JCT

Creator
さうなまん（モイモラ）

Flashでのテキスト表現を追求

検索キーワードをリアルタイムにFlash上のスクリーンに表示するコンテンツ。
ポイントは、言葉の背景を伝えるFlashデザイン。

Title
JCT

Sample URL
http://vgzh.dtdns.net/dotfla/jct/

Archive
jct.zip

File
05

スクリプト
ActionScript 1.0

対応プレーヤー
Flash Player 8以上

制作アプリケーション
Flash 8

発案
インターネットを俯瞰する

「JCT」は NTT レゾナントが運営する goo の検索エンジン上でキーワードとして入力された語句をリアルタイムに取得し、Flash 上のスクリーンに常時流し続けるコンテンツです。この章では最終形態に至るまでのトライ＆エラーを中心に制作過程を解説します。

○ みんなが何を検索しているのか

「みんなが何を検索しているのか知りたいと思ったことはありませんか？」という見出しで始まる「サーチストリーム」というコンテンツがあります。

URL台 **Excite サーチストリーム**
http://www.excite.co.jp/search_stream/

これは検索サイト「Excite」にて入力された検索キーワードをリアルタイムで表示する標準バナー型の Java アプレットです。

右から左へプレーンテキストが電光掲示板のように流れていき、途中でクリックするとそのキーワードの検索結果ページが開きます。この手の“検索キーワードをリアルタイムに提供するサイト”は今ではたくさんありますが、サーチストリームは昔から存在するパイオニア的存在です（私の記憶が正しければ、1999 年にはありました）。

日本のサイトでは、他にも @nifty が提供する「瞬！ワード」などがあります。

URL台 **瞬！ワード**
http://www.nifty.com/search/shun/

サーチストリーム

瞬！ワード

○ Java アプレットから XML データソースへ

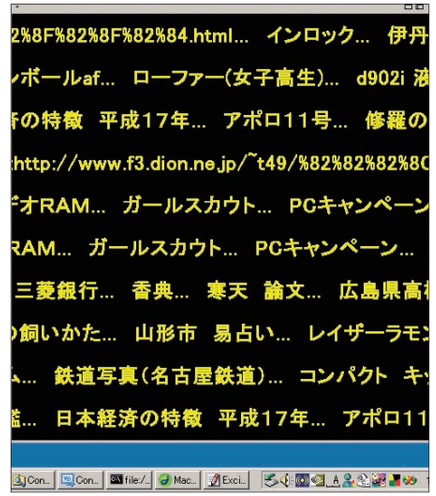
サーチストリームが展開されている HTML ソースを見ると、Java アプレットが埋め込まれていることが確認できます。当時、この部分のソースを空の HTML に複数回コピーし、画面いっぱいに標準バナーサイズのサーチストリームを並べ、アクティブデスクトップに設定し、眺め楽しんでいました。

もの足りなかったのはデザインであり、また展開場所がデスクトップだったため、動作がかなり重いという点でした。どうにかして Java アプレット上を流れるテキストデータを取得し、オリジナルのコンテンツ上で表現したいと挑戦し始めたのが、制作の原点です。

時は過ぎ、XML といった中間データが活躍する時代になりました。「Google zeitgeist」をはじめ、年間キーワードランキングの発表といった検索キーワードを主役とした企画が各種検索エンジンを運営するサイトで目立ち始めています。そういったコンテンツの中で、検索キーワードを RSS 形式で提供しているサイトを見つけました。それが、今回のサンプルで使用した「goo ウェブ検索結果」です。

このデータソースを採用するに至った理由は、その更新頻度にあります。下記ページへアクセスし、リロードしてみると、そのたびに内容が完全に入れ替わっていることが確認できます。また、取得できるデータの数が 20 件と適当だったこと、加えて提供形式が XML だったため簡単に作れるという点が魅力的でした。

アクティブデスクトップに設定されたサーチストリームの集合



URL台 **goo ウェブ検索結果**
http://search.goo.ne.jp/rss/newkw.rdf

goo ウェブ検索結果では XML 形式でデータの取得が可能

```
-<rdf:RDF xml:lang="ja">
 -<channel rdf:about="http://search.goo.ne.jp/rss/newkw.rdf">
    <title>gooウェブ検索 最新キーワード</title>
    <link>http://search.goo.ne.jp/web.jsp</link>
    <description>gooのウェブ検索で今まさに検索されているキーワード情報を発信します。</description>
    <dc:language>ja</dc:language>
   -<items>
     +<rdf:Seq></rdf:Seq>
    </items>
  </channel>
 -<item rdf:about="http://search.goo.ne.jp/web.jsp?MT=%B3%AB%C0%AE%B9%E2%B9%BB&from=newkw_rss">
    <title>開成高校</title>
   -<link>
      http://search.goo.ne.jp/web.jsp?MT=%B3%AB%C0%AE%B9%E2%B9%BB&from=newkw_rss
    </link>
    <dc:date>2005-12-07T12:11:28+09:00</dc:date>
```

