スナッピングを利用した無階層メニュー

小沢邦雄[†]　増井俊之[†]

GUI で現在広く利用されている階層型メニューは大量のエントリを扱うのに便利であるが，階層構造についてあらかじめ充分理解しないとうまく使いこなせないという問題点がある．階層構造を利用せず，全てのメニュー項目を一列に並べて表示し，スナッピングを用いて項目を検索/選択する「全部入りメニュー」を提案する．

FlatMenu: Handling many menu entries without hierarchy

Kunio Ozawa[†] and Toshiyuki Masui[†]

We propose a new GUI widget calld FlatMenu, with which users can handle thousands of entries without using hierarchical data structure. All the entries in FlatMenu are simply laid out as a list, and users can select an item by scrolling the list with fluid snapping behavior.

1. はじめに

図 1 のようなプルダウンメニューや TreeView のような GUI 部品において階層構造が広く利用されている．階層型構造を利用すると小さな領域で多くの項目を扱うことができるが，以下のような不具合も存在する．

(1) 階層の奥の方に存在する項目を見つけるためには項目を開いて探す必要がある．階層構造についてよく理解していれば問題ないが，構造をよく知らない場合，項目が階層構造のどこにあるのかを捜すことが難しい．

(2) アプリケーション毎に項目の名前や位置が異なる可能性があり，項目が存在しない場合もある．

本論文では，このような問題を解決する「全部入りメニュー」を提案する．全部入りメニューは階層構造を利用せず，あらゆるメニュー項目をフラットに並べて配置し，SmoothSnap[1] によるスナッピング操作を利用することにより目的の項目を選択する．

2. 全部入りメニュー

2.1 フラットな構成

全部入りメニューでは，システムが持つあらゆるメニュー項目がひとつのメニュー内に網羅されている (**図 2**)．例えば，Mac の Firefox では「新規タブを開く」という項目が「ファイル」という名前のメインメニュー内にあり，「拡大」という項目が「表示」というメインメニュー内の「ズーム」というサブメニュー内にあるが，仮に Firefox が全部入りメニューを使用していたなら，これらの項目は同じ一つのメニュー内に並ぶことになる．

