくる。一般的なウインドウマネージャでマウスの機能というと、ウインドウを動かす、ウインドウのリサイズがある。この二つの機能をかなえる為には、ion などのようなそういった概念のないウインドウマネージャをつくるというのが考えられる。

6 まとめ

今まで様々なウインドウマネージャや、CUI ツールを紹介してきた。そのなかで evilwm はウインドウの操作をキーボードで行う CUI らしきものであった。だが、その操作性は CUI ツール の方が良かった。それは、CUI は CUI で使うことを前提としていたからで、evilwm は GUI を前提として、 CUI を使用しようとしていたからではないかと思い、 CUI で GUI を操作するようにしたらどうなるか考えた。ここでは、 キーボードでマウスを操

作した場合について、プログラムを作成し、アンケートを取ってみた。その結果、操作性については良いと思われる人はいなかった。ただ、マウスのクリックに関してはそこそこの評定をもらったのに対し、マウス操作をする場合カーソルと同じ方向へ腕が動かないとだめだということにした。

今回、ウインドウマネージャの操作向上ということで話を進めてきたが、ウインドウマネージャは様々な目的でつくられ、様々な目的で使われている。そのため範囲が広く何を作っていいのか分からなかった。この範囲を狭め、どんな目的のために、どんな使われ方をするのかといったことが今後の課題である。

参考文献

[1] 矢吹 道朗, 江面 淳: “ウインドウマネージャ徹底解説”, TECHNO PRESS

[2] 松田 晃一, 歴本 淳一: “入門 X Window”, ASCII, pp15–30

[3] 山口 和紀, 千 旭, 中村 敦司, 新城 靖, 西山 博泰, 古瀬 一隆, 石川桂治, 佐々木 重雄, 林 謙一, 萩原 一隆, 鈴木 孝幸, 黒石 一宏: “The UNIX Super Text[上]”, 技術評論者, pp489-523

[4] Zinnia: “Screen を使う”, http://risky-safety.org/ zinnia/screen/

[5] Matt Chapman: “Window Managers for X”, http://www.PLiG.org/xwinman/

[6] Shirai: “FDclone なページ”, http://hp.vector.co.jp/authors/VA012337/soft/fd/

[7] 木下 凌一, 林 秀幸: “X-Window Ver.11 プログラミング”, 日刊工業新聞社