検索キーワードとは、その人にとって、まさに今、知りたい情報への手がかりです。これらの集合体をプレーンなテキスト（HTML 上）で眺めるのであれば Flash を用いる意味はありません。日本語書体の美しさを全面に出しつつ、それぞれのキーワードが持つ深みをダイナミックに伝える表現として 3D 空間を、また、見知らぬ他人が持っている知識への欲求（検索語句）をぼんやりと眺め、適度な頻度でそれらについて考えてみたいという思いから、スクリーンセーバー上での展開を基本としました。

デザイン
インターネットの世界を表現するインターフェース

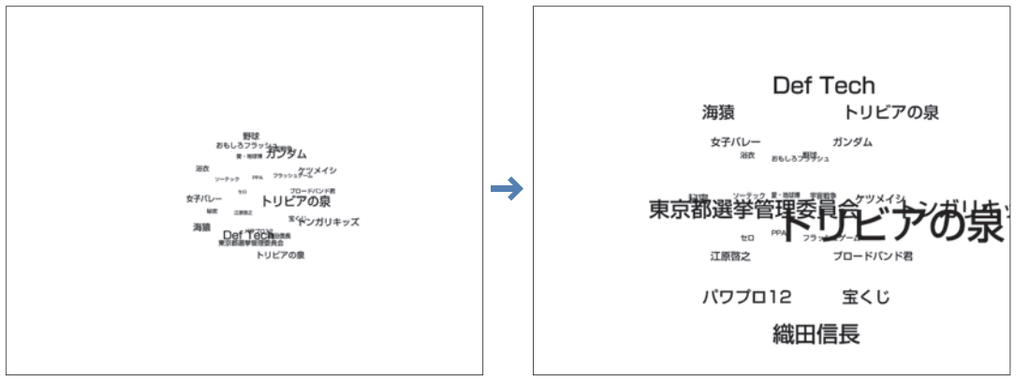
サーチストリームをベースにインターフェースを考えていくことにします。ただ、工場のラインのように右から左へダラダラと流れるサーチストリームは、強弱がなく見る人にとっては平淡に思えるでしょう。ここでは、「検索行為＝もっともインターネット的な活動」という前提にもとづき、インターネットの世界を感じることができるデザインにこだわりました。

3D であふれる期待感を表現

最初のプロトタイプでは、検索ボタンをクリックしたときのユーザーの心情は期待感でいっぱいであるという理由からキーワードを球体状に形成し、心臓のようにドクドクと鼓動する 3D インターフェースを目指して取りかかりました。奥に配置され小さく見えているキーワードと、手前に大きく表示されているキーワードがマウスの座標位置に従って回転し、切り替わる仕様です。

20 件の検索キーワードを新しく取得した 20 件と切り替えるタイミングは画面上の何もない空白部分をクリックする、もしくはリフレッシュといったラベリングのボタンを配置し、それをクリックすることで対応しました。しかし、最終形態として、眺めるだけで操作を必要としない形を望んでいたので、この点がどうしても問題でした。つまり、ボツです。

球体状のキーワード群が回転しながら切り替わる



スピード感を重視

「新しいキーワードを自動で取得し、操作いらずで眺めるだけで済むインターフェース」という条件から最終的にたどり着いたのが、奥から手前にズームする表現です。インターネットの活性を感じてもらえるよう、テンポよくスピード感を重視しました。ダイナミックに次々と迫ってくるド迫力です。さらに、インターネットという不思議な空間へ飛び込んだ錯覚を与えたく、真っ白いキャンパス内にテキストだけが飛び交うシンプルな展開を狙いました。これは、たとえれば『マトリックス』でネオとトリニティーが向かい合い、銃が大量にズコーン！　と現れるシーンのイメージです。

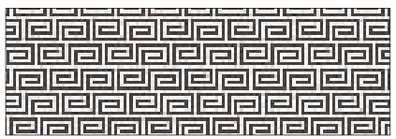
マトリックスバージョン

検索キーワードとしての表情

だいぶイイ感じになってきましたが、何点か気に入らない箇所がありました。まず、白地のキャンパスと単色（黒）で塗りつぶした検索キーワードのコントラストが強すぎるため、飛び交うキーワードを眺めていると思わず酔い止め薬が欲しくなる具合の悪さです。この問題点は、単純に明度を下げることで解決としました。

次に、飛び交うテキスト自体に装飾がほどこされておらず、あまりにシンプルすぎてキーワードの持つ深みが伝えきれていないと感じました。検索する人はそれぞれ、入力したキーワードに対して意味や想いを込めているはずです。ただの単語をテキストとしてそのまま見せるのは NG として、ここではテキストに「脳のしわ」のようなテクスチャーをマスキングし、その背景を表現することにしました。

