# Hyperfridge.com

A Thesis
in TCC 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

By

Douglas C. Ross

March 26, 2002

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses.

Signed_____

Approved_____ Date_____
Technical advisor - Dave Evans

Approved_____ Date_____
TCC Advisor - Bryan Pfaffenberger

## PREFACE

I believe that people learn the most when they must acquire knowledge to reach a goal in which they have a personal interest. The idea for Hyperfridge.com came to me around three o'clock in the morning almost a year ago. I have enthusiastically invested my energies to the end of realizing that moment of inspiration. I have learned more in the process than could be contained in such a modest thesis as this.

I would like to thank my over-indulgent technical advisor, Prof. Dave Evans for allowing me the freedom to pursue my artistic vision in the guise of one doing a technical engineering research project.

I would also like to thank Prof. Brian Pfaffenberger, my TCC advisor. His particular expertise regarding Web technology provided me with a source of genuine sympathy in my labors.

And, thanks to Scott Ruffner and Christina Jackson, CS Dept. System staff, for their advice and help with "Atlas", the CS Dept. server.

## TABLE OF CONTENTS

## ABSTRACT

Magnetic Poetry was the brainchild of artist/musician Dave Kapell. Magnetic Poetry is a social game. Several hundred thin vinyl magnets, each with a single word on it, are placed on some metallic surface in a communal area within a household (often the refrigerator door). Over time, housemates and guests arrange the magnets into short poems or haikus, records of epiphanies, erotic suggestions etc.

Hyperfridge.com recreates this game on a global scale via the World Wide Web. People who visit the Hyperfridge website can drag and drop virtual magnets on a virtual refrigerator door. Their expressions (poems) remain for all future visitors to read or deconstruct for words components to make their own poems. The positions of all the magnets are stored in a database.

On Hyperfridge, visitors may add new words. New words are accepted at a maximum rate of once every tenth of a sidereal day (8,616 seconds, or roughly 2 hours and 40 minutes). Each new word replaces a seldom used word, keeping the total number of word magnets balanced at 500. A spell checker responds to unknown words by suggesting alternatives, but, if the visitor insists, Hyperfridge will accept almost any word (under 26 letters long).

A major design goal for the Hyperfridge project was that it be capable of sustained autonomous operation. Hyperfridge went live on the web on Valentine's Day 2002 and has required only slight adjustments to its internal operations since then.

# 1. INTRODUCTION

This thesis relates in detail the motivation behind, and the methodology of the development of Hyperfridge.com. Hyperfridge.com is a work of conceptual art, idea art, a statement about the nature of human interaction on the World Wide Web. It is refrigerator magnet poetry on a global scale built with a cross-browser compatible DHTML "drag and drop" front end to a MySQL database through server side PHP scripts.

## 1.1 Problem Definition

### *1.1.1 Cyber-culture*

The Internet is a place of fleeting, anonymous yet meaningful human encounters and delayed communication. Anonymity seems to be a sine qua non of casual human interaction on the Web. There is defensiveness about "screen names" and "login ids", aliases people choose like masks at a ball for engaging in Web Forums or sharing Multi-User Dimensions, as expressive as they are obfuscating. There is some tension between the desire to project oneself yet remain unseen.

Of course, we also find forthrightness on the Internet. The biographical Web pages of University faculty, for example, are candid, and we see extremes of mental exhibitionism in the phenomenon of online journals, filled with personal details that invite a kind of intellectual voyeurism. Yet, even in these cases the individuals are hidden. Their personae have become museum pieces and scarecrows. We see images of them. We read what they have written for us but we do not engage them directly.

In Web Forums, where people post information on specific topics of interest and reply to posts from others, we find conversations frozen in time. Personal exchanges in Web Forums become written records of themselves. Personal dialogs may be perused at leisure by observers who need not have participated in their generation, and again the interaction is delayed and mostly anonymous.

### *1.1.2 Magnetic Poetry*

In 1993, about the same time that Tim Berners-Lee was weaving the last threads of the World Wide Web and Marc Andreessen was hacking out the first release of Mosaic, Dave Kapell invented Magnetic Poetry. He was composing a song. He had written lyrics but wanted to try different arrangements of the words, so he wrote them down on paper, cut them out and started to arrange them on a tray. He sneezed and the pieces flew everywhere. An intelligent creative type, Kapell decided to glue the words onto magnets and organize them on a metal cookie tin.

In time, the word magnets found their way to his refrigerator door. Friends came over and started arranging the magnets, leaving new phrases for others to read. It was fun. Kapell was inspired. He made a few "kits", each with a few hundred thin vinyl word magnets, and sold them at a local art fair. Demand quickly exceeded supply. Dave Kapell is now a multi-millionaire. He has sold hundreds of millions of Magnetic Poetry kits [11].

Each kit contains nouns and verbs and adjectives and a few conjunctions to glue sentences together with. You put them all on your refrigerator door in a random way and wait. Over time visitors to the refrigerator will organize the words into short poems or haikus, records of epiphanies, erotic suggestions etc. Constraint fosters creativity. Being

given only a limited supply of concept expressers inspires some ingenious expressions of concepts. Authors enjoy the mischievous thrill of secretly leaving a poem for others to find, or altering an existing poem to give it an ironic twist.

In Magnetic Poetry we have personal, yet anonymous expression and delayed communication of the same kind we find on the Web. Is not the Web then a perfect environment for Magnetic Poetry? It would invite participation on a previously unimaginable scale.

## 1.2 Review of Current Technology and Existing Websites

### *1.2.1 DHTML examined*

Hyperfridge was developed with DHTML rather than a browser plug-in. The most frustrating and difficult aspect of creating dynamic web pages with DHTML is maintaining cross-browser compatibility [7][19]. To fully understand cross-browser compatibility issues as they apply to DHTML it is necessary to understand the nature of the Document Object Model [19], client-side scripting (in a language like JavaScript) [7], Cascading Style Sheets (CSS) [10], and the history of these technologies throughout the amazingly rapid expansion and exploitation of the World Wide Web since the development and release of the first commercial web browsers in 1994 [18].

The DOM is a platform- and language-neutral application programming interface for client-side scripting languages [20]. It provides a structured (n-ary tree), object-oriented representation of the individual elements and content of a web page document with methods for retrieving and setting the properties of those objects. The DOM also provides an interface for dealing with events, allowing scripts to capture and respond to

user or browser actions, allowing DHTML programmers to make web pages that are truly "dynamic" and interactive.

The client-side scripting language of choice for most DHTML programmers is JavaScript, a lightweight interpreted language with object-oriented capabilities, syntactically similar to Java and C++. The general-purpose core of JavaScript (based on the ECMA-262 standard specified in 1999) [5] is embedded in the major web browsers, Netscape, Internet Explorer and Opera [7].

JavaScript has some curious features. It is untyped. The same variable in a segment of code may for example, be assigned an integer value, then a string value and then become a pointer to an object. Also, JavaScript can recursively interpret itself, (through eval(*code*) statements) and it has associatively indexed heterogeneous arrays.

JavaScript is intended to provide maximum functionality with minimum code. Each character of code is another 8 bits that the client must download. The loosely structured nature of JavaScript allows web programmers to write short slick scripts that download quickly, even on low-bandwidth Internet connections.

Cascading Style Sheets are a means of separating presentation from content in web pages [21]. Håkon Lie proposed the first specifications for CSS at CERN, (the birthplace of the World Wide Web) in the early 1990's [10]. He wanted to allow web page designers to control layout using a "common desktop publishing terminology". Nevertheless, early browsers did not utilize CSS. Instead, developers like Marc Andreessen, (who produced NCSA Mosaic in 1994) chose to introduce new "proprietary" HTML tags. Tags like blink, center and font fused description with content, undermining the free flow of generic information through the Internet. CSS has now been accepted in

principle, although the leading browsers implement the specification somewhat differently and retain their idiosyncratic versions of HTML.

Jakob Nielson, a famous pundit of good taste in web design, calls CSS "one of the greatest hopes for recapturing the Web's ideal" [14 : 81].

To a DHTML programmer the significance of CSS is in how it extends the utility of the DOM. CSS allows us to specify such things as the font, color, spacing and position of text elements and to change those specifications dynamically on the server-side based on user input.

In 1994, The World Wide Web Consortium (W3C) was founded by Tim Berners-Lee, (inventor of the Web) [3], at the Massachusetts Institute of Technology, Laboratory for Computer Science in collaboration with CERN, with support from DARPA and the European Commission. The W3C's mission has been to provide some direction for the development of the Web and maintain interoperability across platforms by encouraging dialog between vendors and recommending standards. In spite of the W3C's efforts, shortsighted competitiveness between Web browser developers soon led to chaos. Standards were blatantly ignored as they implemented proprietary functionality to their products. The so-called "Browser Wars" [18] ensued. Netscape introduced the "blink" tag and Internet Explorer countered with "marquee".

The W3C recommended positioning Web page elements via CSS properties, but rather than comply, Netscape brought out the now infamous "layer" tag (a tag which Netscape 6 will not even recognize). Unprecedented by any standard recommendation, Internet Explorer introduced the "document.all" construct for its implementation of

JavaScript. Everyone trying to make their browser just a little bit fancier was tacking bells and whistles on to a de facto DOM without any consideration for interoperability.

When the Hyperfridge project began in the spring of 2001, Microsoft had about eighty four percent of the market. As of February 2002, more than ninety one percent of people browsing the Internet are using a version of Internet Explorer. Less than five percent are using Netscape (see appendix 2). This is little comfort to the DHTML programmer though. That five percent is measured in many millions of users, and hundreds of thousands of people use Opera as well. Simply ignoring them is unacceptable.

Hyperfridge has been made as universal as possible and works in Netscape 4, Netscape 6 (NS4, NS6), Opera 5 and 6 (OP5, OP6), and Internet Explorer 4, 5 and 6 (IE4+). It works in these browsers on PCs (running Windows and Linux) and on Macintosh machines.

### *1.2.2 Has it been done before?*

A deep search of the Internet for existing implementations of a drag and drop Magnetic Poetry concept yielded a few results. A website called Nitric Interactive (under "Diversions, E-Tiles") [9] has word magnets that are movable in a Java applet. Although Java can interface with a server-side database, the word magnets at Nitric Interactive lose their positional information when a visitor exits the site.

It should be noted that although Java is a ubiquitous technology, it is still considered a "plug-in" and not everyone has it installed for use with their browser (in fact IE6 does not come packaged with Java).

Another example exists at “The Magnetic Poetry Fridge” [6], where the programmer chose to use the Flash 5 plug-in to implement the movable magnet idea. The site is visually appealing but has only 52 word magnets and again, although Flash 5 can be used to interface with a server-side database, the word magnets lose their positional information when a visitor exits the site.

A site called “Wingnut – etc” [2] uses ASP to store user entered data but the interface is only loosely based on the concept of refrigerator magnet poetry and does not involve drag and drop behavior.

The most advanced example was located at “pedram.redhive.com” [15]. This programmer (pedram) chose to use a DHTML client side interface with a PHP and MySQL backend. There was no working demo. Pedram did however provide a link to the project source code. Examination of this code revealed significant differences in approach. Notably, pedram chose to use .GIF files (images of the word magnets) instead of dynamically generated text. This greatly inhibits the reconfiguration ability of the system and is very expensive in terms of user bandwidth.