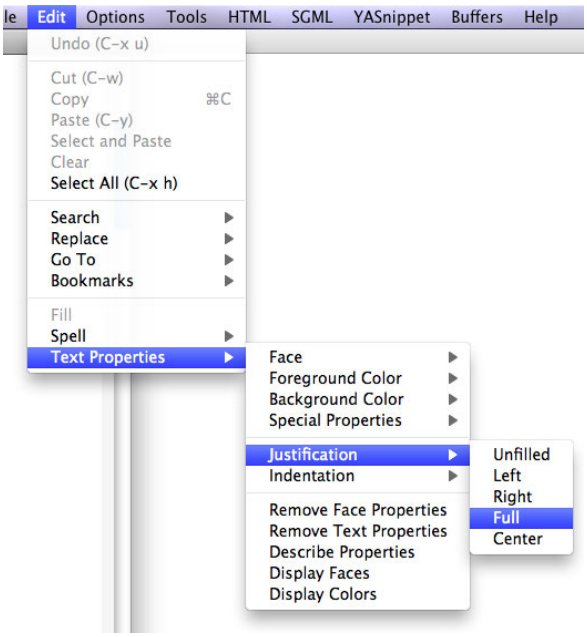

図 1　階層メニューの例
Fig. 1　An Example of Hierarchical Menu

2.2 複数表現の利用

通常のメニューシステムでは，ひとつの機能はひとつのメニュー項目に対応づけられているが，全部入り

† 慶應義塾大学 環境情報学部
Faculty of Environment and Information Studies, Keio University

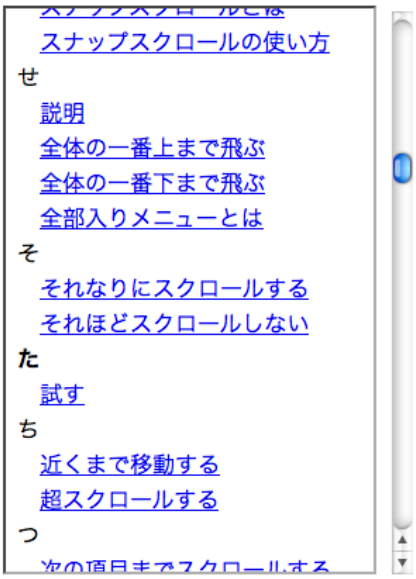

図 2 全部入りメニュー
Fig. 2 Flat Menu

メニューでは，同じ機能に対して複数の表現項目を用意する．例えば，現在のページを印刷するタスクは「プリント」「印刷」「プリンタに出力」など様々な表現が可能であるが，このような全ての表現をひとつのメニュー内に用意しておく．

メニューの作成者が項目の名前をごく少数に絞ってしまうことは，そのシステムのユーザにシステム特有の表現への慣れを強要することにもなりかねないが，全部入りメニューではそれらを絞らないためユーザが思い浮かべた表現のまま直感的に目的の機能を探すことが可能である．

例えば，図 2 では，「それほどスクロールしない」「超スクロールする」など，一般的なメニューではあまり使われないであろう表現も用意している．

2.3 SmoothSnap

全ての項目を一つのメニューに表示する場合，目的の位置までスクロールする方法が問題になる．項目数が膨大なメニューを通常のスクロールバーでスクロールすると，ノブをわずかに動かした場合でも表示は大きく移動するので微調整が難しく，また画面遷移も激しくなるためスクロール中に自分の位置が把握しづらいが，SmoothSnap を利用すればこのような問題が発生しない．SmoothSnap には以下のような特徴がある．

- ノブを微量だけ操作した場合は細かい粒度で連続的に値を変化させる．
- ノブを沢山移動した場合は重要なポイントにスナッピングする．

図 3a の状態からノブをドラッグして少しだけ下に移動させると，通常のスクロールのように連続的にスクロールされ，図 3b の状態のようになる．ノブのドラッグを離さないまま更に下にスクロールすると，**図 4**a のように重要な位置までスナッピングされる．図 3，図 4 の例に使用しているメニューでは項目が五十音順に並んでおり，一気にあかさたなの次の行までスナッピングするよう設定しているので，た行の一番上までスナッピングしている．そのままノブを下げ続けると，メニューは次のな行までスナッピングし，図 4b の状態になる．目的地付近からの微調整を行うためには，一度マウスを離してからもう一度ノブを動かせばよい．

SmoothSnap が持つ以上の特徴により，長大なメニューを目的の項目付近まで一気かつ確実にスクロールし，あとは細かい移動で目的の項目まで容易に辿りつくことができる．

2.4 実　　装

HTML と JavaScript で実装を行った．ノブをクリックした位置をスクロール開始の点とし，そこから現在のマウスカーソル位置までの差分をスクロール距離として，その値に応じてスナッピングを行っている．スナッピングさせるためにはスクロールバーを移動してもメニューの画面遷移が停止している必要があるが，項目のみを表示する iframe とスクロールバーのみを表示する iframe の二つに分割し，二つの iframe 間を JavaScript で連動させることで実現した．

3. 関　　連

3.1 タグ付け

一つの機能が複数の名前を持つのに似た概念として，タグ付けがある．タグ付けはユーザが各々の求めるものを検索しやすいようにカスタマイズするという点においては有用であるが，全ての機能にタグをつけることはユーザの負担になり，そもそも普段使わない機能に，その機能を必要とするときのみ有効となるようなタグを付けることは難しく，それもまたユーザにとっては大きな負担となる．

