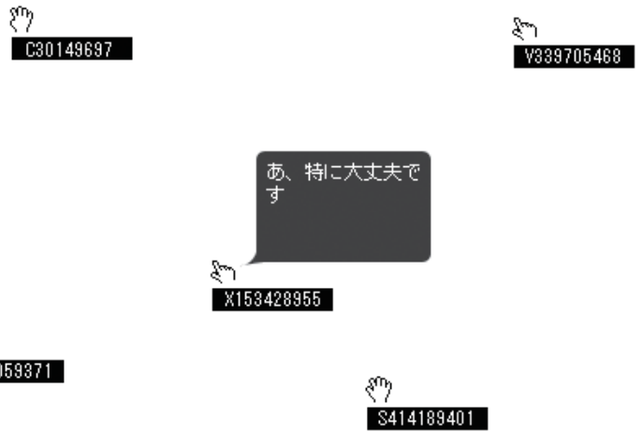


ICQnow

Creator
さうなまん（モイモラ）

FM2、JavaScriptとの連携

リアルタイムにユーザーを把握する。
Flash Media Server 2（FMS2）、JavaScriptとの連携による
モニタリングツール。

Title
ICQnow

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

Archive
icqnow.zip

File
08

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～デザイン
本当のデザイン評価を手軽に得たい

「ICQnow」は、サイト内を巡回しているユーザーのマウス座標をリアルタイムに取得しステージ上にカーソルとして表示するモニタリングツールです。サイト内の HTML テンプレートに JavaScript を埋め込むことで、現在どれぐらいの訪問者がどのページを閲覧し、どのような行動をとっているのか、といった情報をリアルタイムに把握できます。

また、このモニター画面から特定のユーザーを選択し、その人が現在閲覧しているページ上部へメッセージを表示することができます。メッセージを受信したユーザーが返信するとモニター画面上に吹き出しが表示されます。さらにパワーアップすれば、ユーザビリティのテスト用ツール、または EC サイト内でのリアルタイム接客ツール、などといった使用が考えられます。

この章では、Flash Media Server 2（以下、FMS2）と JavaScript を使った連携を中心に全体の構成を解説します。

ユーザーのマウス座標をリアルタイムに取得

ICQnow は、Flash Communication Server（FCS）が登場したときにぼんやりとわきあがったアイデアが原点になっています。当時から、私は Web の制作会社に勤める Web デザイナーで、主にコーポレートサイトの構築などに従事していました。サイトストラクチャーやトップページのデザインなどを担当ディレクターに見せると、利用者の視点でのデザイン評価や設計の見直しが必ず行われます。

そこでは、「ここはボタンとして目立たないから色を強めよう」「この後は製品情報に遷移するはずだからやっぱりリンクを置こう」「初心者ユーザーの訪問が多いだろうからプルダウンのメニューはやめたほうがいいと思わない？」などといった意見を熱くぶつけ合います。しかし、それらのほとんどは乏しい経験上の想像であり、ユーザーテストなどによって立証された事実にはほど遠いものでした。そのためか、ある程度仕上がった成果物をクライアントから突っ込まれても立ちうちできない脆さがありました。本格的なユーザーテストは費用や時間の関係で毎回はできませんし、裏付けのデータも準備できません。

なんとか本当のデザイン評価を手軽に実証できないかと考えたとき、FCS の“リアルタイムに情報をやり取りできる”という特徴に目を付けました。どのページにいるユーザーに対しても、ユーザーのマウス座標を監視して行動を把握できれば、自分のデザインの裏付けが取れると考えたわけです。

結局、そのときはアイデアの実現には至りませんでしたが、2005年12月に発売されたFMS2の価格体系を見て、実現の可能性を感じました。FMS2 の Developer Edition は無償で利用でき、帯域制限がなく、そして最大同時接続数が5から10に増えていたのです。10という数は決して多くありませんが、トラフィックの少ないサイトへこの仕組みを導入するのであれば十分な数といえます。

FCS と FMS2 の製品概要

製品名	エディション	価格	最大同時接続数	最大使用帯域
Macromedia Flash Communication Server MX1.5	Developer Edition	無料	5	0.25Mbps
Flash Media Server 2 Developer Edition	Developer Edition	無料	10	無制限