脳内をイメージするテクスチャー

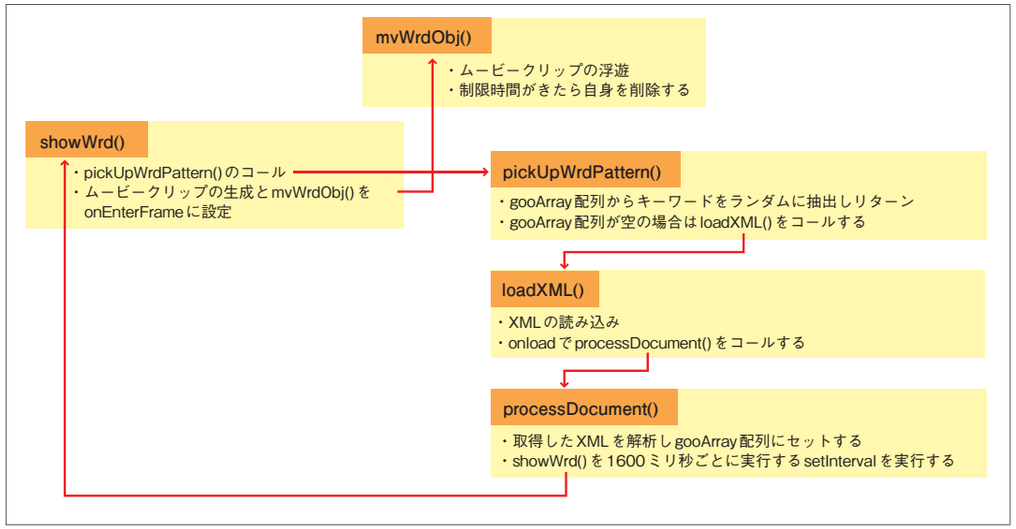


検索キーワードにテクスチャーをマスキング

スクリプト
検索行為の不安定さを表現する

ソースコードはたいへんシンプルで、5 つの関数で全体が構成されています。以下が全体の関係図になります。

5 つの関数の関係図



まずはもっとも重要である一連の動作をキックする記述とその中で全体の展開スピードを制御している部分について解説します。これは、5 つの関数が記述されているフレームの最後の行（初回）と関数 processDocument() 内（2 回目以降）の 2 カ所に記述されています。

SOURCE showWrd() の実行

```
// インターバルの開始
setTimer = setInterval(this, "showWrd", 1600);
```

この部分がキーワードを 20 件ずつ取得し、次々と表示していく関数 showWrd() を 1.6 秒ごとに実行する記述になります。最後の引数（1600）が、全体の展開スピードを制御する部分です。JCT という作品の体感は最後の引数の値次第で激しく変化します。

サーチストリームにもあてはまることなのですが、そもそも、すべてのキーワードを目で追うことは困難です。値を調節し、すべてのキーワードを読ませることも可能です。しかし、たった今通り過ぎていったキーワードを見逃してしまうというストレスは、大量な情報に埋もれている現代の日常生活を皮肉る意味で重要な体験なのではないかと考えました。そこで、「見逃してしまう機会」がたまに発生しうる値として、「1.6 秒」を設定しています。

また、このときの調整時に「JCT」という名称にしました。これは、「接点」「連接」「合流」「連結」といった、つながりを意味する英単語（junction）を 3 文字に省略したものです。

○ さまよう感の演出

検索キーワードとは、ほとんどの場合「その人にとって知らないこと」であると考えられます。また、よほどの検索の達人でない限り、知りたいことをピンポイントで検索できることはないでしょう。検索行為は不安定であり、その不安定さを表現する意味でフラフラ〜っとさ迷い、近付く演出を入れています。その部分が関数 mvWrdObj() です。

SOURCE mvWrdObj()

```
//------------------------------------------
// オブジェクトの浮遊
//------------------------------------------
function mvWrdObj() {
    // テクスチャーの移動
    if (this.death%2 == 0) {
        this.bg_mc._x += 1;
        this.bg_mc._y += 1;
    } else {
        this.bg_mc._x -= 1;
        this.bg_mc._y -= 1;
    }
    // 現在の XY 座標へ反映
    this._x += this.vx;
    this._y += this.vy;
    //
    this.range = 100/this.death*40;
    this._xscale = this.range;
    this._yscale = this.range;
    //
    // ゆれ幅
    this.vx = this.vx+(Math.random()*50-26)/this.range;
    this.vy = this.vy+(Math.random()*30-16)/this.range;
    //
    // 命を削る
    if (--this.death<0) {
        this.removeMovieClip();
    }
}
```

