

SmoothSnap: スナッピングにもとづく微調整可能な GUI 部品

SmoothSnap: Snapping-enhanced Widget for Handling Large Data

増井俊之*

Summary. Browsing a large document using a simple scrollbar is not comfortable, because it is not only difficult for users to locate the information they need, but it is hard to control the knob to display the information at the right place on the screen. We propose a snapping-based widget called SmoothSnap, with which users can easily grasp the structure of a large document, find the information they want, and display it at the right place with minimal effort. SmoothSnap can also be used for fine-controlling a slider and other GUI widgets.

1 はじめに

スライダやスクロールバーは値を設定したりデータをブラウズしたりするのに広く利用されている基本的な GUI 部品であるが、細かい値を設定したり大量のデータをブラウズしようとするとき以下のような不都合がある。

1. 微調整が難しい

たとえば 10 万件の項目を 200 ドット幅のスライダで選択しようとする場合、項目とノブの位置をリニアに対応させると、スライダのノブを 1 ドット動かすだけで 500 件先の項目に表示が飛んでしまうことになる。このため、大量のデータをスライダで選択するためにはノブを微調整するための特殊な仕組みが必要になる。

2. 目的の項目に簡単に到達できない

巨大な Web ページをスクロールバーでブラウズする場合、章や節のタイトルに注目しながらページをスクロールできると便利だが、本文テキストが多い場合はタイトル行が画面にいつも表示されるとは限らないので、どのあたりをブラウズしているのか把握することが難しく、目的の章や節に移動することが簡単ではない。

問題 1. を解決するために、スライダやスクロールバーのノブ位置を微調整するための各種の手法が考案されている。多くの GUI 部品ではスライダやスクロールバーの脇に配置した矢印ボタンでノブ位置を微調整できるようになっているし、部品を拡張することによって解決しようとする研究もある。たとえば AlphaSlider [1] ではスライダのノブを分割し、クリックした位置によって微調整/粗調整を選べる

ようになっている。また FineSlider [5] では、スライダのノブ以外の部分をクリックしたときにスライダをゴムで引っ張るような動きをさせることによりノブ位置を微調整できる。PopupVernier [2] は、スクロールバーのノブ付近に「パーニヤダイヤル」を表示することによってスクロールバーの微調整を可能にしている。

問題 2. を解決するための手法も各種のものが提案されている。Automatic Speed-dependent Zooming (ASDZ) システム [3] では、スクロール操作の速度によってズーム率を変化させることによって大局的情報と局所的情報の連続的なブラウズを可能にしている。Content-aware Scrolling (CAS) システム [4] では、ドキュメントを重要な部分と重要でない部分に分け、重要でない部分はノブを少し動かすだけで沢山スクロールするようにすることにより、文書の重要部分だけを効率的にブラウズできるようにしている。

2 スナッピングを利用した GUI 部品

遠くに旅行する場合、目的地の近くの空港まで飛んでから電車やバスに乗り、最後に徒歩で目的地まで行くことができる。また、遠くない場所に行きたい場合は徒歩や自転車を使うのが普通である。このように、現代社会で移動の必要があるときは、移動する距離によって移動の速度や粒度を変えることによって様々な場所に効率的に移動できるようになっているわけであるが、スライダやスクロールバーにおいても同様の方針を採れば、以下のような簡単な操作で効率的に項目を選択したり表示場所を移動したりできるようになる。

- 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
- 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

Copyright is held by the author(s).

* Toshiyuki Masui, 慶應義塾大学 環境情報学部

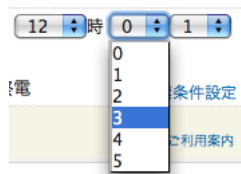


図 1. 「乗換案内」の時刻指定メニュー

このような GUI 操作手法を SmoothSnap と呼ぶことにする。スライダとスクロールバーについて SmoothSnap を実装したものを以下に示す。

2.1 時刻の設定

時/分/秒の設定は $24 \times 60 \times 60 = 86400$ 通りの可能性があるため、これをひとつの標準的なメニューやスライダで設定することは難しい。このため、図 1 のように、メニューを利用して時刻指定を行なうことが多いようである。このサービスでは「だいたい 2 時」程度の指定で十分なことも多いのだが、正確な時刻を指定したいことがあるためにこのような指定方法を採用しているのである。

前章で述べた方針にもとづいて実装した時刻指定スライダを図 2 に示す。スライダのノブを動かそうとすると、マウスをクリックした後のマウスの移動距離が少ない場合は微細な調整が可能であるが、移動距離が大きくなると粒度が粗くなって分単位/時間単位でスナッピングするようになる。

