Using an Existing Game Engine to Facilitate Multi-User Software Visualisation

Claire Knight and Malcolm Munro
Technical Report 8/98

Visualisation Research Group,
Centre for Software Maintenance.

Department of Computer Science,
University of Durham,
Durham, DH1 3LE, UK.

E-mail: C.R.Knight@durham.ac.uk

Keywords: Program Comprehension, Software Visualisation, Virtual Reality, Software Maintenance

Abstract

A specialised subset of information visualisation that is of great interest to software maintainers is software and code visualisation. The techniques of information visualisation can be applied to software because they aim to reduce the complexity of the information and this is a common problem in program comprehension. Three dimensional visualisation enables people to inhabit that visualisation and explore it from within. The work presented in this paper exploits this so that the users can examine the program code by walking through it, exploring it, and viewing it as an analogical abstraction. The tool used to make this visualisation a reality was based on the game engine of a very popular, current, three dimensional first person game. A game engine was chosen because some of the best graphics on computer systems come from the games companies and to create a new system when several already exist was considered to be reinventing the wheel. The particular game used was chosen because (a) there are tools available to allow alterations and extensions to be made to the game and (b) the game was designed for extensibility

1. Introduction

An important activity in maintaining and changing an existing software system is that of understanding the current system. One approach to achieve this understanding (termed program comprehension) is through the use of visualisation techniques. This approach is termed software visualisation.

Information visualisation techniques are useful in software visualisation because a common goal is to reduce the complexity of the data presented to the user. Three-dimensional visualisation creates an environment which people can inhabit and explore. The work presented in the paper exploits this, allowing users to examine the program code under investigation by walking through it. Benford and Mariani [Benford94] write about virtual reality

> *"VR provides the ability to visualise information in a three-dimensional space, to move through it and to interact with it"*

This is an important consideration with this work and a suitably advanced graphical multi-user engine is being used to examine and test various visualisation ideas. Details of the extensions possible with the engine and those that have been implemented to date are presented with the rationale behind them (section 4). A discussion of the results achieved so far and possibilities for further work are presented in sections 5 and 6.

This paper present work carried out to visualise software in a multi-user virtual world but similar principles and ideas can be applied to many other forms of information visualisation.

2. Software Maintenance

Software maintenance is generally considered to be the last phase in the software lifecycle. Whilst this could be argued from the standpoint that it follows the previous requirements, specification and design, it is not as simple as that. During the process of maintenance it may be necessary to go back to any other level of the lifecycle. Maintenance is often overlooked from a managerial viewpoint and software is not seen as a company resource. This very often means that there is not enough support or funding for the software engineers in the company to perform adequate maintenance. This reason alone provides motivation for producing tools which can help the maintainer.

Lientz and Swanson [Lientz80] provide an early definition of software maintenance but later research work at the Centre for Software Maintenance at the University of Durham defines software maintenance as:

> *"Software Maintenance is the set of activities (both technical and managerial) necessary to ensure that software continues to meet organisational needs".*

Empirical studies have produced various statistics as to the true cost of software maintenance, but exact figures apart, all have shown that much time, effort and money is and needs to be used in this phase. Figures produced in these studies show that the maintenance phase consumes between 50% and 70% of the software budget. Any tools that help with maintenance should therefore be of interest to maintainers and companies alike.

2.1 Program Comprehension

One of the major tasks of maintenance is to understand the system being maintained. The understanding can be at different levels, possibly moving progressively towards the detailed code level, but ultimately the code needs to be understood to facilitate any changes that are necessary.

An overview of the types of program comprehension carried out, and aspects of code that can affect it is provided by Robson et. al. [Robson88]. Software visualisation is one technique that can be used for program analysis to aid comprehension. It can be used for both static and dynamic analysis with either

two-dimensional or three-dimensional displays so it has the possibility of being a much-used tool in a maintainer's work.

3. Software Visualisation

This section does not provide an overview of all visualisation techniques and research systems. For a more complete overview refer to Young's work within the Visualisation Research Group [Young96] where many three dimensional information visualisation systems are presented.

For many years the focus of software visualisation has been two-dimensional images. Various aspects of the program code and metrics calculated from the code have been displayed in 2D. A common display was, and still is, based around graph structures. There are problems with this approach to software visualisation, not least because of the complexity of the software. A call graph where the nodes are the procedures or methods and the arcs represent the calls made between them can easily become cluttered and unclear. An example of this is shown in this picture.

