

コロンブス日和

第1回 GyaTV

エンジニアというものは「楽をするためならどんな苦勞も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張らずに楽できるならそれにこしたことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきているのですが、今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきたいと思います。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



GyaTV——Web ページを自動的に表示する



楽するためには作業の絶対量を減らさなければなりません。ソフトウェアを作成する場合は、良い言語やエディタを使って効率を上げるのは基本でしょう。似た操作を繰り返す場合は、次の操作を予測することによって作業量を減らせることもあります。

しかしいくら作業を減らす工夫をしても何もしないことにはかないません。何も作業せずにソフトウェアは作成できませんが、情報を眺めたり収集したりという程度であればまったく手を動かさなくてよい可能性があります。テレビのニュースをつけっぱなしにしておくと、なんとなく世間の新しい情報について知ることができます。記憶したい情報をずっとどこかに表示しておくようにすれば、知らないうちに覚えてしまうかもしれません。

GyaTV(ギャツビー)は、登録しておいたURLをスクリーンセーバのようにランダムに表示させるWebサービスです。原理は簡単で、私が運営しているGyazzというWikiのページに登録したURLのリストから、定期的にランダムにURLを1つ選択してそのページを表示するというものです。Web上のコンテンツの中身を吟味して選択したり観賞したりしているといくら時間があっても足りませんが、自動的に表示されたものを眺めるだけなら楽です。

GyaTVの原理はとても単純ですが、さまざまな応用ができて便利に利用できます。



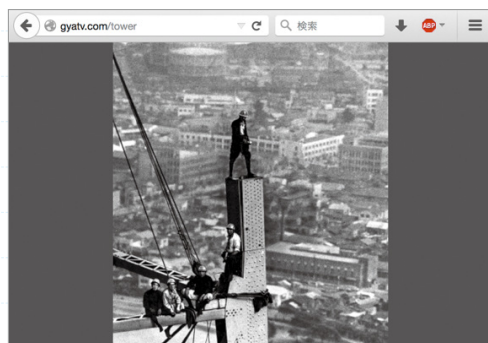
デジタルフォトフレーム

自分のデジカメ写真をランダムに液晶画面に表示する「デジタルフォトフレーム」という製品が流行ったことがあります。こういう製品は最近あまり話題になっていないようですが、デジカメ写真をスクリーンセーバのように表示するのは気持ちが良いものです。

手持ちのパソコンやタブレットコンピュータでGyaTVを動かしておけば、デジタルフォトフレームとして使うことができます。きれいなイラストや写真のまとめがよくWebに投稿されていますが、私は「東京タワー」(図1)や「スミソニアン博物館」(図2)の写真をGyaTVで表示して楽しんでいます。

GyaTV.com/xxx というURLでスライドショー表示を行うためには、Gyazz.com/GyaTV/xxx というWikiページにURLリストを登録します。

▼図1 東京タワーのスライドショー(毎日新聞のサイトの写真をランダム表示)



注1) <http://thinkit.co.jp/free/article/0709/19/>

たとえば図3のように<http://Gyazz.com/GyaTV/tower>にURLのリストを登録しておけば、<http://GyaTV.com/tower>でスライドショーを楽しむことができます。



Wikipediaのランダム表示

あまり知られていないかもしれませんが、Wikipediaページの左側には「おまかせ表示」というリンク(図4)があり、これをクリックするとWikipediaのランダムなページに飛ぶようになっています。

Wikipediaの情報は膨大ですので、ランダムにページを表示したとき、自分が知ってる内容が出ることはほとんどありません。クリックするたびに新しい情報に触れることができるのは楽しいものです。手動でクリックしても楽しいのですが、GyaTVにおまかせページを表示するURLを登録しておけば、一定間隔で新しいページが表示されることになるのでなかなか新鮮です(図5)。



単語帳の表示

「門前の小僧、習わぬ経を読む」と言われるように、何度も同じものを見聞きしていると知らない間に覚えてしまうことがあります。この性質を勉強に利用できます。英単語とその意味、用例、画像を並べたWebページを用意しておき、これをランダムに表示するようにしておけば、ときどき画面をチラ見しているうちに単語とその雰囲気を覚えてしまうでしょう。私はこのよ

▼図2 スミソニアン博物館のコレクション



▼図3 URLのリスト



うなWikiページをたくさん用意しており、これをGyaTVで表示することによって単語力を上げようとしています(図6)。

画像を検索すると単語の用例が直感的に理解できることもあります。「convocation」という単語を普通の辞書で調べると「会合」のような意味が出てきますが、Google画像検索を行うと図7のような写真ばかり出てきます。どうやらこの単語は「卒業式」で使われるらしいということがわかります。こういう画像をずっと眺めていると「convocation」という単語の雰囲気を理解できます。



ニュースの表示

昔は街中にニュースを表示する電光掲示板が

▼図4 Wikipediaの「おまかせ表示」ボタン(<https://ja.wikipedia.org/wiki/特別:おまかせ表示>)



▼図5 ランダムに表示される謎Wikipediaページ



よくあったものです。現在でも新幹線の車両の端ではニュースが表示されていますし(写真1)、ニュースを知るために家でテレビをつけっぱなしにしている人も多いでしょう。

ニュースのような情報ソースは、真剣に見るのではなく、なんとなく聞こえてくるのが好まれるということでしょう。新しいニュース情報をWikiページに書き込んでおけば、GyaTVでいつも最新のニュース情報を表示させておくことができます。



表示順序の制御

GyaTVでは、登録したURLがランダムに表示されますが、“http://GyaTV.com/tower?play=seq”のようにオプションを指定するとリストした順番のとおりに表示できます。旅の写真を順番に表示したい場合や、スライド的に情報を伝えたいような場合はページの順番が大事になるので、このオプションを付けて表示するのが良いでしょう。



情報共有や情報通知

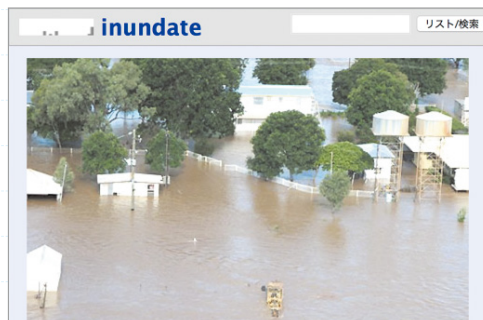
GyaTVページはWeb上にあってどこからでも見えますから、自分のきれいな写真を自動再生するGyaTVページを作って他人に自慢できます。写真を並べて旅日記を作ってあげれば、自分で楽しめるうえに他人に自慢もできるので良いでしょう。

コンピュータを使うのが苦手な老人などにネット経由で情報を伝えたい場合、そのような人の家のiPadなどで常にGyaTVのページを表示するよ

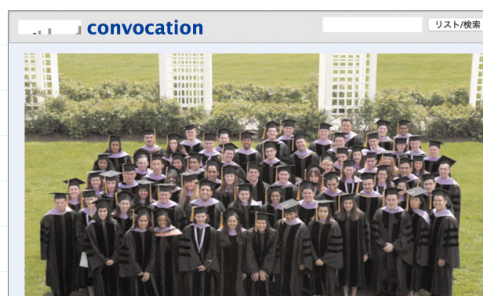
▼写真1 新幹線の電光掲示板



▼図6 「inundate(洪水)」で表示される映像



▼図7 「convocation」という単語の画像検索



うにしておき、情報を伝えたい人がGyazzの内容を書き換えるようにしておけば、常にその人に対するメッセージを表示し続けることができます。



表示時間の指定

GyaTVでは標準では15秒ごとにページが更新されますが、URLの後に数字を書いておけば表示する秒数を指定できます。読むのに時間がかかるページや、長く表示しておきたいページは長時間表示し、急いでたくさんのページを表示したい場合は小さい数字を指定すれば良いでしょう。また、きちんと秒数指定しておけば、YouTubeなどで音楽や動画を順番に表示させることもできます。



置くだけ再生

AndroidスマホやタブレットにはNFCリーダーが搭載されており、SuicaのようなRFIDカード/タグを読み出すことができます。Androidの場合、端末をタグに近付けることによってアプリケーションを起動することもでき

るので、タグを置いた場所に置くだけでGyaTVのページを開くことができます。この機能を利用すると、タブレットやスマホを特定の場所に置くだけでスライドショー／予定表／時計などを表示できるので便利です(写真2)。



自動表示で気づくこと



GyaTVを使っていると次のようなことに気づきます。



時間感覚のあやしさ

15秒ごとにランダム表示を行っている画面をずっと見ていると、なかなか画面が更新されないことにイライラするものです。一方、仕事をしているときに別のマシンでGyaTVを動かしているときは更新の遅さが気にならないどころか、更新が速すぎると感じることもあります。どうやら人間は、自分が注目しているものとしていないものに関して時間感覚がかなり違うのだろーと思えます。小さいと思っていた親戚の子供に久しぶりに会ったら成長ぶりに驚くことがよくあります。あまり興味のないWebサービスが開始したと思ったらいつの間にか終了していることもあります。自分が深くかかわっていないものは進行速度が速く感じられる気がします。つまり、仕事に極度に集中すれば時間の進行が遅く感じられ、離れたところに置いたものは時間の進行が速く感じるのでしょう。このような時間感覚を活用できれば、大事なことには時間をかけつつ、どうでも良いことはすぐに終了させることができるのかもしれません。



乱数感覚

ランダム表示を行うとき、普通の乱数を利用するとランダム感が得られないことがあります。10枚の写真をランダムに表示させると、10回に1回は同じ写真が連続して表示されることになります。これを

▼写真2 置くだけ再生しているところ



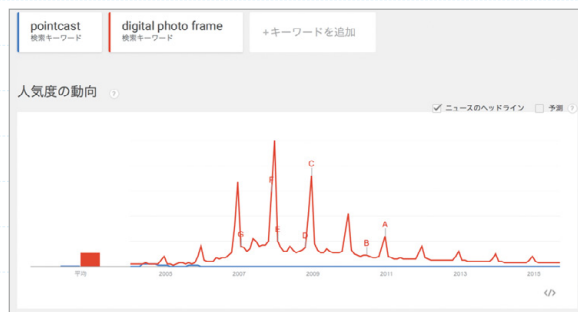
避けるためにはrand()のような乱数関数を利用するかわりに、履歴を考慮したニセ乱数関数を使うのが良いと考えられます。実際、音楽プレーヤのシャッフル再生機能では、本当の乱数の代わりにニセ乱数が利用されているようです。



インターネットの黎明期、ニュースをスクリーンセーバのように配信するPointcastというサービスが注目されたことがありましたが、結局あまり流行しませんでした。また、たくさんの写真を自動的に表示してくれるデジタルフォトフレームも最近あまり流行していないようです(図8)。

これらのシステムがあまり流行らなかった理由はいろいろあるのですが、ネット上の情報を自動表示すること自体は有益なはずです。ぜひGyaTVを使っていろいろ楽しんでみてください。SD

▼図8 GoogleTrendによる情報



コロンブス日和

第2回 Gyump

エンジニアというものは「楽をするためならどんな苦労も厭わない¹⁾」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張らずに楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



長いURLを便利にしたい



あらゆるものがURLで表現可能になりつつありますが、URLで表現する対象が膨大な量になってくると、長いURLを扱う機会が増えてきています。長いURLは取り扱いが面倒でし、記憶するのもたいへんですから、次のような工夫がよく用いられています。



ブックマークを利用する

よく使うURLに手軽にアクセスできるようにするために、ブラウザのブックマーク機能が広く使われています。ブックマーク機能はMosaicのような初期のブラウザでも実装されていたほどポピュラーなもので、あまり機能が進化しないまま現在のブラウザでも利用されています。

ブラウザのメニューからよく使うURLにアクセスできるのは便利ですが、気軽に登録しているとあっという間にメニューが一杯になってしまいます。階層的にブックマークを管理すれば良いのですが、きちんと分類して管理するのは面倒なので、結局利用の機会が減ってくることが多いようです。

ブックマーク情報を、複数のマシンやブラウザで共有しづらいのも面倒です。最近のブラウザではブックマーク情報を複数マシンで同期できますが、こういう設定は面倒なものです。また、ブックマークをブラウザ上に記憶する代わ

りに、クラウド上に記憶して共有するソーシャルブックマークのようなサービスもありますが、ソーシャルブックマークは、Webページにコメントを付ける用途で利用されることが多く、頻繁に利用するURLにアクセスする用途にはあまり使われていないように思われます。



長いURLを短いURLで置き換える

長いURLに簡単にアクセスしたり他人にURLを知らせたりする場合のために、長いURLを短くしてくれるTinyURL、Bit.ly、Goo.glなどのURL短縮サービスがよく使われています。これらのサービスを利用すると、http://pitecan.com/……のような長いURLの代わりに、http://goo.gl/jx7VZy や http://bit.ly/1LnYzAtのような短いURLを利用できます。

URLが短ければ記憶が可能かもしれませんし、他人とURLをやりとりする場合にも便利なのですが、「jx7VZy」のような暗号的な文字列を記憶することは困難です。覚えることができないため、ファイルなどに書いておく必要があるのであれば、長いURLを利用するのとあまり変わらないかもしれません。短い名前を自分で選べるサービスもありますが、好きな名前が使えらとは限りませんし、一度登録したURLを後で変更できないのが普通です。

また、短いURLと長いURLの対応はサーバに記憶されているので、サービスにトラブルがあったり、サービス自体が終了したりすると使えなくなるのは心配です。普通は対応データベースを取得できませんから、データをバックアップしておくこともできません。

注1) <http://thinkit.co.jp/free/article/0709/19/>

▼ 図1 短縮URLサービスでcurlコマンドを試す

```
% curl -w '%{http_code}' http://goo.gl/jx7VZy
<HTML>
<HEAD>
<TITLE>Moved Permanently</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://pitecan.com/">here</A>.
</BODY>
</HTML>
301
%
```

そもそも個人的に使うURLは短くて良いはず。私の周囲で「SFC」といえば慶應義塾大学湘南藤沢キャンパスのことに決まっており、スーパーファミコンのことではありません。自分の周囲で言及することが多いものは、たいてい数文字で表現できてしまうでしょうから、よく使うものから順番に短い名前を割り当てておけば便利でしょうし、情報圧縮や効率化の面でも有利なことは間違いありません。今回は自分用の短い名前を気軽にブックマークとして活用できるGyump(ジャンプ)というサービスを紹介します。



Gyump: 柔軟なURL短縮サービス



URL短縮サービスの1つであるGoo.glにhttp://pitecan.com/を登録するとhttp://goo.gl/jx7VZyのようなURLが利用できるようになります。curlコマンドで確認してみると、図1のようなHTMLとステータス301(Moved Permanently)が返ってくるのがわかります。

Goo.glや、そのほかのURL短縮サービスは、短い名前と長い名前の対応データベースを保持しており、短い名前でアクセスされたとき図1のようなHTMLとステータスコードを返すという単純なしくみで動いているようです。このようなものは自分で作るのも簡単です。

Gyumpは、ユーザが指定した名前を使って任意のURLにアクセスできるようにするサービスです。たとえば「sd」というIDを持つユーザが「map」というキーワードで東京駅の地図にアクセスしたとき、http://gyump.

com/sd/mapというURLを利用できるようにするということです。既存の短縮サービスと異なり、Gyump.com以下の名前は任意のものが使えるので、自分のID(e.g. sd)と短い名前(e.g. map)のような任意の組み合わせを利用できます。

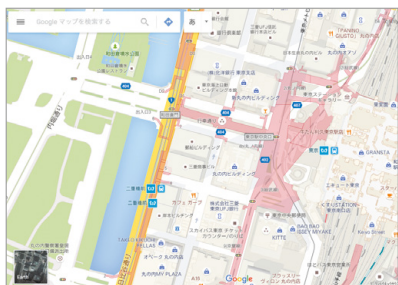
何も登録されていない状態でGyump.com/sd/にアクセスすると、図2のように登録フォームだけが表示されます。

ここで「map」というキーワードと東京駅の地図のURLを入力して登録すると、東京駅のURLが「map」という名前で登録され、登録されたURLが図3のようにリストされます。この状態ではhttp://sd.gyump.com/mapというURLで東京駅の地図にアクセスできます(図4)。

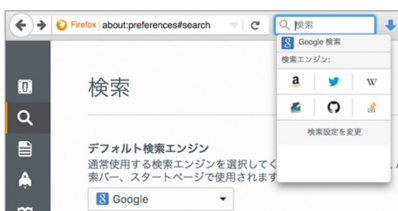
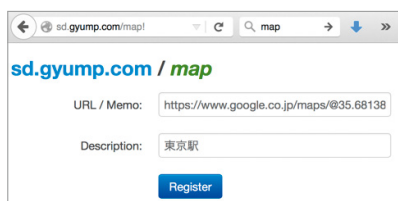
リスト中の「map」をクリックすると編集モードになります。http://sd.gyump.com/map!のように、短いURLの最後に「!」を付けたURLにアクセスすると、直接編集ページに飛ぶことができます(図5)。

▼ 図2 sd.gyump.comの初期状態

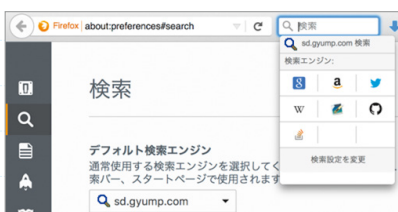
▼ 図3 東京駅を登録後のリスト

▼図4 <http://sd.gyump.com/map>にアクセスして表示される地図

▼図5 編集フォーム



▼図6 Firefoxの検索エンジン選択機能



▼図7 デフォルト検索エンジンとして「sd.gyump.com」を指定

で地図のリストを見ることができます。

「sd」や「map」のような名前は自由に選択できますし、登録URLを後から変更することもできるので、次に示すようにさまざまな使い方ができます。



ブラウザの検索窓に登録

短いとはいっても<http://sd.gyump.com/map>のようなURLをブラウザに入力するのは面倒でしょう。Firefoxなどではopensearch機能を使って検索窓をカスタマイズできるようになっているので(図6)、Gyumpを検索システムとして登録しておけば、検索窓にmapと入力するだけで<http://sd.gyump.com/map>にアクセスできるようになるので便利です(図7)。

このように、Gyumpをデフォルト検索サービスとして登録しておくと、検索枠に「map」と入力するだけで東京駅の地図を開くことができます。Gyumpに登録されていない文字列を入力するとGoogle検索するようになっているので、検索エンジンを切り替える必要は多くありません。



階層的な情報管理

たくさんのURLを扱いたい場合はGyump.com/sd/maps/tokyoのような階層的な名前も利用できます。この場合、Gyump.com/sd/maps/akiba、Gyump.com/sd/maps/shibuyaのような名前を登録しておけば、Gyump.com/sd/maps/



Gyumpの活用例



その1「地図へのアクセス」

初めての場所に行こうとすると、行先をあらかじめGoogle Mapsなどで調べておく人が多いと思いますが、パソコンで調べたURLにスマホからアクセスするためには、なんらかの方法でURLを送る必要があるので面倒です。Google MapsのURLはとても長いのでメールで送るのも手軽ではありません。

私は、行先の地図のURLを常にmyname/mapのような名前でGyumpに登録するようにしており、Gyump.com/myname/mapのショートカットをスマホのホーム画面に登録してあるので、スマホ上でこれをタップするだけで、常に行きたい場所の地図を表示できるようになっています。行先が変わった場合でも、同じURLから目的地の地図が表示されるのでとても便利です。



その2「予定表」

私は予定表をWebで管理しており、Gyump.

com/myname/sのような短いURLでアクセスできるようにしています。予定表のURLは毎月変わるのですが、現在の日付からURLを計算して今月の予定表ページに飛ぶプログラムを用意しているので、常に同じURLで今月の予定表にアクセスできるようになっています。



その3「買い物／メモ」

GyumpのURL登録欄にURL以外の文字列を書いておくと、そのURLに飛ぶかわりに、登録した文字列が表示されるようになっているので、私は買い物メモなどはGyump.com/myname/buyのようなところに書くようにしています。



その4「よく使うショートカット」

よく利用するサービスに飛ぶためのブックマークとしても私はGyumpを活用しており、たくさんの固定URLを登録して使っています。「tenki」で天気予報にアクセスしたり、「jor」でジョルダン乗り換え案内にアクセスしたり、「hon」で本棚.orgのページにアクセスしたり、あらゆる状況で頻繁にGyumpを利用しています。



その5「一時的な仕事での利用」

不慣れなAPIなどを調べて使いたいような場合、マニュアルやブログなどたくさんのページを参照しながら理解を深めるものですが、検索したページをすべて開いたままにしておくとブラウザがタブだらけになってしまいます。このような場合、必要になるかもしれないページをGyump.com/sd/api/1、Gyump.com/sd/api/2のようなアドレスにどんどん登録しておくようにすれば、タブの数などを気にすることなく後で簡単にアクセスできます。



その6「登録URLのチェックとバックアップ」

通常のURL短縮システムと異なり、Gyumpでは登録されたURLのリストを眺めることができますから、そのページをバックアップしておけば

Gyumpが使えなくなった場合でも安心です。



Gyump アドレスのブックマーク

Gyump.com/sd/mapのようなURLをブラウザやスマホでブックマークしたい場合、ブラウザ上でこのようなURLを入力すると、ここに登録されている地図URLにジャンプしてしまうためGyump.com/sd/mapという短いURLをブックマークできません。Gyumpでは、Gyump.com/sd/で表示されるリスト画面の中から「map」を選択するとGyump.com/sd/mapのようなアドレスで登録画面が表示されるので、ここで短いURLをブックマークできます。



Gyumpの利用経験



私はこのシステムを長年活用しているのですが、同様のシステムを活用している人は多くないようなのが不思議です。さまざまなWebページにアクセスしようとするときは、毎回Google検索したり、ブラウザのURL補完機能やブックマーク同期機能を利用したり、Google Mapsのようなサービスが用意している登録機能を利用したり、Evernoteのような情報管理ツールを利用したり、人それぞれにいろいろな方法が利用されているようですが、Gyumpは単純な割に応用が広く、「sd/map」のような文字列さえ覚えておけば、あらゆるブラウザで使えるので安心ですし、他人に口頭でURLを伝えるのにも便利だと思います。

Gyumpのようなシステムはもちろん万能ではありません。「map」のような適切な名前を思いつかない場合もありますし、登録したものを他人に書き換えられてしまう可能性もあります。しかし運用を工夫すればそれほど困ることはありませんから、メリットを活かして活用するのが良いと思っています。

しくみの単純さと効果の大きさを掛けたものを、筆者はコロンブス指数と呼んでいるのですが、単純なしくみにもかかわらず応用範囲が広く有用だという意味で、Gyumpのコロンブス指数はかなり高いと言えるでしょう。SD

コロンブス日和

第3回 Gyamm

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張らずに楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



URLで紐付けられた世界で



先月号の記事でも書いたように、製品にも日記にもレシピにも場所にもURLがついており、あらゆる情報がURLで表現できる時代になってきたと言えるでしょう。

ところが、なぜかメールメッセージにアクセスするのにURLを使うのは一般的ではなく、メール専用のアプリケーションやサービスを利用するのが普通で、何かと不便が多い状況になっています。たとえば、会議やイベントの連絡をメールで受け取ったとき、その内容を表現するURLは存在しませんから、ブックマークできませんし、SNSなどにURLを貼ることができませんし、別のページやWikiなどからリンクを張ることもできません。

メールで個人的に受け取った情報を他人に伝えようとすると、メールを転送したり、SNSにコピー&ペーストしたりしなければなりません。あらゆる情報がWeb上で表現されている現在、情報に直接ブラウザからアクセスできないのはたいへん不便であり、メールで受け取った情報だけが別の世界に存在するよう感じられてしまいます。



なぜメールが使いにくいのか？

メールとWebの相性が悪い原因は次のようなものだと思います。

- ・メールはWebより前から存在しており、独自の環境で扱われていた
- ・メールは個人的な情報のやりとりに使われることが多かったため 公開が前提のWebの世界とは別物と考えられていた

このような事情のため、Webと融合する方法があまり工夫されていなかったのでしょう。

メールの普及はWebの普及よりもずっと前のことですので、しかたがなかったかもしれませんが、Webが発明されてから20年以上経過している現在、いつまでもメールを別世界のまま放置しておくわけにはいかないでしょう。何らかの方法での融合が必要だと思われます。



Gyamm——メールメッセージをWebページ化する



メールメッセージを普通のWebページと同じように扱うことができれば、メールの扱いはずっと便利になるはずです。私はメールを簡単にWebページに変換できるGyammというシステムを作って運用しています(図1)。

Gyammを使うと、メールメッセージをどこからでもアクセスできるHTMLページに変換することができます。



Gyammの使い方

Gyammの使い方はとても簡単です。メールメッセージを[example@gyamm.com]のようなメールアドレスに送ると、送られたメッセージは[http://gyamm.com/example/]から閲覧できるWebページ

注1) <http://thinkit.co.jp/free/article/0709/19/>

ジに変換されます。メールが添付ファイルを含んでいたり、HTML メールだったりした場合でも、普通にWebページとして閲覧できます。

たとえば、受け取ったHTMLメールを[example@gyamm.com]に送ると、図2のようなWebページが生成されて[http://gyamm.com/example/20151119003655]のようなURLでアクセスできます。メールの情報を簡単な手間でWebで公開できたことになります。

添付ファイルがある場合は「▶」の右にファイル名が表示されます(図3)。



Gyammの利用例

Gyammは、次のようなさまざまな用途に利用できます。

✓ 情報の公開／共有

受け取ったメールをそのままの形式で、Webページで公開するのは普通は簡単ではありませんが、Gyammを利用すれば、HTMLメールも添付ファイルつきメールもWebページに変換して、ブックマークしたり他人と共有したりできます。

▼ 図1 SoftwareDesign@Gyamm.comに送られたメールのリスト



▼ 図2 書店からの案内メール



✓ メーリングリストのアーカイブ

xxxx という名前のメーリングリストのメンバとして[xxxx-archive@gyamm.com]のようなアドレスを登録しておくと、メーリングリストのあらゆるメッセージが[xxxx-archive@gyamm.com]に送られ、その結果全メッセージが[http://gyamm.com/xxx-archive/]に蓄積されるので、メッセージのアーカイブとして利用できます。