「12:34:56」のように細かい分/秒までを正確に指定したい場合、まずノブを大きく動かしてスナッピングを活用して目的の時刻に近いキリ時刻 (e.g. 12:00:00) まで移動し、一度マウスを放してから再度ノブを動かすことにより目的の時刻にさらに近いところ (e.g. 12:30:00) まで移動し、... という操作を繰り返すことによって徐々に目的の時刻に近づけていくことができる。

粗い操作をすると大雑把に時刻を指定することができ、細かい操作を繰り返すと詳細時刻を指定することができることになるのは人間の直感に近い。

2.2 大きな文書のスクロール

大きな文書が 1 ページの Web ページになっているとき、その構造を把握しながらブラウジングを行なうことは難しい。図 3 は、著者が執筆した 30 個の Web 記事を並べてひとつのページにしたものの一部をブラウザで表示している様子を示している。ブラウザ画面では大きな Web ページのごく一部しか見えていないため、スクロールを行ないながらページ全体の構造を把握するのはかなり難しい。右側の画面では章タイトル(「第 28 回...」)が見えているので大体どのあたりをブラウズしているのかわかるが、左側の画面はタイトルが見えていないので、ブラウザ画面を見るだけではどの章なのかわから

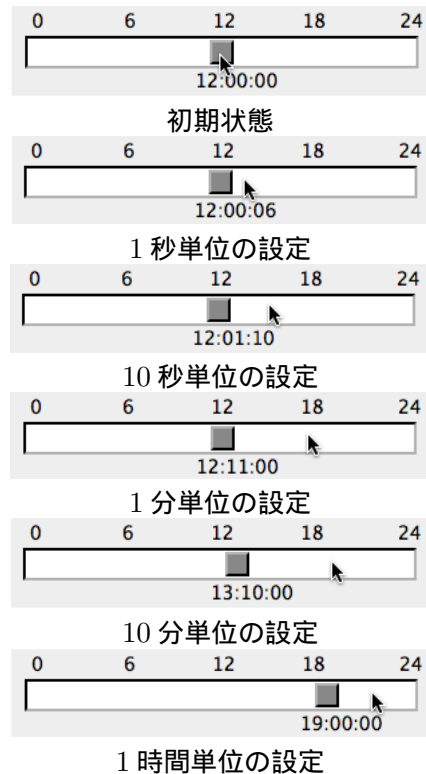


図 2. ノブをドラッグしたときの時刻の値の変化

ず、上下にスクロールしてはじめて位置がわかることになる。

SmoothSnap を利用して同じページをブラウジングしている様子を図 4 に示す。スクロールバーのノブをドラッグしたとき、ノブの移動量が小さい場合は通常の場合と同様にスクロールが行なわれるが、移動量が大きい場合は章や節の先頭でスクロールがスナッピングするため、常に<h2>や<h3>が画面の上部に位置することになり、全体的にどういう章や節で構成されているのかを容易にブラウズして把握することが可能になっている。

3 議論

SmoothSnap を既存システムと比較し、その利点と欠点について考察する。

- 特殊な操作や部品が不要

AlphaSlider[1] や PopupVernier[2] では、標準的なスライダノブ以外に微調整用の特殊な GUI 部品を使用しているが、SmoothSnap スライダもスクロールバーも外見は標準のものと変わりがなく、動作の違いもスナッピングだけであるから、特殊な知識なく利用できることが期待される。

- 直感的な操作



図 3. 大きな Web ページのブラウジング

大雑把な操作をすると大雑把に意味のある反応が起こり、丁寧な操作をすると細かい制御ができるという SmoothSnap の特徴は直感に一致している。

● 可逆的な操作

FineSlider[5] や ASDZ[3] ではユーザの操作に対するシステムの反応が可逆的でないため、スクロールの結果勢い余って目標地点を通り越してしまった場合は逆向きの操作をして戻る必要がある。SmoothSnap ではスクロール位置がマウス移動量の関数になっているため、目標地点を通りすぎてもマウスをもとに戻すだけで前の状態に復帰することができる。

● 汎用性

SmoothSnap によるスクロールは単純であるにもかかわらず CAS[4] と同様の効果を得ることができる。一方、CAS は 2.1 節のような詳細値制御に利用することはできない。