マウスのカーソルをメインに

マウスの軌跡をリアルタイムに反映するモニター画面には、まずはカーソルだけを配置してみました。

しかし、これだとどのカーソルがどのユーザーなのか判別できないため、直下にユニークな ID を加えました。また、ユーザーがサイト内でページ遷移した場合にそのことを通知するアニメーションを発生させたかったので、カーソルの形状を普通の矢印から指に変えています。

さらに、カーソルをロールオーバーすると、そのユーザーについての詳細情報を表示するようにしました。各アイコンの意味は以下のとおりです。

カーソルの変化

カーソルのロールオーバーで表示される詳細情報

JA	System.capabilities.language の値
WINDOWS 2000	System.capabilities.os の値
1400 PX 1050 PX	System.capabilities.screenResolutionX と System.capabilities.screenResolutionY の値
WIN 8,0,22,0	System.capabilities.version の値
TOTAL 1 PAGES	これまでに閲覧したページの総数
HTTP://VGZH.DTDNS.NET/MT/ARCHIVES/2005/12/POST_456.HTML	JavaScript から送られてくる現在閲覧中のページ URL
5.0 (WINDOWS; JA-JP)	JavaScript から送られてくるブラウザとバージョン
HOWDY HO!	選択したユーザーへメッセージを送信する入力欄

いまさらになりますが、ロールオーバーで表示される詳細情報と該当カーソルを関数 lineTo() でつなげてあげれば視覚的にわかりやすくなったかと思います。また、サイト内の全ページを画像として用意しておき、ロールオーバー時に現在閲覧しているページの該当画像をステージ上の背景に設定するという機能を加えたかったのですが、挫折してしまったためキャンパスは真っ白になっています。

スクリプト
サイト内ユーザーのリアルタイム追跡

この作品は２つのFlashファイルと１つのJavaScriptファイルにより構成されており、大きく２つの機能に分けることができます。１つはマウスの座標をリアルタイムに監視する部分であり、もう１つは選択したユーザーとメッセージをやり取りする部分です。この２つのメイン機能に分けて解説していきます。

しかしその前に、必要となるFMS2の準備から行いましょう。

○ FMS2の準備

FMS2を無償で利用できるDeveloper Editionは以下のページからダウンロードできますが、事前にマクロメディアのアカウントであるメンバーシップ登録が必要となります。ログイン後、サーバ製品のFlash Media Server 2にチェックを付けてトライアル版ダウンロード（Windows版/7.17MB）を入手します。

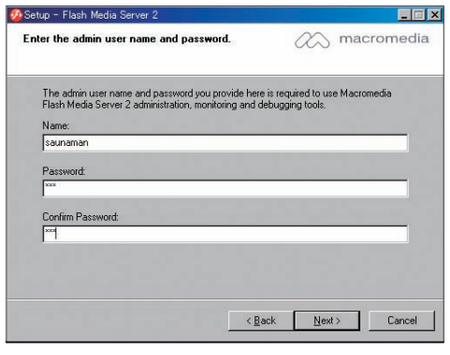
URL抽 **Macromedia Japan—カスタマーサービスセンター**
https://service.macromedia.co.jp/membership/membership_login.asp?Parameter=FC_T

ダウンロードしたインストーラ（FlashMediaServer2.exe）を実行すると、インストールが開始します。インストール中に入力が要求される箇所について、説明しておきます。

● IDとパスワード

まず、FMS2の管理画面へアクセスする際に必要となるIDとパスワードを設定します。自分が覚えていられる組み合わせであれば何でもよいでしょう。管理画面へは、FMS2インストール後にスタートメニューから「プログラム」→「Macromedia」→「Flash Media Server 2」→「Management Console」でアクセスできます。

IDとパスワードを入力

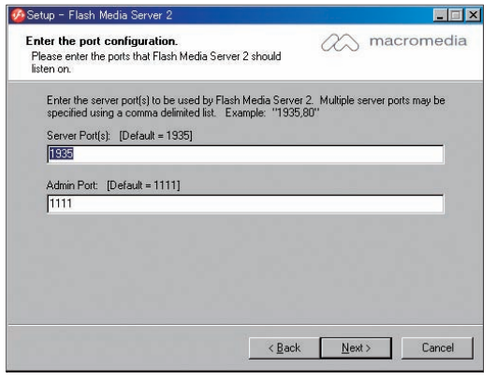


●サーバポート

次に、サーバアプリケーションであるFMS2が通信に使用するポート番号を設定します。Server Portのデフォルトは1935番、Admin Portのデフォルトは1111番です。複数のポートを指定したい場合は、カンマ区切りの入力とします（たとえば1935,8080,80）。