Experiments with word images at the onset of feasibly testing for the Hyperfridge project demonstrated that a site built in such a manner would require at least ten times the amount of user downloaded data. By using a text-based system the total file size of the Hyperfridge interface may be kept below 57 Kilobytes, and the selection of word magnets can be changed instantly.

Also, pedram’s system of data storage is awkward. The user is required to hit a particular key sequence (Shift – S) to store the positions of the magnets. Hyperfridge is more intuitive. Positional data is stored whenever a magnet is dropped on the screen.

Finally, the DHTML in pedram's example will work only in IE4+ and NS4 on a Windows machine.

Interestingly, pedram found a copy of my thesis proposal while doing an Internet search. He came to my personal website (http://www.cyberdoug.info), and found the link there for people to contact me. We have communicated regarding the differences in our projects and the apparent simultaneity of our independent development of the idea. pedram does not intend to bring his project to full-scale operation on the Web [15].

### 1.3 Overview of Contents of the Rest of the Report

The chapters to follow recount the work that went into building Hyperfridge.com. Presented are some of the difficulties encountered writing the cross-browser compatible DHTML, and explanations of how a degree of universality was achieved in the client-side interface. The workings of the server side PHP scripts in coordination with the MySQL database are also detailed. Further, the visual and contextual aesthetics of the Hyperfridge website will be described. The results, including data collected in the form of recorded word arrangements will be given. The paper will conclude with an assessment of the performance of Hyperfridge.com since becoming operational on the Web and recommendations for improvements in the system.

# 2. METHODOLOGY

## 2.1 The Front End

### *2.1.1 The Chits*

The dragable word “magnets” in the Hyperfridge client side Graphical User Interface (GUI), door.htm, are CSS/HTML objects that, in the drag and drop code, are assigned to a variable named “chit” (see appendix 4). In this document these objects will be referred to as “chits”.

Initial prototypes tested the feasibility of using images, .GIF or .JPG files, to generate the visual display of the chits in the GUI. The images required minimal information: only two colors and straight lines for aliased text. Of the two formats .GIF provided the best compression, resulting in an average file size of 1,600 bits per chit. However, the additive effect of the image files quickly lead to unreasonable download bandwidth requirements.

High-speed Internet connections were on the rise in the United States last year. Still, approximately 86% of home Internet users were connecting at 56Kbs or less.

*Table 1.*

| **Internet Use by Connection Speed (in millions)** | | | |
|---|---|---|---|
| **Connection Speed** | **July 2000** | **July 2001** | **Change** |
| High Speed | 8,003 | 17,703 | 121% |
| 56K Modem | 49,666 | 64,290 | 29% |
| 28.8/33.6K Modem | 24,205 | 15,523 | -36% |
| 14.4K Modem | 5,304 | 3,907 | -26% |
| High-speed access includes ISDN, LAN, cable modems and DSL. | | | |

Source: http://www.netratings.com/

There are 500 words at Hyperfridge.com. 500 * 1,600 = 800,000 bits. 800,000 bits on a perfect 56Kbs connection would take over 14 seconds to download. The JavaScript code and basic document HTML of the Hyperfridge GUI require another 38,000 bits. 500 `<IMG>` tags with attributes (width, height, src, id and style) adds another 800 * 500 = 400,000 bits. The total, 1,238,000 bits, would take over 22 seconds to download on a perfect 56Kbs connection. For the millions of users with 33.6Kbs modems, Hyperfridge would take more than 34 seconds to download.

Imagine waiting 34 seconds for any page to come up on your browser.

Moreover, these calculations assume that the 800,000 bits are a single file. In reality, each of the 500 embedded images necessitates a separate request from the server.

Hyperfridge.com resides on "Atlas", an Apache/1.3.22 server, the same server that holds the Computer Science Department website. Atlas is HTTP 1.1 compliant and can respond to requests in pipelined and non-pipelined mode. Atlas is capable of handling roughly 18 requests per second at a concurrency level of 10 simultaneous requests in non-pipelined mode (see appendix 3). Atlas can process a single request, on average, in 567 milliseconds. 500 separate embedded objects would take roughly 28 seconds to queue up and send. Thus, it would take almost 1 minute to get the interface loaded into a non-pipelined browser at average connection speeds. Granted, this is a worst-case scenario. Most modern browsers use HTTP 1.1 in pipelined mode by default.

The solution to the bandwidth problem was `<DIV>` tags. `<DIV>` (Division) tags were added to the HTML 3.2 specification to facilitate CSS operation within the DOM. `<DIV>` tags are used to create new block level HTML elements with novel qualities and

all compliant browsers recognize them, specifically Netscape 4 and 6, and all versions of Internet Explorer since 4.

Here is an example of a Hyperfridge chit.

```
<DIV id="C1" style="position:absolute;left:169;top:511;width:72;">
  example  
</DIV>
```

100 characters = 800 bits of information. `<DIV>` tag chits are 1/3 of the size of .GIF/`<IMG>` chits, and download in one request with the body of the document. The entire interface hovers around 56,000 bytes, with minor size fluctuations from user input. Thus, Hyperfridge downloads in roughly 8-10 seconds for the average user.

Note that the element level “style” attribute is used to specify the position and width properties of individual chits. In addition, all of the chits have common properties that are specified by a document wide embedded style sheet.

```
DIV { border: 1px solid #000000;
      font-weight: bold;
      font-size: 16px;
      font-family: "times new roman", times, serif;
      color: #000000;
      background-color: #ffffff;
      cursor: default; }
```

This technique saves 720,000 bits.

### *2.1.2 Cross-Browser Compatibility*

The JavaScript code (see appendix 4) that allows users to drag and drop the chits was written to be compatible with the browsers used by over 98% of today's Web exploring citizens (see table 2).

*Table 2. Browser Prevalence*

Fri Feb 1 00:05:02 2002 – Thu Feb 28 23:55:04 2002

| | | | |
|---|---|---|---|
| 1. MSIE 5.x |  | 294401110 | (61%) |
| 2. MSIE 6.x |  | 129724397 | (27%) |
| 3. Netscape 4.x |  | 21486675 | (4%) |
| 4. MSIE 4.x |  | 20422222 | (4%) |
| 5. Netscape comp. |  | 4921851 | (1%) |
| 6. Netscape 6.x |  | 3650624 | (0%) |
| 7. Opera x.x |  | 2224971 | (0%) |
| 9. Netscape 5.x |  | 674237 | (0%) |

Source: http://www.thecounter.com

The code is composed of three event driven functions, ("grab", "drag" and "drop") and fourteen global variables. The first four globals are Booleans and they are initialized when the browser loads the page.

```
var NS6 = (navigator.userAgent.indexOf("Gecko") != -1) ? 1 : 0 ;
var OP = (navigator.userAgent.indexOf("Opera") != -1) ? 1 : 0 ;
var IE = (document.all && !OP) ? 1 : 0 ;
var NS = (document.layers) ? 1 : 0 ;
```

These variables are assigned values (1 or 0 == true or false) by means of JavaScript's ternary operator depending on environmental conditions. The code is checking to see what kind of browser it is running in. "indexOf() is a string operator, here used to search for "Gecko" or "Opera" in the read-only userAgent method of the DOM's navigator object. It is checking to see if it is running in Netscape 6 or the Opera browser. It checks for Internet Explorer or Netscape 4 by confirming the existence of their proprietary, non-standard DOM objects, "all" or "layers". Browser specific code in the "grab" and "drag" functions is run based on the conditions of these Boolean flags.

### *2.1.3 Grab*

The purpose of "grab" is to locate the position of any mouse click event that may occur within the document and determine whether or not the position of that click falls within the area of any of the `<DIV>` objects. Recall that each `<DIV>` has an absolute width specified by an element level "style" attribute (different for each word), and a height specified in the document-wide CSS, (18 pixels, 16px font + 1px border top and bottom).

If a `<DIV>` has been clicked, "grab" assigns a reference of that `<DIV>` to the variable "chit". "grab" also increments "z", a running global total (initially 0), and assigns the value of "z" to the chit's "z-index" property. Thus, any clicked chit appears on top of all other chits. CSS objects are positioned by their upper left corners. "grab" gets these values and assigns them to "Xoff" and "Yoff". It assigns the actual click coordinates to "X" and "Y".

"grab" is divided into three browser specific sections (IE and Opera share much of the same syntax).

For Netscape, “grab” and “drag” both take a single argument “e”. “e” is an object in the NS DOM that holds information about events. In NS4, it is also necessary to specifically register events to be monitored by invoking the “captureEvents” method of the “document” object and passing it the particular methods of the Event object desired. The desired events in this code are “MOUSEDOWN”, “MOUSEMOVE” and “MOUSEUP”.

In NS4 we must loop through and search every `<DIV>` object by hand, as it were, checking to see if the Cartesian coordinates of the click event are within its bounds.

Here is the NS4 specific code from “grab”.

```
if(NS)
{
   X = e.pageX ;
   Y = e.pageY ;

   for(var i = document.layers.length - 1; i >= 0; i--)
   {
      var ischit = document.layers[i] ;

      if(( X > ischit.left )
          && ( X < ischit.left + ischit.clip.width )
          && ( Y > ischit.top )
          && ( Y < ischit.top + ischit.clip.height ))
      chit = ischit ;
   }

   if(chit)
   {
      Xoff = X - chit.left ;
      Yoff = Y - chit.top ;
      chit.zIndex = Z ;
      watchit = chit.id ;
      chitwidth = chit.clip.width ;
   }
}
```

Internet Explorer and Opera both utilize the “document.all” syntax, and share a more convenient and intuitive approach to event handling. NS4 implements a trickle

down event model. The first object with the option to handle any event in the NS4 nonstandard DOM is the root node of the document tree. The event then passes down through all child nodes, being handled as applicable. Events in Opera, IE and NS6 (Level 1 DOM compliant browsers) bubble up the document tree. That is, the initial target of any event in these browsers is the first to handle the event. Then, the event continues up the tree to be handled in turn (when applicable) by that objects parent node objects (see appendix 1).

Here is the Opera and Internet Explorer specific code.

```
else if(IE || OP)
{
   watchit = event.srcElement.id ;

   if(watchit)
   {
      chit = event.srcElement ;
      X = event.clientX ;
      Y = event.clientY ;
      Xoff = chit.style.pixelLeft ;
      Yoff = chit.style.pixelTop ;
      chit.style.zIndex = Z ;

      if(IE)
      chitwidth = chit.scrollWidth ;
      else
      chitwidth = chit.style.pixelWidth + 2 ;
   }
}
```

This code is a bit “slick”. The variable “watchit” is assigned the value of the “id” attribute of whatever object is the locus of the document event. Only the chits (and one hidden object) in door.htm have an “id” attribute. Thus if the user clicks on anything other than a chit, the value of watchit will be null and the subsequent test will fail. Notice the accommodation that must be made for syntactic differences, scrollWidth (non standard) in IE vs. style.pixelWidth in Opera. Also, Opera does not consider the border,

(width 1) to be part of the width of the <DIV> so 2 must be added to the value of chitwidth.

The Netscape 6 specific code is more like the code for IE and Opera than it is like NS4. NS6 is more compliant with Level 1 DOM specification, even more so than IE5. Interestingly, as proof of its perfection, NS6 considers the actual text within the <DIV> to be the target of the click event. So, the event must be caught as it bubbles up to its parent node object, the <DIV>.