✓ フロー情報のストック化

宴会などの案内がメールで送られてくることはよくありますが(図4)、こういうフロー情報をWebページのようなストック情報に変換しておくと、後でアクセスしやすくなって便利です。

✓ 情報の集約

何かのトピックに関して、いろいろな人から

▼ 図3 添付ファイル付きのHTMLメール



▼ 図4 宴会案内メールをWebページに変換したもの



メールを受け取ったとき、それを全部[special-topic@gyamm.com]のようなアドレスに転送するようにしておけば、関連するあらゆるメールを[http://gyamm.com/special-topic/]からアクセスできるようになります。アンケートやレポートをメールで集計するような場合に便利です。

✓ 情報の分類管理

仕事に関するメールなど、やらなければならない仕事は全部[masui-todo@gyamm.com]に転送することによって、やらなければならない仕事をすべてGyamm上にリストできます。Gyammではメールが表示されないように指定できるので、終わった仕事に関するメールは非表示にしておけば良いでしょう。同様に、トピックごとに別のGyammアドレスに送るようにすれば、フォルダのように管理できます。



Gyammの実装



Gyammは次のように実装されています。

- ① SMTP を解釈する Gyamm サーバを gyamm.com で動かし、[???@gyamm.com]宛のメールをすべて受け取る
- ② 届いたメッセージを解析してHTMLに変換する

SMTP(Simple Mail Transfer Protocol)はインターネットのメール転送で標準的に利用されているプロトコルで、各種のメール送信アプリケーションやPostfixのようなメール転送エージェント

(MTA)で広く利用されています。Gyammはメールサーバではありませんが、SMTPを解釈してメールサーバのふりをすることによって、Gyamm.comへのメールを受け取って処理しています。



SMTP サーバの実装

SMTP の仕様はRFC821 で定義されています。SMTPサーバには25番ポートを利用して図5のように対話的にアクセスできます(太字がユーザ入力です)

正式なSMTPサーバはメール転送などの処理をする必要がありますが、Gyammサーバはメールを受け取ってデータを処理するだけです。図5のようなやりとりができるようにするために、表1のような一部のSMTPコマンドを解釈するコードを実装しておけば大丈夫です。



メールをHTMLに変換

DATA コマンドによってメールのメッセージの本体が送られます。Subject: やFrom: のようなメールヘッダもこの中に含まれます。メールのヘッダと中身を適切にデコードして解析してHTMLに変換することによって、Webブラウザから読めるようになります。



MIME

日本語を含むタイトルや添付ファイルを利用するときMIME形式へのエンコーディングが利用されています。

メールのSubject(タイトル)に日本語を利用したり、画像や文書などをメールに添付するのは現在あたりまえになっていますが、も

▼ 図5 telnetでメールサーバにアクセスしてみる

```
% telnet gyamm.com 25
Trying 133.242.135.184...
Connected to gyamm.com.
Escape character is '^J'.
220 gyamm.com ESMTP GYAMM
HELO example.com
250 gyamm.com
MAIL FROM: masui@pitecan.com
250 ok
RCPT TO: masui@example.com
250 ok
```

```
DATA
354 send the mail data, end with .
From: masui@pitecan.com
To: masui@example.com

Hello
.
250 ok
QUIT
221 Bye
Connection closed by foreign host.
%
```

▼ 表1 SMTPの仕様の一部

HELO	通信開始
MAIL	送信アドレスを指定
RCPT	受信アドレスを指定
DATA	メッセージ本文を指定
QUIT	通信切断

ととのメールの仕様ではタイトルも本文もASCII文字しか利用できませんでした。インターネットのメールが世の中に普及しはじめた1980年代は、メールの本文にJISコードを利用することにより、なんとか日本語のメッセージを送ることはできましたが、正式な仕様ではないので微妙なところがありました。また、パソコン通信などの非インターネットの世界では、タイトルや本文にShiftJISの日本語テキストが利用されていることが多く、メールの世界はかなり混沌とした状態になっていました。

1990年代のはじめまではそういう状況が続いていたのですが、1996年にMIME規格がRFC2045として制定されたおかげで、既存のメール配信システムを変更することなく、以下のような手法でさまざまなマルチメディアデータをエンコードしてメールメッセージとして送れるようになっていきます。

- ・データ型の指定

Content-Type：フィールドでデータ型や文字コードを指定する

- ・データのエンコード

画像や文書など任意のデータをテキスト形式に変換する方法を定義

- ・データの階層的な構造化

マルチパートというデータ型で複数のデータをまとめて扱う。複数のパートをまとめたデータを上位データのパートとすることもできるので階層的なデータ構造を表現できる

- ・ヘッダの国際化

任意の文字コードをヘッダで利用できるエンコード方式を定義



MIMEのデコード

Gyammでは、MIME形式で送られてきたメールをデコードして、人間が読める形に変換しています。テキストは普通のHTMLテキストに変換し、それ以外のは添付ファイルとして

リストするようにしています。

MIME形式で送られてきたメールメッセージをHTMLファイルに変換するには次の処理が必要です。

- ・ヘッダ文字列のデコード

日本語タイトルなどをデコードする

- ・MIMEパートの分離／デコード

階層的にパートを分離しつつ、データをデコードして添付ファイルなどを取り出す

- ・埋め込み画像の変換

1つのパートに含まれるHTMLファイルから、別のパートに含まれる画像を参照する場合はContent-IDを利用した特殊なimgタグが利用されるので、これを通常のimgタグに変換する



Gyammの将来



Gyammを使うのは簡単なのですが、SMTPやMIMEの扱いのため実装は若干複雑になってしまい、コロンブス指数が低いものになってしまったのは残念なところです。そもそもMIMEフォーマットはマルチメディアデータを無理矢理テキストに埋め込むための苦肉の策であり、最善のものではありませんし、将来のコミュニケーションはすべてWebベースになるでしょうから、メールは次第に廃れていくと考えられ、Gyammのようなシステムも不要になると思われます。しかしメールが完全になくするにはまだ10年はかかるでしょうから、それまでは十分利用価値があると思います。SD

コラム「GitHubにコードあります」

- 本連載でこれまでに紹介したシステムをGitHubで公開しています。ぜひご覧ください。

- ・GyaTV [https://github.com/masui/GyaTV]
- ・Gyump [https://github.com/masui/Gyump]
- ・Gyamm [https://github.com/masui/Gyamm]

コロンブス日和

第4回 Gyazo

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張らずに楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



Gyazo



今回は「Gyazo」という画像キャプチャ／アップロードサービスを紹介します(図1)。

Gyazoは2015年末の時点で、月間ユニークユーザ1,000万人、1日の画像アップロード数70万件という大規模なWebサービスですが、もともとは2007年ごろに私がコロンブスの卵的な発想で開発した小さなツールが大きく育ったものです。初期のGyazoは、パソコンのデスクトップを領域選択・キャプチャして、アップロードする単純なサービスでしたが、最近では動画キャプチャ機能／画像編集機能／コメント機能／検索機能／ブックマーク機能／チーム機能などが強化され、さまざまな用途に利用できる便利なツールに進化しています。

▼図1 Gyazo.comトップページ



注1) <http://thinkit.co.jp/free/article/0709/19/>

Gyazoはもちろん「画像」をもじった名前です。先月までの3回にわたり、「GyaTV」、「Gyump」、「Gyamm」など、「Gy」で始まるシステムを紹介してきましたが、これらはGyazoにちなんだものです。「京都」を「キョト」と発音する欧米人が多いように、欧米人は「gya」や「kyo」などの発音が苦手らしいので「Gy」で始まるドメインは取得しやすいようです。



Gyazoの基本的な使い方



Gyazoはパソコンのデスクトップ画面の一部をキャプチャして、Webにアップロードするツールですが、従来はこのために次のような3ステップの操作が必要でした。

- ・アプリを起動してスクリーンキャプチャを画像ファイル(A)としてセーブする
- ・画像編集ソフトで(A)を開き、必要部分を切り出して別のファイル(B)にセーブする
- ・(B)をWebにアップロードする

一方、Gyazoを利用すると次のような1ステップで済むようになります。このような操作を「Gyazる」と呼んでいます。

Gyazoを起動して画面の一部を選択すると、選択部分が自動的にWebにアップロードされて画像URLが割り当てられる

Gyazoった画像のURLはコピーバッファに保存されるので、Gyazoったあとですぐにチャット画面やメールテキストなどにURLを貼り付

けることができます。

Gyazoにアップロードした画像はWeb上に残るので、あとで資料などとして利用できます。アップロードされた画像のURLは、ハッシュ関数で生成されるので、GyazoったユーザがURLを公開しない限り、他人から画像が見られることはありません。



Gyazoの拡張機能



Gyazoはもともと私が個人的に使うために開発したもので、前述のようなソースコードをGitHubで公開^{注2}していますが、ビジネス化にともなってさまざまな機能を追加したものをGyazo.comで運用しています(いくつかの機能はGyazo Pro[課金版Gyazo]ユーザ限定です)。



編集機能

Gyazoった画像をブラウザ上で編集して文字や図形を追加できます(図2)。



コメント機能

アップロードされた画像にブラウザ上でコメントを付けることができます。「EpsonのHMDを試してみるところ。」というコメントを入力しています(図3)。



検索機能

入力したコメントを使って画像を検索できま

▼ 図2 画像を編集して矢印とテキストを追加



注2) <https://github.com/gyazo/Gyazo>

す。前述の画像や編集後の画像が検索結果に現れています(図4)。



ブックマーク機能

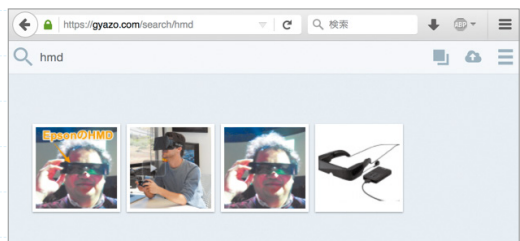
ブラウザで表示されているページをGyazoったときは、WebページのタイトルやURLがコメントに記録されます。また、アプリケーション画面をGyazoったときは「Microsoft Word」のようなアプリケーション名が記録されます。たとえば本誌のWebページをGyazoると、図5のようにページタイトル/ページURL/アプリケーション名(Firefox)が、コメントとして記録されます。

このように、Gyazoを画像付きブックマーク

▼ 図3 コメントの追加



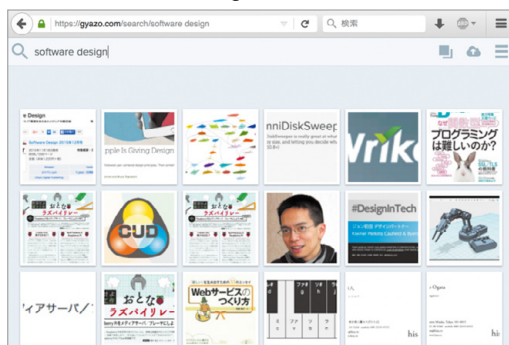
▼ 図4 「hmd」で検索を行ったところ



▼ 図5 本誌のページをGyazoった結果



▼図6 「Software Design」で検索を行ったところ



として利用できますし、WebページのタイトルやURLをもとに画像を検索することもできます。図6の例では、前回 Gyazo った画像に加え、「Software」「Design」というコメントがついた画像がリストされています。



関連画像検索

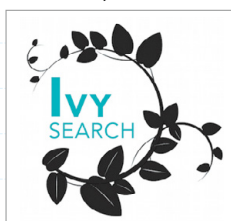
Gyazo.comでは、コメントやアップロード日付を利用して、画像の類似検索を行う「IvySearch」という検索機能(図7)を提供しています。

IvySearchとはGyazoった画像に関連付けて記録される、日付やコメントなどのメタデータを利用して、関連する画像を芋蔓的に検索できるようにしたものです。

図8は「3Dプリンタ」というキーワードで検索された画像の1つを選択したのですが、関連画像として別の3DプリンタやMaker Faireの画像などが自動的にリストされています。これらの関連画像はコメントなどから自動的に計算されます。

ここでMaker Faireの画像をクリックすると、選択された画像が表示され、その下にまた関連画像が表示されます(図9)。今回の画像には「Maker Faire」というコメントが記述されているため、Maker

▼図7 IvySearch



Faireに関連する別の画像がリストされています。

このようにして関連画像をたどることにより、昔 Gyazo った画像を効果的に検索していくことができます。思いがけない画像が関連画像として表示されることも多く、古い画像を探索的にブラウズできます。



Gyazo な一日

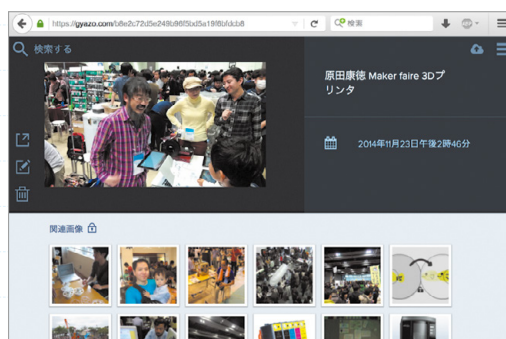


このように、Gyazoを使うと Gyazo ったときのアプリケーションやURLを保存して検索できるのに加え、IvySearchを使って内容が近い画像を簡単に検索できるので、広い応用ができます。私はGyazoの作者ですが信者でもあるので、次のように毎日あらゆる仕事や趣味にGyazoを活用しています。

▼図8 関連画像検索機能



▼図9 関連画像をたどることができる



- ・重要メールが来てたので Gmail ページを Gyazo ってコメントに「TODO」と書いておく (後から「TODO」で検索できるようになる)
- ・新着 Facebook 記事で見つけた面白い Web ページを Gyazo って「読み物」というコメントを書いておく
- ・勉強したい案件のメモを Wiki 上で作成し、Gyazo った画像を貼り付けておく
- ・Web ページを Gyazo ってプレゼン資料を作る
- ・新しい Mac が届いたので、昔 Gyazo っていたメール設定データをもとにしてメールソフトを設定
- ・購入要望の Mac の仕様が Gyazo で送られてきたのでチェックして注文
- ・ちょっとしたアイデアを思いついたのでメモ帳に書いて Gyazo って「アイデア」と書いておく
- ・昼の休憩時間に「読み物」とコメントしてあった記事を読む
- ・ついでに IvySearch で見つかった他の読み物も読む
- ・Facebook に自分の写真がアップされているのを見つけたので Gyazo ってコメントを付けておく
- ・新しいラーメン屋の評判を Gyazo って Wiki に貼っておく
- ・明日の出張のために付近のレストランをチェックして Gyazo っておく
- ・念のため電車の時刻を Gyazo っておく
- ・Skype パスワードを忘れたので「Skype パスワード」で Gyazo 検索したあと、ブックマーク登録されていた EpisoPass ページ^{注3}を使ってパスワードを思い出す
- ・請求書作成が必要だったので古い請求書を IvySearch して修正して印刷
- ・TODO 案件を検索してチェックした後、帰宅
- ・免許証のコピーが必要なサービスに対してデジタル写真を Gyazo ってメールで送付

- ・Peatix^{注4}のチケットとして使われる QR コードを Gyazo っておく
- ・美味しかったワインのラベルを Gyazo ってコメントを書いておく

今回の連載で使っている画像はすべて Gyazo でキャプチャしたものですし、資料の整理にも Gyazo を活用しています。



Gyazo の歴史と発展



Gyazo は、GyaTV や Gyamm、Gyump などと同様に、私がおもに自分で使うために作成したサービスなのですが、公開したところ思いのほか人気が出ました。初期のころは Gyazo.com を自前のサーバで運用していたのですが、アップロードされる画像の量が増えてきたので、クラウドサービスを利用するようにしたところ、サーバ代が家計を圧迫するようになってしまいました。そういう窮状をみかねて Nota Inc.^{注5}の洛西一周氏がサービスを引き継いでくれることになったのですが、そのあともユーザー数やアップロード画像の量は指数関数的に伸び続け、サーバ経費もたいへんなことになってきたので、2012 年ごろから真剣にビジネス化の検討を始め、機能を強化したり課金モデルを作ったりしてきました。現在は私も Nota Inc. の一員となり、Gyazo をグローバルなビジネスとしてさらに発展させるべく、ベンチャーキャピタルから出資を受けて開発を継続しています。

このような Gyazo の発展は、コロンブスの卵的なシステムであっても、大きなビジネスに化ける可能性があることの証明だと考えています。これまで紹介した各種の小さなシステムやこれから紹介するシステムも、ビジネスに発展させられるものがあれば良いと思っています。SD

注3) <http://episopass.com/masui/Skype123456>

注4) <http://peatix.com/>

注5) <http://www.notainc.com/ja>

コロンブス日和

第5回 Gyaki

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張らずに楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっているシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学

お絵描きシステムを忘れていませんか?

今回は「^{ぎゃき}Gyaki」というお絵描きシステムを紹介します。

計算機やWebがこれだけ普及しているのに、お絵描きシステムが世の中であまり使われていないのが不思議です。iPad Pro、Surface Pro、Surface Bookのような魅力的なペン計算機が最近たくさん登場しているのにもかかわらず、ペンを使ったお絵描きシステムは広く活用されていませんし、普通のユーザがお絵描きするシチュエーションも、あまり想定されていないように見えるのは気のせいでしょうか。スマホが流行る以前は、PalmなどのPDA(Personal Digital Assistant: 携帯情報端末)でメモを描く人も多かった気がするのですが、スマホやタブレットでお絵描きする人は以前より減っているような気がしています。

文章を使うよりも絵で説明するほうがわかりやすいことは多いでしょうし、デザインをスケッチしたいことも多いでしょうし、考えをまとめるために図を描きたいこともあるでしょうし、ペンで絵を描きたい機会が多いはずです。実際、紙のノートを活用している人は多いと思われますが、現在のパソコンはマウスやキーボードを使って、テキストや図を編集するのに使うのが普通だと思われており、ペンを使って知的生産活動を行うことはとくに推奨されていません。

Palmのようなペン型PDAが広く使われていたときは、ペンによるお絵描きもそれなりに利用されていたと思うのですが、スマホが普及してペンが駆逐されたために、手書きでメモする機会が減ってしまったように思われます。

しかし、

- ・性能が良いペンコンピュータが普及しつつある
- ・あらゆるブラウザでお絵描き機能が提供されている

という状況の現在、新しいWeb時代のお絵描きシステムが、もっと使われるべきだと思います。

Gyaki——ブラウザ上の手軽なお絵描きシステム

お絵描きアプリやWebサービスはたくさんありますが、達人のための機能を持つものが多く、とくにお絵描きが得意でない普通のユーザが手軽に使うためのものは多くありません。アイデアなどをメモしたくなったとき、最小の手間でお絵描きを開始して安全に保存できるようなお絵描きシステムを簡単に使えるようにしておきたいものです。

パソコンやタブレットで文章を書くときはエディタを起動するのが普通であるのと同様に、お絵描きするときはアプリを起動したりWebページを開いたりする必要があります。しかし絵を描く方法やセーブ方法を簡単化することによって、お絵描きのハードルを下げることはできるでしょう。私は、最小限の機能をもち手軽に利用できる「Gyaki」(楽ギャキ)というシステ

注1) <http://thinkit.co.jp/free/article/0709/19/>

ムを作って使っています。

Gyakiでは、とにかく手軽にアイデアをスケッチすることを目標としているので次のような機能だけを用意しています。

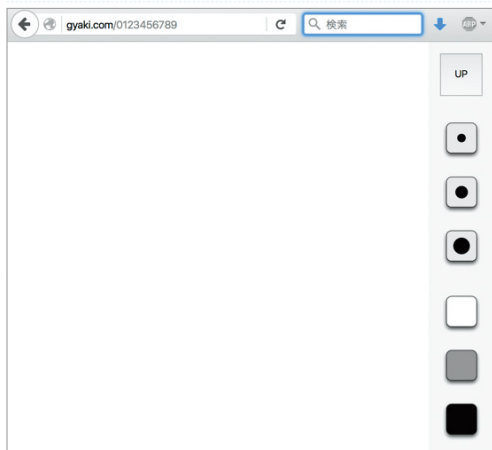
- ・手書きで線を描く
- ・3種類の太さを選択
- ・3種類の色(黒／灰色／白)を選択

一度描いたものを移動することはできませんし、undo機能すらありませんが、紙のスケッチでも同様ですし、たいていの場合においてそれでなんとかなっているわけですから、無理にたくさんの機能を用意する必要はないと割り切っています。

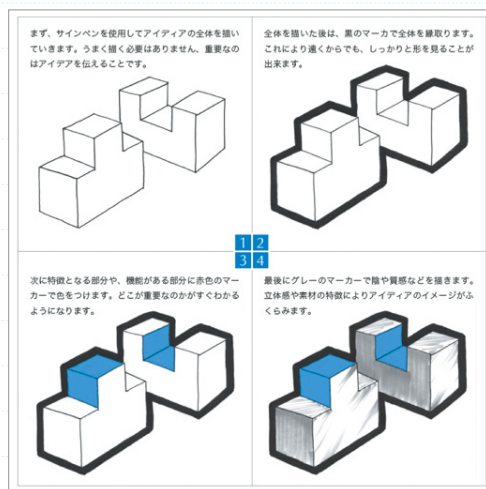
これらの機能だけを用意した理由は、情報科学芸術大学院大学(IAMAS)の小林 茂氏の著書『Prototyping Lab』(オライリー・ジャパン、2010年)で紹介されている「アイデアスケッチ」をスマホやタブレットで簡単に使いたいと考えたからです(図1)。アイデアスケッチは小林氏の同僚であるJames Gibson氏が考案したもので、学内外のワークショップなどを通じて小林氏が普及に努めているものです。

絵が下手な人間でもこの方法でスケッチを書けば、割とまともな感じに見えるという大きな利点があります。

▼ 図2 Gyakiの初期画面



▼ 図1 Prototyping Lab——「作りながら考える」ためのArduino実践レシピ(小林茂著、オライリー・ジャパン、2010年)より注2 作図: 蛭田直



Gyakiの使用例

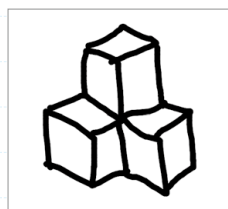


[http://Gyaki.com/\[適当な文字列\]](http://Gyaki.com/[適当な文字列])というURLにアクセスすることでGyakiのお絵描きを始められます(図2)。

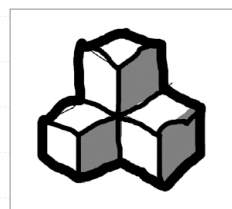
デフォルトのペンを使って、私が箱の絵を描くとこのようなサエない絵になってしまいましたが(図3)、アイデアスケッチで書けばこのようになります(図4)、かなりカッコ良くなることがわかります。

普通の紙でアイデアスケッチを利用する場合は、灰色のコピックや薄墨の筆ペンなどが必要になりますが、Gyakiだともっと気軽にアイデアスケッチを使うことができます。

▼ 図3 デフォルトではこんな具合



▼ 図4 アイデアスケッチ導入!



注2) <https://www.oreilly.co.jp/books/9784873114538/>

前述の Gyaki の URL の「適当な文字列」のところに、Gyazo のユーザ ID (Mac の場合 ~/Library/Gyazo/id に書かれている文字列) を指定すると、アップロードボタンを押したとき、お絵描き結果が自分の Gyazo アカウントに追加されます。URL が多少長くなってしまいますが、本誌 12 月号で紹介した Gyump を使って、短い URL を使用すれば便利です。描いた絵の一部分だけが必要な場合は、Gyazo で選択してアップロードすればよいでしょう。いずれの場合でも、何かを描いてから Web 上にセーブするための手間はかなり少なくなっていますし、Gyazo.com 上で後から検索するのも簡単です。



Gyaki の実装

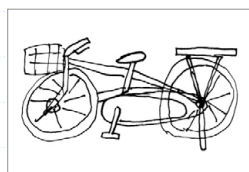
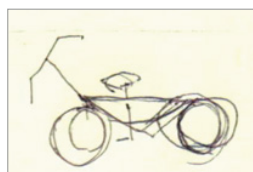


昔のブラウザ上でお絵書きするためには、Flash や Java などのプラグインを使う必要がありましたが、最近のブラウザの JavaScript ではいろいろな方法で何でも描けるようになりました。

- ・ Canvas 機能を使う
- ・ SVG 機能を使う
- ・ WebGL を使う

3次元描画を行うには WebGL が適しており、最近人気の情報視覚化システム「D3.js」では SVG が利用されているなど、用途によって適した描画システムは異なりますが、単純なお絵書きには Canvas 機能を使うのが一番楽です。Gyaki.com では Canvas 機能を使ったお絵描きをサポートしています^{注3}。

▼ 図5 いろんな人に自転車を描いてもらった例



注3) Gyaki のコードを公開しています (<https://github.com/masui/Gyaki>)。



上手なお絵描き方法の考察



白い紙の上いきなり絵を描くのは難しいものです。Gyaki + アイデアスケッチは、考え方をすぐに視覚化するには適しているのですが、きれいな絵を描こうとする場合や奇抜な絵を描くには向いていません。



トレース

お絵描きが得意でない人にいきなり自転車を描かせてみると、まともに書けないことが多いようです(図5)。

しかし「自転車」で Web 画像検索すれば、さまざまな自転車の写真が見つかりますから、それをなぞったりコピー&ペーストしたりすれば、苦勞せずに正しい自転車の絵を書くことができます。最近は多くのものが画像検索で見つかるので、現実世界に存在するものを描く必要性は少なくなっているかもしれません。



補助線の利用

整った表や図を描きたいときは方眼紙を使うと便利ですが、背景として罫目や集中線を利用すると、絵を描きやすくなることがあります。たとえば集中線を背景にして先ほどと同じ絵を描くと(図6)、より正しいパースで絵を描くことができます。



枯尾花システム

現実世界に存在しない物体を描きたい場合や、

▼ 図6 集中線を使って描画してみる

▼写真1 パレイドリアの例^{注4}

新しく形状をデザインしたいような場合は画像検索ができません。しかし、そのような場合でも想像力を喚起するお絵描き支援システムを作ることはできるかもしれません。

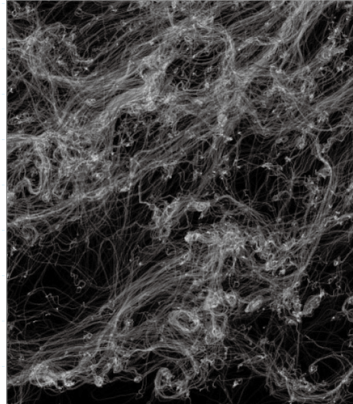
雲の中に動物が見えたり、自動車の前面が人の顔に見えることがよくありますが、このような現象はパレイドリア (Pareidolia) と呼ばれています(写真1)。「幽霊の正体見たり枯尾花^{かれおぼな}」という諺^{ことわざ}がありますが、ススキのパターンが人間の想像力を喚起することがあり得るということなのでしょう。

ランダムな曲線群や直線群の中に関係ない絵が見える現象は、レオナルド・ダ・ヴィンチが指摘していたほどであり^{注5}、『吾輩は猫である^{注6}』の中では迷亭先生が次のように語っています。

レオナルド・ダ・ヴィンチは門下生に寺院の壁のしみを写せと教えた事があるそうだ。なるほど雪隠などに這入って雨の漏る壁を余念なく眺めていると、なかなかうまい模様画が自然に出来ているぜ。君注意して写生して見給えきつと面白いものが出来るから。

枯尾花的パターンを自動生成すると、自分が描きたいものをその中に発見できるかもしれません。たとえば、先進的UIデザイナーの深津貴

▼図7 踊っている人が見える?



