A Partial Implementation of Nelson's Zigzag® ideas for Web Browsers

Last updated Thu Dec 20 00:16:13 GMT 2001

Zigzag® is a novel hypertext paradigm which involves sequences of nodes intersecting in a multidimensional space. See Ted Nelson's Zigzag Home Page for a fuller description.

Getting Started

The software can run as a plain Web page or as an XML data set. The advantage of XML is that the Zigzag data can be distributed *independently* of its presentation medium. The alternative is to translate the Zigzag information into a Dynamic HTML web page. If you have a PC and Internet Explorer version 6 or later, you can run the XML version of the software "out of the box". Otherwise, you should run the ordinary web version.

See the handbook for the latest instructions on downloading and using this software. The rest of the current page describes background work (and demos) that aren't necessarily as up-to-date!

Technical Info

This page describes an XML/XSL/JavaScript implementation of Zigzag's "quantum hyperspace" in a Web browser (IE5, NS6 or Mozilla). The interaction is achieved by dynamic HTML and the data is stored either as XML or JavaScript declarations embedded in an HTML framework.

A Zigzag world is encoded as an XML file (e.g. sample.xml) which is then transformed by an XSL style sheet into a sequence of JavaScript data declarations embedded in a standard Web page. The Web page contains a table of 20 x 20 addressable items which are used as by the JavaScript to display the contents of the appropriate Zigzag cells. (As well as the "pure" XML/XSL which is used directly by Internet Explorer, an HTML version also exists which has been "pre-compiled" by XSLT. This is because Netscape doesn't support XML and Mozilla's XML support doesn't currently allow XSL and JavaScript to inter-operate.)

Note if you have Internet Explorer, but it predates version 6, you will need to install the new XML library

Current work is going ahead on the display semantics: I believe I have these sussed, although they seem to change between the different versions of Zigzag. Since this is not a core part of Zigzag, I am adding a flexible way of expressing presentation rules which will capitalise on the underpinning HTML/CSS/FO document architecture. Watch out for fading text, overlapping layers and animated behaviours :-)

Here is a seminar given to the IAM group on this work.

Demos

For information on how to use these demos, see the Usage section below.

The Schedule Demo (IE5, NS or Mozilla) reverse engineered from the screendumps in Ted's essay *What's On My Mind*. You can see a self-playing ScreenCam presentation of this demo, if you don't have one of these browsers to hand [Download ScreenCam file (2.8Mb)] [Download zipped ScreenCam file (0.9Mb)].

The Holm Family Demo (IE5, NS or Mozilla), faithfully copied from the Linux/Perl original can now be seen without rebooting your machine. Fantastic! Since I couldn't decode the original .zz file, the items and connections are all reverse-engineered, and hence may not be 100% accurate!

The PicZag Demo (IE5, NS or Mozilla), is a test which just demonstrates that (this being a Web browser) you can ZigZag any kind of data: pictures, sound or videos.

The London Underground Demo (IE5, NS or Mozilla), is under construction. It contains the basic infrastructure of the Central London routes of the tubes in terms of the nine major lines (Bakerloo, Central, Circle, District, Hammersmith & City, Jubilee, Northern, Picadilly and Victoria, but not currently including the places where a single line branches). Also included is a dimension for a tour of attractions above ground and a dimension that connects the underground stations with the tourist attractions above. Whenever a tourist attraction is encountered, a Web page is displayed which contains information about that attraction. See underground.zigzag for a version of this demo that can be XML.IMPORTed into Gzigzag version 4.

The Function Demos show how a function can be evaluated as part of the automatic rendering of a cell (IE5, NS or Mozilla), or when requested by a user (IE5, NS or Mozilla).

Usage

This implementation is fairly minimal! It only supports ONE window, and currently implements much of the Zigzag infrastructure directly as JavaScript variables rather than Zigzag cells (e.g. dimensions).

Navigation
: The e, c, s and f keys are supported, as well as the x and y keys to rotate dimensions. Capitalised versions of the letters map onto the SAME actions currently. (Clicking on a cell, or using the arrow buttons and dimension menus in the top region perform the same actions for those more familiar with the Web than Zigzag.) Each cell shows how it is linked to its neighbouring cell with a large (colored) arrow. Small (diagonal) arrows show how the cell is linked in other (unmapped) dimensions. Hovering over one of these arrows indicates the dimension and cell value, clicking on the arrow will move you to that cell and rotate to make that dimension visible.

Tutorials
: Each dataset can have an accompanying tutorial which will be displayed step-by-step in the top left-hand part of the Web page. A change in instructions is triggered when the user navigates to the correct cell with the correct dimensions displayed.

Editing
: The 'Q' key will allow you to edit the contents of the current cell. In IE5, the cell itself will turn editable, and you can make selections, insertions, deletions and changes until you press 'ESC' when the changes will be comitted. In NS/Mozilla, a dialog box will be presented to you to allow you to type in the new cell contents.

Adding Cells
: The 'n' key followed by a direction key (s, f, c, e) will add a new cell and link it to the current cell *in that direction*. The new cell will become the current cell. (You must then type 'Q' to change its contents.)

Adding Dimensions
: The 'D' key will initiate a series of dialog boxes which ask you to type in a name (id to use internally), a description (used in the menus) and color (for the arrows) for the new dimension. The new dimesion is added at ethe end of the dimension list.

Linking Cells
: First mark a cell using the 'm' key. Then go to the cell you want to link it to, and press '-' and a direction key (s, f, c or e). The marked cell will then be linked to the current cell *in the indicated direction*.

Saving
: Scripting languages embedded in a browser are not allowed to access the user's hard disk! Pressing the '!' key will bring up a new window containing an XML representation of the current Zigzag workspace in a textbox. Please copy and paste this text into a text editor and save it yourself. (Sorry about that!)

Evaluation
: Some preliminary work has been done on Zigzag programming, based on Scheme (since ranks are like lists). The 'A' key (for 'Answer') will evaluate the function connected to the current cell in the 'defn' dimension and display the answer in a dialog box.

Common Gotchas
: *Please note in the demos that after choosing a dimension from eiither of the menus, you MUST click on the background to delselect the menu. If you forget to do this then the 'e', 'c', 's', and 'f' keys will be interpreted by the browser to mean "make a new choice from the menu", rather than "move up/down/left/right".*

Useful Files

- Here is an empty Zigzag dataset (IE5, NS or Mozilla), for starting your own examples.
- The Zigzag DTD for verifying your XML file
- The XSL stylesheet for transforming the XML file into an HTML page with embedded JavaScript declarations.
- The external file of JavaScript function definitions that operate on the data in the HTML file.

A Word about Research

Ted has defined an intriguing paradigm for information organisation by mixing two well-known hypertext constructs. Zigzag can be seen to consist of typed links and sequences, however it is the discipline with which the links are applied and the interpretation imposed on their use that makes Zigzag so peculiar (in the sense of individual, outstanding and noteworthy). As well as hacking together a low-impact Zigzag viewer that I can use with other tools (believe it or not I use Excel to create the links in the undergrouynd demo) I am attempting some analysis of the expressivity of these hypertext structures.