さらに，タグ付けは同じタグが付いているものが複数ある場合にはそのタグが付けられたもののリストから一つを選ぶ必要が生じ，そこにはタグを上位，機能を下位とした階層構造がある．それに対して全部入りメニューは機能そのものを探して選択するため，階層構造が生じない．

3.2 索　　引

素早い参照を目的として多くの項目を網羅するという点において，索引との類似性があると考えられる．索引は項目が出現する位置をまとめたものであり，例

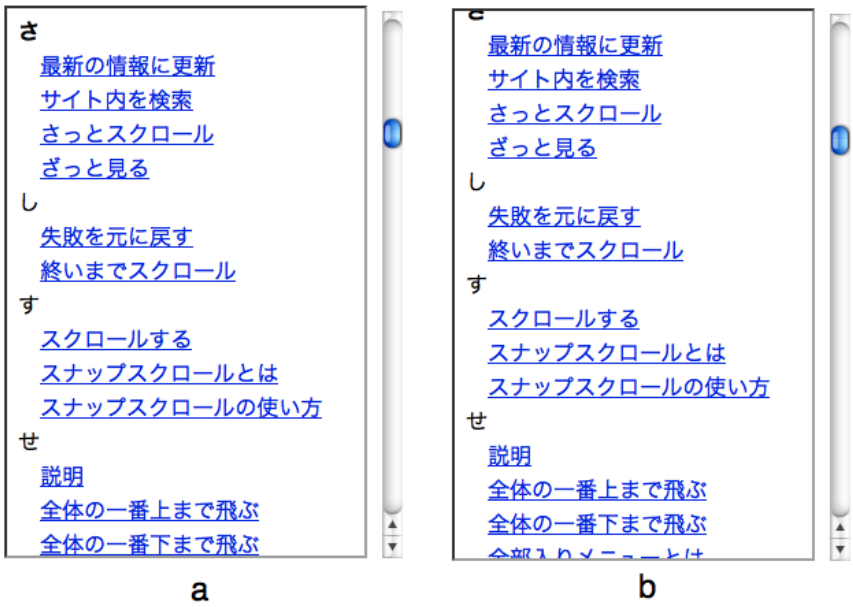

図 3 SmoothSnap で微量スクロール
Fig. 3 a Small Amount of Scroll with SmoothSnap

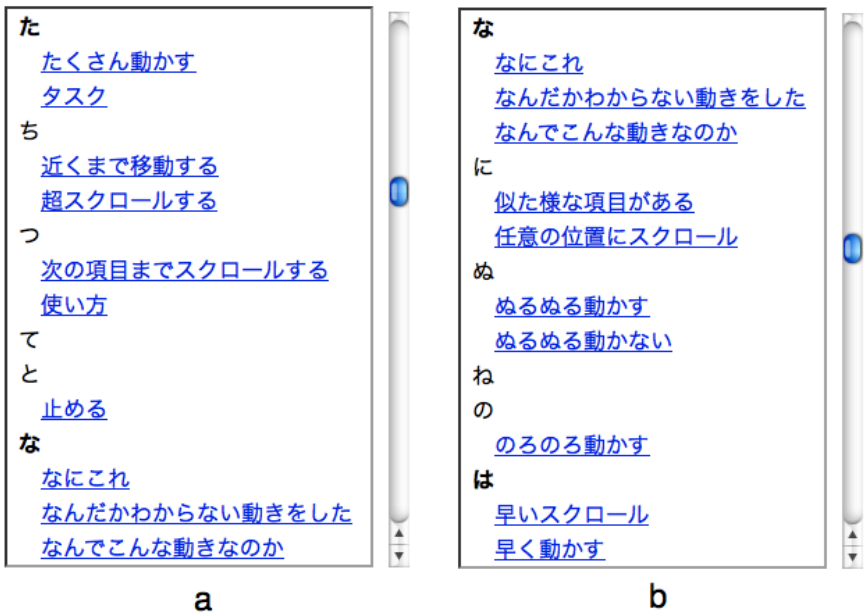

図 4 SmoothSnap でスナッピング
Fig. 4 Snapping with SmoothSnap

えば書籍における索引であれば，その書籍の内容にある程度精通していれば効果的に項目を見つけることができる．

全部入りメニューとの相違点としては，全部入りメニューでは同じ項目に多様な表現でのリンクがなされているという点が挙げられる．索引は，本文中に出現している単語で項目が形成されているので，本文に登場しない言葉を使用して多様な表現をすることは却って混乱を招く．

