Virtual but Visible Software

Claire Knight and Malcolm Munro

Visualisation Research Group, 
Department of Computer Science, 
University of Durham, 
Durham, DH1 3LE, UK. 
{C.R.Knight, Malcolm.Munro}@durham.ac.uk

Abstract

The problem of trying to view and comprehend large amounts of data is a well-known one. A specialised version of this problem, that of visualising program code (often known as software visualisation) has suffered from the field refusing to embrace new ideas and technologies. This paper presents an application of virtual reality to this problem and highlights areas of importance from virtual reality that have been used to good effect when creating graphical abstractions of Java source code. A summary of the achievements of this approach and the relationship to the virtual reality features believed to be most beneficial is provided.

1. Introduction

Visualisation is necessary in today’s arena of mass information generation to provide one way of filtering this volume of data into something more manageable. Virtual reality techniques provide one way of enabling these visualisations and afford many benefits that more traditional visualisation systems lack.

Through the use of three dimensions and suitable metaphors, information spaces or information landscapes can be created in which users have the freedom to explore and browse at will. Such techniques are useful for many types of information visualisation and this paper documents a use of virtual reality visualisation for a subset of information visualisation; software visualisation. In particular, this work focuses on software visualisation for the purposes of aiding the program comprehension process. It is an important area, not least because it is a major part of the maintenance and evolution of software systems. Very rarely is a system coded completely from new, which means existing code fragments (if not more) need to be understood before any effective additions and alterations can be made.

Software visualisation has been around in varying forms for many years but the problem is that the field is still using node and arc structures in two dimensions to visualise ever larger and more complex software systems. The older visualisation techniques have identified deficiencies but little research has been done to create different representations and abstractions to try and address these failings. The use of three-dimensional visualisations in particular has received little attention due to the need to embrace a completely new approach.

The research described in this paper is a foray into this area of three-dimensional software visualisation and shows the feasibility of such an approach. Information about visualisation and virtual reality features that are strongly related is provided. This is then followed by details of the research, which ties in with the features discussed in the earlier sections. The paper is concluded with a view of where the research could be applied elsewhere and then other future research directions.

2. Information Landscape Features

There are many features of virtual reality that can be utilised to good effect when visualising data artefacts but for clarity and conciseness two that have been the main motivating factors in this research are discussed.

2.1 Metaphors

From a virtual reality perspective the metaphors act as a mapping from the concepts or artefacts required to be displayed in the virtual world to their graphical
representation. According to Benford et al. [2] the use of natural metaphors can aid the usability of virtual reality environments.

“... an attempt to exploit people’s natural understanding of the physical world, including spatial factors in perception and navigation, as well as general familiarity with common spatial environments…”

The research being described in this paper has tried to harness natural perceptual skills in this way, through the use of a very strong real world metaphor.

Petitfer and West [3] suggest that the potential power of virtual reality comes from the strength of its metaphor, and the fact that it is closer to natural interaction than many other forms of computer system. They also identify the benefits of natural metaphors, and making use of perceptual and spatial skills learnt and used in the real world in the virtual environment.

“A three-dimensional world metaphor has much more scope for direct human/computer interaction than the two-dimensional desktop because it engages in us those perceptual and spatial faculties that allow us to comprehend our surroundings and to process effortlessly the vast amounts of information that are presented to our senses second by second. It is the potential to directly engage these faculties that is the defining characteristic of virtual reality. As the immersive environment is far richer than the desktop, the metaphors for interaction assume a far greater significance. … The role and management of metaphors for the virtual environment therefore assumes key significance.”

It is obvious from the above that the design of the metaphor used in the virtual reality can play a large part in the usability of that system, both in terms of human computer interaction, and in terms of enabling the user to carry out the required tasks. In this case the comprehension of the software system through the use of visualisation.

What is also of benefit is that in using three-dimensional environments some of the cognitive processing needed for navigation and visual interpretation can be shifted to the sub-conscious as these are activities that are carried out daily with no real conscious thought.