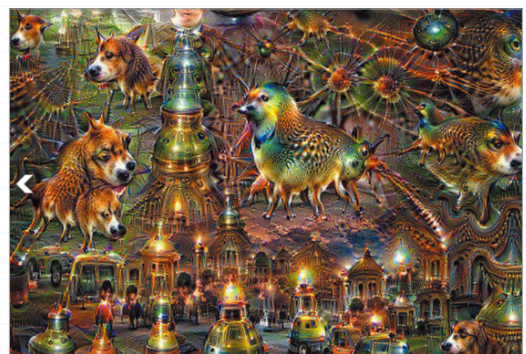
▼図8 トレース結果



之(fladdict)さんが開発した「Jackson Pollock 的スケッチシステム^{注7}」を使うと、絵心がない人でも躍動的な絵を描ける可能性があります(図7、図8)。

枯尾花お絵描きに向けたパターンというのは確かに存在するようですし、描きたいものの分野によって有効なパターンも異なると思われます。

Deep Learningで学習した画像認識システムを利用したDeep Dreamは人間の枯尾花認識機能を計算機でシミュレートしたものと言えるかもしれません(図9)。こういう技法を組み合わせることによって、もっと気軽にお絵描きできるシステムを作りたいと思っています。**SD**

▼図9 Deep Dream(<http://deepdreamgenerator.com/>)

注4) <https://ja.wikipedia.org/wiki/パレイドリア>

注5) <http://www.goodreads.com/quotes/978797-look-at-walls-splashed-with-a-number-of-stains-or>

注6) 青空文庫(http://www.aozora.gr.jp/cards/000148/files/789_14547.html)

注7) fladdict(<http://fladdict.net/blog/2016/01/jackson-pollock.html>)

コロンブス日和

第6回 Dynamic Macro

エンジニアというものは「楽をするためならどんな苦労も厭わない¹⁾」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張っただけで楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学

便利な 予測インターフェース

計算機の上で同じ作業を何度も繰り返さなければならぬことがよくあります。計算機は単純な繰り返し作業が得意なはずですが、つまらない作業を人間が繰り返さなければならぬことは意外と多いものです。たとえばExcelの表の中の負の数字だけアンダーラインを付けたいときはどうすれば良いのでしょうか？ そういう機能はExcelには用意されているかもしれませんが、知らなければ使えませんし、同じような処理であってもシステムに用意されていないかどうかどうしようもありませんから、この手の仕事があったときは泣きながら手作業で処理したり、頑張っただけでスクリプトを書いたりしている人が多いのではないのでしょうか。

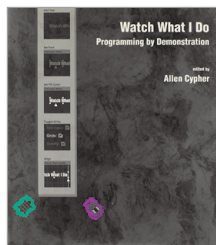
計算機上の操作を効率化するために「予測インターフェース」と呼ばれるシステムが広く使われています。スマホのテキスト入力を効率化する「予測入力システム」やブラウザのURL補

完機能、エディタの補完機能のような簡単な予測インターフェースは最近よく使われていますし、これまでの購入履歴を基にして商品を推薦するシステムなども一種の予測システムと言えるでしょう。予測インターフェース研究の歴史はけっこう古く、1993年には予測インターフェースの研究をまとめた『Watch What I Do』という本が出版されていますし、最近は例示や予測だけでプログラムを作ってしまうという研究も行われています(図1)。

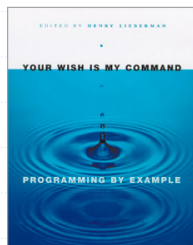
予測インターフェースシステムでは、アプリケーションに関連したデータベースやユーザの操作履歴などを基にして、ユーザの次の操作を予測することによってユーザの仕事を減らす工夫をしています。URL補完の場合はよく使われるURLのデータベースが利用できますし、プログラミング言語に関する情報を持っていれば、ユーザが次に入力する言語キーワードを予測できます。このような固定的なデータベースを用意しておくことも重要ですが、ユーザの操作履歴を予測

のためのデータベースとして利用すると便利です。ユーザが一度訪れたサイトのURLを覚えておけば補完に利用できますし、予測入力システムではユーザが利用した単語やフレーズが次の予測に利用されます。前述のExcelの例のような場合、負の数字にアンダーラインを付ける操作が繰り返されていることをシステムが検出できればユーザ

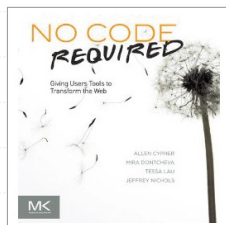
▼図1 予測インターフェースに関する書籍



Watch What I Do: Programming by Demonstration(Allen Cypher, The MIT Press, 1993年)



Your Wish is My Command: Programming By Example (Henry Lieberman, Morgan Kaufmann, 2001年)



No Code Required: Giving Users Tools to Transform the Web(Allen Cypher/Mira Dontcheva/Tessa Lau/Jeffrey Nichols, Morgan Kaufmann, 2010年)

注1) <http://thinkit.co.jp/free/article/0709/19/>

の次の操作を予測できるでしょう。



編集作業の効率化



文章やプログラムのようなテキストを編集するとき、同じような操作を繰り返すことがよくあります。たとえば、連続する行の先頭に記号を挿入したいような場合はカーソルを1行ずつ動かして記号を入力していくのが普通ですが、たくさんの行に対して同じ操作を繰り返すのはたいへんです。行頭に記号を挿入するスクリプトを書けば良いかもしれませんが、一度きりかもしれない処理のためにプログラムを作成するのは面倒ですし、プログラミングの知識が必要です。また、CSV (Comma Separated Values: カンマ区切り) データの桁を並び替えたいときはどうでしょうか？ CSVデータをExcelなどで読み取ってから並びを変えて出力すれば良いかもしれませんが、方法を考えるのも実際に作業するのも面倒です。



キーボードマクロ

このような作業を簡単にするために「キーボードマクロ」という機能が用意されているエディタがあります。キーボードマクロとは、一連のエディタ操作を1つのキー操作に割り当てることにより、複雑な編集操作を楽に実行しようというものです。たとえば「行頭に記号を挿入してから1つ下の行に移動する」という処理を[A]というキーに割り当てておけば、[A]を連打することによって連続する行の先頭に記号を入力していくことができますし、「カンマで区切られた部分を選択して移動してから次の行に移動する」という処理を[B]というキーに割り当てておけば、[B]を連打することによってCSVの桁を入れ替えていくことができます。

キーボードマクロは便利な機能ですが、キーボードマクロの定義開始と終了のための操作が必要であるうえに、処理を正確に登録するのが難しいという問題があります。たとえば前述の

例の場合、「1つ下の行に移動する」処理を定義に含めることを忘れてしまうと正しく動きません。



Dynamic Macro



キーボードマクロの機能をもっと簡単に使えるようにするため、私はキーボードの繰り返し操作から次の操作を予測して、自動的にキーボードマクロとして登録できるDynamic Macroというシステムを作って長年Emacsの上で利用しています。

Dynamic Macroの原理は非常に単純で、

「同じ編集操作を2回繰り返したあとで **CTRL+t** を押すと繰り返された操作が再実行される」

というものです。「二度あることは三度ある」と言うように、同じことが二度あればもう一度あるのは世の中でごく普通のことです。二度実行した操作をもう一度実行することもよくあることです。この方法はたいへん効果的です。



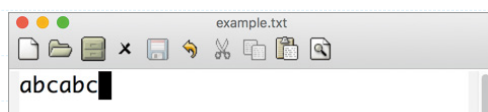
Dynamic Macroの利用例

Emacs上に実装したDynamic Macroの利用例を示します。図2はEmacsでabcbabcと入力したところです。

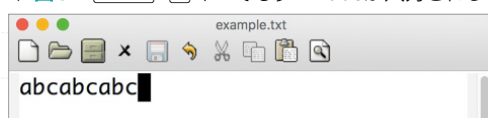
ここで **CTRL+t** を押すと、Emacsのキー操作履歴から「abcの入力」の繰り返しが検出され、キーボードマクロとして登録されて実行され、図3のようにもう1つabcが挿入されます。

再度 **CTRL+t** キーを押すと、図4のようにまたabcが入力されます。

▼ 図2 Emacsでabcbabcと入力



▼ 図3 **CTRL+t** キーでもう1つabcが入力される



これは単純な例でしたが、Dynamic Macroはもっと複雑な編集操作でも使うことができます。

図5のようなテキストの上から2行を図6のように編集したとします。

図6は、行頭に「puts 」を入力してから行末に移動して「」を入力して 次の行に移動するという操作を2回繰り返した結果ですが、ここで **CTRL+t** キーを押すと、この操作の繰り返しを検出されてマクロ登録されて実行されるので、画面は図7のように変化します。

さらに **CTRL+t** キーを何度か押すと画面は図8のように変化します。

このように、複雑な操作であっても、同じ操作を2回繰り返したあとではDynamic Macroで何度でも連続実行できることになります。



Dynamic Macroの特徴

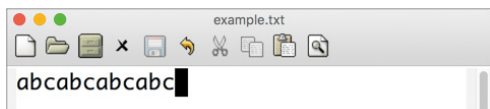
Dynamic Macroは繰り返し操作を効率化するシステムですが、予測インターフェースの考

え方をキーボードマクロに応用したものだともいえます。ユーザの繰り返し操作を基にして次の操作を「予測」し、それをキーボードマクロのように利用できるからです。予測と言っても繰り返し操作からの「予測」ですので、誤った予測をしてしまう可能性はほとんどありません。

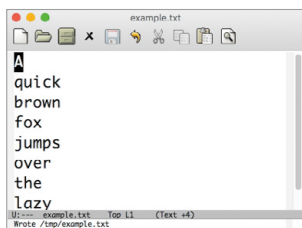
キーボードマクロと比較すると、Dynamic Macroには次のような利点があります。

- ・使うキーが **CTRL+t** 1つだけである
- ・定義の開始と終了を正確に指定する必要がない
→繰り返し操作中のどこで **CTRL+t** を押しても操作が再実行される
- ・操作を行ったあとで繰り返し実行を指示できる
→普通のキーボードマクロを利用する場合、これから繰り返し操作を行うぞ、と意識して登録を開始する必要があるが、Dynamic Macroの場合は操作のあとで繰り返しに気づいて再実行させることができる

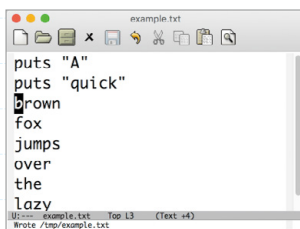
▼ 図4 **CTRL+t** キーでさらにもう1つabcが入力される



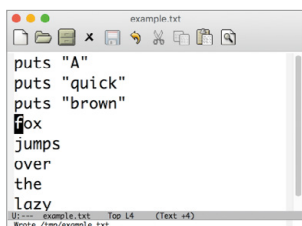
▼ 図5 Dynamic Macroで
もっと複雑な編集操作



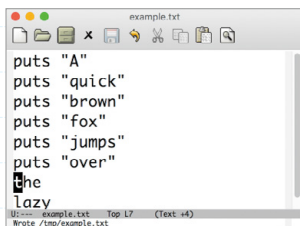
▼ 図6 テキストを編集してみる



▼ 図7 **CTRL+t** キーで入力補完



▼ 図8 **CTRL+t** キーを何度か
押してみる



Emacsでの実装

Dynamic Macroは最初はEmacsの上で実装されました。Emacsでは(recent-keys)という関数を使って、最近のキー操作履歴を知ることができるので、**CTRL+t** が押されたときに、この機能を使ってキー操作履歴を取得し、繰り返し操作が見つかれば、それをキーボードマクロとして登録して実行すれば良いことになります。

dmacro.elを改良したndmacro.el^{注2}というシステムもsnj14さんによって公開されています。ndmacro.elでは 1, 2, 3 と入力したあとで **CTRL+t** を押すと 4, 5, 6, ……のように、連続する文字列を入力していくことができます。

注2) <https://github.com/snj14/ndmacro.el>



Atomでの実装

GitHubが開発しているAtomというエディタが最近プログラマの間で人気が出ています。AtomはJavaScriptとブラウザ技術をベースに作成されたモダンで高機能なエディタで、ユーザーが自由にJavaScriptやCoffeeScriptで拡張機能を作成できます。Atomの拡張機能を利用することによって、Atom上でもDynamic Macro^{注3}を利用できます(図9)。

Atomには(recent-keys)のような履歴保存機能は用意されていないので、addEventListener()のような機能を使って、操作履歴を自力で覚えておくようにしています。



予測インターフェースの難しいところ



予測インターフェースは便利なものですが、あくまで「予測」ですので、システムの予測がユーザーの意図と異なる可能性が常に存在します。ユーザーが1, 2, 3, 1, 2, 3と入力したとき、ユーザーが次に入力したいのが1なのか4なのかはユーザー本人にしかわかりません。高度な予測を行おうとすると、予測を間違える可能性が高くなりますし、複数の予測結果から希望するものを選択する必要が出てくるかもしれませんから、機能と使い勝手のバランスが重要になってきます。

予測入力システムの場合、ユーザーが入力した

い単語が常に第一候補として提示されるわけではありませんが、欲しい単語が候補リストに含まれている可能性が高ければ、それほどユーザーは不満を感じません。一方、正しく予測されることへの期待が大きいにもかかわらず、頻繁に間違った予測が実行されてしまうようであれば、ユーザーの失望が大きいため、予測システムの利用をあきらめてしまうかもしれません。予測インターフェース研究の歴史は長いにもかかわらず、最近まであまり利用されていなかったのは、こういう理由が大きいと思われます。Dynamic Macroの場合、まったく同じ操作を繰り返すだけでするので間違った予測が実行されることはほとんどないのですが、それでも間違えることが皆無ではありません。もっと微妙な予測の場合は、システムが間違った予測をしてしまうことは多いでしょう。

Dynamic Macroのような予測機能は、実世界のさまざまな場所で使える可能性があります。同じ設定で2回動かしたらその設定を繰り返せる洗濯機が売られていたことがありますし、同じフレーズを2回弾いたら何度も繰り返してくれるピアノがあれば便利かもしれません。いろいろな予測機能を有効に使うことによって、世の中の単調作業を何でも効率化していきたいものです。SD

コラム 「エディタと私」

- 私は1980年代からEmacsを利用しており、その上で20年以上Dynamic Macroを使い続けています。
- Emacs以外のエディタも使ってみたい気持ちはあったのですが、Dynamic Macroが使えないエディタの利用は私には考えられないので、ずっとEmacsを使い続けてきていました。最近AtomでDynamic Macroが動くようになったので、これからはAtomを利用するようにしようかと考えています(この原稿もAtomで書いています)。
- EmacsもAtomももともとプログラマ向けに開発されたエディタですが、Atomは30年(!)後発なだけに、Emacsに比べると利用のハードルは低いですし、GUI的な機能もしっかりしており、万人に勧められるものに進化しつつあります。これからはAtom上で誰もが便利に使える他のツールの作成にも挑戦したいと思っています。

▼ 図9 Atomエディタのマクロ「atom-dynamic-macro」



注3) 今回のソフトウェアの公開場所

- ・ Emacs版 Dynamic Macro (<https://github.com/masui/DynamicMacro>)
- ・ Atom版 Dynamic Macro (<https://github.com/masui/atom-dynamic-macro>)

コロンプス日和

第7回 Gyaim

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンプスの卵のようなゴキゲンな発明によって頑張^いって楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽する方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



かな漢字変換システム

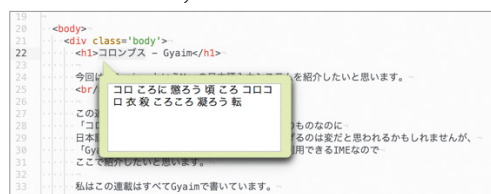


今回は「Gyaim^{注2}」という Mac の日本語入力システムを紹介します。

この連載は、シンプルなのに便利な「コロンプスの卵」的なシステムを紹介する趣旨のものなのに、日本語入力システムのような複雑なものを取り上げるのは変だと思われるかもしれませんが、「Gyaim」は単純な原理にもかかわらず実用的に利用できる IME ですので、ここで紹介したいと思います(図1)。私はこの連載をすべて Gyaim で書いています。

現在のパソコンでは、「かな漢字変換システム」で日本語入力を行うのが普通になっています。かな漢字変換システムは、1978 年に東芝から販売された日本語ワープロで初めて導入されたものですが、パソコンの黎明期から標準的に利用されてきています。かな漢字変換システム以外にもさまざまな日本語入力システムが提案されてきましたが、それほど努力しなくても普通のユーザがとりあえず使えるうえに、熟達すれば高速に入力が可能だという特長があるため、

▼ 図1 Atom で Gyaim を使っているところ



注1) <http://thinkit.co.jp/free/article/0709/19/>

注2) <http://masui.github.io/GyaimMotion/>

かな漢字変換方式の日本語入力が廃れることはなさそうです。



連文節変換

現在のパソコンのかな漢字変換システムでは、いわゆる連文節変換が主流になっています。連文節変換とは、文章の読みをすべて入力してから、漢字混じりの日本語文字列に一気に変換する手法のことで、たとえば「わたしのなまえはなかののです」という入力を「私の名前は中野です」に変換するというものです。「私の」「名前は」のように、息継ぎできる場所で文を区切った単位を「文節」と呼びますが、連文節変換では複数の文節を含む文を一気に変換できるのが特長です。

連文節変換は一見便利そうですが、実は次のようなさまざまな問題があります。

- ・正確な入力が必要
watashinonamae... のような文字列を1文字も間違えずに入力する必要がある
- ・完全な読みの入力が必要
- ・変換誤りの訂正が必須
- ・非力なマシンで利用しにくい

これからのユビキタス社会においては、パソコンの熟練者だけが使える入力手法ではなく、どこでも誰でも簡単に使えるシンプルで柔軟な入力方式が必要で、次の要件が満たされる必要があるでしょう。

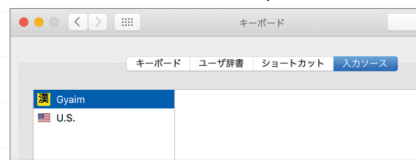
- ・必要なキーやボタンの数が最小限
- ・操作の種類や量が少ない

- ・ユーザがカスタマイズ可能
- ・さまざまな環境で同じ方式が利用可能

▼ 図2 SKK利用例



▼ 図3 日本語入力としてGyaimを選択



このような要件を満たすためには、単純なアルゴリズムに基づいて単純な操作で入力を行えるようなコロンブスの卵的な入力システムが必要だと思われます。



SKK

連文節変換を利用しない場合、かな漢字変換は辞書を参照して読みを漢字に置き換える単純な検索作業に近く、動詞の活用への対応などを除けばそれほど難しいものではありません。

単純なしくみで日本語入力を行うシステムの例として、Emacs上のSKKというシステムがあります。SKKは1987年ごろ、京都大学名誉教授の佐藤雅彦氏が東北大学にいたとき開発したEmacs用の日本語入力システムです(図2)。

SKKには英数字モードと日本語モードがあります。日本語モードでローマ字を入力するとひらがなが入力されますが、「書く」のような漢字を含む動詞を入力したい場合は「KaKu」のように大文字と小文字を混ぜたローマ字を入力します。最初の大文字「K」は変換の開始を意味しており、2つめの大文字「K」は送り仮名の開始を意味しています。SKKの辞書には次のようなエントリが含まれており、

かく /核/格/各/角/画/確/:
kak /書/掛/欠/

「Kaku」と入力された場合は「核」「格」「各」のような漢字への変換が行われ、「KaKu」と入力された場合は「書く」や「欠く」のような文字列に変換されるようになっていきます。つまり変換開始場所や送り仮名の場所をユーザが明示することによって、単純なアルゴリズムでの変換を可能にしているわけです。



Gyaim

SKKは単純なしくみで効率的な日本語入力ができる、コロンブスの卵的な優れた日本語入力システムなのですが、一般的なかな漢字変換システムと同じように、すべての読みを正確に入力する必要があるうえに、漢字や送り仮名についてユーザが明示的に指定しなければならないためユーザの負担が大きく、万人向けとはいえない難いものでした。私は1990年代ごろはSKKを愛用していたのですが、携帯電話の予測型テキスト入力システム「POBox^{注3)}」の開発後はあらゆる場所でPOBoxに準じた入力手法を使っており、MacではGyaimを使っています。

GyaimはMacで動くシンプルな日本語入力システムです。Gyaimは約1,000行のRuby(Ruby Motion^{注4)}を使用)で記述されており、スマホの予測入力システムと同じように利用できます。

Gyaimは次の手順でインストールします。

- ・ <http://masui.github.io/GyaimMotion/> から Gyaim.dmg をダウンロードして展開
- ・ Gyaim.app を ~/Library/Input Methods に置く

このように設定を行ったあと、環境設定画面の[キーボード]→[入力ソース]でGyaimを選択し、追加します(図3)。



Gyaimの実装

IMEの作成には、ユーザインターフェースとかな漢字変換アルゴリズムが必要です。

注3) <https://ja.wikipedia.org/wiki/POBox>

注4) <http://www.rubymotion.com/jp/>

✓ IMEのユーザインターフェース

Macには、IMKit^{注5}というIME作成用ライブラリが用意されており、それを利用してIMEを作成できます。IMKitにはさまざまな機能が用意されていますが、GyaimではIMKInputControllerクラスのhandleEventというAPIのみを利用しています。候補を表示したり選択したりするには、任意のCocoaライブラリを利用できます。

✓ かな漢字変換アルゴリズム

ユーザインターフェースはシステムごとに用意する必要がありますが、かな漢字変換アルゴリズムはAndroidでもMacでも同じものを利用できます。GyaimではRubyでかな漢字変換アルゴリズムを実装していますが、同じアルゴリズムをJavaやJavaScriptで実装すれば、Androidやブラウザなどで利用できます。

GyaimではSKKと同様に読みと漢字の対応辞書を使ってかな漢字変換を行います。たとえば日本語入力モードで「とうきょう」と入力されたとき、辞書から「とうきょう」という読みを持つ単語を検索して、リストして候補として表示し、ユーザの操作で候補を選択してテキストに貼り付けます。

「東京」のあとには「駅」や「大学」のような単語が続くことがあります。「東京駅」「東京大学」のような単語をすべて辞書に登録しておくのはたいへんですので、「東京は地名である」「地名のあとには駅や大学が続くことがある」という情報を辞書に登録しておくことにより、「東京駅」が辞書に登録されてなくても「とうきょうえき」を「東京駅」に変換できるようにしています。

まったく同じ手法で動詞の変化形も扱うことができます。たとえば辞書に、

- ・「書」の読みは「か」である
- ・「書」のあとには力行五段活用語尾がつながる

・「か」は力行五段活用語尾である

といった情報を定義しておけば「かかない」を「書かない」に変換できます。このように、単語の読みと属性、接続情報を定義しておくだけで、それなりに自然言語を入力できますし、日本語以外の文字入力でも利用できます。



Gyaim の特殊機能

Gyaimでは前述のような非常に単純な変換手法を使っていますが、特殊な機能も用意しています。自前のIMEだと好きな機能を自由に実装できるのが楽しいところです。

✓ 単語登録機能

一般的なIMEは単語登録が面倒なことが多いので、頻繁に単語登録を行っている人は多くないでしょう。たいていのIMEにおいて、単語登録は入力とはまったく異なるインターフェースになっていますが、Gyaimでは入力操作と登録操作をほとんど同じにすることにより簡単に単語登録が行えるようになっています。Gyaimでは「選択中の文字列があったりコピーバッファに文字列があった場合は候補として表示する」という単純な方法で単語登録を可能にしています。

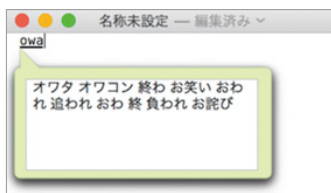
たとえば「\(^o^)/」という文字列を「owata」という読みで登録したい場合(図4)、「\(^o^)/」という文字列を選択またはコピーしておいてからGyaimで「owata」と入力すると「\(^o^)/」が候補の先頭に表示され(図5)、これを選択して確定することにより単語登録が終了します(図6)。

✓ 画像変換

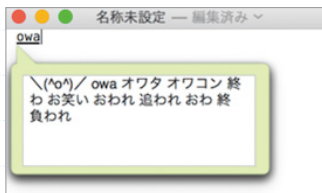
Gyaimでは、辞書の中で文字列の代わりにGyazoの画像ファイル名を登録しておくと漢字の代わりに画像を入力できます。たとえば私の顔画像を「masui」という読みで登録しておくと、

注5) <https://developer.apple.com/library/mac/documentation/Cocoa/Reference/InputMethodKitFrameworkRef/>