Here is the Netscape 6 specific code for "grab".

```
else if(NS6)
{
   if(!e.target.parentNode.id) { chit = e.target ; }
   else { chit = e.target.parentNode ; }

   Xoff = chit.style.left ;
   Yoff = chit.style.top ;
   X = e.clientX ;
   Y = e.clientY ;

   if(chit)
   {
      watchit = chit.id ;
      chit.style.zIndex = Z ;
      chitwidth = (chit.innerHTML.length - 4) * 7 + 16 ;
   }
}
```

Note how the syntax resembles a fusion of NS4, (the "e" object) and IE/Opera, (the "style" object). Also, please excuse the "magic numbers" used to calculate a value for "chitwidth" on the fly. NS6 will not return an accurate value for the width of the chit object. So, this improvised solution calculates a value based on the number of letters of text within. This value remains stable across platforms because the size of the text is specified in pixels in the document-wide Style Sheet.

### *2.1.4 Drag*

"drag" is fairly straight forward. If "grab" was successful then "chit" holds a reference to a `<DIV>` object. "drag" repositions the chit based on the coordinates of the moving cursor, making sure that it not dragged outside of the width or height of the table that defines the perimeter of the GUI. Again, the code is divided into three browser specific sections. The reader should see appendix 4 for a full listing of this code. Here is an example of some of the differences in syntax and DOM implementation that necessitate the specific code sections.

**In NS4:**
```
putX = chit.left = e.pageX - Xoff ;
```

**In IE/Opera:**
```
putX = chit.style.pixelLeft = Xoff + event.clientX - X ;
```

**In NS6:**
```
putX = chit.style.left = e.clientX - parseInt(X) + parseInt(Xoff) ;
```

Note that all event driven functions in the drag and drop code return a Boolean "false". This ensures compatibility with single button mouse platforms like the Macintosh OS. If you hold down the mouse on a Macintosh machine, its default behavior is to pop up a context menu. Returning "false" stops further processing of the mouseDown event, and so prevents the menu, that would otherwise interfere with the drag and drop process, from popping up.

### *2.1.5 Drop*

"drop" is a sneaky DHTML trick. The purpose of "drop" is to free the "chit" variable (by setting it to null) and to send the coordinate information for the dropped chit to the MySQL database via the PHP back end. It works in concert with a 1 pixel wide, 1 pixel tall, embedded image (`<IMG>` object) within a `<SPAN>` object at the bottom of the door.htm. However, the SRC attribute of this `<IMG>` is not the URL of an image file. It is the URL of track.php. track.php is a backend script that opens a connection to the MySQL database wherein all chit data (position, text width, etc.) is stored. track.php updates the X and Y coordinates for the dropped chit based on values passed to it when it is requested from the server. That is, when "drop" dynamically generates the HTML for the `<IMG>` object (and its attributes, including the SRC) and places this into the document (in NS6) or simply changes the value of the SRC (in Opera, IE and NS4).

Here is the HTML for the `<IMG>`.

```
<span id="zap">
<img width=1 height=1
src="http://www.cs.virginia.edu/~dr3f/hyperfridge/track.php"
name="zip">
</span>
```

When "drop" is called, it assembles this string.

```
"http://www.cs.virginia.edu/~dr3f/hyperfridge/track.php?daChit=" +
watchit + "&X=" + putX + "&Y=" + putY
```

And, if it is running in in Opera or IE or NS4, assigns the string directly to the "src" attribute of the <IMG> named "zip".

```
document.zip.src = "http://www.cs.virginia...
```

NS6 requires a modified approach, where the "innerHTML" of the <SPAN> object with the id "zap" is assigned a longer string including the HTML for the <IMG> tag.

```
if(NS6)
document.getElementById("zap").innerHTML = "<img width=1 height=1
src='http://www.cs.Virginia...
```

The sneaky part is this.

```
track.php?daChit=" + watchit + "&X=" + putX + "&Y=" + putY
```

The X and Y coordinates, and the unique identification name of the chit are made part of the URL and thus passed on to the PHP script which is called automatically when the new <IMG> object appears in document.

### *2.1.6 Tests*

Hyperfridge began as an evolutionary prototype. Development followed a pattern of testing and revision. Finding specific browsers on specific platforms on which to carry out testing was mostly serendipity. On the occasion an untested browser/platform would

present itself, a failed test would inspire a revised prototype. The Hyperfridge front end has been tested, and proper operation has been confirmed in the following browsers on the indicated platforms.

*Table 3. Systems on which Hyperidge has been shown to work.*

| | | Windows | | | | | | Macintosh | | Linux |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 95 | 98 | ME | NT(4) | 2K | XP | OS8 | OS9 | RH7 |
| BROWSERS | IE4 | X | X | | | X | | | | |
| | IE5 | X | X | | X | X | | X | X | |
| | IE6 | NA | X | | X | X | X | | | |
| | NS4 | X | X | | X | X | X | X | X | X |
| | NS6 | X | X | | X | X | | | X | |
| | OP5 | | | | | X | | | | |
| | OP6 | X | X | | | X | | | | |

Testing is ongoing, and there is reason to believe that Hyperfridge may work with only minor modifications in the "Konqueror" browser, developed for the Linux Operating system. Konqueror's designers claim that it is complaint with the latest CSS and DOM specifications, HTML 4 and at least 90% of JavaScript [22].

## 2.2 The Back End

### *2.2.1 The Server*

Hyperfridge.com resides on "Atlas", a Unix machine, (Sparc-Sun-Solaris 2.7) running the open source (GPL) server "Apache", (version 1.3.22) with extensions for the GPL scripting language "PHP", (version 4.1.2) and the GPL database package "MySQL" (version 3.23.42).

### *2.2.2 PHP and MySQL*

The coordinates, in pixels from the origin (the upper left of the browser window) of all of the chits that a visitor sees in door.htm are stored in a single MySQL table called "DOOR". Also stored in DOOR is the text of the chit (the word on the magnet) and its width (in pixels), the last time it was moved (Year/Month/Day/Hour/Minute/Second), and its unique identification number (see figure 1).

*Figure 1. Sample from DOOR*

| num | chit | xpos | ypos | moved | width |
|---|---|---|---|---|---|
| 1 | tribulation | 233 | 62 | 20020323042510 | 84 |
| 2 | wing | 94 | 430 | 20020323043714 | 46 |
| 3 | blunt | 1186 | 102 | 20020313051513 | 49 |
| 4 | dawn | 424 | 1044 | 20020226201148 | 50 |
| 5 | fly | 75 | 513 | 20020227102936 | 33 |
| 6 | subterranean | 1269 | 1775 | 20020224180538 | 104 |
| 7 | hedonist | 747 | 454 | 20020306192953 | 72 |

The data in this table is updated and retrieved via four PHP scripts, “chits.php”, “track.php”, and “sawpit.php”, which work in combination with “swap.php” (see appendix 5).

Whenever door.htm is requested from the server, so is chits.php via a server include statement appearing inside the <TD> tags of the table that defines the perimeters of the GUI.

```
<table width=2500 height=2500 cellpadding=0 cellspacing=0 border=0>
<tr><td>

<!--#include virtual="chits.php"-->

</td></tr>
</table>
```

chits.php is a very simple script. It opens a connection to the database, finds the DOOR table, and then runs down the entire table printing chits into door.htm.

```
<?php

$conn = mysql_connect("mysql.cs.virginia.edu", "dr3f", "password") or
die(mysql_error()) ;

mysql_select_db("dr3f", $conn) or die(mysql_error()) ;

$query = "select * from DOOR" ;

$result = mysql_query($query) or die(mysql_error()) ;

while( $row = mysql_fetch_array($result, MYSQL_ASSOC) )
{
      echo '<div id="C' . $row["num"] . '"
      style="position:absolute;left:' . $row["xpos"] . ';top:' .
      $row["ypos"] . ';width:' . $row["width"] . ';">  ' .
      $row["chit"] . '  </div>' ;
}

?>
```

The resulting chits look like this.

```
<div id="C3" style="position:absolute;left:1186;top:102;width:49;">
  blunt  
</div>
```

They download as HTML to the client browser, drag and droppable objects within door.htm document. Notice that a “C” is appended to the unique ID. This is for NS4 which does not allow the “id” attribute to be assigned a value that starts with a number.   is an HTML character entity that translates to a blank space in the browser.

As previously explained, whenever a chit is dropped in door.htm, track.php is requested from the server and the X and Y coordinates of the chit, along with the chit id are passed into it. track.php opens a connection to the database, strips the “C” off of the ID and updates the appropriate row in DOOR.

```
<?php
$conn = mysql_connect("mysql.cs.virginia.edu", "dr3f", "password") or
die(mysql_error()) ;
mysql_select_db("dr3f", $conn) or die(mysql_error()) ;
$noC = substr($daChit, 1) ;
$query = "update DOOR set xpos='$X', ypos='$Y' where num='$noC'" ;
mysql_query($query) or die(mysql_error()) ;
?>
```

In order to maintain a mood of limited selection, key to the idea of Magnetic Poetry, users may add new words to the mix of 500 on Hyperfridge at a maximum rate of

one every 8,616 seconds (one tenth of a sidereal day). The HTML interface for word submission is generated by swap.php based on the value it finds stored in time.txt.

time.txt holds the Unix timestamp (seconds from 01/01/1970 GMT) of the last word submission plus 8,616. swap.php compares this timestamp to the current time. If it has been less than 8,616 second since the last new word was submitted, then swap.php prints the HTML for simple white on black page with the word "Wait" and the number of seconds remaining until the next submission will be accepted. If, on the other hand, sufficient time has elapsed, then swap.php presents the user with a simple HTML form. The form has one text field with space for a 25-letter word and a single button to make the submission (see figure 2). The form sends data by post to swapit.php for evaluation.

*Figure 2.*


*New word submission form.*

There is very little censorship on Hyperfridge.com. swapit.php will allow almost any word to pass. swapit.php will not allow words that contain any characters other than the 26 letters of the alphabet, no spaces, no hyphens, no apostrophizes. Only letters.

swapit.php will not pass words longer than 25 letters long. swapit.php will not pass words less than two letters long where both letters are the same and swapit.php will not pass words of any length where three consecutive letters are the same. swapit.php will not pass duplicate words (words already on Hyperfridge). It is interesting to note that Hyperfridge was live on the web for almost a month before some witty visitor submitted the word "fuck".

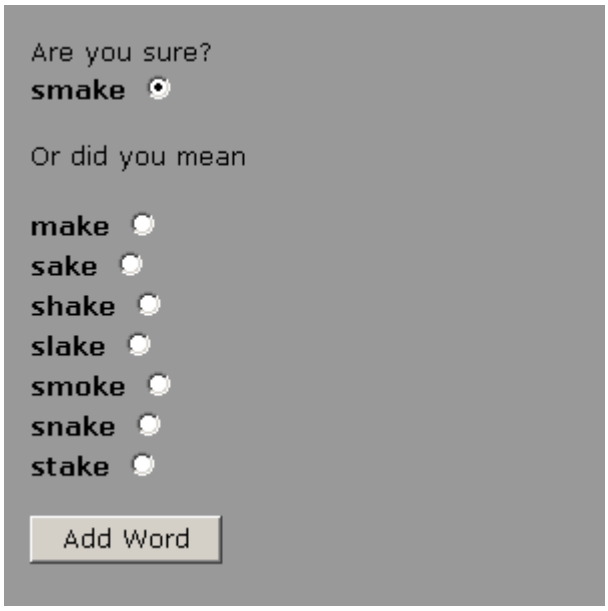
Should any of the aforementioned tests fail, swapit.php generates the HTML for a simple white on black page echoing the attempted submission and appending to it the qualifying phrase "is invalid". Otherwise, if the word is over four letters long it gets passed into Ispell. Ispell (version 3.1.2) is a command line interfaced Unix spell checker running on Atlas [23]. Here is where PHP really shows its worth as a scripting language.

```
$toCheck = "echo $toAdd | /uva/bin/ispell -a" ;
exec($toCheck, $everything, $r) ;
$indic = $everything[1] ;
```

The "exec" function allows the script to pass parameters to another program on the server (Ispell in this case). Each line returned from the program becomes an ordered member of an array.