2.2 Navigation and Orientation

If the virtual reality environment is a representation of the spatial world that we already know then there is a need to model orientation and navigation features found in the real world. In any spatial setting some form of base orientation needs to be found which can then be used for navigation and re-orientation as movement occurs. Hemmje et al. [11] relate this to their database visualisation work although what they write is readily extendible to all spatial visualisations.

“... It is necessary to move, i.e. change position in the context space and explore information visible from each point of view. It is important to achieve an orientation, i.e. to determine the relation between a current point of view (e.g. from an information item) and the whole of an information space.”

Ingram and Benford [4] relate the orientation and navigation processes to the cognitive map the user has of the environment. Cognitive maps can be one of two sorts. Linear maps are based on movement through the space and the observations made during that movement. Spatial maps do not require movement through the space. Generally linear maps are the first created of an environment, and over time the map may evolve to being a spatial map. Exploration rather than guidance through an environment encourages the development of a spatial map.

Petitfer and West [3] relate the problem of losing oneself in virtual space to the systems and metaphors in use today.

“Loosing a cursor on the desktop is one thing, loosing yourself in cyberspace is quite another.”

Three-dimensional worlds are potentially infinite whereas desktops are of generally finite space even if current implementations are able to cover several screens.

The aim, when constructing any sort of virtual reality that requires navigation and orientation, must be to create them so that they correspond with known perceptual abilities of humans. Pesce [12] asserts

“The first perogative in the engineering of a holosthetic environment is: design to avoid disorientation. Disorientation represents a step towards the amputation of the self, and necessarily precedes the dislocation of self that concludes in holosthetic psychosis.”
Another aspect of perceptual orientation, often missed, is that of causality. It provides a continuity of experience in “reality” so by providing such continuity in virtual realities allows natural comprehension, interaction and orientation. This is not implying that the causalities need to model exactly the laws of time and motion, but that the “laws” used in the environment need to be continuous throughout that environment, allowing things to be comprehended, and to an extent, explainable. A ball floating in mid air is considered strange, but provide a context of outer space and the ball’s behaviour is perfectly acceptable!

Solutions to the problem of getting lost in virtual space often take the form of orientation cues, predefined navigable paths, overview maps and the use of natural navigable metaphors. The use of spatial metaphors allows the user to create a mental representation of that world, as they do in reality in moving through their natural environment. What can be missed in these artificial environments are the navigational cues. By creating a rich graphical metaphor suitable information can be provided for users to allow them to navigate their way through the data artefacts.

3. Virtual Reality for Visualisation

A general software visualisation definition by Knight and Munro [1] is:

"Software visualisation is a discipline that makes use of various forms of imagery to provide insight and understanding and to reduce complexity of the existing software system under consideration."

To avoid confusion, this is what is meant when the term software visualisation is used throughout paper. Visualisation has been touched upon in the preceding sections but only in isolation. In this section the virtual reality attributes that can be harnessed, in particular with relation to software visualisation, are discussed with reference to both fields. Just to indicate, visually, the reasons for needing to find different way of presenting information about software Figure 1 shows the calling structure of a well maintained medium sized commercial software system!

![Figure 1 - Traditional software visualisation; a call graph showing the calling interactions between all the functions in the software.](image)

There are several features of visualisations that are well supported in virtual reality environments. One of these is the imposition of form on shapes and colours. Friedhoff and Benzon [5] write:

"The tendency to impose form, whether on the surface irregularities of a cave, or in inkbloths, clouds or shadows, is suggestive of an organising function of vision – an organising tendency so strong that random shapes can trigger the perception of vivid illusions."

This information is something that can be exploited by visualisations and virtual realities. The graphics depicting a scene can be generated with this perceptual knowledge in mind. For example, very little detail can be used on far objects since it does not detract from the virtual space and when investigating the data missing information can be visually filled in by the brain. When more detailed knowledge is required then the user can interact with the virtual space and make that data item the main focus.

