GZIGZAG
A Platform for Cybertext Experiments

Tuomas Lukka & Katariina Ervasti

Introduction

”Computers are fundamentally broken!” lectures Ted Nelson. Unlike many other critics, he also offers ideas for improving the situation. Some of his ideas are currently being implemented by the Hyperstructure Group at the University of Jyväskylä, Finland.This article describes GZigZag, which is currently the Group’s main project. GZigZag is an implemention of ZigZag, a computer paradigm invented by Nelson. The paradigm abandons many concepts currently perceived as central to computing, such as folders, files, and applications. Instead, it offers a more flexible way to arrange information. At this early stage of development GZigZag already has advantages as compared with other computer systems. It could also prove to be a fruitful platform for new kinds of cybertextual writing experiments.

Defamiliarization of files, folders, and applications

Nelson (1999a) offers several reasons for why users face difficulties with present PCs. This article only includes a short attempt to defamiliarize[1] folders, applications, and files from the user’s point of view. It is not an easy task, because folders, applications, and files belong to the first things taught to a beginner, and as it is, they are not often questioned. Hierarchical directories, also referred to as ’folders’, were invented to help finding the right file among many files (Nelson 1999a). Let us imagine that in September 1999 a writer has been writing an article dealing with the impossible

nature of her cat Vilma. In September 2000 she wants to find the article again to edit it for a new purpose. In order to find the article, she opens a folder named 'Vilma'. The folder includes, let us say, twenty files, which are either different drafts of the article, or notes including ideas the writer considered worth writing down at some point of the writing process. The files have names such as vilma3.doc, vilma4.doc, vilfoo.doc and vilmaprob.doc. By the time the writer finished the article she had no time to write out an index explaining the contents of each file. Finally, after opening and closing several files, she succeeds in finding the right file and starts working. When editing, she suddenly remembers that she had a slightly different version of a certain paragraph in another document. Once again, she has to start opening and closing files to find the right one.

Applications, then, are used for performing different tasks with the computer (Nelson 1999a). Problems arise when a user wants to use the same information in many applications. For example, a multimedia author who has manipulated some sound with SoundEdit, might want to use that sound in a multimedia presentation made with Macromedia Director. Since applications do not support all existing file formats, he has to find out which sound file formats Director supports. After a study of sound file formats, he saves the file in a suitable format and imports it to Director. But once again, if he later views the presentation and suddenly remembers another sound sample recorded during the same session, which he might want to use now, there will be no easy way to find that particular file. This is because there is simply no connection between the Director file and the original sound sample, nor between the original sound sample and the second sample from the same session.

These examples demonstrate how the files and folders based model for storing information is insufficient: it does not allow the users to track down the conceptual relationships between interrelated pieces of information, or, different versions of certain content. Files, folders, and applications are easy to understand, but – as Nelson explains – they do not need to be the fundamental concepts of software. The problems described above could be solved by designing software in a wholly different way, starting from different assumptions.

The traditions of Bush and Engelbart

Vannevar Bush and Douglas Engelbart both developed ideas for tools which would improve the working conditions of people performing complicated tasks in the complicated world. Bush, who had noticed the explosion of information in the 1940s already, is famous for proposing Memex, a ”mechanized private file and library” designed to help individual scientists store and handle the growing amounts of information needed in their work (Bush 1945). Engelbart, the developer of NLS, a tool for collaborative work, writes about the augmentation of man’s intellect, which has been the goal of all of his work with computers. By augmenting man’s intellect he means ”increasing the capability of man to approach a complex problem situation, gain comprehension to suit his particular needs and to derive solutions to problems” (Engelbart 1962). The purpose of our work is similar to Bush and Engelbart: we want to find better ways for the production and arrangement of information by creating, as Nelson (1999b) expresses it, ”a high-power personal and media system, with editing and presentation systems that expand the state of art”.

Basics of ZigZag

Defining ZigZag is difficult, because it is so different from any other software in the currently dominant computer paradigm. It is not an application, an operating system, or a platform. It is a new way of putting information into computers, a cross between a database, a filesystem, a personal information manager, and many other things. ZigZag is simply something new and different.