If the submitted word is not in the Ispell dictionary, but there are near misses, then those near misses are filtered by swapit.php and a list of alternative spellings plus the original word are presented to the user as a form.

*Figure 3.*




*Spell checking suggestion dialog form.*

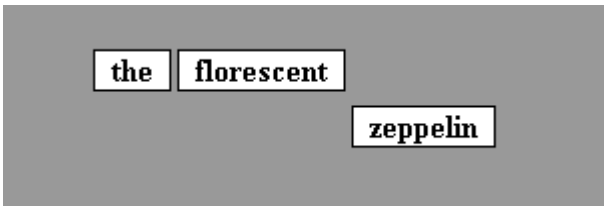
The users initial submission is selected by default.

Next, sawpit.php opens a connection to DOOR and looks for all the chits that are lined up next to each other. That is, not offset from each other on the Y axis by more than their height (19 pixels) and with no more than 19 pixels gap between them on the X axis.

```
$query1 = "select distinct t1.num, t1.chit from DOOR t1, DOOR t2 where
(abs(t1.ypos - t2.ypos) < 19) and (t1.xpos + t1.width + 19 > t2.xpos)
and (t1.xpos < t2.xpos + t2.width + 19) and (t1.num != t2.num)" ;
```

These chits are assumed to be part of a poem structure (see figure 7).

*Figure 4.*




*sawpit.php checks for chits "in use".*

"the" and "florescent" are in use. "zeppelin" is free and may be swapped out for a new word. Of the remaining free words, sawpit.php looks for the word that has not been moved in the longest time. That is the one that will be swapped out for the new word.

Next, the length of the new word is calculated.

```
for( $i = 0; $i < strlen($toAdd); $i++ )
{
	switch( $toAdd[$i] )
	{
		case "a": $chitlen += 8 ; break ;
		case "b": $chitlen += 8 ; break ;
		case "c": $chitlen += 7 ; break ;
		case "d": $chitlen += 8 ; break ;
		case "e": $chitlen += 8 ; break ;
		case "f": $chitlen += 5 ; break ;
		case "g": $chitlen += 8 ; break ;
		case "h": $chitlen += 8 ; break ;
		case "i": $chitlen += 4 ; break ;
		case "j": $chitlen += 4 ; break ;
		case "k": $chitlen += 9 ; break ;
		case "l": $chitlen += 4 ; break ;
		case "m": $chitlen += 12 ; break ;
		case "n": $chitlen += 8 ; break ;
		case "o": $chitlen += 8 ; break ;
		case "p": $chitlen += 8 ; break ;
		case "q": $chitlen += 8 ; break ;
		case "r": $chitlen += 6 ; break ;
		case "s": $chitlen += 7 ; break ;
		case "t": $chitlen += 5 ; break ;
		case "u": $chitlen += 8 ; break ;
		case "v": $chitlen += 8 ; break ;
		case "w": $chitlen += 10 ; break ;
		case "x": $chitlen += 8 ; break ;
		case "y": $chitlen += 8 ; break ;
		case "z": $chitlen += 6 ; break ;
		default: $badword = true ;
	}
}
```

The widths of the component letters are in pixels based on the font size and face specified in the document's CSS.

After the word length is known, sawpit.php writes a new timestamp into time.txt…

```
$waitTime = date("U") + 8616 ;

$daFile = fopen("time.txt", "w") ;
fwrite($daFile, $waitTime) ;
fclose($daFile) ;
```

and updates DOOR. The new word appears in the same position as the one that was swapped out. sawpit.php concludes a successful update by generating the HTML for a simple white on black page with the acknowledgement "Word Added".

"populate.php", a script similar to swapit.php reads words from a text file and was used to initialize Hyperfridge. populate.php assigned X and Y coordinates for each chit and random and set all the timestamps to the moment it was run.

## 2.3 Content and Aesthetics

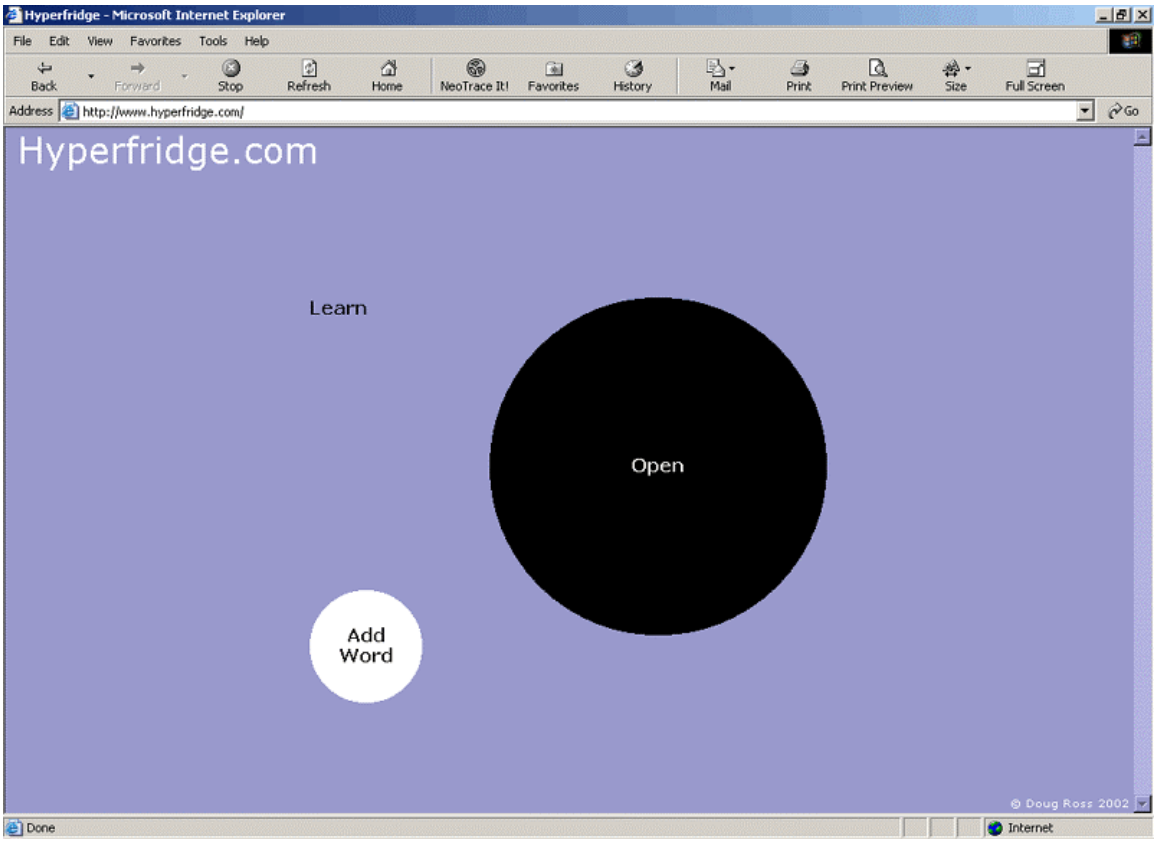
### *2.3.1 The Words*

300 of the 500 initial words on Hyperfridge were extracted from the top 350 words from the British National Corpus list of the 3000 most commonly used words in the English language [24]. Of the remaining 200 words, 150 were chosen by the method of stream of consciousness. The last 50 are simply common suffixes and conjunctions. See appendix 6 for a complete list of the initial 500 words.

### *2.3.2 The Look*

Let us call it ultra-minimal art deco with a touch of Miró.

*Figure 5.*




*The look of the Hyperfridge index.*

The index page is actually a PHP file. When it is requested from the server it generates the HTML for the page with a random, one out of six, color choice for the background. The six are subdued shades simple web-safe colors, #CC9999, #99CC99, #9999CC, #99CCCC, #CC99CC and #CCCC99 (Hexadecimal for Red, Green, Blue, Teal, Magenta and Yellow). The circles are two-color, sliced transparent .GIFs set as the backgrounds to a table (1.8Kb total). The font (specified by a document-wide style sheet)

defaults to Verdana. Arial will be substituted for users without Verdana, Helvetica for those without either or any Sans-Serif font for those without all three.

The entire site is designed to work in resolutions from 640px by 480px up. The links “Add Word” and “Learn” (a brief user introduction) each open a 300X300 window stripped of all attributes except scrollbars. These balance the 300px diameter circle in which the link to the primary GUI, (“Open”) is centered. The GUI itself opens in a 640X480 window, also stripped of all attributes except scrollbars. The background color for know.htm (linked to by “Learn”), the “Add Word” dialogues, and the GUI (door.htm) is a neutral grey, the same shade as the index, #999999.

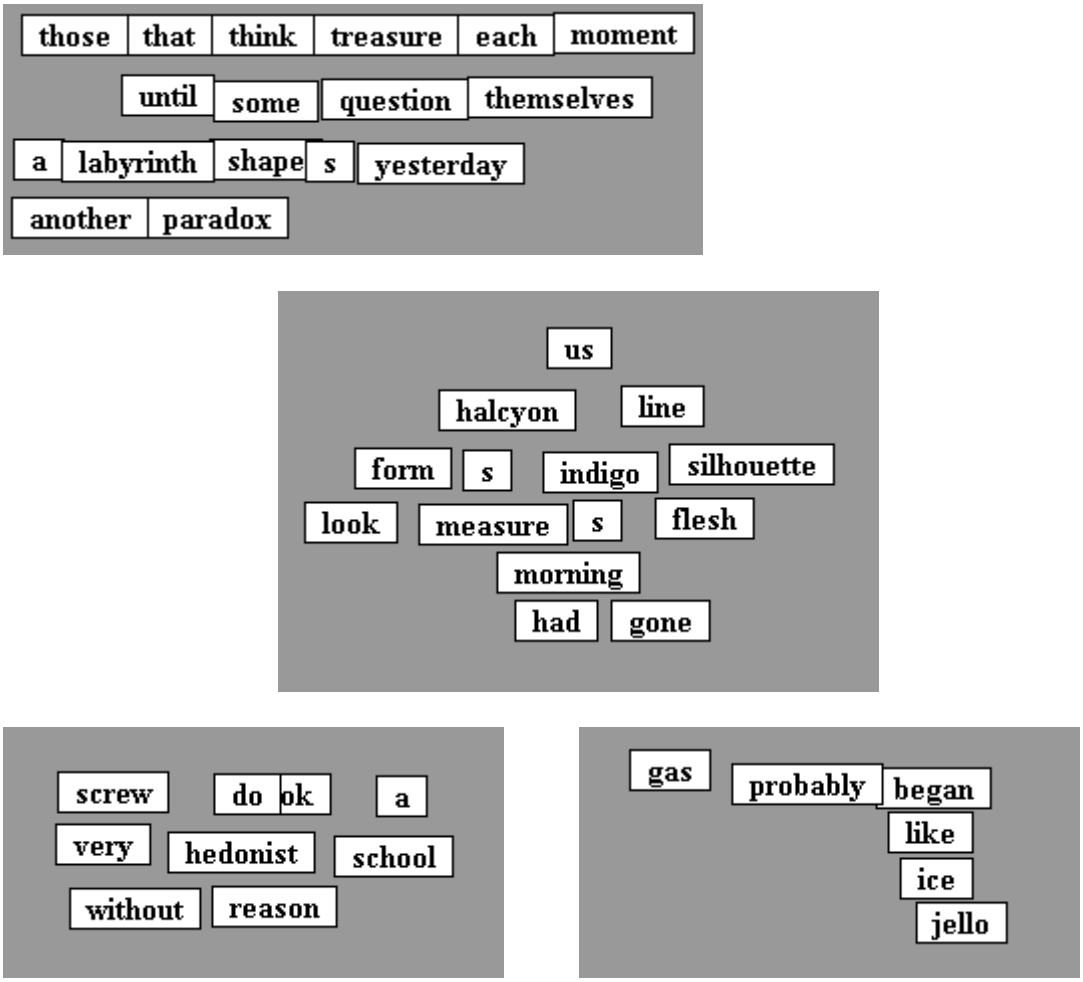
The Hyperfridge look is optimized for simplicity, usability and fast download speed.