● Web 上ですぐ実装できる

SmoothSnap は原理が単純でありブラウザの JavaScript で簡単に実装できる。

● 欠点

SmoothSnap では、ユーザのマウス操作の量とノブの移動量が一致しないため混乱を生じる可能性がある。また、はじめて使う人間がすぐに原理を理解して使いこなすことは困難であると考えられる。最低限の利用法をなんらかの方法でうまくユーザに知らせる方法が必要であろう。

4 結論

普通のスクロールバーで大きな Web ページをブラウズするのは難しいため、巨大な文書を Web に載せる場合は階層的に分割するのが普通である。また、細かい値をスライダで設定することは難しいので、複数の GUI 部品を利用することが多い。SmoothSnap にもとづく GUI 部品は、原理が単純であるにもかかわらず広い範囲で利用することができ、上記のような問題を解決できる可能性があるため、さらに広い範囲の応用について検討したい。

自動スナッピングを利用して大きなページをブラウズする

- Wiredvisionの**選定の記事**を全部つなげたものです。
- スクロールバーに加工されており、スクロール量に応じてパラグラフの先頭や記事の先頭でスナッピングします。
- Firefox以外では動作しないかもしれませんが、それ以外のブラウザの場合は [こちら](#) をお試しください。

第29回 誤魔化す!技術

「誤魔化す」という言葉には悪いイメージがありますが、「情報隠蔽」というと多少聞こえが良くなるかもしれません。 [森の虫に本を隠す](#) という記事で、普通の画像やテキストの中に秘密情報を隠す **ステガノグラフィー** という技術を紹介しましたが、これは内容を誤魔化すことにより秘密を守る技術の一例といえるでしょう。

公開情報を隠す!技術

非公開の秘密情報は、普通に暗号化したりステガノグラフィーを利用したりして隠すことができますが、いったん表に出てしまった情報を後から隠すのはなかなか大変です。情報そのものを消すことは不可能ですから、なんらかの方法によってその情報の内容を誤魔化す方法が必要になります。

正しい情報と似て非なる情報を大量に提供する **数少年メソッド** を使えば、どれが正しい情報なのか判別不能にすることができます。たとえば、内容に問題があるメールを間違えて送ってしまった場合、内容が少しずつ違うメールを大量に送りつければ、どれが最初のメールなのか分からなくなってしまうかもしれません。また、パスワードを間違えてしまった場合は、異なるパスワードを同じように間違えてしまえば、悪用される危険が減るかもしれません。

文字を隠したいときは別の文字を上書きするのが効果的です。左の「増井」のような文字を手取り早く隠したい場合、斜線を引いたりするよりも、右のように別の文字を上書きする方が読みにくくなります。

増井 濶井

(a) 初期状態

情報視覚化の歴史

情報視覚化の研究は1990年ころからXerox PARCなどで盛んになり、PARCでは様々な大量の情報を3次元空間にマッピングすることによって情報をひとつの2次元画面上に表示する手法が主に研究されてきました。3次元空間のデータを眺める場合、遠くにあるものは小さく見えますから、データをうまく配置すれば自然に沢山の情報を表示することができます。たとえば古い情報は3次元空間内の遠くの方に配置し、新しい情報は近くに配置すれば、CGに利用される3次元表示手法を使って近くの新しい情報は大きく/遠くの古い情報は小さく表示することが可能になります。下記の「Perspective Wall」システムでは、沢山のファイルが3次元空間上の「壁」に貼り付けられることによって、注目している範囲の情報だけが大きく見えるようになっています。



1990年当時は高速3次元表示が可能な計算機は高価だったのでこのような研究を行える場所は限られていました。また、視覚化が必要ほど大規模な情報もあり利用できませんでしたが、そのころ提案された情報視覚化手法はほとんど実用的に利用されることはありませんでした。一方、廉価で一般的なパソコンでも高速なグラフィックス表示を行なうことができました。Web上の

(c) もう少しドラッグした状態。<h3>にスナッピングしている。

公開情報を隠す!技術


非公開の秘密情報は、普通に暗号化したりステガノグラフィーを利用したりして隠すことができますが、いったん表に出てしまった情報を後から隠すのはなかなか大変です。情報そのものを消すことは不可能ですから、なんらかの方法によってその情報の内容を誤魔化す方法が必要になります。

正しい情報と似て非なる情報を大量に提供する **数少年メソッド** を使えば、どれが正しい情報なのか判別不能にすることができます。たとえば、内容に問題があるメールを間違えて送ってしまった場合、内容が少しずつ違うメールを大量に送りつければ、どれが最初のメールなのか分からなくなってしまうかもしれません。また、パスワードを間違えてしまった場合は、異なるパスワードを同じように間違えてしまえば、悪用される危険が減るかもしれません。