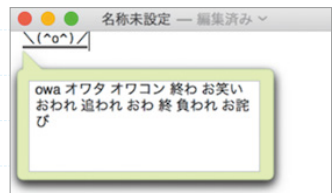
▼図4 「\ (o^o) /」が登録されていない状態



▼図5 「\ (o^o) /」がコピーバッファにある状態で「owa」と入力



▼図6 「\ (o^o) /」を選択。確定すると単語登録される



Gyaimで「masui」と入力することにより顔画像を入力できます(図7)。

✓時刻の入力

Gyaimでは「ds」と入力すると現在時刻が候補に出るようになっており、文章やプログラムの中に手軽にタイムスタンプを記録しておくことができます。数式演算機能のような特殊な機能を追加することも簡単です。こういったちょっとした機能が用意されているIMEはありますが、GyaimはRubyを少し書くだけで特殊な機能でも簡単に追加できるのがうれしいところです。

▼図7 画像変換



✓秘密文字列の入力

ブラウザなどでパスワードやクレジットカード番号を入力するのは面倒なものです、IMEでこれらを覚えておけば簡単に入力を行うことができます。もちろんこれらを生テキストで単語登録するのは危険ですが、秘密の文字列を変換することによりクレジットカード番号が候補に表示されて入力可能にしています。

✓Google変換の利用

Gyaimの辞書は貧弱ですのでうまく変換できないこともよくありますが、そういうときはGoogle入力ツールのようなWeb上の変換APIを利用して変換を行えるようにしています。

IMEは複雑な要素を含んでいるのでGyaimの詳細について書くことはできませんでしたが、とりあえず簡単なIMEをMacで作ることができることが理解いただけたでしょうか。

私のように単純な変換方式を好む人はGyaimのような方法を工夫すれば良いでしょうし、連文節変換が好きな人はGoogleなどの変換APIを利用することもできます。IMEの作成がたいへんだった昔と異なり、現在はIME用のAPI/使いやすい強力な言語/Webの変換API/高度な入出力機能などを利用できますからIME作成のハードルは極めて低くなっていると言えるでしょう^{注6}。IMEを自分で作る試みが、今後もっと増えるといいと思っています。SD

注6) Gyaimのソースコードは <https://github.com/masui/GyaimMotion> で公開しています。

コロナブス日和

第8回 EpisoPass

エンジニアというものは「楽をするためならどんな苦労も厭わない¹⁾」ものだと言われていますが、コロナブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



パスワードの諸問題



さまざまなWebサービスでユーザ認証のためにパスワードが使われています。パスワードはいろいろな問題が多いシステムであり、パスワードにかわるさまざまな認証システムが提案されてはいるものの、すべての点でパスワード認証より優れたシステムは存在しないと言われていたので、パスワードによる認証がほかの方法で置き換えられる時代はすぐには来ないでしょう。

パスワードの最も嫌なところは、覚えておくのがとてもたいへんなことだと思います。パスワードを忘れて困った経験がない人はいないでしょう。いろいろなサービスで同じパスワードを使い回すのは危険ですし、パスワードはときどき変更したほうが安全だと言われていますが、複雑で長いパスワードをたくさん覚えておくことは不可能ですから、仕方なく同じパスワードを使い回している人は多いと思います。

異なるパスワードをすべて記憶することが不可能なのであれば、紙やファイルに書いておけば良いかもしれませんが、パスワード文字列をそのまま記録するのは危険なので、複数のパスワードを暗号化して覚えておくためのさまざまなパスワード管理システムが利用されています。たいいていのパスワード管理システムは1つの「マスターパスワード」を利用してほかのすべてのパスワードを管理するようになっていますが、

マスターパスワードは覚えておかなければなりませんし、パスワード管理システムはどこでも使えるとは限りません。できれば特殊なシステムを使うことなく、複雑な多数のパスワードを利用できるほうがうれしいでしょう。

そもそも強力なパスワードを作成してからそれを覚えたり管理したりするというやり方が間違っているのではないのでしょうか。新しく作ったパスワードを覚えたり管理したりするのではなく、すでに知っていて忘れようがないような秘密の記憶を基にして、複雑なパスワードを生成して使うことにすれば、パスワードを覚えられない問題は解決するはずです。



EpisoPass



子供のころちょっと怪我をしたとか、イジメられた嫌な経験とか、うっかりした失敗を隠していたとか、他人に話したことはないけれども忘れることがないような秘密の記憶というものが誰にもたくさんあると思います。強い体験に基づく記憶は、エピソード記憶と呼ばれ、時間が経っても消えることはありません。一方、数式や電話番号を記憶しようとしてもなかなか覚えられませんし、覚えていたつもりでも時間が経つとキレイに忘れてしまっていたりするものです。このようなものは意味記憶と呼ばれ、エピソード記憶に比べると長期的な記憶が困難です。

パスワードがエピソード記憶だったら良いのですが、体験的にパスワードを覚えるのは無理でしょう。であれば逆に、忘れることがない秘

注1) <http://thinkit.co.jp/free/article/0709/19/>

密のエピソード記憶を基にして、パスワードを生成するようにすれば、秘密で複雑で忘れないパスワードを安心して使えるようになるはずです。

EpisoPassは、ユーザが忘れることがない個人的なエピソード記憶を、複雑な文字列に変換することによって安全なパスワードを生成するシステムです。

パスワード文字列は次の手順で生成されます。

- ・パスワード生成の「種」となる文字列(シード文字列)を用意する
- ・忘れることがない個人的なエピソード記憶に基づく秘密の質問をいくつか作成し、それぞれについて1つの正答と複数の偽答を用意する
- ・質問と回答の組に基づいてシード文字列に換字操作を行う。すべてに正しく回答したとき生成される文字列をパスワードとして利用する

問題文字列とユーザが選んだ回答文字列を結合した文字列を生成し、そのMD5値を基にしてシード文字列を換字することによりパスワードを生成しています。

ブラウザでの利用

図1は私がTwitterのパスワードを生成するために、ブラウザでEpisoPassを利用しているところです。

シード文字列として「Twitter123456」という文字列を指定しており、4個の秘密の質問に対する回答選択に応じて、「Mfveabn574923」のようなパスワード候補が生成されます。

異なる答えを選択するとまったく異なる文字列が生成されます。

シード文字列を「Facebook123456」に変更すると、生成されるパスワードは図2のように変化します。このように、サービスごとに異なるシード文字列を利用することによってさまざまなパスワードを簡単に生成できます。

「いつのパスワードですか」のような質問を用

意しておき、「2016/5」「2016/6」のような選択肢を用意しておけば、毎月異なるパスワードを生成できます。

シード文字列の8文字目が数字である場合はパスワードの8文字目も数字になるなど、シード文字列の文字種に対応したパスワード候補が生成されるようになっています。パスワードとして大文字／小文字／英数字／記号をすべて利用しなければならないサービスの場合は、シード文字列に「PassWord123!@」のような文字列を指定します。

最初の秘密の質問は私の小学校のときの体験に基づくもので、最後の質問は数年前の体験に関するものです。これらは古いエピソード記憶になっているので、私が将来答を忘れることはほとんど考えられませんが、私以外の人間がこのような質問に答えることは難しいので、正し

▼ 図1 EpisoPassの利用例

▼ 図2 EpisoPassによるパスワード生成

いパスワードを得ることはできません。

秘密の質問と答はブラウザで編集でき、右上の[サーバにセーブ]ボタンを押すことにより、シード文字列、秘密の問題、答のリストがサーバにセーブされます。[ファイルにセーブ]ボタンを押すとJSONデータをパソコンにダウンロードでき、パソコン上のJSONデータをブラウザにドラッグドロップすると、サーバにアップロードできます。ユーザはどれが正答かを指定するわけではないので、問題データを見てもユーザのパスワードはわかりません。



Android アプリケーション

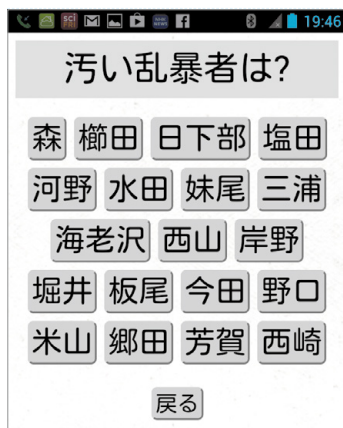


ブラウザからWebサービスを利用する場合、ブラウザとサーバとの間の通信を盗み見されたり、パソコン上の操作を記録されたりする心配を完全になくすことはできません。ブラウザでEpisoPassを使う場合、パスワードはブラウザ内部でJavaScriptにより生成されるので、一度ページを表示したあとはネットワークを遮断してもパスワード計算できるのですが、ブラウ

ザを使わずにパスワードを作成できるほうがより安心でしょう。このため、通信をまったく行わずにマシン単体でパスワード計算を行うためのAndroidアプリも用意しています。ページの右上の[Androidアプリ]ボタンを押すと、現在表示している秘密の問題と答を内蔵したAndroidアプリが、サーバ上でビルドされてダウンロードされます。Android端末でアプリを実行すると図3のような画面が表示されます。シード文字列を設定して[開始]ボタンを押すと図3のように質問が1つずつ表示され、ボタンを押してすべて回答するとパスワードが計算され図4のように表示されます。

回答の選択とパスワード計算はAndroid端末で実行されるため、端末を機内モードに設定するなどの方法でネットワーク接続を遮断した状態でもパスワードを計算できます。EpisoPassをインストールしたAndroid端末を持っていれば常に各種のパスワードを計算できるので、他人のマシンや公共の場所に設置されたパソコンなどでも、容易にTwitterなどのネットサービスを利用できます。

前述の方法でEpisoPassアプリをサーバからダウンロードする場合は、ブラウザ上で秘密の問題をサーバに登録する必要がありますが、秘密の問題をまったくネット上に露出することなくアプリを利用することもできます。秘密の問題を含まないEpisoPassアプリをGoogle Playで公開しているので、これを端末にインストールしたあと、ローカルマシンで作成した秘密の質問を端末に転送すれば、EpisoPass.comからダウンロードしたアプリと同様に利用できます。この手法を使うと秘密の質問が通信路を通ることがないので安全ですが、アプリのセットアップの手間は増えます。



◀ 図3
Androidアプリ
での使用例



◀ 図4
生成されたパス
ワード



EpisoPassの安全性



EpisoPassで選択枝が20個の質問を10個使

用する場合、総当たりでパスワードを生成するには約10兆(= 20^{10})通りの試行が必要になり、大小英文字からランダムに8文字を並べた約50兆(= 52^8)通りの文字列からパスワードを選ぶ場合と同程度の強度になります。総当たり攻撃が可能なオフライン運用ではこのような強度は重要ですが、オンラインサービスでは、パスワード入力を何度か間違えるとサービスがブロックされるのが普通なので、それほど長いパスワードを用意する必要はないでしょう。

秘密の質問を利用する認証は脆弱だと言われることがあります。パスワードをリセットするために「母親の旧姓は?」「最初に飼ったペットの名前は?」のような質問に対してユーザに答えを登録させるサービスがありますが、このような問題は他人が調べたり推測したりすることが簡単ですし、秘密の質問の数は一般的に少ないので、パスワードよりも脆弱なのはたしかです。EpisoPassでは、他人には解くことが難しく自分では忘れないような秘密の質問をたくさん登録しておけば安全です。



秘密の質問の選択

EpisoPassでは、他人が推測することが難しく、自分は決して忘れないようなエピソード記憶を秘密の質問として利用します。忘れないエピソード記憶であっても、次のような性質を持つものは秘密の質問として不適当です。

- ・ 自慢になるもの(何かの機会にうっかり他人に自慢しまう可能性があるもの)
- ・ ネット上に記録が残っているもの
- ・ 他人と情報を共有しているもの
- ・ 趣味や嗜好に関連するもの(他人に推測されやすいうえに嗜好が変化する可能性があるもの)

このようなものではなく、「わざわざ人に話すことはないが自分の記憶に強く残っているような無難なエピソード記憶」を秘密の質問とし

て利用するのが良いでしょう。例に挙げた「鉄条網で怪我した場所は?」という問題の場合、私はこの経験について他人に話したことはありませんし、今後自慢することがあるとは思えませんが、痛い思いをしたことは忘れませんから、問題として適切だと言えるでしょう。



偽答の作成方法

問題の種類によっては偽答の生成が難しかったり、答えが予測できたりしてしまう場合があります。たとえば「好きなスポーツは?」のような質問の場合、たくさんの偽答を用意することが難しいですし、本人を知っていたら想像がつくかもしれませんので問題としては適切ではありません。

一方、答えが人名や地名の場合、正答に似た人名や地名を並べることは簡単です。たとえば「世田谷」が正答であるとき、「目黒」「杉並」のような偽答を用意するのは簡単です。例に挙げた「鉄条網で怪我した場所は?」という問題の場合、似たような地名をたくさん並べることが簡単なのでEpisoPassの問題として適切だと言えるでしょう。

ある単語と同じカテゴリに属する単語を探す「同位語検索」と呼ばれる手法がいろいろ提案されているので、これを利用して正答と同じカテゴリに属する単語を自動的にリストできれば、簡単に偽答リストを作ることができるでしょう。

私は数年前にEpisoPassを開発してから、FacebookやTwitterなどさまざまなWebサービスのパスワードの管理にEpisoPassを使っており、パスワードに関する悩みが完全に解消されました。パスワードで消耗している方はぜひご利用ください。SD

コラム 「ソースコードも公開中」

- EpisoPassは<http://EpisoPass.com>で運用中です。ソースコードも公開しています(<http://github.com/masui/EpisoPass>)。

コロンブス日和

第9回 ExpandHelp

エンジニアというものは「楽をするためならどんな苦勞も厭わない¹⁾」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



ヘルプの憂鬱



複雑なシステムにはオンラインヘルプやマニュアルが用意されているものですが、パソコンやアプリケーションの解説本が大量に売られているところを見ると、ヘルプやマニュアルは十分役にたっているように見えます。

ヘルプが利用されない最大の理由は**必要な項目が見つからない**ことでしょう。たとえばMacを米国時間に設定したいと思って「米国時間」で検索すると図1が表示されます。

これではわけがわからないので「時間」で検索すると、より多くの項目が表示されます(図2)。米国時間の設定には関係なさそうです。

「時刻」や「時計」で検索すれば「日付と時刻」環境設定」という項目が見つかりますが、これで米国時間に設定できるのかはよくわかりませんし、そもそも「米国時間」や「時間」でも検索すべきではないでしょうか。

必要な項目が見つかった場合でも、そこに記述されている方法は自分で実行する必要があります。ユーザはMacを米国時間に設定したいだけであり、時刻やタイムゾーンを設定する方法を知りたいわけではないのに、ヘルプシステムは一般的なやり方しか教えてくれませんか、ユーザは自分で環境設定パネルを開い

て、米国時間に設定する必要があります。

このように、たいいていのヘルプシステムは、

- ・ヘルプ項目が見つからない
- ・自分で操作が必要

という問題があることになります。一方で、最近では普通にWeb検索すればそれなりに有用情報が見つかることが多いため、ヘルプシステムへの期待が低くなってしまっているのだと思われます。



ExpandHelp



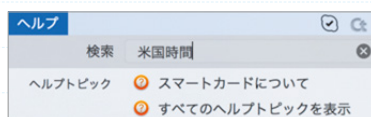
Web検索に苦勞することが少なくなってきたのは、検索エンジンの性能が向上したこともさることながら、情報の量がすごい勢いで増えてきたことによる影響も大きいでしょう。どこかのブログに「Macの時計を米国時間に設定する方法」という記事があれば、普通に「Mac 米国時間」で検索すれば見つかる可能性が高いと思いますが、さまざまな表現で情報が表現されていれば検索に成功する可能性が高くなると言えるでしょう。



Generate and Filter

パズルを解く方法として「Generate and

▼ 図1 米国時間で検索すると……



▼ 図2 さらに「時間」で検索すると……



注1) <http://thinkit.co.jp/free/article/0709/19/>

Test」と呼ばれるアルゴリズムがあります。たとえば「数独」を解こうとする場合、理屈で推論して空きを埋めていくのではなく、あらゆる数字の組み合わせを生成して順番に試してみれば、時間はかかるかもしれませんが必ず最終的に答えが見つかるはずです。これはひどく効率が悪いやり方に見えますが、明らかに間違っている組み合わせを早目に無視するような工夫をすれば、このような簡単なアルゴリズムでもうまくパズルを解くことができる場合があります。

ヘルプの検索でもこれに似た方法を使うことができます。ヘルプデータやWebをキーワード検索する場合、検索対象は固定であり、それを検索するためのキーワードをユーザが工夫するのが常識になっています。たとえば「時間」というキーワードを使って、米国時間に関連した情報を探すことに失敗した場合は「時刻」や「タイムゾーン」のような別のキーワードで再検索を試みるのが普通でしょう。これは結果的に「(時間|時刻|タイムゾーン)」のような正規表現で検索したのと同様ですから、ユーザにとっては面倒な話ですが、Web検索でもファイル検索でもこういうやり方があたりまえになっているので、これをとくに苦痛だと思っていない人も多いかもしれません。

しかし本当にユーザにとって便利なヘルプ情報を提供しようとするならば、ユーザに検索語を工夫させるのではなく、「米国時間にセット」でも「タイムゾーンを設定」でもヘルプエントリを検索できるように、あらゆる表現でデータを用意しておくほうが親切です。たとえばヘルプシステム側に「タイムゾーンを(米国|英国)(時間|時刻)に(設定|セット|変更)する」のような正規表現を用意しておき、これを使ってあらゆるヘルプ表現の組み合わせを生成し、ユーザの与えた検索語でフィルタリングすることになれば、「米国時間にセット」でも「タイムゾーンを設定」でもヘルプが見つかるようになるはずです。

Generate and Testであらゆる解の候補を生

成してから解のチェックをするのと同じ要領で、あらゆる説明文字列のパターンを生成しつつ、その中でユーザの検索文字列とマッチするものでフィルタリングを行う「Generate and Filter」方式を使えば、ユーザはキーワードについて頭を使うことなく検索に成功する可能性が高まります。このような手法をExpandHelpと呼ぶことにします。

展開されたデータは巨大になってしまう可能性があります。最近のパソコンでは多少大きなデータを利用しても問題ありませんし、Generate and Testの場合と同じように実装を工夫すれば、実際にすべての文字列を展開せずに検索を行うこともできます。



ヘルプ項目の実行

ヘルプ項目の検索に成功した場合、記事を参照しながらユーザが自分で操作を行う必要があります。この手間をなくすには、ユーザが指定した検索キーワードを基にして、ユーザの意図に合致する操作コマンドをシステムが生成して実行できるようにすれば良いでしょう。

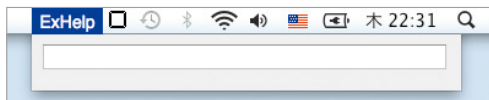
「米国時間」でヘルプ検索に成功してユーザの意図どおりの結果が出た場合、ユーザの意図は正しく検索システムに伝わっていることになりますから、そのままシステムに実行を指示できます。タイムゾーンを設定するtimezoneというコマンドが存在したとすると、「米国」のようなパラメータを使って、「timezone 米国」を起動できます。



Mac版ExpandHelp



このような考えに基づいて、Mac版のExpandHelpを試作してみました。MacのExpandHelpはメニューバーに常駐するアプリケーションで、キーワードを入力するとそれに関連したヘルプが表示されます(図3)。たとえば「3:45」と入力すると、3:45という時刻に関連するコマンドが

▼ 3 Mac版ExpandHelpの起動

メニューに表示されます(図4)。

ここで「時刻を3:45にセットする」を選ぶと **date** コマンドが実行されて、システムの時刻が3:45にセットされます。

ExpandHelp のヘルプデータは図5のような配列として表現されています。

「**〔時計|時間|時刻〕**を(0|1|2|3|4|5|6|7|8|9|10|11|12):(0|1|2|3|4|5)(0|1|2|3|4|5|6|7|8|9)(AM|PM)に**〔セットする|設定する|あわせる〕**」を展開したパターンのうち3:45にマッチするものが表示され、「**時刻を3:45にセットする**」を選択するとsetdate(3,4,5)が実行されます。



GitHelp



Mac版のExpandHelpは毎日使うようなものではないにもかかわらずヘルプデータの用意は面倒なので、結局あまり使わなくなってしまいました。万人向けの一般的なヘルプデータを用意するのはたいへんですが、自分が毎日のように使う複雑なシステムの利用を助けてくれるシステムならばもっとうれしいだろうと考え、Gitのヘルプシステム「GitHelp」を作ってみました。

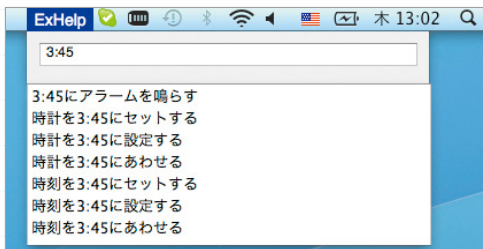
▼ 図5 ExpandHelpのヘルプデータの構造

```

['@weather')の天気を読む',
  'open http://tenki.jp/forecast/point-#{S1}.html "',
  ['@restaurant']のレストランを読む',
  'open http://#{S1} "',
  ['(時計1時刻)を(011213141516171819101112)時に(セッターを設定する(あわせる))',
    '# date {S12:00}',
    '/@01/ ', '# 数字を入力したときだけ利用',
    '(011213141516171819101112)時に(時計1時刻)時に(セッターを設定する(あわせる))',
    '# date {S12:00}',
    '/@01/ ',
  ['(時計1時刻)を(011213141516171819101112): (011213141516171819)(AM/PM)に(セッターを設定する(あわせる))',
    'setdate {S12:#{S3},{S4}}',
    '/@01/ @-310-01/ ',
  ['(時計1時刻)を(011213141516171819):45Cに(セッターを設定する(あわせる))',
    'setdate {3,4,5}',
    '/@01/ @-310-01/ ',
    '# delete mail from #{S13}',
  ['@app]というクロスアプリケーションプログラムを(消す/止める/停止する/殺す/落とす)',
    'kill -s {S13} "',
  ['(完ッ)している(Java系/スクリプト系)プログラムを(インストール)',
    'ps -ef ',
    '#現在のアレクサリのファイルをリストする',
    'terminal "ls -la" ',
  ['@station]駅名(#{station})駅までの電車の(時刻/時間)を読む',
  'open http://www.juden.jp/area/line/na/gi/nori.cgi?r=top&ok=ok&2-agg=&ok1=1#{S1}&ok2=1#{S2}&Cnoy=085-検査&cg=1"',
  ['@twittercard]のtwitterを読む',
  'open http://wtiller.com/#{S1}',
  ['FireFoxブラウザを起動する(動かす/使う/走らせる)',
    'open -a firefox "',
  ['@ls])というファイルをFireFox(ブラウザ)で開く',
    'open {S13} -a firefox "',
  ['(最新の(最も古い)ばんだ新しい)#{@apps}ファイルを開く(見る)',
    'file -mfind -o -#{S1} -f 1970 -h head -11; 'open \\#{file} "',
  ['大きな#{@apps}ファイルをリストする',
    'terminal mfind -o -#{S1} -b 10 1 head -11' ]

```

▼ **図4** 関連するコマンドが表示される



バージョン管理システム「Git」は、最近プログラマの間で広く使われていますが、基本概念もコマンド構成も複雑なのが難点です。Gitには100個以上のサブコマンドが存在するうえに各サブコマンドの仕様も複雑です。バージョン履歴のログを表示する「`git log`」コマンドのマニュアルページは2,000行近くもあり、これだけでも機能を理解するのは困難になっています。

Gitのコマンドに詳しいユーザでも、ちょっと標準と異なることをやりたい場合には、Gitの機能をどう組み合わせれば良いのか悩むことがあります。ユーザのさまざまな要求に対して良い解決手順を記述しておき、ExpandHelpの要領で検索できるようにしておけば便利です。GitHelpはGitに関するExpandHelpをコマンドラインから使えるようにしたシステムです。

たとえば「昨日のREADME.mdを見たい」と思っても、標準的なGitコマンドや引数でこれを実行することは簡単ではありません。git

log コマンドでREADME.mdの編集履歴を参照し、昨日のコミットIDを見つけて「git cat-file -p (コミットID):README.md」のようなコマンドを起動すれば良いのですが、これはなかなか面倒な作業で少し操作を覚えるのはたいへんです。実はコミットIDを取得しなくても「git cat-file -p "@{yesterday}":README.md」というコマンドで実行できるのですが、こういうマイナーな機

能があることを知っているユーザは多くないでしょう。

GitHelpでは、「昨日の(#{pat})ファイルを表示する」という説明パターンと「git cat-file -p "@{yesterday}":#{ \$1 }」というコマンドパターンのペアを用意しておくことにより、「昨日」「README」のようなキーワードを指定するだけで必要なコマンドを検索してリストできます。

```
$ githelp 昨日 AD
[0] 昨日の[README.md]ファイルを表示する
% git cat-file -p "@{yesterday}":README.md
[1] 昨日の[data/adddelete.rb]ファイルを表示する
% git cat-file -p "@{yesterday}":data/
adddelete.rb
[2] 昨日の[data/afteradd.rb]ファイルを表示する
% git cat-file -p "@{yesterday}":data/
afteradd.rb
$
```

ここではコマンドの候補がリストされているだけですが、-x オプションを指定すると候補を選択して実行させることができます。

```
$ githelp 昨日 AD -x0
# GitHelp

**Gitのコマンドの使いこなしを支援**

### 解決したい問題
...
```

GitHelpのヘルプデータはリスト1のような配列になっています。各配列要素の1行目がヘルプ文字列のパターンになっており、2行目が

▼ リスト1 GitHelpのヘルプデータ

```
[
  [
    "昨日の[#{files.join('|')}]]ファイルを表示する|見る",
    'git cat-file -p "@{yesterday}":#{ $1 }'
  ],
  [
    "[#{files.join('|')}]]ファイルを(#{numbers.join('|')}))
    日前の(もの|バージョン)と比較する",
    'git diff HEAD "@{#{ $2 } days ago}" #{ $1 }'
  ],
  [
    "[#{files.join('|')}]]ファイルをひとつ前のコミットと比較
    する",
    'git diff HEAD^ #{ $1 }'
  ],
  [
    "(#{numbers.join('|')}))日前から(変化した|変わった|変更され
    た)ファイルをリストする",
    'githelp-changed ¥#{ $1 } days ago¥'
  ],
]
```