# 3. CONLUSIONS

## 3.1 Live on the Web

Hyperfridge.com completed final beta testing and went live on Valentine's Day 2002. The results have been amusing. Here are some samples of user input over the last 40 days.

*Figure 6.*

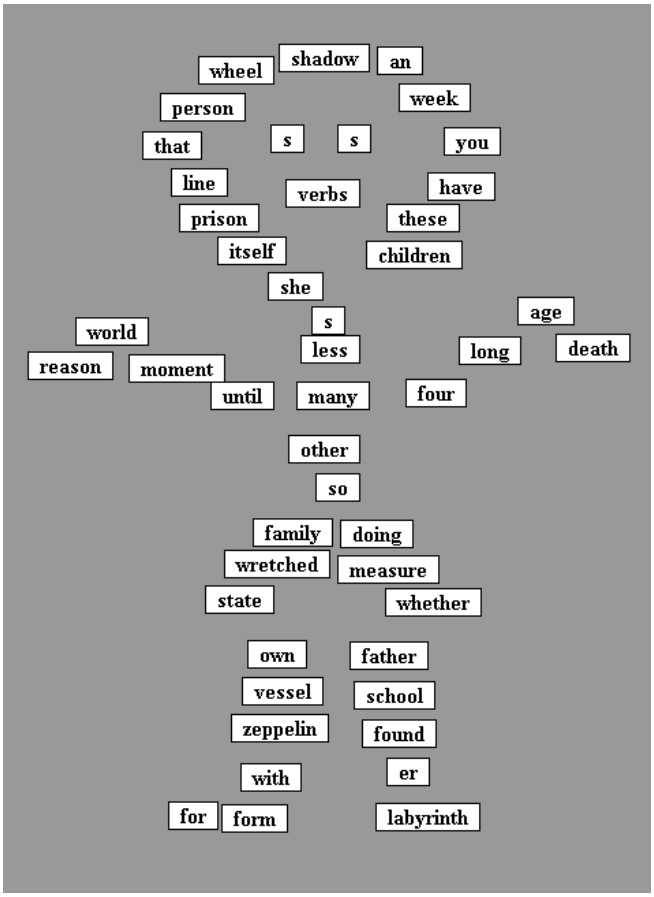



*Sample poetry.*

In the same time, users have added 38 new words. Here are the words in the order they were added.

Tribulation, trunk, beagle, knowledge, perspicacious, verbs, supercalifragilistic, sleep, mom, hold, sake, send, feel, facetious, wing, blunt, dawn, fly, hedonist, silhouette, doorway, lesbian, freaks, malarkey, yooveeay, traffic, test, listen, stranger, exhilarating, cyberdoug, trust, explain, fuck, gelatinous, mediocre, armadillo, happy

*Figure 7.*




*Unexpected results.*

On February 20th, this surprising construction appeared.

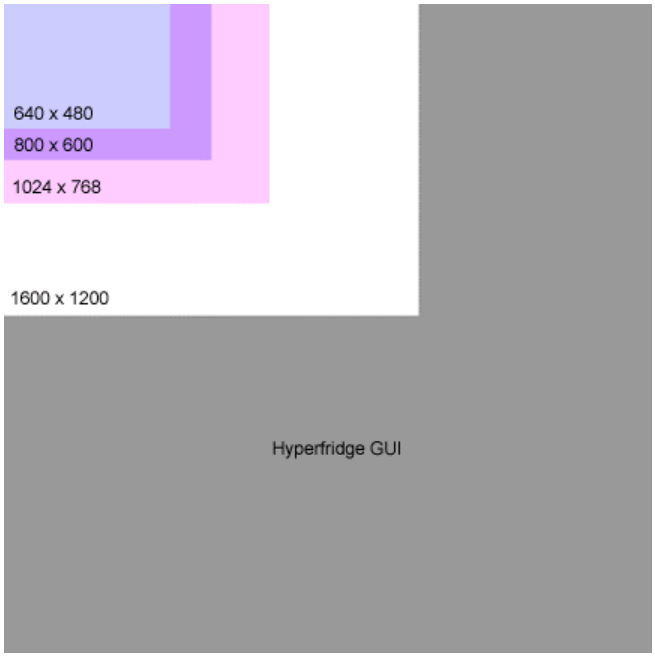
### 3.2 Possible Improvements

Some problems with the Hyperfridge website have revealed themselves since it went live on the Web.

1. Chits are migrating from the lower right of door.htm to the upper left.
2. Common words are being replaced with obscure words.

The migration of chits towards the origin is a direct result of average screen resolution. The table that defines the perimeters of door.htm is 2,500 pixels wide and 2,500 pixels high. The majority of those browsing the Internet have monitors with 800px by 600px resolution (see appendix 2). Even high-resolution monitors, 1,600px by 1,200px, are not large enough to display the whole surface of the GUI without scrolling.

*Figure 8.*




*Common screen resolution vs. the Hyperfridge GUI*

Visitors are moving the chits to make poems in the areas where they became inspired. The page loads with the origin in the upper left corner. Whatever is visible then is likely to inspire the visitor first.

A possible solution to this problem would be making the chit and door sizes relative and dynamic. If all chits were visible on opening the page we would likely see more balanced placement of poems. This is no small task. However, it is theoretically doable with DHTML.

Common words are being depleted because arcane polysyllabic words are more fun to submit. Ultimately Hyperfridge could become unusable for lack of simple connecting words like “have”, “what”, “and”, and “there”. To curb this progression an extra field could be added to DOOR in the MySQL database. The most useful words could be marked “permanent” and sawpit.php could be written to exclude these from the list of replaceable ones.

Both of these improvements will be looked into.

### 3.3 Final Thoughts on DHTML

Making Hyperfridge cross browser compatible took months. At times, it seemed that every new browser/platform test revealed another unforeseen contingency necessitating a revision. At the time the project began, there was very little documentation available on the Netscape 6 DOM implementation. Much of the code for that part of the drag and drop program was hacked out by listing the methods of various

document objects with “for/in” statements and trying them out with what seemed like appropriate input until it worked.

In general, the project would have been much simpler if the code were written for a single standardized implementation of the document object model. Writing for Netscape 4 and 6, and Internet Explorer 4 plus, and Opera ended up requiring more than three times as much code as it would have taken for any single one of them.

Non-compliance with recommended standards has limited the end users experience of the modern commercial Internet because programmers (and companies) would rather deliver simple pages that are guaranteed to work across all browsers, than create elaborate and engaging pages that are sure to fail embarrassingly at some point.

An alternative approach is now becoming popular. Until now, Flash, the browser plug-in developed by Macromedia software had been little more than a toy. Its only purpose was playing vector image animation. Macromedia changed all of that with the release of Flash 5. They have incorporated a powerful scripting language, ActionScript, based on the ECMA-262 specification into Flash and given it back-end and database connectivity features like XML sockets. Seeing the importance of what are now being called “Web Applications”, Macromedia is positioning itself to challenge Sun Microsystems, the developers of Java, as the leading developer of platform independent, server connected programs. With ActionScript, Flash becomes a versatile tool for building client-side front ends to databases and server-side programs. The new Flash also supports streaming multimedia.

A project like Hyperfridge, developed in Flash, would require much less code and be guaranteed to work on the client side for anyone with a Flash player installed.

The browser wars may be over, and it does seem as though Microsoft has won (see appendix 2). Many people see plug-ins like Flash as just more confusion. However, the author of this document sees Flash 5 in particular as a viable new product opening a great deal of potential for new forms of expression and interaction on the Internet.

Flash has built in functions for dragging and dropping objects. Dynamic sizing is easy with vector graphics. If he had to do Hyperfridge all over again, he would use Flash.

He may just do it for kicks.

## Bibliography and Works Cited


[1] Atkinson, Leon. Core PHP Programming. Prentice Hall PTR, Upper Saddle River, New Jersey, 1999.

[2] Beer, David. "Wingnut - Etc." 14 Apr. 2000. 14 Oct. 2001. <http://www.wingnut-etc.com/poetry>.

[3] Berners-Lee, Tim. Weaving the Web. Harper Collins, New York, New York, 1999.

[4] Di'Aradia. "Magnetic Poetry." Unknown. 14 Oct. 2001. <http://www.diaradia.com/magneticpoetry.html>.

[5] ECMA. "Standard ECMA-262, ECMAScript Language Specification." 1999. 14 Oct. 2001. <http://www.ecma.ch>.

[6] GF Design. "The Magnetic Poetry Fridge." GF Design Las Vegas . Unknown. 14 Oct. 2001. <http://www.gfdesignlv.com/flash/the_fridge.htm>.

[7] Flanagan, David. JavaScript (the Definitive Guide). O'Reilly & Associates, Sebastopol, California, 1998.

[8] Greenspan, Jerry and Bulger, Brad. MySQL/PHP Database Applications. M & T Books, New York, New York, 2001.

[9] Kapell, Dave. "nitric interactive, e-tiles." Unknown. 14 Oct. 2001. <http://nitric.com/interactive/>.

[10] Lie, Håkon Wium and Bos, Bert. Cascading Style Sheets (Designing for the Web). Addison Wesley, United States, 1999.

[11] Magnetic Poetry inc. "Online Magnetic Poetry." Unknown. 23 Oct. 2001 <http://www.magneticpoetry.com/>

[12] Meloni, Julie C. Essential PHP. Prima Publishing, Roseville, California, 2000.

[13] Musciano, Chuck and Kennedy, Bill. HTML & XHTML (the Definitive Guide). O'Reilly & Associates, Sebastopol, California, 2000.

[14] Nielson, Jakob. Designing Web Usability. New Riders Publishing, Indianapolis, Indiana, 1999.

[15] Pedram. “pedram.redhive.com.” 29 Aug. 2001. 14 Oct. 2001. <http://pedram.redhive.com/>.