Perception and the cognitive mapping process (covered from a virtual reality perspective in §2.2) are important for visualisation because of the cognitive processes involved in the comprehension of the underlying data. The build up of knowledge in a virtual reality visualisation happens on many levels; navigation of the graphically depicted space, navigation of the data represented, orientation within the graphics and finally orientation within the dataset. Gardner [6] summarises this perceptual learning through experience and exploration as:

"These perceptual associations are not so strange. People pick up a lot of rules as they go through life. Not necessarily hard and fast rules;
people use a kind of fuzzy logic. When we see something, we relate it to other things.”

In deciding to move in a particular direction there is very often the ulterior motive of wanting to move from one location to another (i.e. in search of the next data item), not just to move along that route. It is the perception of the space/world and accompanying that, the data, which is part of the decision about which path to take. Generally the physical world properties are not compelling in their own right, other than to avoid obstacles when moving from one place to another!

In presenting spatially arranged and located information, i.e. visually, there is the benefit that the human perception skills can be used for part of the comprehension. This moves some of the comprehension load away from the conscious cognitive processing. The double navigation and orientation load of virtual reality visualisations could be seen as a burden but it can be useful. The two mapping processes, that of virtual space movement and of the data comprehension, can feed each other and help to build up a solid mental representation of both.

Dieberger [13] ties together metaphors and navigation in information landscapes and writes

“When talking about navigation in information spaces we automatically use metaphors but we do not fully use these metaphors. Metaphors are incomplete mappings from a source to a target domain thus carry certain restrictions. We should see these restrictions not as obstacles but as devices to communicate structure to our users.”

and later in the paper

“I get the impression that we don’t use metaphors to their full potential in navigation. They are not only vehicles to make something easier to understand, but – especially in the case of information spaces – they are also structuring devices.”

Such factors have been incorporated into the metaphor design carried out during the course of this research. They can, if the metaphor definition is well thought out and thorough, provide the capability of creating powerful metaphors. These metaphors can then convey not only the data of the information landscapes, but provide navigation and orientation information that is based on that information. This means that evolving data sets with suitably evolving visualisations (the visualisation definition needs to support this evolution) can still remain familiar because the landmarks, and landscape to data relationships will evolve with the data thus providing a degree of cognitive familiarity.

Using three dimensions for visualisation adds an element of familiarity and realism into systems. The world is a three-dimensional experience and by making the visualisation more like that world means there is less cognitive strain on the user. This in turn makes the system easier and more comfortable to use because of all the experience and knowledge the user has built up elsewhere. In using three dimensions the depth cues that make the world, and the visualisation, appear three-dimensional can be used as part of the visualisation. This means that the aim of the visualisation to aid the comprehension of complex phenomena can be achieved without adding unnecessary complications because of the visualisation used.

One of the main problems for software visualisation (and other forms of information visualisation) is of trying to create a tangible representation of something that has no inherent form. Therefore the aim is to visualise the intangible in an effective and useful way. Effective and useful here refer to the visualisation being able to increase the understanding of the user whilst reducing the perceived complexity. Walker [7] comments on the software being the intangible part of information systems when he writes

“Some aspects of an information system are tangible, but a major component is the software which is an abstract and invisible collation of computer instructions.”

Chapin and Lau [8] also recognise the intangible nature of software

“Furthermore, software is intangible, and it is only the representation of the software which can be communicated between people and between people and computers.”

An important point to be drawn from this is the communication aspect. Since software is intangible and each programmer has his own mental representation then an effective visualisation can also act as a common frame of reference. In discussing pieces of the software either informally between colleagues or formally in meetings, if the participants do not concur over the code being discussed the discussion may as well not take place. Visualisation of the software can provide not only a graphical representation of the piece of code under
discussion (for clarity over the section being discussed), but also allow the discussion to take place in the realm of that visualisation; in the virtual space itself. This means that the discussion can be based around the visualisation and the code it represents rather than the piece of code. In doing this, the visualisation has provided a starting point for common understanding.

4. Software Visualisation; Software World

The discussion presented in the preceding sections was used as a driving force behind the research done to lead to the creation a three-dimensional visualisation of Java source code. This software visualisation was implemented in a virtual reality system and automation of this process has been achieved.