まず検索キーワードを内包する MC には death という属性が attachMovie() による生成時に与えられます。読んで字のごとく「死」ということで、このプロパティは画面上にその検索キーワードが存在する時間を内容として持ちます。つねにデクリメントされ、最終的には removeMovieClip() されます。この値が 0 以上のときに実行されるのが、中間に記述されている、さ迷う演出部分のスクリプトです。

また、関数内の上部にある「// テクスチャーの移動」以下の条件文では検索キーワードにかけているマスク画像（テクスチャー）を１ピクセル単位で移動しています。毎フレームごとに動かすことで、マスクが固定されず、画面手前に迫ってきた際に検索キーワード内で何かチロチロと動いている印象を出しています。これは、前述の「脳のしわ」の意味をさらに強調するためのアクセントです。

○フィニッシュ

この作品はスクリーンセーバー形式にすることで完結します。職場で働く私がタバコを吸いに休憩へ行き、戻ってきたときに目に飛び込んでくる、といった利用シチュエーションを想定して作ったからです。

スクリーンセーバーにせず、インターネット上で公開する場合には関数 loadXML() 内の XML 読み込み部分をコメントし、以下の１文を追加します。

SOURCE loadXML() 内の XML 読み込み部分をコメント

```
//gooXML.load("http://search.goo.ne.jp/rss/newkw.rdf");
gooXML.load("./goo.xml");
```

同時に、関数 pickUpWdPattern() 内の条件文を以下のように変更し、PHP プログラムを中継するようにします。

SOURCE pickUpWdPattern() 内の条件文を変更

```
if (gooArray.length<1 || gooArray == undefined) {
    clearInterval(setTimer);
    myLoadVars = new LoadVars();
    myLoadVars.onLoad = function(success) {
        loadXML();
    };
    myLoadVars.load("./getGooRDF.php");
    //
    n = 1;
    return false;
} else {
```

Flash のセキュリティポリシーの関係上、goo が提供する newkw.rdf はサーバに上げた途端、直接読み込むことができなくなります。このため、自分のサーバ内にプロキシとして働くプログラム（getGooRDF.php）を用意し、このプログラム経由で XML を取得する仕様に変更する必要があります。

PHP プログラム内（以下）では、Flash の代わりに XML を読み込み、goo.xml としてファイルの書き出しを行っています。

SOURCE getGooRDF.php

```
<?php
$URL="http://search.goo.ne.jp/rss/newkw.rdf";

$aurl = fopen($URL,"r");
```

次のページに続く

```
while(!feof($aurl)){
    $xml .=fgets($aurl,4096);
}
fclose($aurl);

$fp = fopen("./goo.xml", "w+");
fseek($fp, 0);
flock($fp, 2);
fputs($fp, $xml);
fclose($fp);

echo "phpResult=true";
?>
```

えらそうに講釈させていただきましたが、実際のモノを見ると課題がまだまだあると思われるでしょう。スクリプトレベルではテクスチャーによるマスキングを追加した時点で動作が重くなっています。また、完成直前にはデータソース自体にも不満を持ってしまいました。

Excite のサーチストリームでは「フィルターあり」と「フィルターなし」の２つのタイプが用意されています。「フィルターなし」版では、検索で使用されたキーワードに対して検閲をかけずにロウデータが流れます。こちらのモードではエログロなワードが大半を占め、猟奇的なオカルトワードなどにも遭遇します。インターネットという場では、人は内面をさらけ出す傾向にあると思います。とすると、みんなの心の奥底にあるドス黒い何かをのぞき込んでいるような「フィルターなし」版こそインターネットの姿を象徴しているのではないでしょうか。

goo の「ウェブ検索結果」では健全的なキーワードにしか遭遇せず、そういった点ではインターネットを俯瞰しきれていないのではないかと思います。

TIPS SWF からスクリーンセーバーへ

私は、データプッシュ型のコンテンツにはスクリーンセーバー上での展開が一番向いていると考えます。一度、設定してしまえば、あとは適度な頻度でふれあうことができ、アクセスいらずの身近さがあるからです。そういったコンテンツを Flash で作ってしまえば、よけいな操作を要求することなく、必要な情報だけを配信する作品が簡単に実現できます。

SWF 形式からスクリーンセーバーを作ってくれるソフトはたくさんありますが、無料という条件から２つほど紹介しておきます。

● Light Saver

たいへん優秀な、お気に入りソフトです。Flash Player 8 を対象に書き出された SWF でも取り込め、どんな形式でも「.src」の実行形式に出力してくれます。背景の透明度を変えるといったカスタマイズも可能です。

URL http://www.forest.impress.co.jp/lib/dktp/wpprscr/scrsvr/lightsaver.html

● fla:ver

Mac 版のスクリーンセーバーも書き出すことができます。さらに高機能な有料の Professional Edition も用意されています。

URL http://flaver.jp/