[16] TheCounter.com. “Global Statistics.” 14 Oct. 2001. 14 Oct. 2001. <http://www.thecounter.com/stats/>.

[17] Unknown. “The Dubya Virtual Magnetic Poetry Page.” The Adam Clymer Fan Club. Unknown. 14 Oct. 2001. <http://www.adamclymerfanclub.org/vmagpoetry.php>.

[18] Veen, Jeffrey. The Art & Science of Web Design. New Riders Publishing, Indianapolis, Indiana, 2000.

[19] Williamson, Heather. Writing Cross-Browser HTML. Apress, Berkeley, California, 2000.

[20] W3C DOM WG. “Document Object Model.” W3C Architecture Domain. 14 Oct. 2001. 14 Oct. 2001. <http://www.w3c.org/DOM/>.

[21] W3C. “Web Style Sheets Homepage.” 31 Aug. 2001. 14 Oct. 2001. <http://www.w3c.org/style/>.

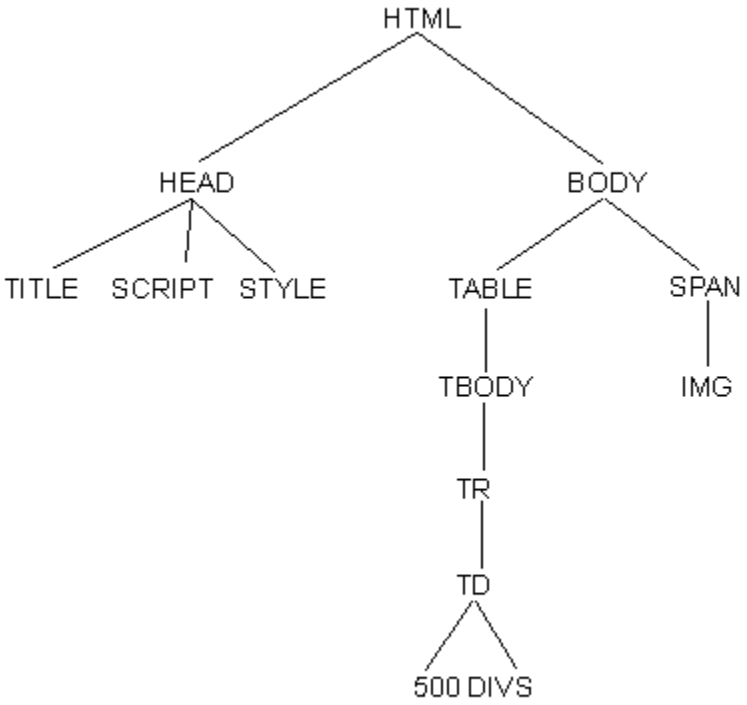
[22] Konqueror. “Konqueror - the Web Browser.” 23 Mar 2002. 15 Apr 2001. <http://www.konqueror.org/>.

[23] Ispell. “International Ispell” 01 Feb 2002. <http://fmg-www.cs.ucla.edu/fmg-members/geoff/ispell.html>.

[24] British National Corpus. “British National Corpus” 15 Jan 2002.02 Nov 2001. <http://www.hcu.ox.ac.uk/BNC/index.html>.

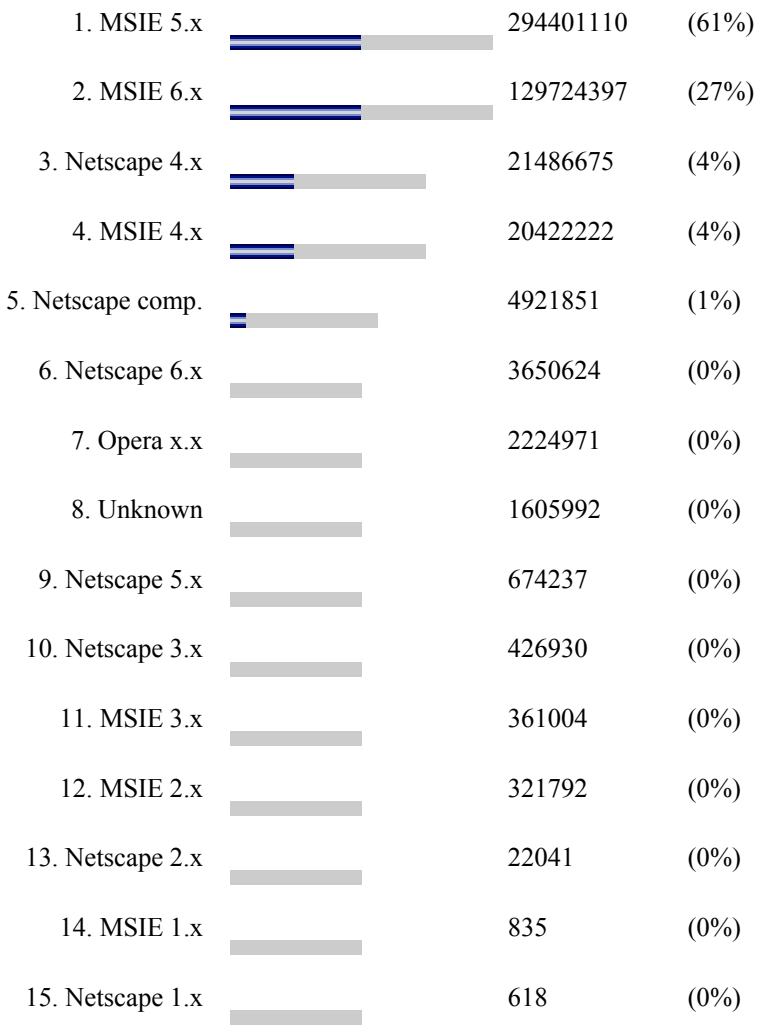
# Appendices

## 1. Object Tree for door.htm



## 2. Browser and OS Statistics

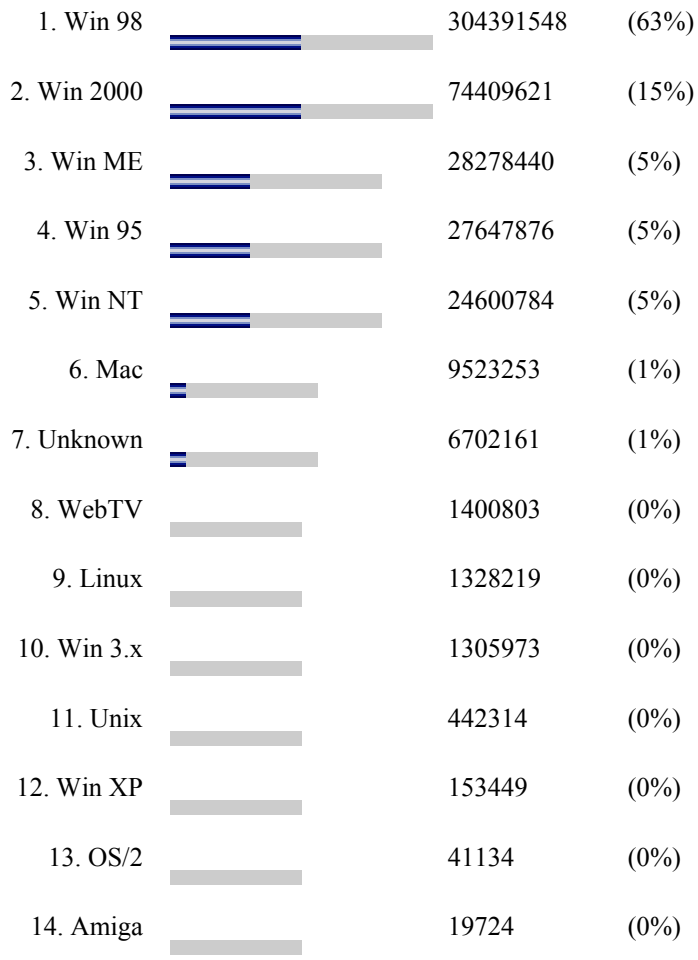
**Browser Stats**
Fri Feb 1 00:05:02 2002 - Thu Feb 28 23:55:04 2002  28.0 Days



| | | |
|---|---|---|
| 1. MSIE 5.x | 294401110 | (61%) |
| 2. MSIE 6.x | 129724397 | (27%) |
| 3. Netscape 4.x | 21486675 | (4%) |
| 4. MSIE 4.x | 20422222 | (4%) |
| 5. Netscape comp. | 4921851 | (1%) |
| 6. Netscape 6.x | 3650624 | (0%) |
| 7. Opera x.x | 2224971 | (0%) |
| 8. Unknown | 1605992 | (0%) |
| 9. Netscape 5.x | 674237 | (0%) |
| 10. Netscape 3.x | 426930 | (0%) |
| 11. MSIE 3.x | 361004 | (0%) |
| 12. MSIE 2.x | 321792 | (0%) |
| 13. Netscape 2.x | 22041 | (0%) |
| 14. MSIE 1.x | 835 | (0%) |
| 15. Netscape 1.x | 618 | (0%) |

Source: http://www.thecounter.com

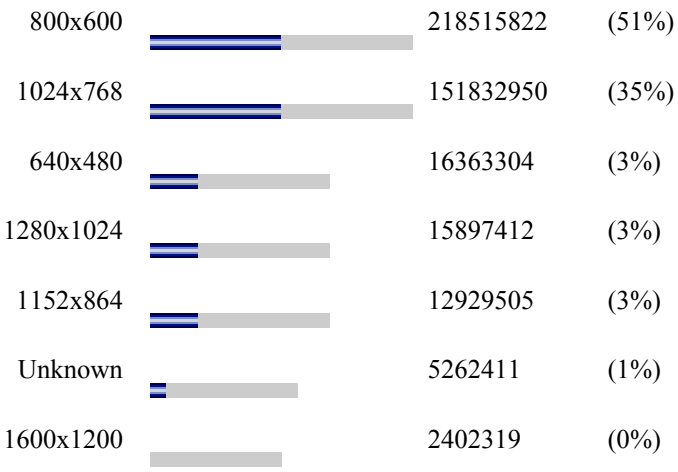
**OS Stats**
Fri Feb 1 00:05:02 2002 - Thu Feb 28 23:55:04 2002  28.0 Days



| | | |
|---|---|---|
| 1. Win 98 | 304391548 | (63%) |
| 2. Win 2000 | 74409621 | (15%) |
| 3. Win ME | 28278440 | (5%) |
| 4. Win 95 | 27647876 | (5%) |
| 5. Win NT | 24600784 | (5%) |
| 6. Mac | 9523253 | (1%) |
| 7. Unknown | 6702161 | (1%) |
| 8. WebTV | 1400803 | (0%) |
| 9. Linux | 1328219 | (0%) |
| 10. Win 3.x | 1305973 | (0%) |
| 11. Unix | 442314 | (0%) |
| 12. Win XP | 153449 | (0%) |
| 13. OS/2 | 41134 | (0%) |
| 14. Amiga | 19724 | (0%) |

Source: http://www.thecounter.com

**Resolution Stats**
Fri Feb 1 00:05:02 2002 - Thu Feb 28 23:55:04 2002  28.0 Days



| | | |
|---|---|---|
| 800x600 | 218515822 | (51%) |
| 1024x768 | 151832950 | (35%) |
| 640x480 | 16363304 | (3%) |
| 1280x1024 | 15897412 | (3%) |
| 1152x864 | 12929505 | (3%) |
| Unknown | 5262411 | (1%) |
| 1600x1200 | 2402319 | (0%) |