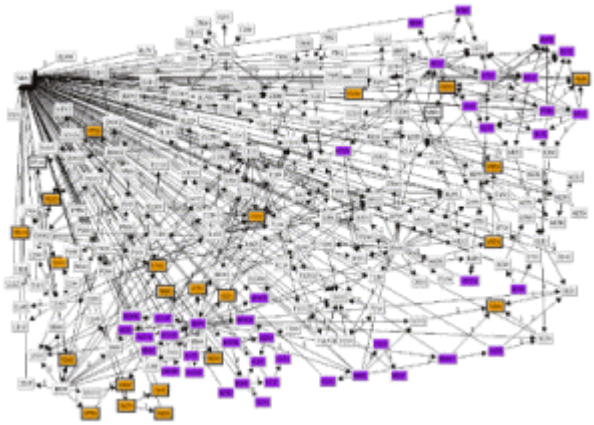

Figure 1

This call graph actually represents the calling information from a highly maintained commercial system. If there is this much overcrowding of information from a reasonable sized, but well maintained, piece of software then it is clear that new techniques for the investigation and visualisation of the code are needed.

Because of the problems with many two dimensional representations, especially with any real amount of data, it was considered sensible to consider the use of three-dimensional visualisations.

Some of the three dimensional work was a direct extension of the two-dimensional [Young96]. The extra dimension was used only as an extra dimension for the nodes and arcs so that some of the crossover of arcs was eliminated. Other work moved away from directly visualising graph structures and used other representations to show the same sort of information. Metaphor and analogy are common ways of moving from the program code to the visualisation in these cases.

Young [Young97] moved completely away from a nodes and arcs display in his work on representing call graphs. He writes

> *"CallStax makes full use of the extra dimension afforded by VR to maximise the amount of information available and the flexibility for displaying and interacting with that information."*

The visualisation does not try and show the information as a network but as paths through the network. The picture shows all the calls stacks for a specific piece of code, with the qsort item selected as the focus of interest. The user is able to interact with the representation by selecting the focus. The display then updates itself appropriately.

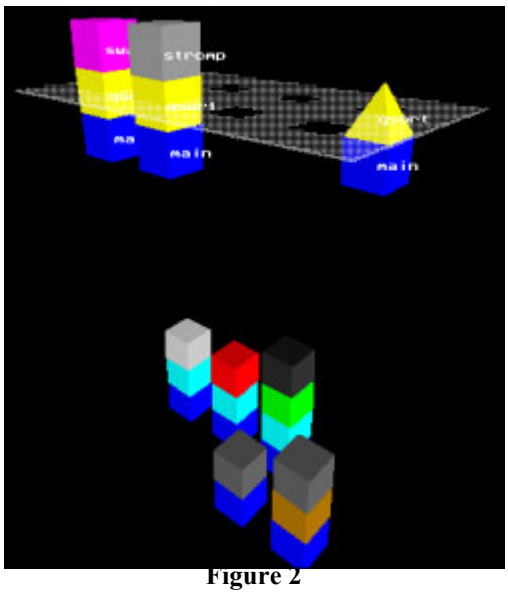

Figure 2

Although the progress made so far is good there are still more things that are desirable from these software

visualisations. Making use of virtual reality just to have nice pictures is not using the technology to its limits. Virtual reality enables the visualisations to be more dynamic, interactive and immersive (without headsets and expensive equipment). Being able to be inside the software and exploring it from within, and allowing the user to choose the direction and amount of exploration they want to do is seen as a next step. Metaphors can be used to represent both the high level structure of the program code and detailed pieces of the code. One such analogy is that of a building (or buildings) where the rooms and corridors represent the high level structure and items with in the architecture represent the code detail. The benefit of this analogy is that exploring a building, looking out of windows and using stairs and lifts is a natural part of daily life for most people. One of the advantages of virtual reality is that it enables abstraction from the actual code but, if designed and implemented correctly, the meaning can still be evident. Multi-user communication is also desirable for these immersive virtual realities because it allows fellow workers to share and discover knowledge whilst exploring the software. Being inside the world and able to go to certain areas of the code whilst explaining something to a colleague can be invaluable.

4. The Game Engine

The game engine used with this work is Quake 2 [IdSoftware] which was released in December 1997. Since then there have been several updates made available via the World Wide Web. Very quickly the existing on-line Quake community built up a large knowledge base for Quake 2 playing and editing.

Id Software designed Quake 2 to be extended. It is easy to play and distribute new levels. Some of the source code (not the main game engine) has been made available allowing people who can understand and program C code to make many different twists on the original game. There have been many tools created that enable new maps (levels) to be created for the game, tutorials about the properties that can be used in the levels and tutorials for extending or altering some of the code used in the game.