文字を隠したいときは別の文字を上書きするのが効果的です。左の「増井」のような文字を手取り早く隠したい場合、斜線を引いたりするよりも、右のように別の文字を上書きする方が読みにくくなります。

増井 濶井

文具用品のプラスチック株式会社は、葉書などに印刷された住所や名前などの個人情報を読めなくするための **ケンパン** というスタンプを販売しています。ケンパンは下のようなボタンをもつスタンプです。



左のような文字の上にケンパンを押すと右のようになり、確かに文字がかなり読みにくくなるのがわかります。

何かが存在することを示したり存在に気付いたりすることは簡単ですが、存在しない事物をうまく扱うことは簡単ではありません。	何かが存在することを示したり存在に気付いたりすることは簡単ですが、存在しない事物をうまく扱うことは簡単ではありません。
---	---

(b) わずかに下にドラッグした状態

第9回 なんでも自動処理

同じような操作を何度も実行しなければならないとき、手間を省くために操作を自動化したくなります。たとえば、テキストファイルの行の先頭に「>」という文字列を追加していきたいとき、何度も「>」をタイプするのは面倒ですから、なんらかの方法で自動化できれば便利です。エディタに「n行にわたって行頭に文字列を追加する」という機能がほしいのですが、様々な自動化要求すべてに対応することは不可能ですから、特殊な自動化処理を行いたい場合は、なんらかの方法でユーザが自力でプログラムを作成して計算機に伝える必要があります。

ユーザが「操作マクロ」を定義できるエディタは多いですが、本格的なプログラミングが可能なEmacsのようなエディタもありますが、こういったプログラミングは範囲が狭く難しいものから、たいしてはあきらめておつて作業するのが普通だと思います。しかし、手軽にちょっとした自動化処理を指示できると嬉しいでしょうから、誰でも簡単なプログラミングを行えるようになる **エンドユーザプログラミング** や **視覚プログラミング** という考え方が登場しています。

本格的なプログラミングは難しいかもしれませんが、料理のレシピのような手順書であれば簡単に書いたり利用したりできます。大抵の人はアナログ時計のアラームをセットできるわけですから、機械さえ用意すれば誰でも簡単なプログラミングができるようになる可能性は充分あるといえるでしょう。

エンドユーザプログラミング

操作手順のような簡単なプログラムを、誰でも手軽に作って利用できるようにしようというのがエンドユーザプログラミングの考え方で、テキストだけ利用するよりも図を併用した方がわかりやすいことが多いので、**ビジュアルプログラミング** がよく採用されています。ビジュアルプログラミングの研究は長い歴史がありますが、まだ本格的なプログラミングに利用されているとはいえません。しかし、昔ながらの前では **Mega** というビジュアルプログラミングシステムが広く利用されています。Logo **Mindstorms** のような教育システムでもビジュアルプログラミング言語を採用することによって、プログラムの敷居を下げるのに成功しています。

Macintoshには **Automator** という簡単なビジュアルプログラミングシステムが搭載されており、アプリケーション操作を自動化するプログラムを手軽に作るようになっています。下図は、人間を表現するアイコンへファイルをDrag&Dropすることによってその上にファイルをメールで送るというプログラムをAutomatorで作成したものです。これは **Macアイコンシステム** で構築されているものですが、Automatorを使うことのように非常に簡単に実現することができます。

(d) かなり下方までドラッグした状態。<h2>にスナッピングしている。

図 4. ドラッグによるスナッピング

参考文献

- [1] C. Ahlberg and B. 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] Y. Ayatsuka, J. Rekimoto, and S. Matsuoka. Popup Vernier: A Tool for Sub-pixel-pitch Dragging with Smooth Mode Transition. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'98)*, pp. 39–48. ACM Press, November 1998.
- [3] T. Igarashi and K. Hinckley. Automatic Seed-dependent Zooming for Browsing Large Documents. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST2000)*, pp. 139–148. ACM Press, November 2000.
- [4] E. W. Ishak and S. K. Feiner. Content-aware Scrolling. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST2006)*, pp. 155–158. ACM Press, November 2006.

- [5] T. Masui, K. Kashiwagi, and G. R. Borden. Elastic Graphical Interfaces for Precise Data Manipulation. In *CHI'95 Conference Companion*, pp. 143–144. Addison-Wesley, May 1995.