Source: http://www.thecounter.com

## 3. Server Benchmark

```
This is ApacheBench, Version 2.0.34-dev <$Revision: 1.89 $> apache-2.0
Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright (c) 1998-2002 The Apache Software Foundation, http://www.apache.org/

Benchmarking www.cs.virginia.edu (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Finished 1000 requests

Server Software:        Apache/1.3.22
Server Hostname:        www.cs.virginia.edu
Server Port:            80

Document Path:          /
Document Length:        22453 bytes

Concurrency Level:      10
Time taken for tests:   56.689414 seconds
Complete requests:      1000
Failed requests:        971
   (Connect: 0, Length: 971, Exceptions: 0)
Write errors:           0
Total transferred:      22714775 bytes
HTML transferred:       22545775 bytes
Requests per second:    17.64 [#/sec] (mean)
Time per request:       0.567 [ms] (mean)
Time per request:       0.057 [ms] (mean, across all concurrent requests)
Transfer rate:          391.29 [Kbytes/sec] received

Connection Times (ms)
            min  mean[+/-sd] median   max
Connect:    0     0    0.3 0 7
Processing: 154   565  237.0 516 1629
Waiting:    19   175  107.4 155 857
Total:      154   565  237.0 516 1629

Percentage of the requests served within a certain time (ms)
  50%    516
  66%    619
  75%    690
  80%    749
  90%    889
  95%   1021
  98%   1186
  99%   1295
 100%   1629 (longest request)
```

Thanks, Cliff Woolley.

```
<script language="JavaScript">
<!--

var NS6 = (navigator.userAgent.indexOf("Gecko") != -1) ? 1 : 0 ;
var OP = (navigator.userAgent.indexOf("Opera") != -1) ? 1 : 0 ;
var IE = (document.all && !OP) ? 1 : 0 ;
var NS = (document.layers) ? 1 : 0 ;

var chit = null ;
var watchit ;
var chitwidth ;
var X ;
var Y ;
var Z = 1 ;
var Xoff ;
var Yoff ;
var putX ;
var putY ;

var moved = false ;

function grab(e)
{
   Z++ ;

   if(NS)
   {
      X = e.pageX ;
      Y = e.pageY ;

      for(var i = document.layers.length - 1; i >= 0; i--)
      {
         var ischit = document.layers[i] ;

         if(( X > ischit.left )
             && ( X < ischit.left + ischit.clip.width )
             && ( Y > ischit.top )
             && ( Y < ischit.top + ischit.clip.height ))
         chit = ischit ;
      }

      if(chit)
      {
         Xoff = X - chit.left ;
         Yoff = Y - chit.top ;
         chit.zIndex = Z ;
         watchit = chit.id ;
         chitwidth = chit.clip.width ;
      }
   }
   else if(IE || OP)
   {
      watchit = event.srcElement.id ;
```

```
      if(watchit)
      {
         chit = event.srcElement ;
         X = event.clientX ;
         Y = event.clientY ;
         Xoff = chit.style.pixelLeft ;
         Yoff = chit.style.pixelTop ;
         chit.style.zIndex = Z ;

         if(IE)
         chitwidth = chit.scrollWidth ;
         else
         chitwidth = chit.style.pixelWidth + 2 ;
      }
   }
   else if(NS6)
   {
      if(!e.target.parentNode.id) { chit = e.target ; }
      else { chit = e.target.parentNode ; }

      Xoff = chit.style.left ;
      Yoff = chit.style.top ;
      X = e.clientX ;
      Y = e.clientY ;

      if(chit)
      {
         watchit = chit.id ;
         chit.style.zIndex = Z ;
         chitwidth = (chit.innerHTML.length - 4) * 7 + 16 ;
      }
   }

   moved = false ;
   return false ;
}

function drag(e)
{
   if(chit)
   {
      if(NS)
      {
         putX = chit.left = e.pageX - Xoff ;
         putY = chit.top = e.pageY - Yoff ;
         if(putX < 0)
         putX = chit.left = 0 ;
         if(putY < 0)
         putY = chit.top = 0 ;
         if(putX > 2500 - chitwidth)
         putX = chit.left = 2500 - chitwidth ;
         if(putY > 2481)
         putY = chit.top = 2481 ;
      }
      else if(IE || OP)
      {
```

```
         putX = chit.style.pixelLeft = Xoff + event.clientX - X ;
         putY = chit.style.pixelTop = Yoff + event.clientY - Y ;
         if(putX < 0)
         putX = chit.style.pixelLeft = 0 ;
         if(putY < 0)
         putY = chit.style.pixelTop = 0 ;
         if(putX > 2500 - chitwidth)
         putX = chit.style.pixelLeft = 2500 - chitwidth ;
         if(putY > 2481)
         putY = chit.style.pixelTop = 2481 ;
      }
      else if(NS6)
      {
         putX = chit.style.left = e.clientX - parseInt(X) +
         parseInt(Xoff) ;
         putY = chit.style.top =  e.clientY - parseInt(Y) +
         parseInt(Yoff) ;
         if(putX < 0)
         putX = chit.style.left = 0 ;
         if(putY < 0)
         putY = chit.style.top = 0 ;
         if(putX > 2500 - chitwidth)
         putX = chit.style.left = 2500 - chitwidth ;
         if(putY > 2481)
         putY = chit.style.top = 2481 ;
      }
   }

   moved = true ;
   return false ;
}

function drop()
{
   if(watchit && moved)
   {
      if(NS6)
      document.getElementById("zap").innerHTML =
"<img width=1 height=1
src='http://www.cs.virginia.edu/~dr3f/hyperfridge/track.php?daChit=" +
watchit + "&X=" + putX + "&Y=" + putY  + "'>" ;
      else
      document.zip.src =
"http://www.cs.virginia.edu/~dr3f/hyperfridge/track.php?daChit=" +
watchit + "&X=" + putX + "&Y=" + putY ;
   }

   chit = null ;
   return false ;
}
if(NS)
document.captureEvents(Event.MOUSEDOWN | Event.MOUSEMOVE |
Event.MOUSEUP) ;

document.onmousedown = grab ;
document.onmousemove = drag ;
document.onmouseup = drop ;
```

# 5. PHP

## track.php

```
<?php

$conn = mysql_connect("mysql.cs.virginia.edu", "dr3f", "password") or
die(mysql_error()) ;

mysql_select_db("dr3f", $conn) or die(mysql_error()) ;

$noC = substr($daChit, 1) ;

$query = "update DOOR set xpos='$X', ypos='$Y' where num='$noC'" ;

mysql_query($query) or die(mysql_error()) ;

?>
```

## chits.php

```
<?php

$conn = mysql_connect("mysql.cs.virginia.edu", "dr3f", "password") or
die(mysql_error()) ;

mysql_select_db("dr3f", $conn) or die(mysql_error()) ;

$query = "select * from DOOR" ;

$result = mysql_query($query) or die(mysql_error()) ;

while( $row = mysql_fetch_array($result, MYSQL_ASSOC) )
{
      echo '<div id="C' . $row["num"] . '"
      style="position:absolute;left:' . $row["xpos"] . ';top:' .
      $row["ypos"] . ';width:' . $row["width"] . ';">  ' .
      $row["chit"] . '  </div>' ;
}

?>
```

## swap.php

```
<?php

$daFile = fopen("time.txt", "r") ;
$nextSwap = fread($daFile, filesize("time.txt")) ;
fclose($daFile) ;

$now = date("U") ;

$waitTime = $nextSwap - $now ;
```

```
if( ($waitTime) > 0 )
{
      echo '<html><head><title>Wait</title></head>' ;
      echo '<style type="text/css">h1 { font-size: 16px ; font-family:
      verdana, arial, helvetica, sans-serif ; }' ;

      echo 'body { scrollbar-base-color: #000000 ; scrollbar-arrow-
      color: #ffffff ; }</style>' ;

      echo '<body bgcolor="#000000" text="#ffffff"><table width=100%
      height=100%><tr><td align=center>' ;

      echo "<h1>Wait<br><br>$waitTime Seconds</h1>" ;

      echo '</td></tr></table></body></html>' ;
}
else
{
      echo '<html><head><title>Add Word</title></head>' ;

      echo '<style type="text/css">.ss { font-family: verdana, arial,
      helvetica, sans-serif ; }' ;

      echo 'body { scrollbar-base-color: #999999 ; scrollbar-arrow-
      color: #ffffff ; }</style>' ;

      echo '<body bgcolor="#999999" text="#000000" topmargin=0
      leftmargin=0 marginwidth=0 marginheight=0>' ;

      echo '<table width=100% height=100% border=0 cellpadding=0
      cellspacing=0>' ;

      echo '<tr><td align=center><form name=daForm method=post
      action="swapit.php"><input type=hidden name=editMode value=0>' ;

      echo '<input type=text name=daWord size=25
      maxlength=25><br><br><input class="ss" type=submit value="Add
      Word"></form>' ;

      echo '</td></tr></table></body></html>' ;
}

?>
```