それを実行するためのコマンド文字列になっています。

README.md、Rakefile、...のようなファイルがGitに管理されている環境において、ユーザが「githelp 昨日 AD」を実行すると、files()の結果が['README.md', 'Rakefile', ...]のような値になるので、最初のパターンは「昨日の[README.md|Rakefile|...]」ファイルを表示する|見る)のようになります。この正規表現を展開した結果得られる文字列の中の「昨日のREADME.mdファイルを表示する」という文字列がユーザの与えたキーワード「昨日」「AD」とマッチするので、それに対応するコマンド「git cat-file -p "@{yesterday}":README」を実行できるようになります。

このようにGitHelpでは、やりたいことの一部を示すだけでヘルプ情報を検索し、実行まで行うことができます。



ExpandHelpの実装



今回はRubyで正規表現展開ライブラリre_expandをまず実装し、それを使ってExpandHelpを実装しています。

ExpandHelpはアイデア自体は単純なのですが、正規表現の展開が必要なのでre_expand

も含めると、残念ながら実装はちょっと複雑になってしまいました。しかし、ヘルプなどの検索を行うときにGenerate and Testのような方法が有効だということに気づいたのはコープスの卵的だと思っています。GitHelpは実際毎日のGit活用には有用なので、ヘルプデータをもっと整備して毎日使っていこうと考えています。

GitHelpは<https://github.com/masui/GitHelp>で公開しており、「gem install githelp」で簡単にインストールできるので、ぜひご利用ください。SD

コロンブス日和

第10回 DragZoom

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学

大規模データの検索と閲覧

世の中にはさまざまな大規模データが存在しますが、きちんとした管理者が存在するデータは階層的な構造で管理されているのが普通です。地名や電話番号やドメイン名などは厳格に階層的に管理されていますし、図書館の蔵書は分類番号を使って階層的に管理されています。個人的なデータであっても、現在のPCのファイルシステムでは、フォルダを使って階層的に管理しなければなりません。

一方、世界最大のデータベースであるWebは階層的に管理されていませんし、手持ちの書籍のような小規模データは手間をかけて階層的に管理するほどでもないの、適当に管理している人がほとんどだと思います。しかし、管理された大規模階層データは数も種類も圧倒的に多いので、現在も将来もこれらを簡単に検索したり閲覧したりする方法が重要であることは間違いありません。

明示的な階層が存在しないリストのようなものでも、あたかも階層が存在するように扱うことができる場合があります。たとえば辞書の場合、「aで始まる単語」「bで始まる単語」のように分類を行い、その下に「aaで始まる単語」「abで始まる単語」のような階層を考えれば階層データと同じように扱うことができます。つまり、ソートが可能ならあらゆるデータは階層的なデータとして扱うことができると言えるでしょう。

階層データの閲覧

Webのようにリンクで自由に結合された構造のデータと比べると、階層的に管理された情報は対話的に閲覧／検索するのが比較的簡単です。図書館で分類に基づいて本を搜したり辞書で単語を搜したりといった作業は日常的なものですし、計算機で階層メニューやフォルダ階層などを操作することは一般的になっています。

階層的な構造を持つ大規模データを閲覧するためにさまざまなインターフェース手法が利用されています。小さな計算機画面に大量データを全部表示することは不可能ですから、なんらかの方法でデータの一部だけを表示する工夫が必要です。対話的に表示部分を変更しながら大規模データを閲覧するために次のような方法が広く使われています。

- ・少しずつ順番に見る(e.g. スクロール、辞書や書籍をバラバラめくる)
- ・キーワードなどでフィルタリングして表示量を減らす
- ・階層的に選択を繰り返してだんだん細部を表示する(e.g. ファインダー、階層メニュー)
- ・注目点を部分的に拡大する
- ・なめらかに直線的にズームする(e.g. GoogleMaps)

フィルタリングにより絞り込んだデータをスクロールして閲覧するなど、これらを組み合わせた手法もよく用いられています。

注1) <http://thinkit.co.jp/free/article/0709/19/>



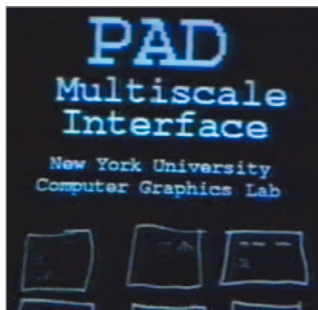
ZUI



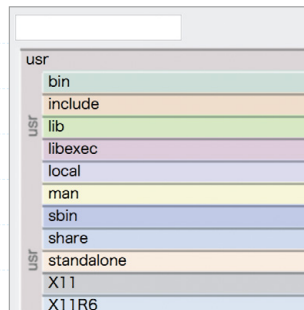
階層メニューを使ったりフォルダ選択を繰り返したりして大規模データを絞りこむ方法は、データの一部の選択を繰り返すことによる非線型なズームの一種といえます。これに対し、CGでよく使われる「Perlin Noise」で有名なKen Perlinは、2次元画面上に表現したデータ全体をなめらかに直線的に拡大したり縮小したりすることによって大規模データの閲覧をできるようにする「Pad」というZUI(Zooming User Interface)システムを1993年に提案し、研究者の間でかなり話題になりました(図1)。Padのような単純な方法を使えば拡大／縮小操作と移動操作だけであらゆる大規模な階層データを楽に閲覧できるわけですから、CLI、GUIのつぎに来るのがZUIだと期待され、さまざまなシステムや製品プロトタイプが作成されました。

ところが結果的にPadやその後継システムは世の中で流行ることがなく、ZUIの試みはほとんど忘れ去られてしまいました。流行らなかった理由はいろいろあるのですが、2次元画面を自由に拡大縮小して目的の情報を得ることは普通のユーザには難し過ぎたということが大きな理由の1つでしょう。また当時はマウスホイールはまったく普及しておらず、ズームのための標準的な操作が存在しなかったことも関係しているかもしれません。

▼ 図1 ZUI(Zooming User Interface)システム



▼ 図2 DragZoom(灰色部分にはエントリが隠れている)



現在はGoogle Mapsなどでズーム操作はお馴染みになっています。地図の場合は操作対象が具体的で誰にでも比較的わかりやすいためズーム操作が問題なく利用されているようですが、地図以外への応用は進んでいませんし、スクロール操作とズーム操作が衝突したり操作を間違えることはよくあります。2次元画面のズームを利用したインターフェースが今後流行する可能性は低そうです。



DragZoom



Padのような純粋なZUIは残念ながら普及に成功しませんでした。スマホのような小さな画面で大規模データを閲覧／検索するためにズームとフィルタリングが有効なことは間違いありませんから、これらを組み合わせれば誰でも使える簡単なインターフェースを工夫すればズームインターフェースが日の目を見ることがあると思います。

私はスクロールバーを拡張してズームやフィルタリングを可能にした「LensBar」というシステムを長年提案しているのですが、これを簡単にした「DragZoom」というシステムを紹介します。



階層データのズーム

図2は、フィルタリングとズーム操作によってファイルを検索しようとしているところです。最初はUNIXのファイル構造のうちrootに近い

部分だけが表示されています。灰色の横線は、エントリが隠れていることを示しています。

libexecをクリックして右にドラッグするか、スマホなどの場合は指でタッチしてから右にドラッグすると図3のようにリストがズームされます。

さらに右にドラッグを行うと、図4のようにすべてのファイルが

見えるようになります。

階層構造の要素を選択して拡大してブラウジングする方法はTreeViewや階層メニューなどでもお馴染みですが、DragZoomでは左右ドラッグでなめらかにズームインのレベルを調整していることになります。

複数の指を利用できるスマホやタブレットではピンチ操作でもズームレベルを変えることができます。



ドラッグによるスクロール

上下にマウスドラッグを行うと、スマホのスクロール操作と同じように画面をスクロールできます(図5)。

スクロール操作とズーム操作は完全に可逆的なので、スクロールしたりズームしたりしたあとでカーソルや指をもとの位置まで戻すと画面は最初の状態に戻ります。



フィルタリングとズームの組み合わせ

テキスト入力枠に「ruby」と入力すると、図6のように「ruby」を含むファイルやディレクトリだけが表示されます。ファイル名の左側にはフォルダ名も縦に表示されているので、どのフォルダのファイル名がマッチしたのかがすぐわかります。

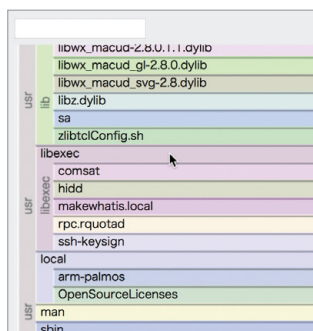
この状態では「ruby」を含まないファイル名はまったく表示されていませんが、「/usr/bin/ruby」の部分ズームすると図7のように「usr」「include」なども見えるようになります。DragZoomでは、リストの各エントリに「重要度」を設定しており、マウスや指を移動したときは^{いち}閾値を変化させて閾値より大きな重要度を持つ行を表示するようになっています。「usr」や「include」のように階層のrootに近いファイルには大きな重要度を与えているため、キーワードにマッチされていなくても表示されるというわけです。



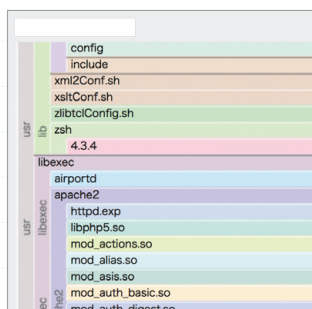
階層がない場合

辞書データのように階層構造が存在しないデータでもズーム検索できるようにするため、DragZoomでは仮想的な階層を利用して、たとえばa,b,c,d,e,f,gというフラットなデータがあるときは、それぞれに対して1,2,1,3,1,2,1のような重み(重要度)を与えておきます、ユーザが操作する閾値を超えたも

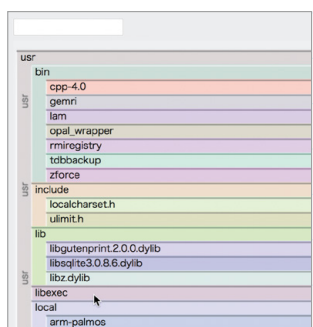
▼ 図3 リストがズームされる



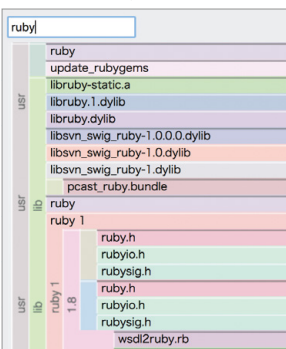
▼ 図4 すべてのファイルが見られるようになる



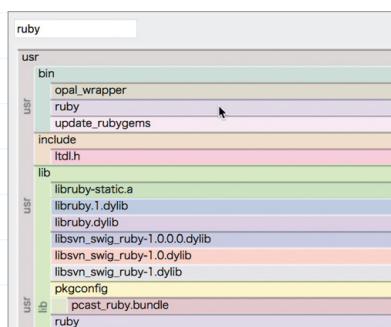
▼ 図5 マウスドラッグでスクロール表示できる



▼ 図6 「ruby」を検索



▼ 図7 ズームしながらファイルを探す



▼ 図8 フラットな英語辞書をブラウズしてみる



▼ 図9 ドラッグ操作



のだけ表示されますから、ユーザのズーミング操作で閾値を変えることによってd,b/d/f,a/b/c/d/e/fのように表示が変化することになります。

DragZoomでフラットな英語辞書をブラウズすると初期画面は図8のようになります。

ここでドラッグ操作でズームレベルを変化させると表示は図9のようになります。

さらにズームすると単語の意味が表示されます(図10)。

キーワードマッチングにあいまい検索機能を利用すると、多少間違ったキーワードを指定しても最も近いエントリが表示されます(図11)。



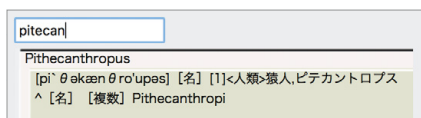
音楽ファイルの検索

図12は私の手持ちの音楽ファイルで「day」を

▼ 図10 単語の意味が表示される



▼ 図11 あいまい検索



検索してみたところです。このように、ファイルでも辞書でも音楽ファイルでもDragZoomで簡単に検索できることがわかります。



ズーミングシステムの課題



スクロールバーを使わずズーミングとフィルタリングだけで簡単に大規模データの閲覧や検索ができることはたいへんメリットがあるはずです。多くの大規模データは階層的に構造化されていますからDragZoomは広範囲なデータの検索に利用できるのですが、残念ながら私はまだこの手法を普及させることに成功していません。

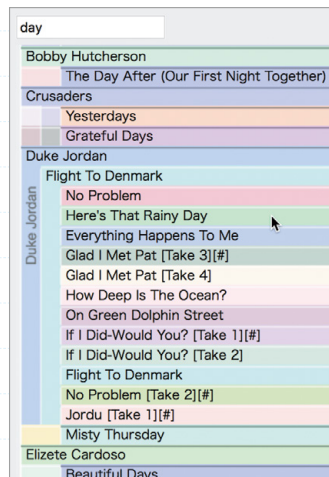
有用性は十分なのに普及がうまくいかないのは、左右ドラッグでズーミングを行うというインターフェースに慣れるのが難しいことと、従来の方法でもとくにひどく困ることがないという理由によると思われます。DragZoomを使わなくても普通のテキスト検索や階層メニューなどで階層データを取りあえず眺めることができるのであれば、わざわざ新しい手法に乗り換えようとする人は少ないでしょう。

しかしDragZoomの方法は慣れればかなり便利なのはたしかです。いろいろなサービスで地味に使えるようにしたうえで、今後これをはじ

めとするさまざまなズーミングインターフェースが普及してほしいのだと願っています。

DragZoomのデモおよびソースコードは、<http://DragZoom.com/> および<http://GitHub.com/masui/DragZoom>で公開しているのでご利用ください。SD

▼ 図12 「day」で検索



コロナブス日和

第11回 Gear

エンジニアというものは「楽をするためならどんな苦労も厭わない^{いそ}」ものだと言われていますが、コロナブスの卵のようなゴキゲンな発明によって頑張るって楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学

2つのボタンだけで 操作できる Gear

今回は、大規模な階層情報をズーム検索する DragZoom というシステムを紹介しました。DragZoom はパソコンやタブレット上のマウスや指の操作でズームを行うことによって大規模データの検索を行うシステムですが、ユビキタスコンピューティング環境ではいつでもポインティングデバイスが使えるとは限りません。料理しているとき、歩いているとき、通勤電車で立っているとき、車を運転しているときなど、ポインティングデバイスを使いづらい状況はよくありますが、そういった状況でも大きな階層情報をうまく検索できる方法があれば自由に音楽や番組などを選択できるので便利です。今回は、2つのボタンを使うだけで大規模な階層データを簡単に検索できる「Gear」というシステムを紹介します。

階層データの段階的 なナビゲーション

DragZoom のように GUI を使って階層データをなめらかに検索する方法はこれまでたくさん研究されてきているのですが、残念ながらデスクトップやブラウザ上で広く利用されているものはほとんど存在せず、階層構造を段階的にたどりながらブラウズするやり方が一般的になっています。たとえば Mac のデスクトップ画面で

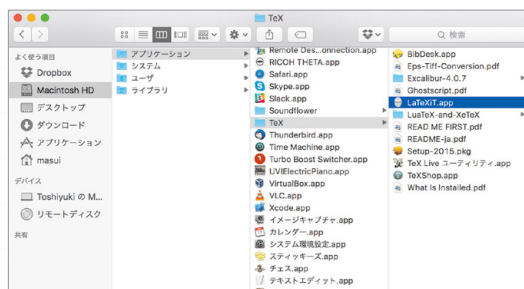
ファイル进行操作するプログラム(ファインダ)では、ファイルの階層構造を視覚化してナビゲーションを可能にする方法がいくつか用意されていますが、どの方法でもマウスやキーを操作して階層構造の親子関係や兄弟関係を段階的にたどっていくことにより階層型ファイルシステムのナビゲーションを行うようになっています(図1)。

たとえば1つ下の階層のフォルダの内容を見たい場合、マウスでダブルクリックしてフォルダを開いたり、矢印キーを使って注目行を移動させてから **[Enter]** キーを押すといった方法がよく使われています。

現在のパソコンのファイルはすべて階層的に管理されており、ほとんどのファイルにデスクトップからアクセスできるようになっています。パソコンの中の100万個のファイルの中から、ファイルを1つ選ぶのはたいへんはなはずですが、実際のパソコン操作においてそれほど苦労している人は多くないでしょう。

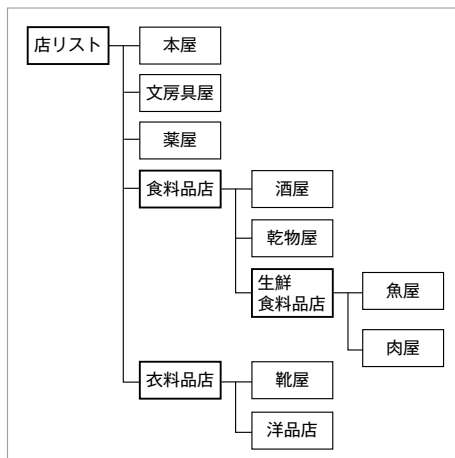
データがきれいに階層的に管理されている場合、それほどたくさんの操作をしなくても必要

▼ 図1 Macのファインダでアプリケーションを段階的に検索しているところ



注1) <http://thinkit.co.jp/free/article/0709/19/>

▼ 図2 あるファイルシステムの階層例



な情報を見つけることができるのが普通です。実際、パソコンのデスクトップ画面でマウスを使って階層のトップから階層をたどって特定のファイルを探すのに必要な操作は多くても10回程度でしょう。

キーボードで階層構造をナビゲーションする

図2のような階層を持つファイルシステムをキーボードでナビゲーションすることを考えてみましょう。

これがファイルシステムの構造になっている場合、Macのファインダでは▲▼◀▶という4個の矢印キーでナビゲーションを行うことができます。

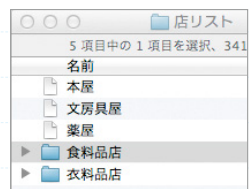
「店リスト」をファインダで表示して、「本屋」を選択すると、表示は図3のようになり、3回▼を押すと、図4のように「食料品店」が選択されます。

「食料品店」は下位階層を持っているので、ここで▶キーを押すと図5のように下位階層が表示されます。さらに▼キーを押すことによって「酒屋」を選択したり、「生鮮食料品店」を選択してから▶を押すことによって、図6のように下位階層を表示できます。

▼ 図3 「店リスト」をファインダで表示して「本屋」を選択



▼ 図4 3回▼を押して「食料品店」を表示



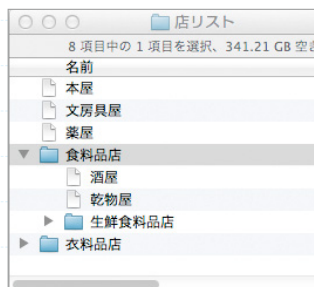
また、図6の状態でも◀キーを押すと下位層の表示を消し、図5のような状態に戻すことができます。このように、Macのファインダでは▲▼◀▶という4個のキーを使って階層データのナビゲーションを行うことができます。テレビのリモコンやジョグダイヤルでも階層構造データのナビゲーションのためにほぼ同様の手法が利用されていることが多いようです。

Gear——2つのキーだけで階層構造をナビゲーション

前述のように4個のキーを使って階層データのナビゲーションを行うシステムはたくさんありますが、実はキーを2つしか使わなくても同様のナビゲーションを行うことができます。

Gearでは▲と▼という2つのキーだけを利用してナビゲーションを行います。Gearで「店リスト」を表示すると、ファインダの場合と同じリストが表示されます(図3)。▼を3回押すと前の例と同じように「食料品店」が選択されます(図7)、そこで操作を中断して一定時間待つと「食料品店」の下位層が自動的に展開され、

▼ 図5 「食料品店」で▶キーを押して下位の階層を表示



▼ 図6 「生鮮食料品店」で▶キーを押して下位の階層を表示

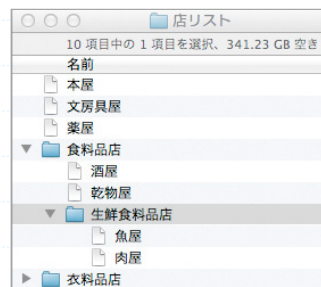


図8のようにその最初の要素が選択されます。

ここで▼を2回押して「生鮮食料品店」を選択したまま一定時間待つと、図9のように下位層が自動的に展開され、最初の要素である「魚屋」が選択されます。

つまり、▶のようなキーを押さなくても、一定時間待つことによって同様の効果が得られることになります。

図7のように食料品店を選択した状態から時間を置かずに▼を押すと、下位層は展開されず、図10のように「衣料品店」が選択されます。またここで操作を止めて一定時間待つと下位層が自動的に展開され、図11のように「靴屋」が選択されます。

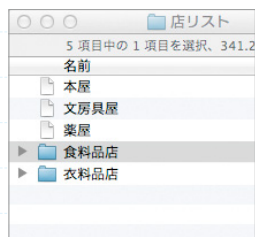
図9の状態から▲を押すと、下位層は自動的に閉じられて図8の状態に戻ります。さらに▲を押すと「食料品店」の下位の層も閉じられ、図7の状態に戻ります。また、図9の魚屋が選択されている状態から▼を2回押すと、「食料品店」の下位層は自動的に閉じられて図10の状態になります。

まとめると、

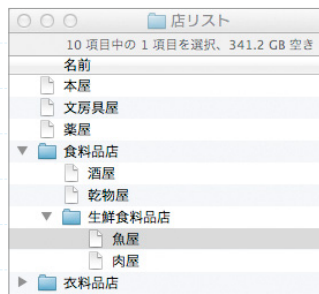
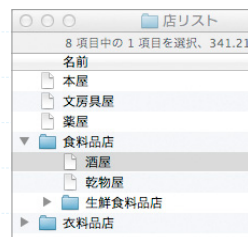
- ・選択した項目に下位層が存在するときキー入力を行わずに待つと下位層が自動的に展開され、下位層の最初の項目が選択される
- ・項目リストの端を選択しているとき、さらに▲▼を押すと下位層は閉じられて1つ上の層の項目が選択される

という2つの工夫により、▲と▼だけで階層デー

▼図7 「店リスト」から「食料品店」を選択する



▼図8 一定時間で下位階層が表示される



◀図9 一定時間で下位階層である「魚屋」が表示される

タを自由にナビゲーションすることが可能になるというわけです。



ブラウザでの実装



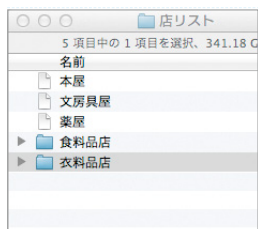
前述の例は小さな階層的データの上でのGearナビゲーションでしたが、実際には巨大な階層データでも同様のナビゲーションができます。図12のスクリーンショットはブラウザ上に実装したGearを使って各種コンテンツをナビゲーションしているところです。



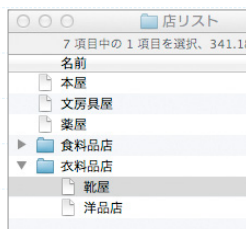
Gearの利点

入力装置は単純であるにこしたことはありません

▼図10 すぐに▼を押すと「衣料品店」が表示される



▼図11 「衣料品店」で少し待つと「靴屋」が表示される



▼図12 WebブラウザへのGear実装



せん。2個のセンサであらゆる操作が可能なのであれば左右の回転操作や何かを押したり引いたりする単純な装置でも操作が可能だということになるので、さまざまな環境で利用が可能だということになります。Gearの操作は▲▼という2つのキーしか必要としないため、圧力センサや回転ダイヤルなどを利用した各種の実装ができます。

たとえばMacの入力装置として写真1の「Hyper Mate」という製品が市販されており、これをGearの入力装置として利用できます。

また写真2は、パドルに貼った2個の圧力センサの値を▲▼に割り当てています。板を押したり引いたりすることによってコンテンツのナビゲーションができます。

このように、現状のGUIでは利用されていないような単純なデバイスでもGearの入力装置として利用できるのが大きなメリットです。



Gearの利用環境



Gearの操作には2個の入力装置しか必要としませんから、普通の入力装置が使いにくいような場所でも利用できます。たとえば食卓の裏にさりげなくスイッチや圧力センサを2つ並べておけば、それらを操作するだけでBGMや番組を切り替えることができますでしょう。テレビの前のソファの手すりにローラーを付けておけば、それを回すだけであらゆるコンテンツを選

択できるでしょう。

私は自宅の机の前のサブモニタで常にGearブラウザを動かしており、マウスホイールを使ってコンテンツを選択して視聴しています。パソコンで音楽や動画を再生しながら仕事をしている人は多いと思いますが、手近にGear端末があればマウスホイールを回すだけであらゆるコンテンツを簡単に視聴できるのはとても快適です。

たくさんのボタンを必要とする入力装置ではどうしてもボタンに合わせた設計が必要になり、装置の形状が制約を受けてしまいますが、Gearの場合にはそもそもどこでコンテンツを視聴したいのかを最初に考え、その場所に最も適した入力装置の形態を考え、それらをGearのコントローラに利用するという順番でデザインができることになります。これまでの電子機器は機械の都合にあわせて装置の形状がデザインされるのが普通でしたが、それを根本的に改善できる可能性があると言えるでしょう。

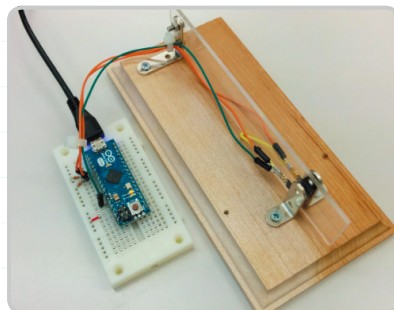
Gearを見たとき「これの何が面白いの?」という顔をする人がいます。確かに、一度Gearを使ってみるとGearの挙動はまったく自然に感じられるので、どこが新しいのかわからなくなってしまうのかもしれませんが、これまでボタン2個だけで階層的データをナビゲーションする方法が普及していないことを考えると、Gearはまさに「コロンブスの卵」のような手法だと言えるでしょう。

Gearは確かにコロンブスの要素を持つインタラクション手法であり、どこでも簡単に情報検索を行うための強い味方となります。Gearのためのオシャレな入力デバイスを考えていきたいと思っています。SD