4.1 Ways to Customise the Game

There are three main ways to extend or alter the game as released by Id Software.

- Changing the appearance of the player (beyond the variations provided as part of the game)
- Creating new maps (levels)
- Altering the code of the game to achieve different things during play

The are two ways to change the appearance of the player. Changing the underlying model used and changing the "skin" (essentially the clothes of the player). In the version of Quake 2 released on the CD there are two models; male and female. There are 15 predefined skins for the male model and 10 for the female. Since then, a downloadable upgrade has added another model, the Cyborg, and appropriate skins. The skins are image files and can be changed to make the characters clothing appear differently. Model changes require a modelling tool and are much more sophisticated.

Maps are basically text files containing a description of the world (in that map) which have to then be put through three utility programs that convert the information into a format suitable for the game engine. There are several freeware and shareware programs that provide an easier way to create levels than as text. These provide a CAD style drawing interface of the three dimensions and very often a 3D view as a guide to the designer.

Altering the code provides many avenues for new and different behaviours from the game. Most changes are made to alter one or two aspects of the gameplay rather than trying to alter everything. Changes can be made to weapons, monsters, players and some aspects of the game although most of the detailed game information (such as client-server code or sprite handling) is encapsulated within the game engine.

A simple change that many users of Quake 2 use within the realms of the normal game is that of a configuration file. This is a text file that sets up the necessary key bindings for play. Whilst the menus within the game allow such configurations to be done (to an extent) more experienced players use the configuration file to refine the standard options. In some cases (such as games running with alterations to the code) the configuration file or the standard console are the only ways to be able to use some weapons

or commands. Being able to define such configuration is useful for multi-user systems so that each user can define controls they are happy using.

4.2 Extensions Considered

There are many ways in which the game as it stands can be altered for use as a multi-user software visualisation environment. Some extensions can be made on their own but others require extensions in other ways to be effective. The most common of these is having to alter the code to support items placed in the maps or an alteration of the philosophy behind the use of the system.

It is possible, based on the player's model and skin attributes to make changes using code so that they, for example, always have keys to certain doors. In this way a map can be designed based on different coloured door locks with certain users having certain keys. It also means that the keys do not have to be left around the level for users to pick up – this would remove any control over where each sort of user could explore. This distinction based on appearance of the person within the virtual world is one of the important features mentioned by Benford and Mariani [Benford94].

Maps represent a very large extension. Without extending any other part of the system a map representing a piece of software can be loaded into the standard Quake 2 game engine and explored from there. As standard each player has the basic blaster weapon but this can be useful for triggering items in the map. No further weapons can be used because unless they are placed in the maps for players to pick up then they are unusable.

By editing the code and creating new models new weapons/tools can be added. Torches and lasers could be useful in a visualisation system. Torches could be used to highlight areas of interest, both to trigger messages and to point out areas of importance to other users. Another variation on this theme is a laser that can be used as a long-range pointer, and can help when directing other users around a particular map. Another area where altering the code can be useful is to create specialised in-game displays of various pieces of information. Id use this technique in single player Quake 2 with help messages appearing when the user reaches certain areas of the map.

4.3 Extensions Implemented

The bulk of the extensions that have been investigated to date involve creating and using maps. A graphical editing tool has been used to carry out this work so the three dimensional view shown in this image was created using orthographic projections. This wireframe view is very cluttered and is really only used as an indication of how the map looks. To really get the full effect of the map and especially to test it Quake 2 is used.

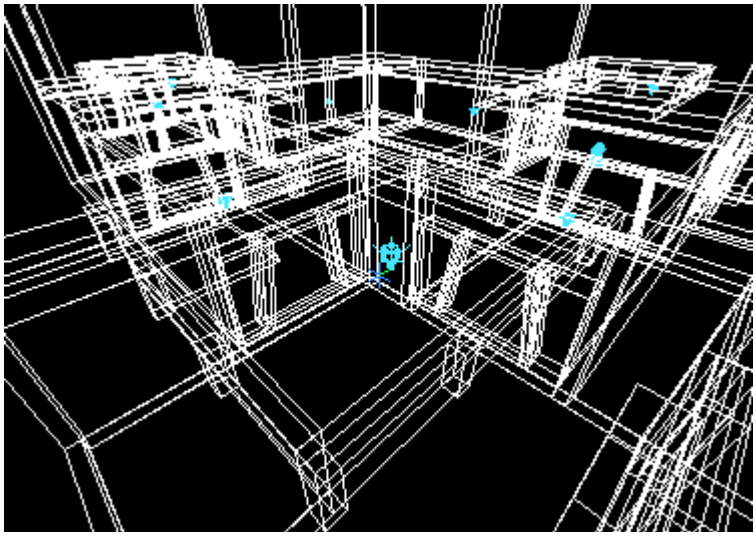

Figure 3

All attributes of the virtual world are specified in the map. The layout, the way the level will look with textured graphics and any extra entity specific properties are all defined using the editor.