さらに，索引はあくまで項目の出現位置をまとめたものなので，項目が複数ヶ所に出現する場合は複数ヶ所へのリンクが明示されており，メニュー項目からのリンク先が機能そのものであるがゆえに，一項目に対応するリンク先が常に一箇所である全部入りメニューとは異なる．同じ位置へ至る複数の道筋があるものではない．

4. 議　　論

全部入りメニューの利点と欠点について考察する．

4.1 利　　点

- 可逆的である

プルダウンメニューなどの，階層メニューを使用した既存の GUI は，しばしば不可逆的な動きをするが，階層構造がなく一つのメニューしかない全部入りメニューではそのような現象は起こり得ない．また，SmoothSnap を使用していることにより，スクロールし過ぎて目標を通りぎ過てしまった時もマウスをもとに戻せば復帰できるという可逆性も持っている．

- 目標に辿り着きやすい

メニュー自体が一つしかないので，他のところにあるかもしれない，と複数のメニューを切り替えながら探す必要がない．コンテキストメニューやクリック箇所毎に固有のメニューなど，気が付きにくい位置に隠れてしまって見つけられないことがない．また，一つの機能に辿りつく道筋が複数あるので，目的とする機能を見つけやすい．非常に単純に考えるならば，一つのメニュー内に同じ機能を持つ項目が 3 つあれば，辿り着きやすさは 3 倍になる．

- 見やすい

メニューが一つしかなく，尚且つ先述の通り，大きくスクロールした場合には重要なポイントにスナッピングするので，視点の移動が非常に少なくて済む．

- 実装しやすい

構造としては項目を一つのメニュー内に全て羅列するのみ，それを SmoothSnap でスクロールするだけと非常にシンプルで，SmoothSnap も原理が単純なため HTML と Javascript の知識さえあれば簡単に実装することができる．

4.2 欠　　点

- 項目の有無が判断できない

元から存在していない機能についての項目は無い，もしくはわずかに言及している項目があるかもしれないが，それをユーザが予見することは難しい．作成者側にしてみても，搭載していない機能についての項目は想定しづらいと思われる．また，搭載されている機能の項目に関しても，ユーザが作成者の想定外の表現で探した場合は見つからないが，たまたまユーザの思い浮かべた表現が漏れているだけなのかどうかをユーザが判断するためには別の表現をいくつか試してみなければならない．

ただし，以上の欠点は既存の階層メニューにおいても同様に発生する問題であり，特に全部入りメニューにおいてのみ生じる問題ではないと付け加えておく．

- 作成者側の手間が増える

多彩な表現を思いつく限り網羅しておく必要があるので，作成者の手間が増える可能性がある．

5. まとめと展望

通常であれば，膨大な数のメニュー項目をひとまとめにしてしまうとその中から特定の項目を見つけ出すのには手間がかかるために，何らかの方法で分割するが，全部入りメニューを使うことでそれらの手間を大きく軽減することができ，また，分割することによって生じる問題を解消できる．

今回は Web 上での実装を行ったが，全部入りメニューの概念はデスクトップアプリケーション等にも幅広く使用できるシステムだと考えられるので，更なる範囲の応用を検討していく所存である．

参　考　文　献

1) 増井俊之. SmoothSnap: スナッピングにもとづく微調整可能な GUI 部品. ソフトウェア科学会 インタラクティブシステムとソフトウェア (WISS2009): December 2009.