▼写真1 HyperMate



▼写真2 圧力センサを利用した入力装置



コロンブス日和

第12回 SmoothSnap

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



手際よく値を設定したい

パソコンやスマホのアプリ上で各種の値を設定することがありますが、スライダで細かい値を正確に設定することは困難です。Mac OS Xの色設定画面のように0~255の数値を設定するだけの場合でもマウスで正確に値を設定するのは難しいので、図1のように数字を入力するテキストボックスが別に用意されています。

秒単位で時刻を設定したいような場合は事態はさらに絶望的です。設定可能な値は $24 \times 60 \times 60 = 86,400$ 通りあるにもかかわらず、スライダのドット数は数百程度しかありませんから、1ドットスライダを動かすだけで値が何百も飛んでしまうことになります。

このため、細かく時刻を指定したい場合は時間/分/秒を別々に指定するのが普通です。

値を細かく設定したい場合がある一方、大まかな値をすぐに設定したい場合もあります。たとえば時刻を10時ちょうどに設定したい場合、前述のようなシステムでは時間を10に/分を0に/秒を0に設定する必要がありますが、このようなキリの良い時刻を指定したい場合でも分や秒まで細かく指定するのは面倒です。キリの良い値はもっと簡単に設定できるようになってほしいものです。

注1) <http://thinkit.co.jp/free/article/0709/19/>

このように、一般的なスライダやスクロールバーでは細かく値を調整したり大まかな値を設定したりすることが簡単ではないのですが、スライダやスクロールバーの挙動を工夫すればこういった問題を解決できる可能性があります。



微調整のインターフェース

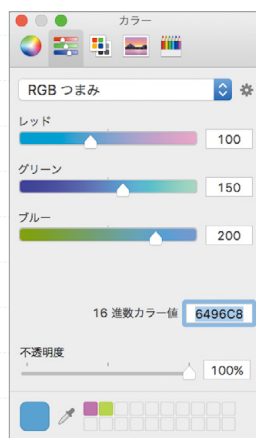
GUIで値を微調整するためのさまざまなインターフェースが昔から研究されています。

普通のスライダで秒単位で時刻を設定することが不可能なのであれば、微調整専用のインターフェースを追加すればよいでしょう。スライダやスクロールバーにボタンを追加し、ボタンを押したら最小単位ずつ位置を移動させられるようにしたものがよく使われていました(図2)。

スライダのノブを拡張して、微調整を可能にしたシステムもあります。たとえば90年代にメリーランド大学で開発された

AlphaSliderというGUIツールでは、スライダのノブを上下に分割し、ノブ上でクリックした位置によって微調整/粗調整を選べるよう

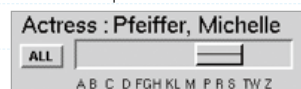
▼ 図1 数字を入力するテキストボックス



▼ 図2 ボタンで微調整可能なスライダ



▼ 図3 AlphaSlider



になっています(図3)。

▼ 図4 FineSlider

私が昔開発したFine Sliderというシステムでは(図4)、スライダのノブ以外のバックグラウンドをクリックするとゴム紐

のようなものが出現し、マウスでゴムを引っ張るような動きをすることによってノブ位置の微調整を可能にしていました。マウスカーソルがノブの近くにあるときはノブがゆっくり動き、遠くにあるときは速く動きます。

iPhoneの音楽再生アプリでは(図5)、再生時刻を指定するためのスライダをタップしてから指を下にずらしてスライドすると、ずらさないときよりも細かく時刻を調整できるようになっています。



場所をおおまかに指定する

キリの良い時刻を簡単に指定したり図形をきちんとそろえたりするためには、微調整とは異なる手法が必要です。

図形を動かして隣の図形にぴったりそろえるような操作をしたいときは「スナッピング」というテクニックがよく使われます。スナッピングとは、ドラッグ中の図形が別の図形やグリッドの近くに来たときそこにぴったりそろえるように移動するもので、ユーザが正確でない操作をしても意味のある場所に移動させることができるので便利です。

スナッピングは図形編集ではお馴染みのテクニックですが、キリの良い値にパラメータを設定するような場合でも利用できます。10:00かきりに時刻を合わせたい場合や、次の曲や次のチャプタに動画を移動させたいような場合にも使えます。音楽や動画を見る場合でも、次の曲やチャプタの1秒前から再生したいような場合、スナッピング機能で次の場所に移動してから微調整操作で1秒戻りたくなるでしょう。

つまり、スナッピングのようなおおまかな操作の恩恵を受けつつ微調整も可能であるという巧みなインターフェースが欲しいところです。

▼ 図5 iPhoneの場合



SmoothSnap



遠くに旅行する場合、目的地の近くの空港まで飛行機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ます。

このように、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さまざまな場所にうまく移動できるようになっているわけですが、スライダやスクロールバーにおいても同様の方針が使えればいいでしょう。

iPhoneの微調整機能もあまり直感的とはいえませんし、そういう機能があることに気付く表示もありませんから、この機能を知らない人も多いと思われます。もっと単純に、現在と同じ方法で自然にスライダやスクロールバーを操作するだけにもかかわらず、おおまかなスナッピング動作や微調整を可能にするためにSmoothSnapという手法を開発しました。

SmoothSnapでは次の方針で値や位置をコントロールします。

- ・ノブを大きく移動した場合は重要なポイントにスナッピングする
- ・ノブを少しだけ動かした場合は細かい粒度で連続的に値を変化させる



時刻を細かく設定する

前述のように、時／分／秒の設定は86,400通りの可能性がありますから、これを1つのスライダで設定することは困難です。このため、乗換案内のように時刻指定が必要なサービスでは、図6のようにメニューを利用して時刻指定を行うことが多いようです。

SmoothSnapに対応した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あと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り、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っているので、スライダのノブを動かすだけで秒単位で時刻を設定できます(図7)。

「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ます。

おおまかな粗い操作をするとキリ時刻を指定でき、細かい操作を繰り返すと詳細時刻を指定できることになるのは、微妙な操作をするほど細かい調整が可能なので、人間の直感に近く感じられます。



大きな文書のスクロール

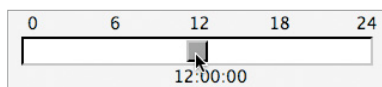
大きな文書が1ページのWebページになっているとき、その構造を把握しながらブラウジングを行うのはたいへんです。

図8は、私が昔書いた30個のWeb記事を並べて1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ます。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、スクロールを行わずにページ全体の構造を把握するのはかなり困難です。

右側の画面では章タイトル(「第28回……」)が見えているのでだいたいどのあたりをブラウズしているのかがわかりますが、左側の画面

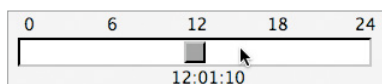
▼ 図7 ノブのスライドでさまざまな時刻を設定する



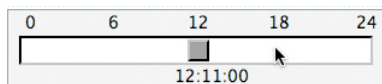
初期状態



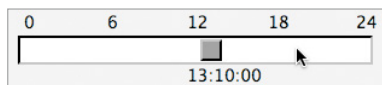
1秒単位の設定



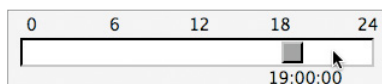
10秒単位の設定



1分単位の設定



10分単位の設定

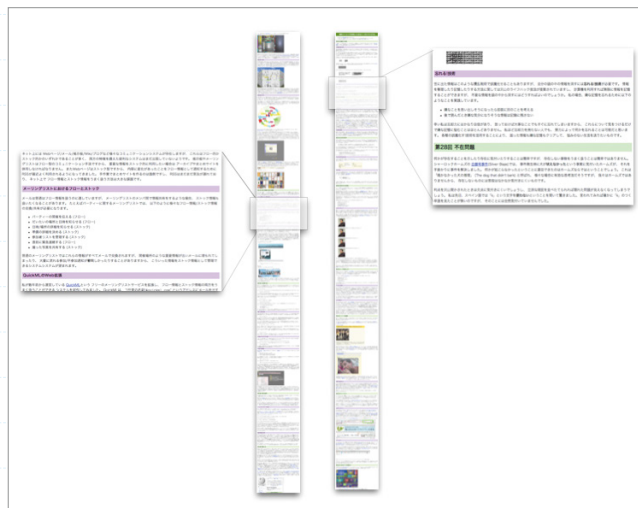


1時間単位の設定

▼ 図6 時刻設定の例



▼ 図8 Web記事のスクロール



▼ 図9 ページ先頭

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

- Wireddvisionの[排印の記事](#)を全部つなげたものです。
- スクロールバーに紐づいており、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。それ以外のブラウザの場合は [こちら](#) をお読み下さい。

第29回 誤魔化す!技術

「脱色化」という言葉には悪いイメージがありますが、「情報隠蔽」というと多少聞かえが良くなるかもしれません。森の中に木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、これに内容を脱色化することにより秘密を守る技術の一例といえるでしょう。

公開情報公開法(技検)

※公開の秘密情報は、普通に増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逆ですから、なんらかの方法によってその情報を元に戻すことが出来ればよいですね。

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断が困難**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、内容が少しずり済メールを大量に送りつけば、どれが最初のメールなのかわからなくなってしまうかもしれません。また、パスワードを漏洩してしまった場合は、異なるパスワードを同じように漏洩させてしまえば、原則的な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。

▼ 図10 先頭から少し下にドラッグした状態

「木を隠す」という記事で、智恵の画家やアーティストの中に秘密情報を送るステガノグラフィーという技術を紹介しましたが、これは内容を隠蔽化する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。

正しい情報と似て異なる情報を大量に提供する偽少年メソッドを使えば、どれが正しい情報なのかわかりにくくなることがあります。たとえば、メールに問題があるメールを間違えて送ってしまった場合、内容が少しず違うメールを大量に送りつけば、どれが最初のメールなのかわからなくなってしまうかもしれません。また、パスワードを漏洩してしまった場合は、異なるパスワードを同じように漏洩させてしまえば、悪用する危険が大幅に高まるかもしれません。

文字を消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「潜井」のような文字を手取り早く消したい場合、鉛筆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断りになります。

増井 隆寛

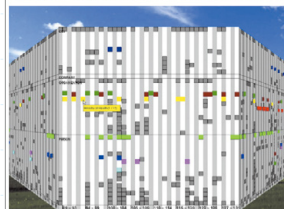
文具用品のプラスステーション株式会社は、 票書などに印刷された住所や名前のような個人情報を読み取するための ケシ
ボン というスタンプを販売しています。 ケシボンとは下のようなボタンをもつスタンプです。

BOOK VARIOUS VARIOUS VARIOUS VARIOUS
 MORE MORE MORE MORE MORE
 MORE VARIOUS VARIOUS VARIOUS VARIOUS
 MORE MORE MORE MORE MORE

▼ **図 11** もう少しドラッグした状態。<h3>にスナッピングしている

情報視覚化の歴史

情報視覚化の研究は1990年ごろからXerox PARCなどで盛んになり、PARCには様々な大量の情報も3次元空間にマッピングすることによって情報むけの2次元空間上に表示する手法が試みられてきた。3次元空間のデータを見る場合、遠くにあるものは小さく見えますが、データをうまく配置すれば自然に元の情報を見ることができます。たとえば新しい情報は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。下図の「Perspective View」システムでは、元のファイルも3次元空間上の「樹」に結び付けられることによって、注目したい範囲の情報が大きく見えてくるようになります。



1990年当時は高価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なえる場所は限られていました。また、視覚化が必要なほど大規模な情報もあり利用できませんでしたから、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、最近はどうなパソコンでも高速に3Dグラフィックス画面を動かすことができますし、Web上の

▼図12 かなり下のほうまでドラッグした状態。
<h2>にスナッピングしている

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ぬとき、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中の先頭行に“*”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。マシナに“行頭にわたって行頭に文字列を追加する”のような機能があればいいのですが、基本的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ないたい場合は、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与える必要があります。

ユーザが「**魔作マクロ**」を定義できるエディタは多いですし、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。こうしたプログラミングは数値が高くないのもですから、たいしてはあきらめて**こつこつ**作業するのが普通だと思えます。しかし、**手帳**にちょっとした自動化処理を指示できると嬉しいから、誰でも簡単なプログラミングを行えるような**エンドユーザプログラミング**や**視覚プログラミング**という考え方が唱えられています。

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し、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、環境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です。テキストだけでなく図形をも活用した方向けやすさというのがいいので、**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、音楽家の間で**Max** 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し、**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

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れるようになってます。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、その人のアドレス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 [前ページ](#)のシステムで提案されているものですが、Automatorを使うとこのように非常に簡単にAutomatorに実装することができます。

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かわからず、上下にスクロールして初めて位置がわかることになります。

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9～図12に示します。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われますが、移動量が多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>が画面のトップに位置することになり、全体的にどのような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できます。

SmoothSnapの期待

AlphaSliderのような手法では、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

用していますが、SmoothSnap スライドもスクロールバーも外見は標準のものと変わりがなく、スナッピングと微調整の挙動が違うだけですから、特殊な前提知識なく利用できることが期待されます。

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のも利点でしょう。SmoothSnapは筆者のWebページ²で実際に使ってみることができます。汎用ライブラリとしてはまだ準備できていませんが、要望が多ければ開発したいと思っています。

スライダやスクロールバーが発明されてから何十年もたっていますが、画期的な改良は行われていないようです。しかしまだまだコロンブスの卵的な細かい改善は可能だと思いますので追及していきたいと考えています。SD

注2) <http://www.pitecan.com/SmoothSnap/>

コロンブス日和

第13回 廃れるページ

エンジニアというものは「楽をするためならどんな苦労も厭わない^{注1}」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張ってるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



Web ページの古さの視覚化



Webで面白い情報を見つけたとき、よく見たらとても古いページだということに気づいてびっくりすることがあります。びっくりするだけだとまだ良いのですが、SNSなどで紹介したあとで古さを指摘されたりすると恥ずかしいものです。

世の中のたいていのものは古くなるとゴミが溜まったり変色したりするものなので、古さになんとなく気づくのが普通ですが(写真1)、ファイルやWebページのようなデジタル情報は古くなくても見栄えが変わらないため、こういう失敗をしやすいと言えるでしょう。

情報が劣化しないことは、デジタルデータの大きな特長なのですが、古さが直感的にわかったほうが都合が良い場合もあります。古くても価値が変わらない情報なら気にする必要はないのですが、ニュースや技術情報のように新しさが重要な場合、間違って古い情報を参照しない

ように気を付けなければなりません。今回はWebページの古さを直感的に感じることができるようにするための工夫を紹介します。



廃れるリンク

はこだて未来大学の塚田浩二氏は、古いページへのリンクが汚く見える「廃れるリンク^{注2}」というシステムを開発しました。

図1のようなWebページは古い情報へのリンクを含んでいるのですが、これを見るだけではどのリンクがどれだけ古いのかはわかりません。廃れるリンクシステムを使うと、図2のように古いページへのリンク文字列が汚く表示されるため、リンク先のページが古いことが明らかにわかります。

廃れるリンクシステムはプロキシサーバとして実装されています。廃れるリンクのプロキシサーバ経由でWebページにアクセスした場合、アクセス先のページの古さを取得したあとで、古さに応じてリンクのCSSを調整することによって古さを表現しようとしています。

▼写真1 明らかに古い壁



▼図1 古い情報へのリンクを含むWebページ



▼図2 廃れるリンクを利用した結果



注1) <http://thinkit.co.jp/free/article/0709/19/>

注2) <http://mobiqutious.com/dying-link.html>



廃れるバックグラウンド

リンク先の古さを視覚化するだけでなく、ページそのものも古く見せると良いでしょう。自分のWebページの場合、作成時刻や編集時刻を利用して、きめ細かく古さを表示できます。

図3は私が昔作成した「廃れるバックグラウンド」というシステムです。古さに応じてランダムドットの数を増やしていますが、あまり美しいとは言えないようです。

図4はページ内の各部分のアクセス状況によってバックグラウンド表示を変えてみたものです。gimpで自動生成したシミのような画像と日付を表示することによって、古さを直感的に理解させようとしています。

廃れるバックグラウンドシステムは、バックグラウンド画像がイケてなかった点や、古いページが読みにくくなってしまった点であり評判がよくなかったため、真面目に運用せず放置状態になったまま現在に至っています。

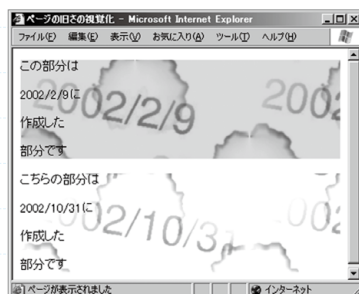


廃れるページ



「廃れるリンク」も「廃れるバックグラウンド」も10年以上前のシステムなのですが、手間の割に便利さが不足していたり見栄えが悪かったりしたため、日常的に使ってはいませんでした。一方、Web上の情報がますます増えてきた現在、古い情報を新しいものと勘違いする危険は増えていますし、古さの理解は以前よりも重要になっている気がするので、「廃れるバックグラウンド」と同様のシステムを次の方針で作りなおしてみることになりました。

▼ 図4 バックグラウンド画像を工夫してみたもの



- ・ブラウザ拡張機能として実装
- ・ページの古さを最初の数秒だけ全面に表示

任意のWebページに対して古さを視覚化するためには、どこかになんらかの細工が必要になるわけですが、最近のChromeやFirefoxはブラウザ拡張機能を簡単に作れるようになってきたので、最近はこの利用するのが良い気がしています(図5)。

従来はブラウザ拡張機能を作るのはかなりたいへんでしたが、最近FirefoxでもChromeでも同じ形式で開発ができますし、開発の手間もかなり小さくなっています。



ブラウザの挙動を制御する方法

✓ ブラウザ拡張機能

ブラウザの挙動を変えたり機能を追加したりするために、さまざまな「ブラウザ拡張機能」が利用されています。ブラウザ拡張機能とはブラウザアプリケーションの見栄えや動きを変更したり拡張するためのもので、ブラウザの挙動を制御するためのたくさんのAPIを利用して開発されるものです。たとえばブラウザのタブ

▼ 図3 古くなったWikiページ



▼ 図5 廃れるページ



の挙動を変更したい場合は、タブの挙動を変更するAPIを利用することになります。

便利なブラウザ拡張機能を使っている人は多いでしょうが、ブラウザ拡張機能を自分で作って活用している人は少ないと思います。

ブラウザ拡張機能を作る方法はブラウザごとに異なっていましたし、APIが膨大で理解するには努力が必要ですし、作成方法がしょっちゅう変わるので対応がたいへんだったりしたからです。

たとえばFirefoxの場合、

- ・ Chrome XUL を利用して開発
- ・ Python ベースの cfx コマンドで add-on SDK を利用して開発
- ・ Node ベースの jpm コマンドで add-on SDK を利用して開発
- ・ Web-ext コマンドで開発

のように開発方法が追加されたり変化したりしており、最新の開発手法について行くのは苦労したものです。

✓ ブックマークレットと Greasemonkey

ブラウザの挙動に変更を加えるのではなく、Web ページの内容を利用したり修正したりするためには「ブックマークレット」や「Greasemonkey」などのシステムが利用されてきました。ブックマークレットとはブラウザのブックマークとして登録した JavaScript プログラムで、普通のブックマークを呼び出すのと同じ方法で呼び出して実行するものです。

ブックマークレットはブックマークメニューなどから手動で起動する必要がありますが、Web ページに対して自動的になんらかの処理を行いたい場合は Greasemonkey というシステムが従来よく利用されていました。

Greasemonkey は Firefox の拡張機能で、ユーザが Greasemonkey に登録した JavaScript コードを自動的に呼び出して利用できるというもの

でした。

私の場合、拡張機能の API を勉強するのは面倒だけれど、自動的に Web ページを操作したいことは多かったので、ブックマークレットや Greasemonkey をよく利用していました。

たとえば、パスワードを要求する Web サービスに対して、本誌 2016 年 6 月号で紹介した EpisoPass のような「なぞなぞ認証」でログインを可能にするような Greasemonkey スクリプトを作って使っていましたし、面白いページを Gyump (2015 年 12 月号) に登録するためのブックマークレットは頻繁に使っています。

このように、これまでは、

- ① ブラウザの挙動を変えるためにブラウザ拡張機能を利用
- ② Web ページを手動で操作するためにブックマークレットを利用
- ③ Web ページの自動処理化のために Greasemonkey を利用

のような使い分けをするのが普通だった気がします。

現在見ている Web ページに対する処理を指定できるブックマークレットというのは便利なくみなのですが、そもそも Web ページの URL を記憶するためのブックマーク機能を Web ページの操作に利用するというのはわかりにくいですし、登録も起動も簡単ではないので世の中で広く使われるようにはなっていないようです。本来、別の Web ページにジャンプする機能と今見ている Web ページに手を加える機能はかなり性質が異なりますから、同列に扱うのは適切でない気がします。

一方、ブラウザ拡張機能というのは、ブラウザの見栄えや挙動を変えるというはっきりした意図が明らかですし、インストールや管理も簡単なので、そういう用途にはこちらの方が向いていると思われます。最近は開発事情が若干改善しており、それほど苦労しなくても Java

Scriptで拡張機能を作成できるようになってきたので②も③もブラウザ拡張機能を利用して問題ない状況になりつつあります。ブラウザ拡張機能の作成が簡単になってきた現在、もっとさまざまなブラウザ拡張機能が出現してきてほしいものだと思います。

現在、どこでもEpisoPassが使えるようにするための拡張機能を開発中なのですが、拡張機能をうまく使うとEpisoPassがかなり便利になりそうです(図6)。

こういう状況なので、廃れるページもブラウザ拡張機能として実装するのが良いと思われます。

✓ 廃れるページの実装

廃れるページの原理はとても簡単で、

- ①Webページの時刻を取得する
- ②古い場合は全画面を古い感じにして古さを表示する

というだけです。②については「廃れるバックグラウンド」で利用していたような画像を<div>タグで重ねるだけなので簡単ですが、時刻を取得するのは少し注意が必要です。

Webページの作成時刻はdocument.lastModifiedで取得できるはずなのですが、これがうまく取得できないページがあります。また、動的に生成されるページでは現在時刻が設定されてしまう場合があります。私はLivedoor Blog^{注3}で時々ブログを書いているのですが、このサービスではブログ記事ページの作成時刻は常に現在時刻になってしまい、記事内容が古くてもページは最新という状況になってしまいます。本当の古さを知るためにはきちんと内容を調べる必要がありますが、これは難しいので、ページ中に出現する日付をページの古さだと解釈することにしています。

✓ 利用結果

廃れるリンクも廃れるバックグラウンドも、

▼ 図6 Amazonのログイン画面でEpisoPassを利用する拡張機能



▼ 図7 「廃れるページ」の利用結果



いろいろ用意が必要な割に効果がパツとしないせいか、流行するということはありませんでしたし、私自身も利用していませんでした。しかしブラウザ拡張機能を使ったものは、インストールが簡単でし、ページの古さがすぐにわかるので、今のところ満足して利用しています(図7)。ページの古さの表示はウザいこともあるでしょうが、数秒待てば完全に消えてしましますし、古いページにアクセスする機会はそれほど多いわけではありませんから、なんとか便利さとウザさのトレードオフが我慢できるレベルになっている気がします。

廃れるページの拡張機能は、ChromeやFirefoxのストアに「sutare_extension」という名前で登録してあるので簡単にインストールできます。またソースをGitHub^{注4}で公開しているのでご利用ください。SD

注3) <http://masui.blog.jp/>

注4) <https://github.com/masui/SutareExtension>

コロンブス日和

第14回 HashInfo

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



HashInfo



次のような作業をやりたいと思ったことはないでしょうか？

- ・同じファイルがあちこちにないかチェックしたい
- ・重要なファイルだけバックアップしたい
- ・自分のアイデアを日付つきで記録したい

実はこのような仕事は、1つの単純な方法を使って実行できます。

パソコンやWebには膨大なファイルがありますが、同じファイルがあちこちに冗長に置かれていることも多いでしょう。

私の場合、同じファイルを何度もダウンロードしてしまったり写真や動画をいろんな場所にコピーしてしまったりすることがよくありますし、バックアップのつもりでコピーしたデータをさらにバックアップ対象にしてしまった結果、バックアップのバックアップ(のバックアップの……以下同様)のような無駄なデータを作ってしまうことすらあります。

こういう問題を防ぐためには、同じファイルがすでに別のところに存在するか調べれば良いわけですが、すべてのファイルの中身を比較することは現実的ではありません。しかし、ファイルの「ハッシュ値」を利用すれば、あるファイルが独自の(ユニークな)ものなのかどうかを比較的簡単に判別できます。



ハッシュ関数とハッシュ値

デジタルデータを一定サイズの数値に変換する関数をハッシュ関数と呼び、計算された値をそのデータのハッシュ値と呼びます。

最近のプログラミング言語では`a['abc']`のような連想配列が使えるのが普通ですが、これを通常の計算機上に実装する場合、“abc”のような文字列に対してハッシュ関数を計算し、その数値を添字として利用することによって通常の配列と同じように扱う手法がよく利用されています。

ハッシュ値のサイズが小さい場合は、異なるデータから同じハッシュ値が計算されてしまう(ハッシュ値が衝突する)ことがあります。ある程度大きなハッシュ値を生成する適切なハッシュ関数を用意すれば、ハッシュ値が衝突する可能性はほぼゼロにできます。また、ハッシュ値からもとのデータを計算することもほぼ不可能にできます。

ハッシュ関数はさまざまなものが考えられますが、現在はMD5やsha1というハッシュ関数が広く利用されています。このようなハッシュ関数は次のような特徴を持つため、暗号化アルゴリズムなどで広く利用されています。

- ・ハッシュ値からもとのデータを知ることはできない
- ・異なるデータのハッシュ値が同じ値になることはない

ファイルデータから計算されるハッシュ値の情報(HashInfo)をうまく利用することにより、

注1) <http://thinkit.co.jp/free/article/0709/19/>

最初に述べたようなさまざまな有用な機能を実現できます。



重複ファイルの視覚化



ハッシュ値の特徴を利用すれば、手持ちのすべてのファイルのハッシュ値をあらかじめ計算しておくことにより、あるファイルが別のところすでに存在するかを調べることができます。たとえば、movies/abc.mp4という動画とbackup/xyz.mp4という動画のハッシュ値が同じであれば、abc.mp4とxyz.mp4は同じファイルだということがわかるので、たとえば両方をバックアップする意味はないと判断できます。

