Computers & Graphics 37 (2013) 202–213

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Using visualization for visualization: An ecological interface design approach to inputting data ☆

H. Wright [a,*], C. Mathers [a], J.P.R.B. Walton [b]

[a] Department of Computer Science, University of Hull, Hull HU6 7RX, UK
[b] Numerical Algorithms Group Ltd, Jordan Hill Road, Oxford OX2 8DR, UK

ARTICLE INFO

Article history:
Received 11 January 2012
Received in revised form
10 January 2013
Accepted 20 January 2013
Available online 9 February 2013

Keywords:
File organization
Information search
Ecological interface design
Pattern recognition
Scientific visualization

ABSTRACT

Visualization is experiencing growing use by a diverse community, with continuing improvements in the availability and usability of systems. In spite of these developments the problem of how first to get the data in has received scant attention: the established approach of pre-defined readers and programming aids has changed little in the last two decades. This paper proposes a novel way of inputting data for scientific visualization that employs rapid interaction and visual feedback in order to understand how the data is stored. The approach draws on ideas from the discipline of ecological interface design to extract and control important parameters describing the data, at the same time harnessing our innate human ability to recognize patterns. Crucially, the emphasis is on file format discovery rather than file format description, so the method can therefore still work when nothing is known initially of how the file was originally written, as is often the case with legacy binary data.

1. Introduction

Visualization provides scientists, researchers and engineers with an invaluable tool for understanding their data. Since coming to the fore in 1987 [1], work to improve the usability of visualization systems has variously addressed the problem of data representation [2], technique selection [3], and satisfying certain goals or interpretation aims [4–6]. More recently, improvements have been made in securing the provenance and reproducibility of visualizations [7,8] and in tackling the complexity of using visualization software [9,10]. However, the problem of data input continues to receive comparatively little attention, even though it is estimated by some expert practitioners to consume up to 90% of the effort when visualizing clients' data [11].

In this paper we present a novel approach to inputting visualization data that uses visualization at the earliest stages, even before the file structure is completely understood. After summarizing conventional approaches we apply the principal features of ecological interface design (EID)—work domain analysis, the abstraction hierarchy of means–ends relations, and constraints arising from the work domain—to describe the problem of file input. This approach encourages thinking about the constituent activities of file input as a set of distinct methods that can be combined in multiple ways, rather than the traditional approach of constructing multiple self-contained recipes. We go on to demonstrate our software implementation in the IRIS Explorer visualization package [12] with two interpretations of binary data carried out *a priori*. The paper concludes with a review of the intended scope of the work and its perceived contribution.

2. Existing approaches to inputting data

Data intended for visualization falls into two broad types: it exists either (1) in some pre-defined format or (2) in some user-defined or unpublished format. Herein may lie the reason for the lack of attention paid to this problem, since the former situation is generally considered to be resolved whilst the latter appears unresolvable in general. It is indeed tempting to assume that inputting data according to some pre-defined standard is a straightforward matter, but sometimes it is not. Visualization software that supports multiple file formats usually requires a different reader function for each format, so the development effort required can be substantial. In the field of chemistry alone, for example, there are over 50 different file formats in use [13].

There are several different types of visualization software but all employ essentially similar approaches to the support of pre-defined formats. Turnkey visualizers that are designed to work without customization by the user generally have a number of file formats that they can import; systems begin by supporting a small subset of definitions which grows over time with each new

☆ This article was recommended for publication by M. Chen.

* Corresponding author. Tel.: +44 1482 465247; fax: +44 1482 466666.
E-mail address: h.wright@hull.ac.uk (H. Wright).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.01.013

release. The class of general-purpose, customizable tools known as modular visualization environments (MVEs; [14]) all include a set of reader modules for common, pre-defined file formats, though this set may differ from one MVE to another. As with turnkey systems, it is common for the number of reader modules to increase as a system matures due to the contributions of vendors and aficionados.

Where the generating application writes data in some user-defined, unsupported or unpublished format, the functionality of the visualization system must be extended, either by programming code to read the file or describing its format using a built-in tool. For example, IRIS Explorer's API allows users to incorporate their own code into the visualization system by encapsulating it within a series of 'wrappers' [12]; VTK [15] provides various reader classes that the user can extend to fulfill their needs. Examples of built-in tools for file input are IRIS Explorer's QuickLat [12] and DataScribe [12] tools, the Data Prompter and General Array Importer which are components of Open DX [16] and the AVS/Express Add File Import Tool [17].