The map definition is based on objects (or brushes and entities as they are officially known) so that each item in the world can have specialised attributes in an object-based way. At the moment there is no way of extending or inheriting properties from objects.

This next image shows the same view with the textures turned on (in the editor). The display is much less cluttered because the brushes that represent the walls then become solid. Some of the display is incorrect when this is done in the editor. For example, the white box is actually a light entity and will not been seen as an object in Quake 2, although its placement in the map will have an effect on the lighting in this room.

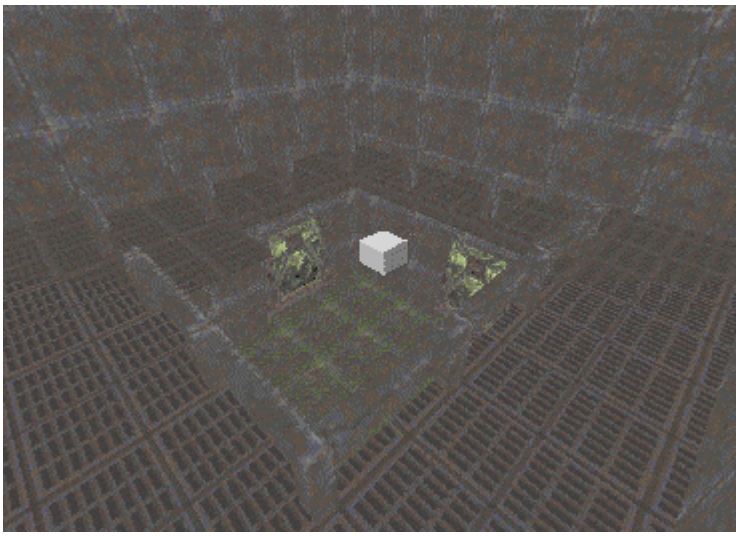

Figure 4

In addition to creating new maps the Quake 2 program code that Id Software make available has been altered. This code allows users to make changes to the way the game/system works and is therefore provides a powerful way of extending the system to support aspects of visualisation and multi-user virtual worlds that are of no use to the game. Simple extensions such as displaying messages (both to a specific user and to all connected to the system) and reading in text files containing visualisation information have been implemented and tested.

5. Discussion

The images shown in the previous section are based on a very simple call graph structure. This basic structure (seen from the top) can be seen in this image. Whilst this current layout and implementation has the same problems as the call graphs previously done in three dimensions it provides the basis for future work. The demonstration world is being used to investigate the multi-user aspects, ways of displaying detail in the world and incorporating different means of interaction for the user.

Figure 5

Another representation not seen in these images is that of using ladders to move through the class hierarchy of the code under consideration. All of these, and other, representations, different parts of the code chosen for display and different layout algorithms still need more work.

The results obtained so far are encouraging. The maps created to represent portions of source code are all usable inside the Quake 2 environment, and combined with the recompiled extendible code provide the basis for a completely configured and controlled visualisation environment.

Altering the Quake 2 code to provide information, that would be useless in the game setting of Quake 2, is also useful because it allows instructions or extra information about the software visualisation to be presented to the users.

The interface to quake is based on the first person view; i.e. the images shown on the screen are meant to be what the avatar sees inside the world. This makes the interface intuitive, and using the mouse for head

movements enables a large degree of visibility. The field of vision can also be configured. By default it is 90° but this can be extended to 160° which is closer to the natural human field of vision. The problem with extending the field of vision is that because the user is restricted to using a computer monitor the edges of the image then start to appear distorted.

Due to many multiplayer features already in the game not too much has had to be done to enable simplistic communication between users. A "talk" and a "teamtalk" command are available to users that allows them to type their comment which is then broadcast to all players or their team respectively.

Another useful means of communication when two avatars are within each other's view is to use gestures. There are five implemented in Quake 2, two of which are of use in this situation. Pointing and waving can be used in a combination to direct another user to a different part of the map or to indicate that one user should follow the other. In combination with talking this can be an effective means of user interaction.