Cells, dimensions, views, and applitudes

A ZigZag structure consists of cells and dimensions. A ’cell’ is the basic unit of information in ZigZag. A cell can contain an information unit of any kind, for example a text string (e.g. ”Vilma”), an image (e.g. a picture of Vilma), or sound (e.g. recorded ”Meow!” by Vilma). Cells can be connected to each other along ’dimensions’, which are referred to with names such as *d.1, d.cursor*, *d.jokes*, or *d.comments*. The number of dimensions

is not restricted, and it is easy to create new dimensions. For example, if Ville wants to write down comments concerning several different cells, he could use *d.Ville*-comment for connecting his comments to the cells. On each dimension each cell has two ends, a negative end and a positive end, and on each dimension only one cell can be connected to one end. This means that on each dimension a cell can have at most two neighbours: a predecessor, which is connected to the negative end of the cell, and a successor, which is connected to the positive end of the cell.

Figure 1 shows a simple structure. In the figure, cells are represented by rectangles, and neighbours along a dimension by a line.

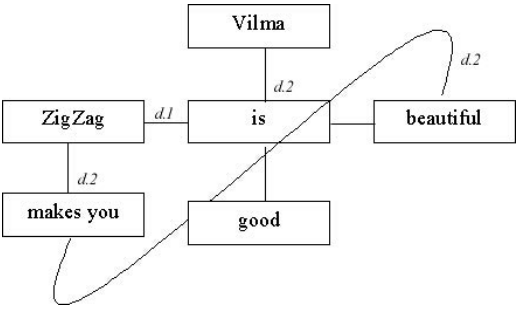

Figure 1. Seven cells, connected to each other along the two dimensions *d.1* and *d.2*

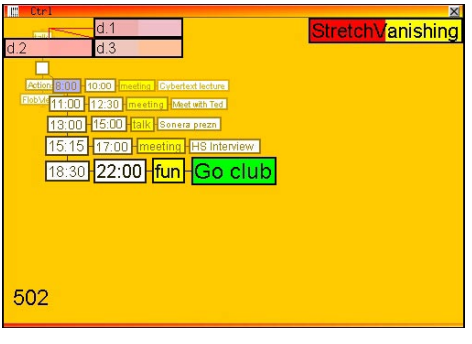

Figure 2. The most important events of a day in a general view

There are several different visualizations (views) of the ZigZag structure. The views range from 'general views' that are useful for looking at all kinds of structures to 'specific views' that are useful for only one particular kind of structure. For example, Figure 2 shows a generic view of a structure that represents a day schedule.

Figure 3, then, shows a specific view of the same structure. The underlying data is exactly the same, but the specific view – designed especially for the purpose – interprets the structure and draws the events in a visually more intuitive way. Looking at another structure through this view would usually not make sense, because the view is designed especially for this structure.

Combining the specific view, such as the schedule view above, with special operations for editing such a structure (for example dragging the start and end times with the mouse) makes an 'applitude'. Thus, an applitude consists of views and operations designed for a particular purpose. Even though the term 'applitude' resembles the term 'application', there is an important difference: in ZigZag nothing is separate, and applitudes, unlike applications, can be combined with each other, as the following example will show.

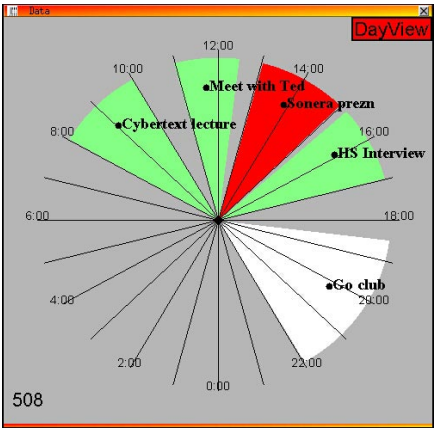

Figure 3. The same events in a specific view

Example: Address book and Family tree in GZigZag

One of the first examples which Nelson has used to demonstrate ZigZag is the Holm Family Demo, a family tree prepared for his talk at the University of Oslo, to show how he is related to one of the professors at the University. Here, a variant of Nelson's original example is used, combined with an address book.