Help is therefore at hand, but tackling the problem at the programming and file description level ignores a key element: users of scientific visualization often know something of how their data may look, or have useful clues as to how it is stored or was generated. They may be able to recognize when the representation is faulty, but such recognition usually begins only after the data has been read in, when the visualization process itself is underway. Making better use of partial knowledge *during* file input has motivated a new approach—a mechanism to input data on a per-solution basis, incorporating interaction combined with visual feedback to guide the process. This philosophy contrasts with existing methods that require prior knowledge of the file's format in order to make any attempt at reading it. Furthermore, because the process is intentionally iterative there is no tendency for the whole reader to collapse entirely when any small detail is overlooked. The result is a set of tools which, whilst they can be used in the conventional way to apply known formats, can also be used to mine visually for unknown file storage parameters. This includes binary files for which, if nothing is known, nothing can easily be gleaned. It is this latter property which proves the most valuable, in some circumstances yielding complete solutions for file input problems that would otherwise be completely intractable.

3. Inputting data—the ecological way

Classically the ecological approach occurs in continuous-process control, where operators oversee and adjust outputs by direct interaction with information coming from the production plant (for examples see [18–21]). This section first describes the foundation principles of EID and then applies these to the problem of data file input.

3.1. Ecological interface design in brief

The EID framework was devised to reduce the rate at which human errors arise during the control of complex systems and to mitigate their effects should they occur [22]. The framework recognizes three cognitive control mechanisms, namely skills, rules and knowledge (SRK; [23]), which in turn give rise to skill-based, rule-based and knowledge-based behaviour (SBB, RBB, KBB). The EID approach aims to support interaction at the lowest appropriate level of control, whilst at the same time providing support for higher levels, as needed [22]. Thus for the most part the user will simply be reactive to signals provided by the display (SBB), and preferably will act on the display directly. At a higher level the use of rules may be prompted by the emergence of familiar scenarios (RBB) or, more rarely, problem-solving activity (KBB) will be undertaken on the system as a whole [24]. In this way, efficiency can be maximized during normal operations, whilst at the same time sufficient information remains available to address unusual situations safely.

A valid question at this point is how EID, conceived in order to reduce the effects of human error, is relevant to the problem of file input. If we adopt the conventional approach of a file reader working from an established format then the reader will either work if the format is correctly understood or fail if it is not. The human has little to do in this situation, other than look for another reader. However, the approach we will take is an unconventional one of testing various hypotheses about the type of the data, the size of an array and the meaning and use of the variables. Such a *trial-and-error* approach may well already be the technique of last resort for inputting unknown file formats but can be impossibly slow when the parameter search space is very large and the support tools are weak. As such we can see parallels with the principles of EID: the need for efficiency (in this case, to test many hypotheses); the need to correct a previous assumption or action; and, the need for visibility of the current (possibly imperfect) overall system state. This property of goal-oriented behaviour resulting from perception of the environment is central to the well-known ecological theory of visual perception [25], from which EID in turn derives its name. A key difference, however, is that the natural ecology (our observed surroundings) is visible whereas the system state is normally invisible. Characterizing system state as a collection of variables and their inter-relationships is therefore central to the EID approach, and accomplishing this for data file input is the subject of the next two subsections.

3.2. Work domain analysis

Work domain analysis (WDA) aims to present a system at different levels of abstraction, the so-called abstraction hierarchy (AH). Levels are linked by means–end relations that answer the questions Why?, What? and How? to achieve a particular goal [26]. For example, interpreting a byte sequence in a particular way (the 'How?') is the means to the end of understanding the numerical values in a file (the 'Why?') and the concept (the 'What?') relating the two is the primitive type of the data (Fig. 1(a)). The means–end relations are not anchored to any particular level: manipulating the number and sizes of an array's dimensions, for example, is the means, along with the length of the chosen primitive type, of ensuring it accounts for all the values in a file (Fig. 1(b)). To enable flexible support in unanticipated scenarios, the AH aims to capture all such links and, importantly, to show how decisions may interact. Work domain analysis is thus very different to task analysis, which more closely

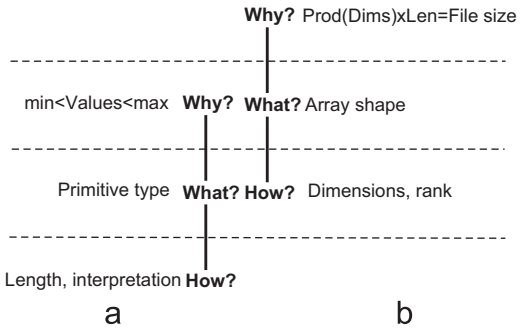

Fig. 1. A selection of the purposes and functions pertaining to data input, organized as means–end relations. Means–end relations answer the goal-oriented questions 'Why?', 'What?', and 'How?' (cf. [26, Fig. 7.10]).