FMS2は、ここで指定されたポートを使ってクライアントとの接続を確立させます。これらのポートを使ってもファイアウォールやプロキシサーバ越しのユーザーと接続ができない場合には、最終的にHTTPトンネリングによる接続を試みます。ポート番号はサーバの環境に合わせて設定してください。通常はデフォルトのままでよいと思われます。

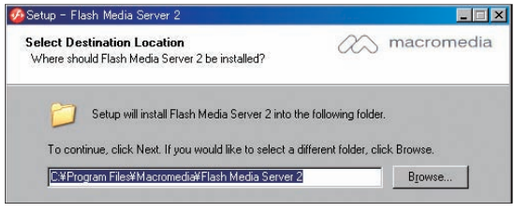
使用するポート番号の指定



●インストールパス

FMS2本体をインストールするパスもデフォルトのままでよいでしょう。ここで指定したディレクトリ内のapplicationsフォルダ以下がFMS2アプリを定義する重要な場所となります（たとえば、C:¥Program Files¥Macromedia¥Flash Media Server 2¥applications¥以下）。FMS2を使ったアプリケーションを作成するたびに、ここで新規フォルダを作成する作業が発生します。

FMS2本体をインストールするパス



インストールが完了したらFMS2アプリを定義するディレクトリへ移動し、「icqnow」という名前の空フォルダを作成します。これでFMS2の準備は完了です。

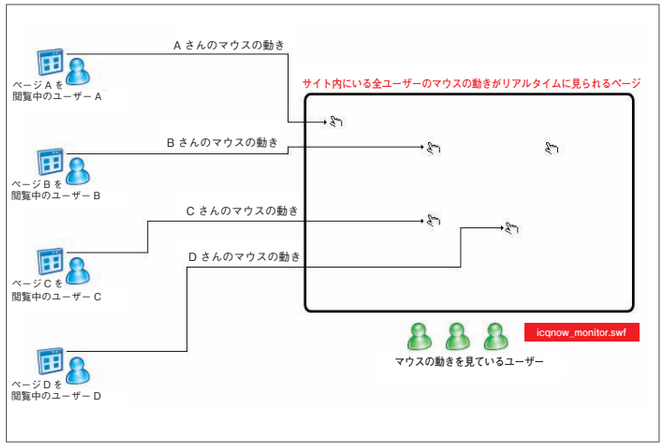
○ マウス座標のリアルタイム監視

それではマウスの座標位置をモニター画面へリアルタイムに送信するパートから解説します。この部分での展開は、右図「マウス座標監視のイメージ」になります。

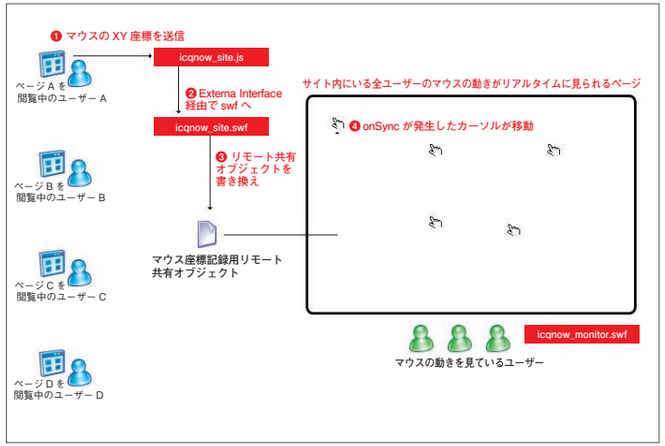
サイト内を巡回しているユーザーが４人いるとして、それぞれが見ているページ内で自分のマウスを動かすたびに、モニター画面である icqnow_monitor.swf ではカーソルがリアルタイムに移動します。

これを実現するための裏の仕組みが、右下図「マウス座標監視のフロー」です。これは、A というページを閲覧しているユーザーのマウス座標が送信されている例です。

マウス座標監視のイメージ



マウス座標監視のフロー



ページ A には icqnow.js という JavaScript と icqnow_site.swf というクライアント用の SWF をインクルードさせます。icqnow_site.swf は縦横 1 ピクセルの大きさで、スクリプトのみ記述されています。これらはページ A に限らず、マウス座標をモニタリングしたいすべてのページに共通して埋め込ませるファイルとなります。私の場合は、Movable Type を使ったブログのテンプレートにこれらの要素を記述し、リビルドすることで、ラクに埋め込んでいます。