The structure of the address book is simple: it is a list of names and addresses. The names are listed along the dimension *d.2* in alphabetical order. The addresses are connected to the names along the dimension *d.1*. Figure 4 shows the (incomplete) address book in the Row View.

Since the list of relatives is long, only a subset of it can be seen on the screen at a time. In Figure 4 the cursor is on the cell "cousin 1". Moving the cursor downwards would cause more cells below the cell "grandfather 2" (on dimension *d.2*) to become visible. Next, the address book is combined with the family tree, which is represented by a slightly more complicated structure, shown in Figure 5.

The two dimensions *d.marriage* and *d.children* are used to represent the family tree. Siblings are connected along *d.children*, and married couples along *d.marriage*[2]. An extra cell ("+") is used on dimension *d.marriage* to make the structure symmetric, and the list of children from this marriage (on dimension *d.children*) starts from that cell. Figure 5 and Figure 6 show two different views of the structure. The Row View, as Figure 5 shows, enables dealing with one family at a time. The Vanishing View, shown in Figure 6, gives a more thorough picture of the family tree as a whole.

It is important to understand that the same cells are used to represent the relatives in both the address book and the family tree, but the connections related to the two structures are on different dimensions. In the default views only two or three dimensions (x,y,z) can be shown on the screen at a time. As can be seen from Figure 4, the dimensions used for viewing the address book are x=*d.1* and y=*d.2*. Rotating the dimensions to x=*d.marriage* and y=*d.children* shows the family tree, as in Figure 5.

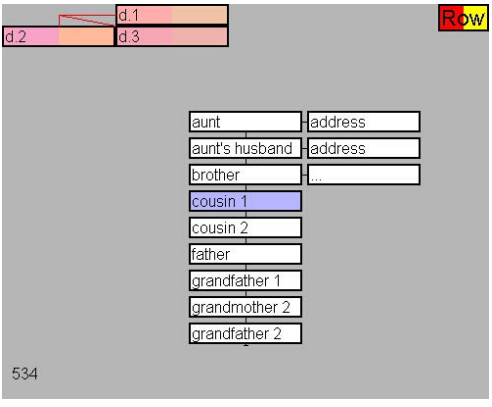

Figure 4. The address book in the Row View

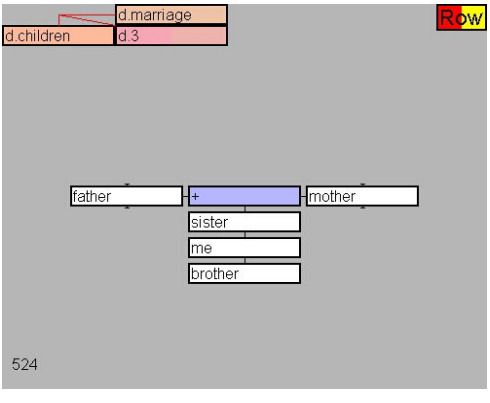

Figure 5. One family of the family tree in the Row View

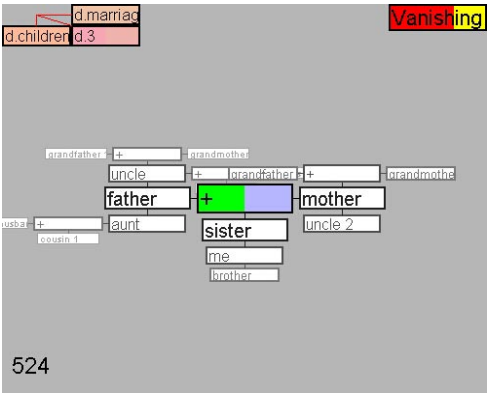

Figure 6. The family tree in the Vanishing View

The address book and the family tree could still be combined with further structures. For example, a cell representing a certain person could also be connected to photographs and emails related to that person. Simply, in GZigZag everything can be connected with everything else, based on the user's associations. One consequence of the ZigZag structure is that all connections are bidirectional, which means that navigating between related bits of information is easy.

Utility of GZigZag

ZigZag offers several advantages compared to existing computer systems. To begin with, as Nelson usually remarks after showing his Holm Family Demo, "we did not create a genealogy program". Modeling a complicated structure, such as the family tree, on usual computer systems would require one to create a specific program for that purpose. Modeling a complicated structure using GZigZag requires one only to create some new cells and connect them along dimensions.