The work done so far succeeds in displaying software and program code in a way that allows those investigating it to actually be inside the code. The layouts used so far also hide some of the information from view at any one time, and the user is able to choose the focus of investigation by exploring the portion of the map they are interested in. Whilst one of the maps is based on simplistic call graph layout the future work aims to overcome many of the difficulties found with other sorts of software visualisation.

6. Conclusions

The preliminary results of this work have been encouraging which means that there are several avenues for future work. The main areas seen as the way forward at the moment are

- Designing skins to identify different classes of user. These skins should be usable with male and female users of the system so coloured clothing is currently seen as a good way forward.
- Once classes of user have been decided upon and appropriate skins created then the communication can be extended to allow not only global communication (as text) or person to person communication (with gestures) but to team communication via selectively displaying messages based on the class of the user.
- Not only the communication can be based on the class of the user. Areas of maps can be tailored for certain users so that detail can be hidden from those users that do not need to be aware of it.
- Increasing the size and complexity of the maps displayed and the code used in the visualisation. No real world computer system is as small as these test cases so once design decisions have been made in the test maps then attempts at scaling up and automation are necessary. This work may then involve altering some of the display and visualisation decisions made. Consideration of the items in the world and their properties is also important. Should one user interact with an object in the world then (for a static investigation) that object should remain unchanged because of other users inhabiting the virtual reality.
- The Quake 2 code that is available for extending the virtual world has great potential. A number of different additions based on communication (visually and textually) and also to the general environment are being considered and these need to be implemented to test their suitability.

The work done to date has provided software visualisation within a multi-user virtual reality. The interaction and exploration between users has been limited (due to the limited nature of the visualisation) but is promising. Some means of communication are discussed above and with the implementation of extra tools based on pointers and torches the visualisation system has the possibility of becoming useful to maintainers in their work. Not only is the environment useful for many users, it allows a greater freedom in the creation and exploration of software visualisations.

References

[Benford94] S. Benford, J. Mariani.
Populated Information Terrains: Virtual Environments for Sharing Data
Research report CSCW/4/1994, Centre for Research in CSCW, Lancaster University, 1994. ftp://ftp.comp.lancs.ac.uk/pub/reports/1994/CSCW.4.94.ps.Z

[Benford97] S.Benford, C. Greenhalgh, D. Lloyd.
Crowded Collaborative Virtual Environments
Proceedings of the ACM CHI 1997.
http://www.acm.org/pubs/articles/proceedings/chi/258549/p59-benford/p59-benford.pdf

[Gaver95] W. W. Gaver.
Oh What A Tangled Web We Weave: Metaphor and Mapping in Graphical Interfaces
Proceedings of the ACM CHI 1995, Short Papers.
http://www.acm.org/sigchi/chi95/Electronic/documnts/shortppr/wwg2bdy.htm

[Hemmje93] M. Hemmje.
A 3D Based User Interface for Information Retrieval Systems
Proceedings of IEEE Visualization '93, Workshop on Database Issues for Data Visualization, San Jose, California, October 25-29, 1993.
ftp://ftp.darmstadt.gmd.de/pub/VISIT/papers/hemmje/IEEEVIS93.ps.gz

[IdSoftware] http://www.idsoftware.com/

[Lientz80] B. P. Lientz, E. B. Swanson.
Software Maintenance Management
Addison-Wesley, 1980. ISBN 0-201-04205-3

[Lin95] S. Lin
Metaphors, Architectures, and Cyberspaces – An Introduction
http://research.umbc.edu/~slin1/paper/page1.html

[Roberston93] G. G. Robertson, S. K. Card, J. C. Mackinlay.
Information Visualization Using 3D Interactive Animation
Communications of the ACM, Vol. 36, No. 4, April 1993, pp57-71.

[Robson88] D. J. Robson, K. H. Bennett., B. J. Cornelius, M. Munro.
Approaches to Program Comprehension
University of Durham, Computer Science Technical Report 11/88

[Young96] P. Young.
Three Dimensional Information Visualisation
University of Durham, Computer Science Technical Report 12/96
http://www.dur.ac.uk/~dcs3py/pages/work/document/lit-survey/IV_Survey/index.html
and the Visualisation Research Group at http://www.dur.ac.uk/CSM/VRG

[Young97] P. Young, M. Munro.
A New View of Call Graphs for Visualising Code Structures
University of Durham, Computer Science Technical Report 03/97
http://www.dur.ac.uk/~dcs3py/pages/work/documents/tr-03-97/tr-03-97.html