図1は、手持ちのパソコンのすべてのファイルのハッシュ値を計算し、それをTreemapという手法で視覚化してみたものです。同じファイルが複数ある場合、そのファイルを赤く表示しています(研究室に所属していた池滝俊太氏の卒業論文による)。

池滝氏は整理が得意なのか、重複するファイルは多くはないようですが、それでもかなりの部分に色がついている(ファイルが重複している)ことがわかります。

ハッシュ情報を複数の人間で共有すれば、自分が持っているファイルを他の人も持っているかがわかります。図2は、研究室の数人の学生に協力してもらって、他人のパソコンまで比較対象を広げて同じ計算をしたものです。多くの学生はMacを利用しており、Macのシステムファイルは全員に共通であるため、図1に比べると左下部分が赤くなっています。

このように、ハッシュ関数を利用すると自分が持っているファイ

ルの素性がなんとなく見えてくると言えるでしょう。ファイルの中身をいくら調べてもこういうことはわからないわけですが、他のファイルや他人のファイルとの比較によりファイルの特徴が見えてくることになります。



重要情報のバックアップ

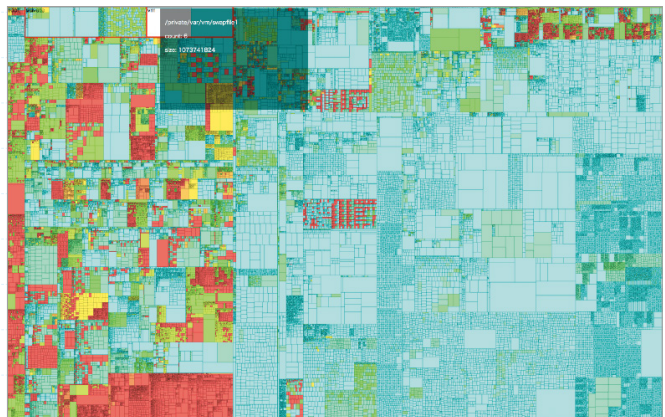


前述の方法を使うと、複数のマシンでHashInfoを計算することによって、さまざまなマシンで同じファイルが共有されているかがわかります。システムファイルやメジャーなアプリケーションのファイルは多数のマシンで共有されているはずですが、あまり重要でない大きなファイルでもいろんなマシンで共有さ

▼ 図1 自分のパソコン内で重複しているものを赤く表示したもの



▼ 図2 他人とのファイル共有の視覚化



れている可能性があります。

たとえばNode.jsでプログラム開発をする場合、node_modulesというフォルダの下にたくさんのライブラリがダウンロードされますが、このようなファイルはバックアップをする必要はありません。

ブラウザでWebからデータをダウンロードするとき、大きなアプリケーションのプログラムファイルも重要なファイルも同じフォルダに入ってしまう、広く配られているファイルと重要なファイルを区別することは困難ですが、HashInfoを利用するとその違いが明白になるので、どのファイルが重要なかがわかるようになります。

一方、コンパイル結果やログファイルなどは重要ではないにもかかわらず、他人が同じファイルを持っている可能性は低いと思われます。

「他人と共有していない」「自動生成されるファイルではない」といった条件を満たすファイルは重要である可能性が高いので、こういう性質をもつファイルだけをバックアップすることにしておけば、無駄なバックアップを防ぐことができるでしょう。



ファイルの存在証明



特許や論文になるかもしれない新しいアイデアを思いついたときなど、ある時点においてある情報を自分が持っていたことを証明したい場合があります。

情報がいつ作成されたかを個人的に覚えておくためにはファイル作成時刻を記録しておけば良いのですが、ファイルの中身や作成時刻は後から偽造できるので、確かにその時点でその情報が存在したということを他人に納得させることはできません。デジタル情報がいつ作成されたか証明するためには、信用できる外部の誰か

に認証してもらう必要があります。

ある時点である書類が存在したことを証明するためには昔から「公証役場^{注2}」が利用されています。公証役場の業務の1つに「確定日付の付与」というものがあります。これは、私文書に「確定日付」を付与し、その日付にその文書が存在したことを証明するものです。

デジタルデータを印刷して書類にしておけば、その存在を証明するために公証役場を利用できますが、情報を印刷して公証役場に持って行くのは面倒ですし費用もかかります。

最近はデジタル情報の存在を証明するために「電子公証システム^{注3}」を利用できるようになったので、公証役場に行かなくても公的な認証を得られるようになりました。これは朗報なのですが、かなりの費用がかかるのが難点です。

ある時刻にあるデジタルデータが存在したことを証明する技術のことをデジタルタイムスタンプ^{注4}技術と言います。電子公証システム利用は高価であり利便性の問題もあるので、Seiko^{注5}やアマノ^{注6}のような会社がデジタルタイムスタンプ技術を提供するサービスを提供しています。

しかし、大事なデータを民間企業に渡すのは心配ですし、サービスがどれほど信頼できるかわかりません。実際、現在運営を中止してしまったサービスもありますし、10年後／20年後も確実に利用できる保証はありません。

電子公証システムや民間の証明システムを利用する場合は証明したいデータの提示が必要ですが、情報が漏れることは心配かもしれません。その時点でその情報が存在することは証明したいけれども情報の中身は公開したくないかもしれません。簡単で良いアイデアを思いついたときなど、特許をとるまで内容は誰にも公開したくないでしょうが、そのアイデアを自分が思いついた日時については記録しておきたいでしょう。

注2) <http://www.koshonin.gr.jp/a2.html>

注3) <http://www.koshonin.gr.jp/de2.html>

注4) <http://www.imes.boj.or.jp/japanese/kinyu/2000/kk19-b1-4.pdf>

注5) https://www.seiko-cybertime.jp/product/easy_time_stamp/

注6) <http://www.e-timing.ne.jp/product/timestamp/characteristic/type/>

つまり「情報を持っていることは証明するが、情報そのものは公開しない」方法が必要になります。



HashInfoによる デジタルタイムスタンプ

生のデジタル情報の存在証明を行う代わりに、その情報のハッシュ値の存在証明を行うことにすれば、情報そのものを渡す必要がないのでより安全だと考えられます。

たとえばある問題の解法を発見したとき、たとえば「問題AはBという方法で解決できる」といった文字列のハッシュ値を公証役場や電子公証サービスに登録しておけば、ハッシュ値からもとの文字列を計算することはほぼ不可能なので、文字列そのものを登録しなくても証明を行うことができます。

電子公証サービスや民間のデジタルタイムスタンプサービスを利用しなくても、一般的なブックマークサービスやブログサービスを利用することによって無料でハッシュ情報の存在を証明できるかもしれません。

たとえば秘密情報のハッシュ値がABCDEFという値になるとき、<http://example.com/ABCDEF>というURLをはてブのようなブックマーク登録サイトや任意のブログサービスに記録しておけば、ABCDEFという情報がどの時点で存在していたかがそれらのサイトで公開されることになりますから、Aという問題を解くための情報が、ある時点で存在したことが証明されることになります。

1つのサービスだけに登録すると、そのサイトの管理者に偽造される可能性もあるでしょうが、関連のないたくさんのサービスに同じ情報を登録しておけば、すべてを偽造することは不可能と考えられるので、その情報がその時間に存在したことをかなり確実に主張できると思われます。

つまり、時刻を証明したい情報があるときは必ず次の手順をとることにしておけば、確実に

データの存在証明を行うことができるでしょう。

- ・情報を秘密の場所に格納する
- ・情報のハッシュ値を複数のサイトに登録する

普通のテキストエディタを使っている場合でも、気合いを入れてセーブした場合には上記の処理を自動的に行うようにしておけばいいかもしれません。秘密にしておくべき元データを安全な場所に格納しておけば良いでしょうし、暗号化したデータをクラウドなどにもセーブしておけばさらに安全かもしれません。



HashInfo.net



今回はHashInfoを利用する3種類のアプリケーションを紹介しましたが、多数の人間でファイルのハッシュ値を共有するデータベースがあればさまざまな用途に利用できることが期待できるので、誰もがHashInfoを活用できるようにするためのHashInfo.netというサイトを運用したいと考えています。

一方、こういうサービスではプライバシーの問題に気を付ける必要があります。素性が怪しいファイルを他の誰かが持っていることがわかれば問題になる可能性もありますし、「名寄せ」的に個人が特定される可能性があるので、誰がどのハッシュを持っているかという情報は十分に注意して扱う必要があるでしょう。しかし、プライバシーやセキュリティに十分配慮すればHashInfoが有用な機会は多いはずです。

Googleは、ファイルの中身よりもファイル間のリンクを重視することで検索精度を向上させることに成功しましたが、ファイルの中身よりもファイルの重複を重視することによって大きな効果を得られる可能性もあるでしょう。HashInfo.netの運用などによって有益なシステムを構築できればと考えています。SD

コロナブス日和

第15回 フラット整理法

エンジニアというものは「楽をするためならどんな苦勞も厭わ^{いと}ない^い」ものだと言われていますが、コロナブスの卵のようなゴキゲンな発明によって頑張^いって楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



整理術の紹介



今回は単純で役に立つ整理のためのテクニックを紹介したいと思います。私は整理が極端に苦手、計算機ファイルの整理や小物の整理に毎日苦しんでいます。そういう人間が整理法について語るのとは変かもしれませんが、何かが下手な人ほどそれを克服するために苦勞しているはずですから、良い方法を発明する可能性は高いはずです。



階層脳の恐怖



引き出しや^{たす}筆筒の中に置き場所を決めて小物を整理したり、階層的なフォルダを使ってファイルを整理したりしている人は多いと思いますが、現代においてそれは最善の整理方法でしょうか？

なんでも階層的に分類して格納するものだと思っていたり、ファイルは階層的なフォルダで管理するのが当然と考えている人は「階層脳」を持っているのだと思われます。

以前の記事で、大規模な階層的データを検索したりナビゲーションしたりする方法として、「DragZoom」や「Gear」のようなシステムを紹介しました。大きなデータは階層的に構成されていることが多いためこのようなシステムが有用なわけですが、実はこれらのシステムで扱っていたデータはすべて専門家によって用意された定形的なデータであり、個人が扱う雑多なもの

ではありませんでした。



階層管理の問題点

住所や電話番号のように明らかに階層的な構造を持つものを階層的に管理するのは難しくありませんが、さまざまな属性を持つ雑多なデータや物品を階層的に管理するのは難しいものです。A社のBという製品に関する問い合わせ書類はどういうフォルダに入れるべきでしょうか。A社フォルダでしょうか？ B製品フォルダでしょうか？ 問い合わせフォルダでしょうか？

また、旅行先で買った便利なナイフはどの箱に入れるべきでしょうか。思い出箱でしょうか？ 文房具箱でしょうか？ 旅行用品でしょうか？

属性がはっきりしない場合もあります。動物を分類しようとするとき、コウモリは獣に分類すべきか鳥に分類すべきか困るかもしれませんが、分類作業にはこういう「コウモリ問題」がつきものです。図書館のように専門の職員がいる場所であっても、変な分類をされていて困ることもあります。世の中の多くのものは階層的に管理することが難しいので階層構造を仮定した検索手法や整理法はうまくいきません。



フラット整理術



Webの巨大なデータは階層的に管理されていませんが、Googleのようなテキスト検索でたいていの情報は見つかる気がするので、階層的でないことを問題だと思っている人はいないでしょう。つまり「検索できれば階層的整理は

注1) <http://thinkit.co.jp/free/article/0709/19/>

必要ない」ということになります。

Webの黎明期にはWebページを階層的に整理するサービスがたくさんありましたが、現在はほとんどなくなってしまいました。階層的に分類する作業がたいへんになり過ぎたからかもしれないませんが、階層的な構造がそれほど有用と思われなくなってきたからかもしれません。

Webを階層的に整理するサービスでは、コウモリ問題に対応するため、同じデータをあちこちに置くという方法がよく使われていました。たとえば横浜のラーメン屋のページは、「横浜」というカテゴリにも「ラーメン」というカテゴリにも登録されるようになっていました。一方、現在は「横浜」「ラーメン」のようなキーワードで検索すれば簡単に情報にたどり着けますし、横浜レストラン情報やラーメンランキングのようなページがいくつでもありますから、階層的な分類を行う検索サービスの存在価値は現在かなり小さくなっていると思われます。

Webページを階層管理できないということは、個人のファイルや所有物を階層管理できないのと同じで、均質でわかりやすい構造を持つデータ以外は階層管理にはむかないということになるでしょう。個人のファイルや所有物の場合、レストラン情報のような整理情報を期待することはできませんが、そもそもすべて自分が一度は見たものですからそういうものは必要ありません。昔見たはずの情報が再度見つけれられるようになっていけば問題ありません。なんでも階層的に管理すべきであるという「階層脳」から脱却し、フラットな構造でもうまく検索できる方法を利用すべきだと考えられます。



計算機情報のフラット整理



次のような方法を使えば、階層的にいていなくても分類を行わなくても雑多なファイルを適切に管理できます。

✓ ファイルにユニークな名前を付けて1つの場所で管理する

たとえば「A社のBという製品に関する問い合わせ書類」をメールで受け取った場合、この「製品B問い合わせ.pdf」のようなファイルのハッシュ値を計算して01234567.pdfのようなファイル名に変更し、1つの場所に保存します。ファイル名はユニークであれば何でもかまいません。

✓ ファイルにタグやコメントを付ける

ファイルを後で検索できるようにするために、01234567.pdfに対して「A社」「製品B」「問い合わせ」のようなタグを登録します。必要であれば詳しいコメントや関連情報なども書いておきます。

✓ 似たタグやコメントを持つファイルに関連ファイルとして表示する

タグやコメントのテキスト検索によってファイルを検索することができますが、似たタグやコメントを持つファイルが関連ファイルとして自動的に検索されるようにしておくとう便利です。A社や製品Bに関する文書が見つかったとき、その文書に関連する文書もリストされると便利です。

こういう検索／管理システムをゼロから作るのは面倒ですが、以前紹介したGyazo(2016年2月号)を使うと簡単にこの方法を使うことができます。Gyazoでは、キャプチャした画像にタグやコメントを付加したり関連画像を検索したりすることが簡単にできますから、次の手順でファイルをフラットに管理できることになります。

- ・ファイルをユニークな名前に変換して特定の場所に置く
- ・ファイルの中身を表示してGyazoでキャプチャし、新しいファイル名やコメントを記入する

私は、保存したいファイルがあったとき、その中身のMD5ハッシュ値を使ったユニークなファイル名に変換してAmazon S3の特定の場

▼図1 各種申請書をGyazoで検索したところ



所に格納してから、ファイルの中身をGyazoでキャプチャしたものにそのURLをコメントとして書くようにしています。

この手順をルーチンワークにしておくと、ファイル名や階層的ファイル管理についてまったく悩むことがなくなります。たとえば私は各種の申請書を書く場合、Gyazo.comで「申請書」を検索するとたくさんの古い申請書やテンプレートが見つかるので、それを開いて修正することによって新しいものを作成しています(図1)。

ファイルはクラウドに置かなくてもかまいませんが、クラウド上で一括管理しておくのが何かと便利です。Amazon S3などを利用するのが面倒な場合はDropboxなどを使うと良いでしょう。

身のまわりのフラット整理

前述の方法であらゆるファイルをフラットに管理できるようになるのですが、書類や小物はどうすれば良いのでしょうか？ 実は書類や小物をもっと簡単な方法で楽に整理できます。



クリアファイル整理法

書類の管理は面倒なものです。普通の家庭でも、製品のマニュアルだの保証書だのクレジットカード明細だの領収書だの保険の書類だの、医療費明細だの、たくさんの書類を整理する必

要がありますが、こういったものをきれいに分類して管理するのはたいへんです。

雑多な書類を楽に提案する方法として、1993年ごろ野口悠紀雄氏が提案した「超整理法^{注2)}」という書類整理手法がかなり話題になったことがあります。「超」と言っていますがこれは極めて単純な整理方法で、

- ① タイトルを書いたA4の封筒(角形2号封筒)に資料を入れて棚に並べる
- ② 使った資料は必ず一番端に戻すという作業を繰り返す

だけです。この結果、

- ・よく使う資料が端に集まる
- ・使わない資料は逆の端に集まる

ことになりますが、よく使う資料は端にある可能性が高いのですぐ見つかりますし、逆の端にたまってきた使わない資料は廃棄を検討できるというわけです。

ここでは分類も階層的管理も使われていませんから、階層脳な人はこれでうまくいくことに驚いたかもしれません。野口氏は最近のインタビュー(『PHP Business“The 21”2016年11月)でも「目当ての情報がすぐ見つかるのであれば、わざわざ整理に時間や手間をかけることはありません。その方法論の基本は、『分類しない』ことです」と語っており、この整理手法を現在も利用しているようです。

「超整理法」は確かに簡単うえに分類の必要がないという大きなメリットがあるので非常に話題になりました。野口氏をはじめ、現在もこの方法を利用している人も多いようですが、この方法には次のような問題がある気がしています。

- ・封筒にタイトルを書くのが面倒臭い
- ・封筒の中身が見えない
- ・古いファイルを探すのに時間がかかる

注2) 「超」整理法——情報検索と発想の新システム 野口悠紀雄(著)、新書判、248ページ、1993年11月25日、ISBN978-4-12-101159-6(<http://www.chuko.co.jp/shinsho/1993/11/101159.html>)

- ・古いファイルの存在に気づかず同じファイルを作ってしまう可能性がある
- ・角形2号封筒のサイズはA4書類よりかなり大きいのでA4用の箱や棚に入れにくい

現在は計算機やネットワークが十分普及しており、「超整理法」が執筆された1993年とは状況がかなり異なっていますから、私は次のような「クリアファイル整理法」のほうが良いと思っています。

- ①大きく番号を書いた透明クリアファイルに書類を入れる
- ②クリアファイルを番号順に並べる
- ③番号と中身の対応をテキストファイルやWikiなどに書いておく

私は数年に渡ってこの方法を実践しており、実用的に利用しています。番号を最初は手書きしていましたが、最近は印刷したシールを貼っています(写真1)。

これは非常に単純な方法ですが、次のようなメリットがあります。

- ・タイトルやキーワードを手書きする必要がない
- ・テキスト検索すればどの番号のクリアファイルに何が入っているかすぐわかる
- ・大きさがA4にそろっているのでA4用の箱や棚で整理しやすい
- ・透明なので中身を確認しやすい
- ・番号順に並んでいるので、整理／検索しやすい
- ・書類の属性やコメントなどをいくら記述してもかまわない

超整理法の時代にはタイトルや中身の情報を書くために封筒を利用する必要がありました、たくさんの情報を書くことはできませんでした。クリアファイルには番号だけ書いておけばいいわけですし、テキストファイルにはいくらでも情報を書いておくことができます。

この「クリアファイル整理法」はあまりにも単純なので、こんなものでうまくいくのか？ こういう

方法をサポートする文房具や検索システムが売られているのではないかと考えたのですが、意外とそういう製品は存在しませんでしたし、この方法はとてもうまくいくことがわかりました。

番号と中身の対応を書くのが面倒だという問題はありますが、封筒に手書きするよりは楽ですし、あとで検索するのが楽であることを考えるとテキストファイルに書くほうがはるかに便利でしょう。

▼写真1 書棚の中のクリアファイル



箱整理

クリアファイルに入れにくい小物は、番号を書いた箱に入れるようにすればクリアファイルの場合と同じように整理を行うことができます。箱や引き出しでものを整理する場合、箱のカテゴリを決めて分類して格納したくなるものですが、箱の中身をきちんと記述しておくことに気を付けておけば、きちんと分類する必要はないでしょう。たとえば「土産」とか「食器」とかいった箱を用意してきちんと分類したくなるかもしれませんが、「土産物の食器」はどちらに入れれば良いかわかりません。適当な番号の箱に入れてから、その番号のところに「Xさんにもらったドイツのワイングラス」などと記述しておけば後から探すのに苦労することがありません。



フラットへの移行

現状の開発環境は、階層型ファイルシステムが前提になっているのがほとんどですので、残念ながらあらゆるファイルを今回紹介した方法で管理することは困難です。しかし、これまで整理が難しかった雑多なファイルに関しては明らかに有効だと感じているので、徐々にフラットな整理方法に移行していこうと思っています。SD

コロンブス日和

第 16 回 Scrapbox(1)

エンジニアというものは「楽をするためならどんな苦労も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



Scrapbox の紹介



2017 年 1 月号ではさまざまなファイルや書類を「フラット」に管理する方法について説明しましたが、今回はさまざまな情報をより効果的に整理し共有できる「Scrapbox」というシステムを紹介します。



理想の情報整理システムとは



2017 年現在における理想的な情報整理法とはどのようなもののでしょうか？ 現在のテクノロジー状況をふまえて要件を考えてみます。

✓ ブラウザを使ってネット上で情報を管理する

ネットやブラウザを使えない環境は現在ほぼなくなりましたから、情報整理は当然これらを活用するべきでしょう。

✓ 一カ所で集中管理する

ネット上の 1 つの場所にあらゆるデータを保存してブラウザからアクセス可能にするのが良いでしょう。

✓ 気軽な入力と編集

ブラウザ上での入力／編集手法は極力単純にしておく必要があるでしょう。

✓ タグを活用する

先月号でも解説しましたが、個人的な雑多なデータを階層的に管理することは常人にはとても難しいので、タグを使って管理するのが良いと思われます。

✓ 他人との情報共有

Web 上に置いた情報を簡単に共有できると便利でしょう。

現在、Web 上で情報を整理したり共有するためのさまざまなシステムが利用されていますが、上記の要望をすべて満たすものはありませんでした。



Scrapbox

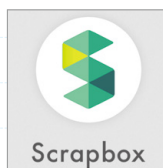


私は、Web 上に気軽に情報を書いて簡単に共有できる Scrapbox というシステムを開発して利用しています。長年に渡って「Gyazz」という名前の Wiki を作成して使っていましたが、Scrapbox はこれを大幅に改良して前述の要件をすべて満たすようにしたものです(図 1)。

Scrapbox は次のような特長を持っています。

✓ ブラウザ上で情報編集／整理

Scrapbox ではあらゆる情報を Web 上に置き、ブラウザで閲覧や編集を行います。このためあらゆるパソコンやスマホなどから



◀ 図 1 Scrapbox
(<https://scrapbox.io/>)

注 1) <http://thinkit.co.jp/free/article/0709/19/>

同じデータにアクセスできます。

✓ グループで利用

ScrapboxのページにはユニークなURLがついており、個人で利用することも共有して利用することもできます。

✓ 強力な編集機能

ScrapboxのページはEmacsのようなテキストエディタと同じようにブラウザ上でWYSIWYG (What You See Is What You Get)的に編集を行います。編集されたテキストは自動的に保存されるので保存ボタンなどを押す必要はありません。

✓ リアルタイム同時編集

ScrapboxではGoogle Docsと同じように、複数ユーザが同時にリアルタイムにWYSIWYG編集することができます。

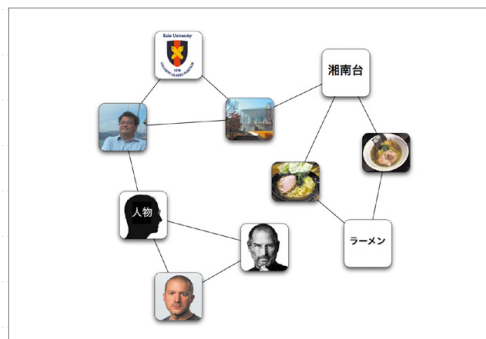
✓ リンクの活用

Scrapboxではタグを利用して情報管理を行います。ScrapboxはWikiなので、「増井」というページに「人物」というタグを書くとこれは「人物」というページへのリンクになりますが、このとき「人物」ページから「増井」ページへのリンクも同時に作成されるようになっています。このため「増井」のような名前を忘れたとしても「人物」ページから「増井」ページへのリンクを見つけることができます。

このような「逆リンク」を利用すると、特別なタグ検索機能を用意しなくても、タグ的なものをすべて独立したWikiページとして作成するだけでタグを用いた検索が利用できるようになります。

また、Scrapboxではリンク先ページからリンクされているページ(2ホップ先のページ)までを関連ページとして表示できるようになっているので、「Steve Jobs」のページも「人物」ページにリンクされている場合「増井」のページから「Steve Jobs」のページを直接参照できます(図2)。

▼ 図2 リンクの構造



たとえば「湘南台」のページを表示したときは、湘南台のラーメン屋のページや私のページは表示されますがSteve Jobsのページは表示されませんし、Steve Jobsのページを表示したときは「人物」つながりで私のページやJonathan Iveのページは表示されますが、慶應大学や湘南台のページは表示されません。

✓ 代表画像

リンク先ページを表示するとき、ページに含まれる画像を表示できるようになっているので、関連ページの視認性がよくなっています。

✓ その他各種の便利機能

各種の簡単なタグ記法、プログラムコードをきれいに表示する機能、簡単な記法による文字装飾、テキストと同じ大ききで画像を文中に埋め込むアイコン表示機能など、たくさんの便利な機能が搭載されています。



Scrapbox 利用例



ScrapboxはWeb^{注2}から利用できます。Scrapboxを長年に渡って利用してきた例を紹介します。



研究室での利用

私の研究室では数年に渡ってScrapboxを利

注2) <http://scrapbox.io/>

▼図3 研究室のトップページ



用しており、2016年末現在、約8,000ページが作成されています。研究室に所属する学生の興味はいろいろであり、論文や研究トピックや開発Tipsのような情報に加え、ラーメン情報もアニメ情報も部品情報もイベント情報もすべて同じところに置いてあります。雑多な情報が何千ページもあるとたいへんなことになりそうですが、とくに分類を行わなくても関連ページやカテゴリをページとして関連付けておくだけで、これらの情報が適切に分類管理されるのが便利です。

研究室のScrapboxのトップページは図3のようになっています。研究関連情報からアニメ情報、ラーメン情報までかなり雑多な情報が並んでいることがわかります。

ここで「AD620」という部品に関するページを選んで表示すると、その部品の詳しい情報が表示され、似た部品など関連研究のページが表示されます(図4)。この部品ページには「ストレンゲージ」「オペアンプ」というページへのリンクが定義されているので、同じページへのリンクを持つ「キッチンスケール」や「LMC660」が関連ページとして表示されているというわけです。