**swapit.php**

```
<?php

$daFile = fopen("time.txt", "r") ;
$nextSwap = fread($daFile, filesize("time.txt")) ;
fclose($daFile) ;

$now = date("U") ;

if( ($nextSwap - $now) > 0 ) { exit ; }
```

```
$toAdd = strtolower($daWord) ;

if( ereg("[^a-z]", $toAdd) || ( strlen($toAdd) > 25 ) || (
strlen($toAdd) < 1 ) || (( strlen($toAdd) == 2 ) && ($toAdd[0] ==
$toAdd[1])) )
{
      echo '<html><head><META HTTP-EQUIV="refresh" CONTENT="3;
      URL=swap.php"><title>-(!)-</title></head>' ;

      echo '<style type="text/css">h1 { font-size: 16px ; font-family:
      verdana, arial, helvetica, sans-serif ; }' ;

      echo 'body { scrollbar-base-color: #000000 ; scrollbar-arrow-
      color: #ffffff ; }</style>' ;

      echo '<body bgcolor="#000000" text="#ffffff"><table width=100%
      height=100%><tr><td align=center>' ;

      echo "<h1>$toAdd<br><br>Is Invalid</h1>" ;

      echo "</td></tr></table></body></html>" ;

      exit ;
}

for( $i = 0; $i < strlen($toAdd) - 2; $i++ )
{
      $x = $i + 1 ;
      $y = $i + 2 ;

      if( ($toAdd[$i] == $toAdd[$x]) && ($toAdd[$x] == $toAdd[$y]) )
      {
            echo '<html><head><META HTTP-EQUIV="refresh" CONTENT="3;
            URL=swap.php"><title>-(!)-</title></head>' ;

            echo '<style type="text/css">h1 { font-size: 16px ; font-
            family: verdana, arial, helvetica, sans-serif ; }' ;

            echo 'body { scrollbar-base-color: #000000 ; scrollbar-
            arrow-color: #ffffff ; }</style>' ;

            echo '<body bgcolor="#000000" text="#ffffff"><table
            width=100% height=100%><tr><td align=center>' ;

            echo "<h1>$toAdd<br><br>Is Invalid</h1>" ;

            echo "</td></tr></table></body></html>" ;

            exit ;
      }
}

if( (strlen($toAdd) > 4) && ($editMode == 0) )
{
      $toCheck = "echo $toAdd | /uva/bin/ispell -a" ;
```

```
exec($toCheck, $everything, $r) ;

$indic = $everything[1] ;

if($indic[0] == "&")
{
      $uno = strlen($toAdd) + 8 ;

      $dos = substr($everything[1], $uno) ;

      $dos .= ",xxx" ;

      $i = 0 ;

      for( $token = strtok($dos, ","); $token != "xxx"; $token =
      strtok(",") )
      {
            $uncool[$i] = $token ;
            $i++ ;
      }

      $j = 0 ;

      for( $i = 0; $i < count($uncool); $i++ )
      {
            $uncool[$i] = ltrim($uncool[$i]) ;

            if( ereg("[^a-z]", $uncool[$i]) )
            {
                  ;
            }
            else
            {
                  $cool[$j] = $uncool[$i] ;
                  $j++ ;
            }
      }

      $self = basename($PHP_SELF) ;

      echo "<html><head><title>-(?)-</title></head>\n" ;

      echo "<style type=\"text/css\">\n.ss { font-size: 13px ;
      font-family: verdana, arial, helvetica, sans-serif ; }\n" ;

      echo ".ssf { font-family: verdana, arial, helvetica, sans-
      serif ; }\n" ;

      echo "body { scrollbar-base-color: #999999 ; scrollbar-
      arrow-color: #ffffff ; }\n</style>\n" ;

      echo "<body bgcolor=\"#999999\" text=\"#000000\">\n<table
      width=100% height=100%>\n<tr><td align=left>\n" ;

      echo "<form name=daForm method=post action=\"$self\">\n" ;

      echo "<input type=hidden name=editMode value=1>\n" ;
```

```
            echo "<span class=\"ss\">Are you sure?<br>\n" ;

            echo "<b>$toAdd</b> <input type=radio name=daWord checked
            value=\"$toAdd\">\n" ;

            if( count($cool) > 0 ) { echo "<br><br>Or did you
            mean<br>\n" ; }


            for( $i = 0; $i < count($cool); $i++ )
            {
                  echo "<br><b>$cool[$i]</b> <input type=radio
                  name=daWord value=\"$cool[$i]\">\n" ;
            }

            echo '<br><br></span><input class="ssf" type=submit
            value="Add Word"></form>' ;

            echo "\n</td></tr></table></body></html>" ;

            exit ;
      }
}

$conn = mysql_connect("mysql.cs.virginia.edu", "dr3f", "password") or
die(mysql_error()) ;

mysql_select_db("dr3f", $conn) or die(mysql_error()) ;

$query0 = "select chit from DOOR where chit = '$toAdd'" ;

$result0 = mysql_query($query0) or die(mysql_error()) ;

$row0 = mysql_fetch_row($result0) ;

$dupe = $row0[0] ;

if( $dupe == $toAdd )
{
      echo '<html><head><META HTTP-EQUIV="refresh" CONTENT="3;
      URL=swap.php"><title>-(!)-</title></head>' ;

      echo '<style type="text/css">h1 { font-size: 16px ; font-family:
      verdana, arial, helvetica, sans-serif ; }' ;

      echo 'body { scrollbar-base-color: #000000 ; scrollbar-arrow-
      color: #ffffff ; }</style>' ;

      echo '<body bgcolor="#000000" text="#ffffff"><table width=100%
      height=100%><tr><td align=center>' ;

      echo "<h1>$toAdd<br><br>Already Exists</h1>" ;

      echo "</td></tr></table></body></html>" ;

      exit ;
```

```
}

$query1 = "select distinct t1.num, t1.chit from DOOR t1, DOOR t2 where
(abs(t1.ypos - t2.ypos) < 19) and (t1.xpos + t1.width + 19 > t2.xpos)
and (t1.xpos < t2.xpos + t2.width + 19) and (t1.num != t2.num)" ;

$result1 = mysql_query($query1) or die(mysql_error()) ;

$query2 = "delete from TEMP" ;

mysql_query($query2) or die(mysql_error()) ;

while( $row = mysql_fetch_array($result1, MYSQL_ASSOC) )
{
      $ins = $row["num"] ;
      $ins2 = $row["chit"] ;
      $query3 = "insert into TEMP (tnum, tchit) values ($ins, '$ins2')"
;
      mysql_query($query3) or die(mysql_error()) ;
}

$query4 = "select num, chit from DOOR left join TEMP on TEMP.tnum =
DOOR.num where TEMP.tnum is NULL order by DOOR.moved" ;

$result2 = mysql_query($query4) or die(mysql_error()) ;

$row = mysql_fetch_row($result2) ;

$toSwapOut = $row[0] ;

$toRemove = $row[1] ;

if( !$toSwapOut )
{
      echo '<html><head><META HTTP-EQUIV="refresh" CONTENT="3;
      URL=swap.php"><title>-(!)-</title></head>' ;

      echo '<style type="text/css">h1 { font-size: 16px ; font-family:
      verdana, arial, helvetica, sans-serif ; }' ;

      echo 'body { scrollbar-base-color: #000000 ; scrollbar-arrow-
      color: #ffffff ; }</style>' ;

      echo '<body bgcolor="#000000" text="#ffffff"><table width=100%
      height=100%><tr><td align=center>' ;

      echo "<h1>All Words<br>In Use</h1>" ;

      echo "</td></tr></table></body></html>" ;

      exit ;
}

$waitTime = date("U") + 8616 ;

$daFile = fopen("time.txt", "w") ;
fwrite($daFile, $waitTime) ;
```

```
fclose($daFile) ;

$chitlen = 16 ;
$badword = false ;

for( $i = 0; $i < strlen($toAdd); $i++ )
{
      switch( $toAdd[$i] )
      {
            case "a": $chitlen += 8 ; break ;
            case "b": $chitlen += 8 ; break ;
            case "c": $chitlen += 7 ; break ;
            case "d": $chitlen += 8 ; break ;
            case "e": $chitlen += 8 ; break ;
            case "f": $chitlen += 5 ; break ;
            case "g": $chitlen += 8 ; break ;
            case "h": $chitlen += 8 ; break ;
            case "i": $chitlen += 4 ; break ;
            case "j": $chitlen += 4 ; break ;
            case "k": $chitlen += 9 ; break ;
            case "l": $chitlen += 4 ; break ;
            case "m": $chitlen += 12 ; break ;
            case "n": $chitlen += 8 ; break ;
            case "o": $chitlen += 8 ; break ;
            case "p": $chitlen += 8 ; break ;
            case "q": $chitlen += 8 ; break ;
            case "r": $chitlen += 6 ; break ;
            case "s": $chitlen += 7 ; break ;
            case "t": $chitlen += 5 ; break ;
            case "u": $chitlen += 8 ; break ;
            case "v": $chitlen += 8 ; break ;
            case "w": $chitlen += 10 ; break ;
            case "x": $chitlen += 8 ; break ;
            case "y": $chitlen += 8 ; break ;
            case "z": $chitlen += 6 ; break ;
            default: $badword = true ;
      }
}

$query5 = "update DOOR set chit='$toAdd', width=$chitlen where
num=$toSwapOut" ;

mysql_query($query5) or die(mysql_error()) ;

$ipNum = getenv("REMOTE_ADDR") ;

$browserType = getenv("HTTP_USER_AGENT") ;

$query6 = "insert into SWAPIN (ichit, ip, browser) values ('$toAdd',
'$ipNum', '$browserType')" ;

mysql_query($query6) or die(mysql_error()) ;

$query7 = "insert into SWAPOUT set ochit='$toRemove'" ;

mysql_query($query7) or die(mysql_error()) ;
```

```
echo "<html><head><title>-( )-</title></head>" ;

echo '<style type="text/css">h1 { font-size: 16px ; font-family:
verdana, arial, helvetica, sans-serif ; }' ;

echo 'body { scrollbar-base-color: #000000 ; scrollbar-arrow-color:
#ffffff ; }</style>' ;

echo '<body bgcolor="#000000" text="#ffffff"><table width=100%
height=100%><tr><td align=center>' ;

echo "<h1>Word Added</h1>" ;

echo "</td></tr></table></body></html>" ;

?>
```

## 6. 500 words

a, a, a, a, a, a, able, about, across, action, actually, after, again, against, age, ago, alchemy, all, almost, already, also, although, always, am, among, an, and, another, any, anything, are, area, as, at, available, away, back, be, became, because, been, before, began, behind, being, best, better, between, big, blood, body, book, both, business, but, by, came, can, car, case, catalyst, century, cephalopod, certain, chant, child, children, church, clear, come, company, control, could, council, country, court, cyber, day, death, development, did, different, difficult, do, does, doing, done, door, down, during, each, ed, ed, effect, else, end, enough, er, er, er, est, est, even, ever, every, evidence, eyes, face, fact, family, far, father, fathom, find, first, five, flesh, florescent, for, form, found, four, from, full, further, gas, gauntlet, gave, gesture, get, getting, ghost, give, go, going, gone, good, got, graphic, great, group, had, halcyon, half, hand, has, have, he, head, headache, health, help, her, here, hieroglyph, high, him, himself, his, home, house, how, however, howl, i, i, ice, icon, idea, if, illuminate, illusion, important, in, including, incognito, indigo, information, ing, ing, interest, into, is, it, its, itself, jejune, jello, jerk, jet, job, just, kind, knew, know, known, labyrinth, large, last, later, lava, law, leather, less, level, liaison, life, like, line, liquid, little, little, lizard, local, long, long, look, love, ly, ly, made, main, major, make, man, many, market, may, me, mean, measure, members, men, metal, might, million, minister, moment, money, monolith, months, moon, moot, more, morning, most, mother, much, mushroom, must, my, nail, name, nasty, need, never, new, next, night, no, not, nothing, now, number, obelisk, obey, obscene, obscure, observation, of, off, office, often, oh, old, on, one, only, or, orangutan, orbit, ornament, other, others, our, out, over, own, pancake, paper, parachute, paradox, part, particular, particularly, party, people, perhaps, period, person, pickle, place, point, police, policy, political, porcupine, position, possible, power, power, prison, probably, problem, process, psychedelic, public, pyramid, question, quintessence, quite, r, r, rabid, rather, real, really, reason, report, research, right, room, rubber, rusted, s, s, s, s, s, s, s, s, s, saccharin, said, same, saw, say, says, school, screw, second, see, seemed, seems, seen, sense, service, services, several, shadow, shall, shape, she, should, side, since, six, sky, slippery, small, so, social, society, some, something, sometimes, sort, special, squid, staff, state, still, subterranean, such, suck, surf, system, table, take, taken, taking, tattoo, television, tell, ten, than, that, the, the, the, the, their, them, themselves, then, there, therefore, these, they, thing, think, third, this, those, though, thought, three, through, thus, time, times, to, today, together, too, took, towards, toxin, treasure, tune, two, ultimate, ultra, under, undulate, until, up, upon, us, use, used, using, vegetable, very, vessel, view, voice, want, war, was, water, way, we, weed, week, well, went, were, what, wheel, when, where, whether, which, while, who, whose, why, will, with, within, without, woman, women, words, work, world, would, wretched, xiphoid, y, y, y, y, y, yeah, year, yes, yesterday, yet, yield, yoke, you, young, your, zeppelin, zombie, zone, zoo,