● マウス座標監視のフロー

以下、先の図（マウス座標監視のフロー）の番号にそって処理を見ていきます。

❶ マウスの XY 座標の送信［icqnow.js］

ページ A がリクエストされると、window の onload 時にマウス移動でイベントが毎回発生するよう、初期化の作業が行われます。このイベント内で実行させるのが関数 sendInfoToFlash() で、引数にはマウスの X と Y 座標、閲覧中のページ URL、ブラウザ名とバージョンをわたします。

❷ ExternalInterface 経由で SWF へわたす［icqnow_site.swf］

icqnow.js から実行している sendInfoToFlash() の本体は Flash 内で定義されています。外部の JavaScript からシンプルに呼び出せるのは ExternalInterface クラスを使って相互の通信を実現しているためです。ExternalInterface クラスのおかげで Javascript と Flash 間のやり取りは本当に楽になっています。使用するにあたって、ここでは以下の３行を記述しているだけです。

SOURCE ExternalInterface クラスの使用

```
import flash.external.ExternalInterface;
import mx.utils.Delegate;
ExternalInterface.addCallback("sendInfoToFlash", this, sendInfoToFlash);
```

この後、「function sendInfoToFlash(x, y, url, bname, bversion) {}」といった感じで、通常どおりの関数宣言を行っています。これだけで外部の icqnow.js から呼び出すことが可能となります。

❸ リモート共有オブジェクトの書き換え［icqnow_site.swf］

この各ページに埋め込まれる icqnow_site.swf では、何よりも先に FMS2 アプリケーションである「icqnow」への接続を行います。「icqnow」とは前述のインストール完了後に定義したフォルダの名前です。

接続成功後、getRemote 関数を使ってサーバ上の共有オブジェクトを操作できるように準備します。しばらくするとブラウザ上でマウス移動のイベントが発生し、外部から sendInfoToFlash() が呼び出されます。関数 sendInfoToFlash() 内では FMS2 アプリケーション「icqnow」への接続が確立しているか確認し、接続していた場合にはリモート共有オブジェクトである users_so.data.saveInfo プロパティに、引数でわたってきたマウス座標などを書き込みます。

❹ onSync が発生しカーソルを移動［icqnow_monitor.swf］

カーソルが実際に動く画面である icqnow_monitor.swf でも、最初に行う処理は FMS2 アプリケーション「icqnow」への接続となります。

接続完了後、やはり、リモート共有オブジェクトを関数 getRemote() で取得しています。icqnow_site.swf と違うのは、このあとに onSync のイベントハンドラを定義している点です。この onSync イベントハンドラは接続したリモート共有オブジェクトの内容が変更されるたびに呼び出されます。これは、前述した icqnow_site.swf が users_so.data.saveInfo プロパティへマウス座標などを書き込むという行為に該当します。onSync 内では新たに書き込まれた内容を読み込み、要素別に分解します。

users_so.data.saveInfo プロパティにはさまざまな情報が入っていますが、ほとんどがカーソルをロールオーバーすると表示される詳細情報なので、単にカーソルの MC へ分解した要素をわたすだけとなります。重要なのは、カーソル直下に表示されるユニーク ID です。この ID と同じ名前の MC がステージ上にない場合は attachMovie() によるカーソルの新規作成となります。JavaScript からわたされたマウスの X と Y 座標は、最終的にこの MC が受け取り、カーソルをその位置へと移動します。