一方、藤沢のラーメン屋のページを選んで表示すると、藤沢やラーメン屋のページがたくさん関連ページとして表示されます(図5)。部品情報もラーメン屋情報も同じ場所に書いてあるのですが、リンク関係がまったく違っているのので別のクラスタとしてうまく管理できています。

▼図4 1つの部品ページを開くと関連する情報がすべて表示される



UIPedia

前述の例は研究室内の人間だけが参照できるページでしたが、ユーザーインターフェースに関連する論文やシステムなどをScrapboxで公

開いています(図6)。UIPediaページは誰でも閲覧できますし、参加も可能になっています^{注3}。

文献情報を管理するさまざまなシステムが利用されていますが、文献データベースでは入力できる情報の属性が限られているのが普通です。Scrapboxを使うと、タイトルや著者名のよう

多くの機能を盛りこんだために現在のScrapboxの仕様はある程度は大きなものとなっていますが、多くのアイデアは「コロンブスの卵」的であり、仕様のシンプルさは保てています。次号では、Scrapboxのさらに詳しい利用法および実装について解説します。**SD**

著者の写真や弟子筋情報、配偶者情報を記述したり、普通の文献整理システムではできないことができるので味のある情報ページにできます。



家族情報

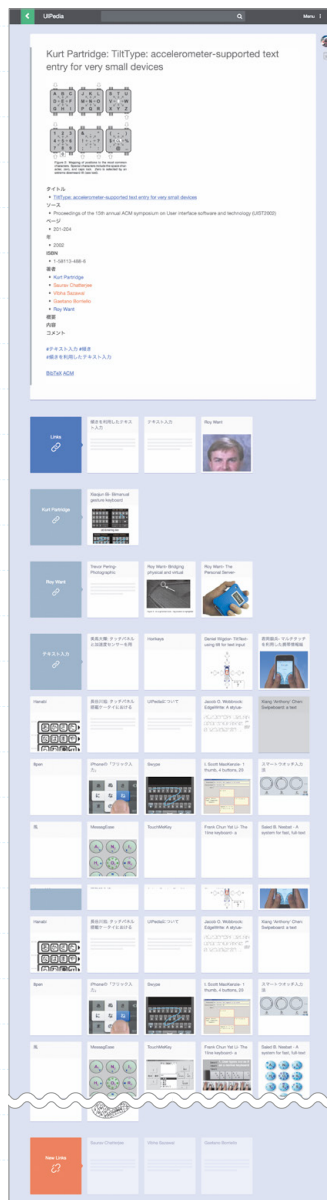
私は個人的なメモや予定表、TODOなどをすべてScrapboxで管理しており、現在6,000ページほどが作成されています。また家族間で共有したい情報もScrapboxで管理しています。親戚の連絡先、銀行口座情報、各種の契約情報、予定表など、家族間で共有したい情報は意外とたくさんあるものです。こういった情報は住所録やスケジュール帳のようなアプリケーションで管理している人が多いと思いますが、家族関連情報はすべて1つの場所に置いておけば何かと便利です。私の場合、家紋の情報、家系図、引っ越し履歴など家族に関連するさまざまな情報を書いていたら簡単に100ページを越えてしまいました。

私はこの考えに基づくシステムを10年以上利用しており、Scrapboxはその最新版です。

▼図5 藤沢のラーメン屋のページを開いたところ



▼図6 入力インターフェースの研究ページ



注3) <https://scrapbox.io/UIPedia>

コロンブス日和

第 17 回 Scrapbox (2)

エンジニアというものは「楽をするためならどんな苦勞も厭わ^{いと}ない^い」ものだと言われていますが、コロンブスの卵のようなゴキゲンな発明によって頑張って楽できるなら、それに越したことはないでしょう。私はコンピュータ上の簡単な工夫で楽をする方法を考えるのが好きで、長年にわたっていろんなシステムを開発してきています。今回の連載では、私がこれまで開発して長年実際に利用しているような単純かつ便利なシステムをたくさん紹介していきます。

Author 増井 俊之(ますい としゆき) 慶應義塾大学



Scrapboxの記法



先月号では、シンプルで強力なWikiシステム「Scrapbox」の概要を紹介しましたが、今回はScrapboxの機能についてより詳しく紹介したいと思います。

Scrapboxでは、簡潔なマークアップ記法により、他のページへのリンクを張ったり、外部Webページへのリンクを張ったり、画像や動画を表示したり、文字修飾を行ったりすることができます。現在広く利用されているHTMLは「HyperText Markup Language」の略であり、テキストの装飾やリンクなどを指定するためにや<a>のような各種のマークアップ(タグ)が利用されています。

HTMLはあらゆるブラウザで利用できるのですが、仕様が巨大で覚えるのが大変ですし、単に文字を太くするだけのために「」「」のような記述をしなければならないのは面倒です。このため最近では「Markdown」のような簡易マークアップ言語が人気があるようです。MarkdownはHTMLに比べると記述が簡潔です(リスト1)、GitHubなど多くのサービスで利用できるので便利です。

一方、Markdownで画像やリンクを表現するための言語仕様は、かなり複雑なため覚えるのが困難です。たとえば、リスト1の「[タイトル](URL)」のような記法は私にはなかなか覚えられません。実

際にマークアップはHTMLを併用して使われていることもあるようなので、Markdownの存在価値を私は疑問に感じています。



Scrapboxのマークアップ

Scrapboxでは極力簡単に目的を達成できるような簡単に強力なマークアップ記法を採用しています。基本的には、

- ・あらゆるマークアップは角カッコ・([...])で表現する
- ・記述の制約をできるだけ緩くする

という方針に基づいています。

内部リンク

テキスト内の文字列を[]で囲むと、囲んだ部分がその名前のページへのリンクになります。たとえばScrapboxの文中で「増井」と記述すると「増井」というページへのリンクになります。

外部リンク

外部のWebページにリンクを張りたいときは、

▼ リスト1 Markdownの記述例

Gyaaim

- * [MacRuby](http://www.macruby.org/)で作ったMac用のIMEです。
- * 数百行のRubyで実用レベルのIMEを作れることを示すものです
- * 単純な変換しかできませんがそれなりに使えます
- * 見栄えはInterfaceBuilderで簡単に変更できます
- * 変換アルゴリズムやIMEの動作はRubyで簡単に変更できます
- * 自前のIMEをいろいろ作りましょう!
- * 変換手法はSlimeと同じです
- * 予測機能はありません

インストール

注1) <http://thinkit.co.jp/free/article/0709/19/>

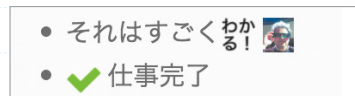
▼図1 [増井.icon]の表示結果



▼図2 [増井.icon*10]の表示結果



▼図3 Scrapboxページへの適用結果



[URL タイトル]のように記述します。これはタイトルのようなHTMLに変換されます。[タイトル URL]のようにURLとタイトルを逆順に書いても同じ結果になるので、順番について気にする必要はありません。

✓ 画像や動画の埋め込み

テキスト中に[画像URL]と書くと、その場所に画像が表示されます。また[画像URL リンクURL]と書くとURLへのリンクを持つ画像が表示されます。この場合も、URLの順番は逆でも大丈夫です。画像URLの代わりにYouTubeやVimeoのURLを書くとページに動画が埋め込まれます。

✓ 文字修飾

[* ...]のような記法で太字や斜体を表現できます。[* abc]と書くと**abc**のように太字になり、[/ abc]と書くと*abc*のように斜体になり、[/* abc]と書くと***abc***のように太字の斜体になります。



アイコン記法

[増井.icon]と書くと、「増井」というページの代表画像が、アイコンのように行の中に表示されるようになっていきます(図1)。

アイコンをたくさん並べることもできます。たとえば[増井.icon*10]と記述すると、アイコンが10個並びます(図2)。

このような「アイコン記法」を利用すると、Scrapboxページを絵文字のように利用できて便利です(図3)。

Wikiページ上で複数の人が編集を行った場合、

▼図4 コード記法



誰がどの部分を書いたのかわからなくなって困ることがありますが、このようなアイコンを常に書く習慣をつけておけば、そういう問題が起きにくくなります。



コード記法

ちょっとしたプログラムをWeb上で公開したいとき、Gistのようなサービスにプログラムの断片を貼ることがありますが、Scrapboxページの中にコードを記述できます(図4)。

コード記法したデータは「https://scrapbox.io/api/code/prog-exercises/モンテカルロ法で円周率を求める/pi.js」のようなURLでアクセスすることができます。



その他の便利な機能



アウトライン編集

Scrapboxのページでは、行頭に空白文字を入れてインデントすることによって行の階層構造を指定できますが、階層構造を利用してアウトライン編集を行うことができます。



行の古さの表示

Wiki ページを長年利用していると、新しく記述を追加しても気付きにくいことがありますし、どの部分が新しいのかわからなくて困ることがあります。Scrapbox では、行の左端に表示される灰色の矩形の大きさが行の古さがわかるようになっているので、記述の古さがすぐわかります。図5の例では、「藤沢店」あたりの記述が他の行よりも新しいことがわかります。



プレゼンテーションモード

スライドを使ったプレゼンテーションを行うときはPowerPointやKeynoteのような専用ソフトウェアが利用されるのが普通です。これらのソフトウェアは高機能で便利なのですが、ブラウザと連動して利用することはできないので困ることもあります。

Scrapbox には「プレゼンテーションモード」という機能があり、ページを使ってスライド表示できます(図6)。私は授業の資料をすべてScrapboxで作成しています。



入力補完

Scrapbox でたくさんのページを作っていると、ページ名で混乱することがあります。ページが存在するかどうかは検索すればわかるのですが、ハッシュタグやカッコの後でページ名を

入力しようとするとき、すでに存在するページ名に対して曖昧検索を行って候補を表示するようになっているので、似て異なるページを作ってしまう可能性を減らしています。



ソート機能

Scrapbox のページは、編集時刻の順に表示したいことがあるでしょうし、アクセス時刻順に表示したいこともあるでしょう。Scrapbox ではさまざまな指標でソートできるようになっています。



ピンどめ機能

どういう方法でソートした場合でも、古いページはリストの下の方に移動してしまうことになります。重要なページや最初に見るべきページについては常にトップに固定されるようにするための「ピン機能」を用意しています。ピンどめされたページは常にプロジェクトの先頭に表示されます。



各種の設定

ブログなどに利用するScrapboxのページは誰に対しても公開したいでしょうし、個人的なメモは誰に対しても秘密にしておきたいでしょう。Scrapboxでは、ページを公開するかどうかを設定可能であり、またページを編集可能なユーザーを設定できるようになっているので、内容に応

じて細かくアクセス制御を行うことができます。また、ページの見栄えも細かい設定が可能になっています。



Scrapboxの実装

Scrapbox の編集機能や同時編集機能は、Atom、Gitの実装を参考にしてReactなどで

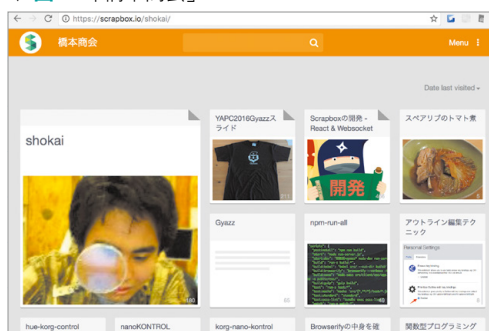
▼ 図5 行の古さを表示



▼ 図6 テキストとプレゼンテーションモードでの表示



▼図7 「橋本商会」



すべて自前で実装しています。実装の詳細は、橋本翔氏のページ^{注2}で解説されています(図7)。

Scrapboxの利用例

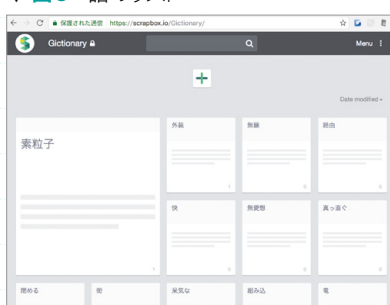
Scrapboxの利用例は先月号でもいくつか紹介しましたが、再度紹介しておきます。



個人メモとしての利用

個人用のちょっとしたメモは、パソコンのローカルファイルとして保存している人が多いと思いますが、ローカルにファイルを保存すると別のマシンやスマホから見ることはできませんから、ちょっとした情報でも Scrapbox のようなクラウド上に保存しておくのが良いと思います。私は電話帳も予定表も TODO リストもすべて Scrapbox 上で管理しているので、スマホでもパソコンでも同じように情報を管理できています。昔はデータを同期したり、共有ファイルとすることが多かったようですが、完全に一本化しておくのが一番便利だと感じています。

▼図8 語のリスト



▼図9 「素粒子」のエントリ



✓ ブログとしての利用

各種のブログサービスが利用されていますが、Scrapboxを公開設定すれば、簡単なブログのように利用することもできます。

✓ 日本語入力辞書

昨年5月号で「Gyaim」という日本語入力システムを紹介しましたが、私はGyaimで使われる辞書の管理にScrapboxを使っています(図8、図9)。

辞書は単純な形式ですが、間違いや追加があればWiki上で簡単に修正できるのが便利です。多くのユーザで共有して利用すれば、質の高い辞書に育てていくことも可能でしょう。

2回にわたって汎用で使いやすいWiki「Scrapbox」を解説しました。現在のScrapboxはかなり複雑なシステムになってしまったので実装はコロンブス的とは言いがたいですが、ユーザから見ればその利用方法は、コロンブスの卵的だと感じられると思います。情報を階層的に関連するのをやめ、あらゆる情報をクラウドに置くことによって情報整理に関する精神の重荷はかなり軽減されますし、将来の情報共有にもとても有用だと思います。ぜひScrapboxをいろんな用途に活用ください。SD

注2) 「橋本商会」橋本翔氏のブログ <https://scrapbox.io/shokai/Scrapboxの開発%20-%20React%20&%20Websocketで作るリアルタイムWiki>

コロンブス日和

最終回 発見プログラミング

「コロンブスの卵」的なシステムをいろいろ紹介してきたこの連載も今回が最終回になりました。コロンブスといえば「発見」ということで、最終回の今回はプログラミングと発見にまつわる話をします。

Author 増井 俊之(ますい としゆき) 慶應義塾大学

発想と発見

新しい発想を得るためには人間が時間をかけてやり方を考えたり試行錯誤したりする必要があります。前回紹介した「Scrapbox」は、少しでも関連のある情報を柔軟に思い出しやすくすることによって、発想活動を支援します。もし発想活動の一部でも計算機にやらせることができればアイデアを出す効率がアップするかもしれません。

プログラミングと発見

失敗から何かを発見したというエピソードはたくさんあります。もちろんただ失敗すれば良いというものではなく、たくさんの試行錯誤をした結果、普通と異なる挙動を見だしたことが発見につながったということでしょう。実はプログラムを動かすことによって、さまざまな発見ができます。こういう作業を発見のプログラミングと呼ぶことにします。

今回は、前回紹介したScrapboxの上に書いたJavaScriptコードをブラウザで実行してみたいと思います。Scrapboxにはコード記法というものがあり、プログラムなどのテキストデータをブロック表示してファイルとして外部からURLでアクセスできるので、これを使えばScrapbox上に書いたプログラムをブラウザ上で実行できます。CanvasやSVGを利用するとJavaScriptでブラウザ上に絵を描くことはでき

ますが、初期設定などが必要で面倒なところもあるので、描画などを極端に簡単にしたP5.jsというシステムを利用することにします。

数え上げによる答えの発見

算数の問題を解くときは、理屈を使って計算するのが普通です。たとえば「鶴と亀があわせて30匹いて、足の数は全部で100本でした。鶴と亀は何匹いたでしょう?」のような「つるかめ算」を解く場合、「全部が鶴だとすると足の数は60本で、全部が亀だとすると120本になるから……」のように算数的に解くやり方もありますし、

$$t + k = 30$$

$$2t + 4k = 100$$

のような1次方程式を使うこともできます。数学の問題はこのように解析的に解くのが普通ですが、図1のような絵当たりプログラムを書けば、数学的知識がなくても力技で答を得ることができます。

つるかめ算を解くのにこういうプログラムを書く人は少ないと思いますが、このような方法

▼ 図1 Scrapboxでつるかめ算を解く(tsurukame.js)

```
tsрукame.js
for(t=0;t<=30;t++){
  for(k=0;k<=30;k++){
    if(t+k==30&&t*2+k*4==100){
      alert(`鶴${t}匹, 亀${k}匹`)
    }
  }
}
```

のほうが有利な場合もあります。もし問題が「足
の数が100本以上でした」のようなものであつ
た場合、1次方程式で普通に解くことはできま
せんが、上のような方法ならすべての解を網羅
できてしまいます。つまり、論理的に解けなく
ても、簡単なプログラムを書くことによって発
見的に解が得られてしまうことになります。

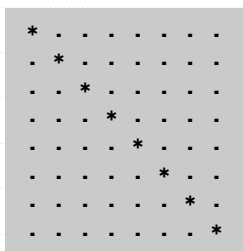


乱数で答えを発見

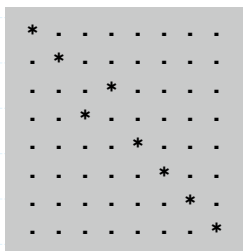


つるかめ算の場合は解の候補をすべて数え上
げれば答えを発見できるはずですが、総当たり
で発見が難しい場合は、乱数を使って試行錯誤
的に答えを捜すこともできます。たとえば
8-Queen問題を解きたいときは、再帰的に解を
探索するアルゴリズムを使うのが常識的ですが、
ランダムに行や列を入れ換えることを繰り返す
だけでも意外と簡単に解を得ることができます。

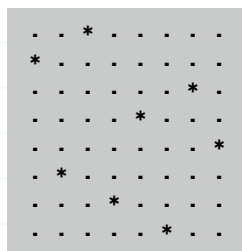
まずこのようなデータを用意します。



2つの行をランダムに交換してみると、次の
ように少しだけ良い結果が得られることがあり
ます。



これを何度も繰り返してよさげなものを残し
ていくと、最終的に次のような答えを得ること
ができます。



人間は数え上げ操作が苦手ですから、
8-Queenのような問題に対しては試行錯誤的に
取り組んで答えを見つけることが多いですが、
試行錯誤を計算機に実行させることによって、
解を発見できる場合も多いでしょう。



乱数の性質の発見



乱数を使って問題を解く場合、JavaScriptの
Math.random()のような「一様乱数」関数がよく
利用されます。これと呼ぶと、0から1までのラン
ダムな実数が返ります。たとえばrandom()
が0.1以下の値を返す確率は1/10です。

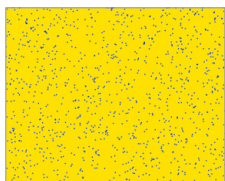
ここまでは普通の話ですが、図2のように
random()で生成した座標を使って点を表示し
てみると図3のようになります。一様な乱数と
いうと図4のようなイメージを連想してしまいま
すが、本当の乱数はこうならないことが発見で
けるのが面白いところです。

乱数はシミュレーションやゲームでよく利用
されるものですので、乱数とはどのようなもの

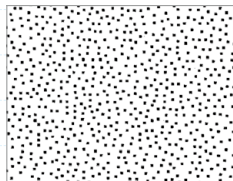
▼ 図2 乱数で生成した座標に点を打つ(random.js)

```
random.js
function setup(){
  createCanvas(400,400)
  fill('yellow')
  strokeWeight(0)
  rect(0,0,400,400)
}
function draw(){
  fill('blue')
  rect(random(400),random(400),2,2)
}
```

▼図3 生成された点



▼図4 一様な乱数



かについては、エンジニアであれば誰でもなんとなくイメージを持っていると思いますが、乱数をたくさん生成して視覚化してみるとさまざまな発見があったりします。



乱数の足し算

いくつかの乱数の平均をとると何が起ころうでしょうか？ ランダムなものをいくら集めてもランダムに決まっているだろうと思うかもしれませんが、実は一様な乱数をたくさん足したも

▼図5 乱数を何回か足した結果のヒストグラムを計算 (normal.js)

```
normal.js
function setup(){
  createCanvas(400,400)
  histogram = []
  for(i=0;i<100;i++){ histogram[i] = 0
    strokeWeight(0)
  }
}
function draw(){
  val = 0
  for(i=0;i<10;i++){ val += random(1/10)
    histogram[Math.floor(val * 100)] += 1
    fill('yellow')
    rect(0,0,400,400)
    fill('blue')
    for(i=0;i<100;i++){
      rect(i*4,400-histogram[i],4,histogram[i])
    }
  }
}
```

▼図6 計算結果



のは一様乱数にならず、正規分布に近づくことが中央極限定理として知られています。2つのサイコロをふった場合、その数の和が2や12になる確率よりも7になる確率のほうが大きいだろうということは直感的にわかりますが、一様乱数を足した場合も同じだということはよく考えないとわからないかもしれません。そういう場合、何も考えなくてもプログラムで実行してみればそのような性質はすぐ発見できます(図5、6)。



乱数の掛け算

乱数をたくさん足すと正規分布になるわけですが、では乱数をたくさん掛けるとどうなるのでしょうか？ 実は乱数を何度も掛け算したものは「^{べきぶんぶ}冪分布」になります。図7のプログラムを実行すると図8のような冪分布になります。リニアスケールで青い点を、対数スケールで赤い点を表示しています。

世の中には「ロングテール」と表現される冪分布がたくさんありますが、どうやらランダムな事象が掛け算的に繰り返されるとロングテールの現象が発生するようです。



勝負の発見

まったく五分五分の勝負を繰り返した場合で

▼図7 乱数の掛け算のプログラム例 (beki.js)

```
beki.js
function setup(){
  values = []
  createCanvas(400,400)
  strokeWeight(0)
}
function draw(){
  value = 1.0
  for(i=0;i<10;i++){ value *= random() }
  values.push(value)
  values = values.sort((a,b) => b-a)
  fill('#ddd'); rect(0,0,400,400)
  for(i=0;i<values.length;i++){
    fill('#00f'); rect(i,400-values[i]*10000,2,2)
    fill('#f00'); rect(50*Math.log(i),
      300-10*Math.log(values[i]),2,2)
  }
}
```

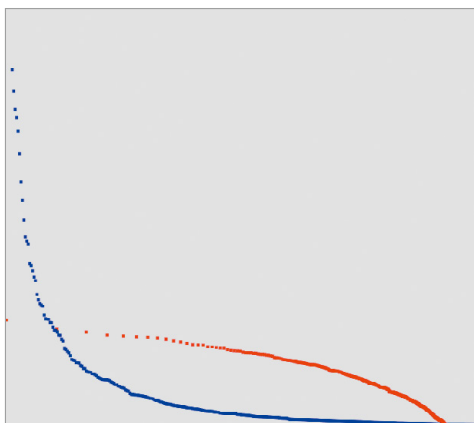

も意外な片寄りを発見できます。

2人に配ったトランプの札を1枚ずつ出して勝負する「戦争ゲーム」というものがあります。このゲームは単純であり面白いものではありませんが、そもそも勝負はつくものなのでしょうか？ Scrapbox上で勝負するプログラムを図9のように書くことができます。

P5.jsのdraw()がメインプロセスになって勝負の結果を表示しています。2人のプレーヤはplayer()というgeneratorで表現されており、2つのプレーヤプロセスがsetup()で初期化されています。player()もメインプロセスも無限ループになっていますが、yieldでメインプロセスと通信することにより同期しながら対戦が行われます。generator機能を利用すると、このようなシミュレータを柔軟に記述できるので便利です。

このプログラムを実行すると、意外にもすぐに決着がついてしまうことがわかります(図10)。最初の手札は同じですし、ランダムにしているにもかかわらずあつという間に総取り状態になってしまうということは感慨深いものがあります。条件がまったく同じであっても大きく勝ったり負けたりすることが発見できたこと

▼ 図8 実行結果



▼ 図10 勝負の途中

player1: 12 13 5 10 11 9 4 6 9 1 8 2 8
player2: 7 11 6 7 2 1 5 13 10 3

になりますが、ほんの少しのきっかけで運命が変わってしまうのは怖いものだと思います。



終わりに



今回紹介したプログラムはすべてScrapbox上に置いてあり(<http://scrapbox.io/prog-exercises/>)、クリックするだけでブラウザ上で実行できます。Scrapboxは情報整理にたいへん便利なシステムですが、そのうえでプログラムを気軽に作成してさまざまな発見ができることでさらに広い応用が考えられると思います。

18回に渡ってコロンプスの卵的な実用的なシステムをいろいろ紹介してきました。私はこれらを毎日活用していますが、読者の皆さんもぜひ試してみてくださいと思います。SD

▼ 図9 戦争ゲームのプログラム(sensou.js)

```
sensou.js
var cards1, cards2, table = []
function* player(cards){ // プレーヤプロセス
  while(true){
    var result = yield cards.shift() // 手を出す
    yield // 結果を待つ
    result.forEach((v) => cards.push(v))
  }
}
function setup(){
  cards1 = new Array(13) // 1~13のカードを用意
  for(var i=0;i<13;i++) cards1[i] = i+1
  cards2 = cards1.concat() // 配列をコピー
  shuffle(cards1,true)
  shuffle(cards2,true)
  p1 = player(cards1)
  p2 = player(cards2)
  createCanvas(400,400)
  frameRate(10) // 毎秒10回勝負する
}
function draw(){
  if(cards1.length == 0 || cards2.length == 0) return
  var hand1 = p1.next().value // プレーヤの手を取得
  var hand2 = p2.next().value
  table.push(hand1); table.push(hand2)
  shuffle(table,true) // 場の札をシャッフル
  v = hand1 == hand2 ? 0 : // 勝敗判定
    hand1 == 1 && hand2 == 2 ? 2 :
    hand1 == 2 && hand2 == 1 ? 1 :
    hand1 > hand2 ? 1 : 2
  switch(v){
    case 0: p1.next([]); p2.next([]); break;
    case 1: p1.next(table); p2.next([]); table = []; break;
    case 2: p1.next([]); p2.next(table); table = []; break;
  }
  clear()
  text('player1: '+cards1.join(' '),10,10)
  text('player2: '+cards2.join(' '),10,30)
}
```