The technical implementation details are presented, followed by a discussion of the metaphor used. This is then followed by some examples and then revisiting the main points covered in the first part of the paper and showing the benefits they have brought to this work.

4.1 Implementation Details

The implementation has been done with the MAVERIK [9] GNU virtual reality toolkit. It is a C toolkit for constructing single user Virtual Reality applications. It is a mid-level virtual environment creation tool because it allows the creator great freedom over what can be created and managed but it does not require them to write routines to deal with the intricacies of drawing three-dimensional graphics.

MAVERIK provides an infrastructure in which a (complex) virtual environment can be created, managed, viewed, interacted with and navigated around but does not make any further assumptions about the environment or its uses. Since it is provided as a programming level toolkit (C code) then there are no virtual environment issues forced upon the world being created, although through the use of C code it has the ability to deal directly with an environment’s data structures and algorithms. This is only if the creator makes the MAVERIK world aware of them in the initialisation code!

In order to implement the objects required by the Software World metaphor a library of MAVERIK code has been written. This library is then used by the generated C code to describe (in the implementation done to date) a district of a city in that world. In creating a generic library it means that less code needs to be “written” for each automatic generation of code, that changes can be centralised, and even that objects used in different metaphors can be shared. It also means that the effort to describe the metaphor in terms that MAVERIK can understand and process is a once only task.

4.2 Software World in more detail

In order to be able to coherently create and use the virtual reality environment for the purposes of program comprehension a metaphor that creates a constrained environment for mapping the data to be visualised is required. The development process of this Software World metaphor was necessary because the mapping needed to be complete and logical to enable full utilisation of the visualisation process. As an overview, the mappings used for the Software World are that the entire software system being visualised is the world, cities represent source files that may contain one or more classes which themselves are represented as districts and finally methods are shown as buildings.

The mapping that was created within the confines of the metaphor became known as Software World because of the representations chosen and because within its mappings it had the ability to encompass the entire software system under consideration. Java source code attributes have the nice feature that they can be organised into a clean hierarchical structure and all attributes must be contained within this structure. The metaphor definition relies on this to make logical sense to those using it and has been created with this fact in mind. More details of the mappings chosen and the rationale for these can be found in [1].

The city metaphor has often been applied to visualisation problems but the motivating factors for its use as part of the Software World metaphor were

- It fitted into the overall scheme of world, country, city, district and building used to provide a consistent but scaleable and evolvable visualisation.
- The use of abstract visualisation features has already received a small amount of attention in the software visualisation field and a real world visualisation was decided to be worth investigation.
- The use of a city as a container mechanism for a mid-level Java attribute worked well with long term evolution, which from a software perspective is an important consideration. Since it is known the software will change a visualisation that can meaningfully change with it is of benefit to those who use the system.
Using a city metaphor directly allowed the legibility features identified by Lynch [10] (and also covered by Ingram and Benford in [4]) to be easily incorporated to show their use in information gathering situations.

It may well be the case that other metaphors, mappings, representation, abstractions may well be suitable for this visualisation of Java source code. It is important to bear in mind that the use to which the visualisation is to be put, and the underlying data, have a far greater influence on the appropriateness of any of these factors than is often recognised. These will govern the perceived success from both design (mappings and representations) and usability perspectives.

4.3 Software World Examples

Some examples of the views that can be seen in an implementation of the Software World when applied to real source code are included to show some of the points that have been presented and discussed in the previous sections of the paper.

Figure 2 shows an overview of the source code of a reasonably sized Java class, which is itself part of a much larger Java system. From this point of view the user of the visualisation is able to get an appreciation of the number of methods in the class, those methods that have a large line count, and which are private methods. All of this overview information provides, at a glance, useful contextual information when the user goes and investigates any of the methods in more detail. The use of block structures for layout (blocks and roads), a central garden and an enclosing fence have been used to try and aid the navigation and orientation of the user of the visualisation system.