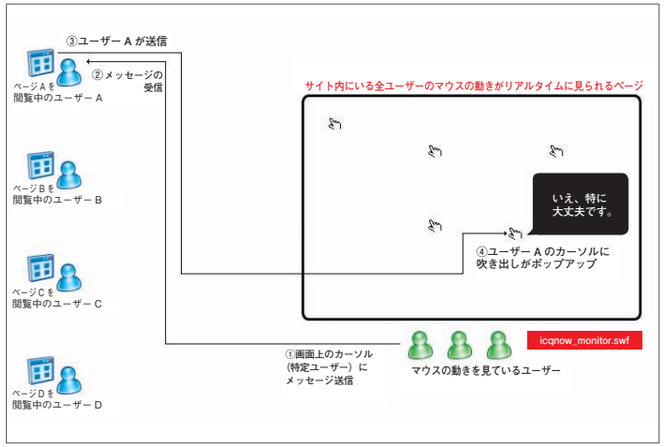
○ 選択ユーザーとのメッセージのやり取り

続いて、特定のユーザーとメッセージをやり取りする部分について解説します。

マウスの座標位置を監視するだけで作品を完結してもよかったのですが、それだと、ただのストーキングツールになってしまうため、メッセージング機能を追加しました。

この機能の全体概要は右図のとおりです。

メッセージ送受信のイメージ



メッセージング機能を実現するために、受信メッセージと返信用テキストボックスを内包する div タグを以下のような形で全ページに記述しておきます。

SOURCE 受信メッセージと返信用テキストボックスの記述

```
<!-- メッセージ送受信の要素を識別するためのタグ -->
<div id="icqnow_msg" style="position:absolute; visibility:hidden;">
</div>
```

● メッセージ送受信のフロー

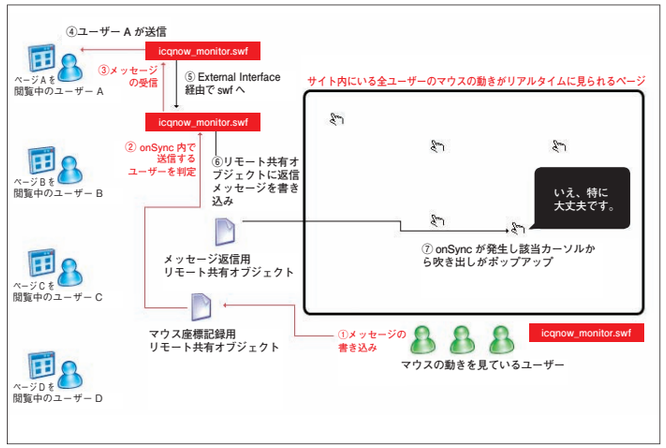
メッセージの送信は、カーソルが表示されている icqnow_monitor.swf から始まります。まず、このモニター画面を見ているユーザーがステージ上にあるカーソルのどれかをマウスオーバーします。すると、そのカーソルの持ち主であるユーザーの詳細情報が画面左上に表示されます。この詳細情報の下部にはテキストボックスがあり、ここに送りたいメッセージを入力し送信します（上図の①の部分）。

このメッセージはサイト内の全ユーザーに送られるのではなく、選択したカーソルの持ち主だけに届きます。届いたメッセージはそのユーザーが現在見ているページ上に返信用のテキストボックスと一緒に表示されます（前ページの図の②の部分）。

返信内容を入力し送信すると、送信したユーザーに該当するカーソル上に吹き出しがポップアップされます。吹き出し内に表示されるのが返信されたメッセージです。

これらの一連の展開は右図のとおりです。以下、この中の番号にそって処理を見ていきます。

メッセージ送受信のフロー



❶ メッセージの書き込み［icqnow_monitor.swf］

テキストボックス内にメッセージを入力して送信ボタン（HOWDY HO!）を押すと、すでに接続済みのリモート共有オブジェクトへ２つの情報をセットします。

SOURCE ユニーク ID とメッセージテキストの情報をセット

```
users_so.data.nowid = info_mc.nowid;
users_so.data.dirmsg = info_mc.msg.text;
```

セットする内容は、どのカーソルかを示すユニーク ID と入力したメッセージの内容です。

❷ 送信するユーザーの判定［icqnow_site.swf］

icqnow_site.swf には、メッセージ受信用の onSync イベントハンドラが定義されています。

SOURCE onSync イベントハンドラ

```
users_so.onSync = function() {
   if (this.data.nowid == id) {
        this.data.nowid = "";
        displayMsg(this.data.dirmsg);
   }
};
```

ここで行っているのは、たった今変化のあったリモート共有オブジェクト内のユニーク ID と自分のユニーク ID が一致しているかどうかのチェックです。一致していた場合は、自分宛てのメッセージということで JavaScript へメッセージをわたす関数 displayMsg() を実行します。

SOURCE displayMsg()

```
function displayMsg(msg) {
    ExternalInterface.call("showMsg", msg);
}
```

ここでもJavaScriptとのやり取りにExternalInterfaceクラスを使用しています。前述のパターンではJavaScriptからFlashを呼び出していたのに対し、上記displayMsg()内ではFlashからJavaScriptの呼び出しになっています。