Remarkably, there are no separate files and applications in ZigZag. As seen above, the same cells can simultaneously be parts of different structures without any restrictive boundaries. Thus, a multimedia author using GZigZag would not have the same problem of file formats as the multimedia author described in the example above. He would not have to bother to find out what file formats a multimedia authoring applitude supports, since it supports everything. Because the same cells can be used in various structures, also updating information becomes easier. Updating the last name of a newly married aunt both in the family tree and in the address book requires updating only one cell.

ZigZag also offers a more flexible way to arrange information than the conventional model of files and folders. A certain piece of information is found by following connections the user has previously made based on his associations. Hence, a user of GZigZag does not need to remember any file names in order to find the information he is looking for. The above mentioned writer looking for the document containing an interesting paragraph about Vilma and the multimedia author looking for the sound file could simply follow a connection made previously.

Finally, ZigZag separates the structure and the visualization of information. This is somewhat similar to HTML 4.0 and CSS (Cascading Style Sheets), but ZigZag generalizes the concept: all structures and all visuali-

zations are possible. The same GZigZag structure can be used in different media from mobile phones to the immersive virtual reality of the CAVE, because different visualizations can be constructed to take full advantage of each medium.

Conclusion and future work

The primary purpose of this article (which is the first publication of the Hyperstructure Group) has been to present a short summary of our ongoing work. Because we are dealing with such a different view of computing it is very difficult to find ways to express our ideas – indeed, the difficulty of explaining new ideas to people has been one of the main problems of Nelson's broader Xanadu project – and it is even harder to try and imagine all the possible ways to use this model. There seems to be, however, a great potential for all kinds of participatory and interactive writing experiments in this model, where reading, writing, rewriting, restructuring, and programming all happen in the same space.

In the near future we will focus on developing a stable, working GZigZag on the Java platform, and, a cellular language Clang which would make programming in GZigZag environment easier. We are also planning a network protocol for exchanging cells between computers. Furthermore, we continuously develop applitudes for several purposes in order to learn more about the system[3]. We believe that, ultimately, Nelson's original and creative ideas should be understood and implemented. Our long-term goal is to develop a computer system we would like to use ourselves.

Acknowledgements

We would like to thank Theodor Holm Nelson and Marlene Mallicoat for collaboration. All Nelson quotes without reference are from private discussions with him. We would also like to thank the other members of the Hyperstructure Group: Tuukka Hastrup, Antti-Juhani Kaijanaho and Vesa Parkkinen.

NOTES

1. According to Fowler, defamiliarization is the use of a strategy to force us to look at familiar things in a critical way, to see the absurdity of a familiar object. Criticism, as Fowler sees it, is not a negative practice. The basic motivation for criticism is ”healthily sceptical inquisitiveness”, which can give a stimulus to develope things for the better (Fowler 1986, 34–35, 42).
2. If a person has been married several times, a mechanism called cloning is used to represent this in the structure. However, this is beyond the scope of this article.
3. Everyone interested is welcome to test and work on the current version of GZigZag, which can be downloaded from the project’s homepage: <http://gzigzag.sourceforge.net/>. GZigZag is a free software project: the source code is released under the LGPL license and interested parties are welcome to join our mailing list.

REFERENCES

Bush, Vannevar (1945) ”As We May Think”, *The Atlantic Monthly* 176:1 (July), 101–108. Available as electronic document: http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

Engelbart, Douglas (1962) *Augmenting Human Intellect: A Conceptual Framework*. Stanford Research Institute Summary Report. Available as electronic document: http://www.histech.rwth-aachen.de/www/quellen/engelbart/ahi62index.html

Fowler, Roger (1986) *Linguistic Criticism*. Oxford: Oxford University Press.

Nelson, Ted (1999a) *Ted Nelson's Computer Paradigm, Expressed as One-Liners*. Electronic document: http://www.sfc.keio.ac.jp/~ted/TN/WRITINGS/TCOMPARADIGM/tedCompOneLiners.html

Nelson, Ted (1999b) *ZX Views*. Electronic document: http://www.xanadu.com/FW99/ZXviews.html