Figure 3 shows a more detailed view. The user has navigated from the overview shown in Figure 2 down to the street level to be better able to investigate the information shown at this level. Here the road and various blocks can be seen, with the boundary fence in the distance. The colour of the buildings represents whether the method the building represents is private or not (private is a keyword in Java and indicates that the method cannot be accessed outside of the class in which it is declared). The height is a representation of the number of lines of code. Each building has a minimum height of one storey and a roof, but for each extra ten lines of code an extra storey is added to the building. The doors also provide information about the method. All buildings have a blue door – the reason being that a method may have no parameters and there still needs to be a logical way for a user to enter the building when exploring the environment. Parameters are shown in number and type (at a basic level) by extra doors. All method parameters are shown by a door with either yellow or green paintwork; the formal type of the parameter determines the exact colour.

Figures 2 and 3 have shown two very different views of the visualisation implementation of Software World. Whilst much of the mapping is not shown these views provide an indication of the appearance of the environment and how it makes use of metaphor, navigation and orientation.
4.4 Relation to Virtual Reality

The visualisation aims to make use of legibility features of landscapes for navigation and orientation purposes. It is therefore hoped that the environment can be learnt through usage and exploration. This familiarity can then be harnessed for several uses related to the use of the virtual reality as a knowledge discovery tool.

Because there is a build up of environmental awareness then as changes happen to the buildings and cities in the environment it can be noticed by users of the visualisation. The metaphor has a process by which such changes can be graphically identified but this awareness can be enhanced by the users having a prior knowledge and hence an expectation of what they should see in relation to their current location and orientation. This is of most use as the visualisation evolves as the underlying code being visualised undergoes changes and maintenance and itself evolves over time.

There is also the use of navigation and orientation features for the discovery of knowledge through the exploration of the data. Program comprehension literature has highlighted that there needs to be ways of orienting the user when exploring program code artefacts and ways of navigating around those facets. This navigation needs to support both an exploration viewpoint and a directed query viewpoint. In the use of the Software World to display the Java code artefacts this can be supported through the use of virtual reality as an implementation mechanism and through the emphasis placed on the legibility features of environments which the metaphor provides.

4.5 Virtual Reality Limitations

There are several areas in which the state of the art (on desktop machines) of virtual reality limits the implementation of this visualisation. An example of this is a coherent multi-user environment in which communication, both textual and audio-visual, between users is seamlessly integrated with the visualisation. That is not to say that the technology does not exist in some form in some virtual reality system. What it does mean is that the particular technology does not serve the lower end of the market or that the inclusion of it in a system that requires automation requires effort that far outweighs any benefits it might bring.

A real issue at the moment is the implementation of the complete Software World including distributed users within the same graphical environment, communication within this environment, the display of two dimensional images and text when not near an artefact and the level of graphics achievable on a desktop. All of these have been addressed in various ways in the virtual reality and collaborative virtual environment literatures but this work prioritises automation above these. For a company to think any form of visualisation tool is worth time and effort investment it would require, especially in the software field, some assurances as to the ease of creating them is vital.

5. Conclusions

This paper has described an application of virtual reality technology to the problem of visualising data artefacts. In particular it has tried to show that the use of such techniques can be very beneficial when dealing with large, complex and changing data sets such as source code.

Since the data required to be shown is large, the metaphor mapping was designed with this in mind and it shows the future possibilities of applying visualisation techniques in virtual reality environments with many other sources of data. The work presented, whilst being tailored towards the problems and issues of software visualisation, has application beyond it into many forms of information visualisation. The creation of tangible and explorable scene from inherently intangible data has been shown to be feasible and hence such techniques can be applicable to other information sources. This is not to say the each new visualisation would be able to use, for example, Software World mappings. But, if the main effort required is in designing a metaphor that works with the data and the task to which the virtual reality visualisation is to be put then surely it is worth that initial investment.

Acknowledgements

This work has been partly financed by an EPSRC studentship and is partly financed by an EPSRC ROPA research grant; VVSRE, Visualising Software in Virtual Reality Environments.

References