❸ メッセージの受信［icqnow.js］

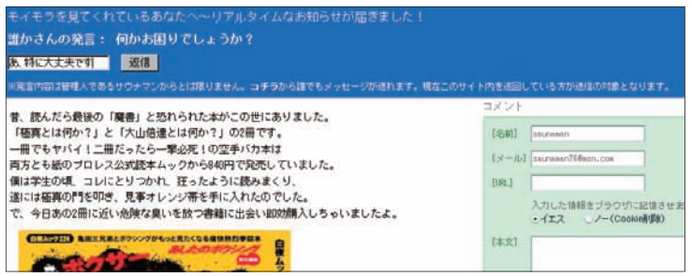
icqnow_site.swfから呼び出されるshowMsg()は、受信したメッセージと返信用テキストボックスを表示するための関数です。この2つに必要となるHTMLコードをicqnow_msgというIDのdivタグ（innerHTML）にセットし、visibilityをvisibleにすることで画面上に表示させています。

❹ ユーザーAの返信

❺ExternalInterface経由でSWFへわたす［icqnow.js］

返信用テキストボックスにメッセージを入力し、返信ボタンをクリックするとExternalInterface経由でicqnow_site.swf内のsetJSmsg()を実行します。わたすパラメータは返信用テキストボックスに入力されたメッセージそのものです。

返信内容を入力中



❻ リモート共有オブジェクトに情報を書き込む［icqnow_site.swf］

返信メッセージを受け取ったicqnow_site.swf内のsetJSmsg()ではメッセージの返信用に用意したリモート共有オブジェクトjsmsg_soへ2つの情報をセットします。

SOURCE リモート共有オブジェクトに情報をセット

```
jsmsg_so.data.jsmsg = jsmsg;
jsmsg_so.data.nowid = id;
```

内容は、返信したメッセージと自分がどのカーソルかを示すユニークIDです。

❼ 吹き出しの表示［icqnow_monitor.swf］

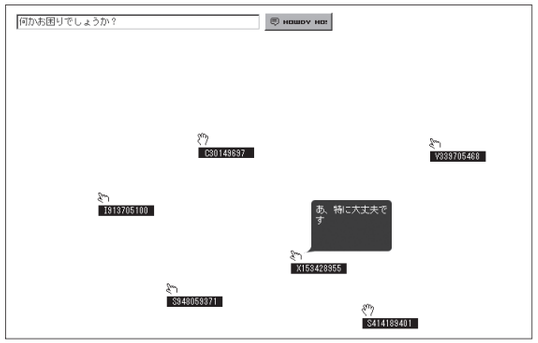
アンカーであるicqnow_monitor.swfは事前に返信リモート共有オブジェクトjsmsg_soを取得し、onSyncイベントハンドラを定義しておきます。

SOURCE onSyncイベントハンドラの定義

```
jsmsg_so.onSync = function() {
    if (this.data.nowid != undefined && this.data.jsmsg != undefined) {
        var msgMc = _root[this.data.nowid].msg_mc;
        msgMc._visible = true;
        msgMc.msg.text = this.data.jsmsg;
    }
};
```

ここではリモート共有オブジェクト内に記録されたユニークIDの該当カーソルをターゲットに吹き出しを表示させ、その中にメッセージをセットしています。

サイト内のユーザーにメッセージを送信



○ 今後の展開

そもそも、これを作ったきっかけは自分のデザインの正しさを証明するためだったのですが、いろいろやっていたら、それ以外の展開ができそうな気がしてきました。

まず、毎回ブラウザを開いてモニター用のページへアクセスするのは面倒なので、「GIZMO」というアプリケーションを使ってデスクトップアプリに変換しました。これにより、デスクトップ上でマウス移動の様子がモニタリングできるようになりました。このようなデスクトップ上での展開であれば、サイト訪問を通知する音をさまざまなバリエーションで鳴らせるだけでもよさそうです。音で聞くアクセスログといったところでしょうか。ある日モニタリングするための画面を開きっぱなしにしたまま朝を迎えると、画面上に残っている大量のカーソル群がおもしろい形を作っていました。花火のようなイメージです。

また、散在するカーソルを眺めているとマウス移動ひとつにしても、何か性格や傾向が出ているような気がしました。これを1日単位で記録し続ければ、パッと見ただけで、その日のトラフィックが把握できるサイコロジー的視点のログツールができるかもしれません（妄想ですね…）。

今回のリアルタイムという方向で攻める以外にも、アクセスログの見せ方にはまだまだ可能性があるのではないでしょうか。