Keg Master: a Graph-Aware Visual Editor for Scene Graphs

Hannah Slay, Dr Bruce Thomas, *Dr Rudi Vernik, *Matthew Phillips

School of Computer and Information Science
University of South Australia
Mawson Lakes Boulevard, Mawson Lakes 5095, South Australia

*Command & Control Division
Defence Science and Technology Organisation
West Avenue, Edinburgh 5111, South Australia

Hannah.Slay@unisa.edu.au Bruce.Thomas@unisa.edu.au Rudi.Vernik@dsto.defence.gov.au
Matthew.Phillips@dsto.defence.gov.au

Abstract

This paper describes the manner in which papers should be formatted for papers adhering to the ACS series, *Conference in Research and Practice in Information Technology*. The abstract should be a maximum of 250 words and should clearly identify the content of the paper. The text in the abstract should be in style "abstract text". Note that the best way I have found to get the styles correct is to copy the text into a copy of this template rather than attempting to change the original file.

Keywords: Information Visualisation, multi display environment.

1 Introduction

The goal of this research was to design a visual editor for a 3-dimensional representation of a graph. Existing visual editors for scene graphs are not graph-aware, and therefore cannot manipulate or save a representation of the underlying graph data structure (as opposed to the scene graph) for use or analysis by other tools. Our requirement was for the tool to support the importation of both the graph data structure and its presentation into our existing multi display environment. An extensive search of existing off-the-shelf products was performed and many applications were tested, but the applications we found were all either too expensive or did not provide all the tools we considered essential in scene graph creation software. The decision was then made to create our own application.

The criterion for our target application was a visualisation environment that provided the user with a mechanism to create graphs visually and save them using the Virtual Reality Modelling Language (VRML). The motivation for selecting this language was that it was already supported by the existing elements of our multi display environment, namely InVision and FOCAL. The application had to allow users to automatically arrange the elements of their graph, but also allow them to manually move elements to new locations (to prevent occlusion of essential information, etc). Most importantly, when an element was moved to a new location, all of its connections needed to be dynamically resized and appear to move with the element.

As mentioned previously, InVision and FOCAL represent two key technologies in the DSTO's multi-display framework. InVision has traditionally supported two dimensional models and views of data. With the Augmented Reality (AR) plugin, it now has the tools necessary to allow users to display and interact with three-dimensional models. The primary motivation for creating Keg Master was therefore to provide a mechanism of bridging the gap between the traditional two-dimensional views supported by InVision and the three-dimensional display capabilities provided by the AR plugin and FOCAL.

In this paper, we first present related work to our field of study, followed by an overview of InVision. Next we provide an overview of the FOCAL. We then move on to give an overview of the Keg Master system. We then follow this with some concluding remarks.

2 InVision

InVision is a research initiative through which the Information Technology Division (ITD) of the DSTO is conducting research into tools and techniques for the rapid assembly and deployment of information visualisation solutions (Goodburn, Vernik, Phillips and Sabine 1999; Pattison, Vernik, Goodburn and Phillips 2001). InVision uses an infrastructure framework with pluggable components to deploy a large range of visualisation solutions. A key goal of InVision is to facilitate the integration and coordination of a wide variety of disparate information visualisation view types through the research, design and prototype development of an open, component-based software architecture hosted on the Java 2 platform.

This architecture also supports the location, collection and modelling of the information to be visualised, and the use of knowledge-enabled components (primarily software agents) to ensure that the deployed visualisation solution is able to adapt to changes in the deployment environment. Research infrastructure for the exploration

of these research goals is provided by a prototype JavaBean implementation of selected elements of this architecture.

InVision uses the concept of workspaces to support the management and integration of views for particular individuals or roles (see Figure 1). These views specify how the information in an underlying data model will be represented. InVision models can be thought of as attributed graphs that capture information about a set of "things", the various relationships between these "things", and sets of attributes that describe the "things" (eg size, colour, name). The modelling approach is generic and so can be used to support the visualisation of a wide variety of artefacts such as software systems, intranets, organisational structures, social networks, or financial systems.

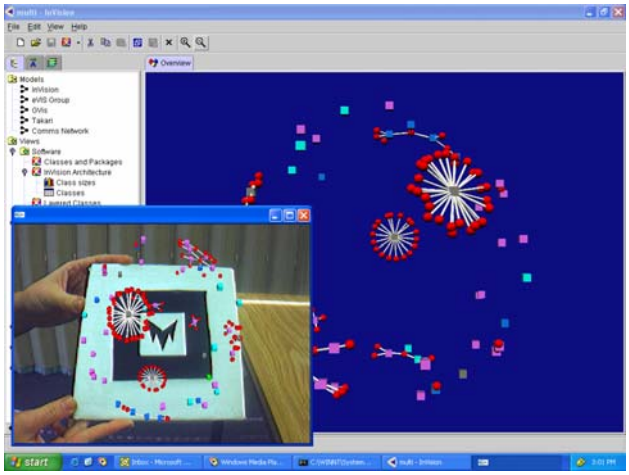

Figure 1: InVision Workspace

A variety of visual forms can be used to represent various viewpoints of the model. These may include cluster graphs, charts (scatter, bar, kiviat), tabular grids, or textual representations (or any combinations of these). InVision already has components that allow for the specification and use of a wide variety of visual representations. These views can be displayed on one or more conventional display devices (eg workstation screen, large screen projection). Interaction with these more traditional visual forms is via direct manipulation interfaces using devices such as a mouse.

Augmented Reality Visualisation (ARVIS), a component in the InVision framework was create to allow any of the views in an InVision workspace to be displayed through an AR tangible interface (Slay, Thomas, Vernik and Phillips 2002). AR refers to the process of overlaying computer generated images onto the real world. By using a head mounted display placed on the user's head, it is possible to completely replace the user's view of the real world with this computer-enhanced world. Numerous different interaction devices are used in AR applications, ranging from traditional interaction devices such as mice and keyboards to more environment-specific devices such as pinch gloves and twiddlers. ARVIS supports the use of fiducial markers as its tangible user interaction device. A fiducial marker is a black and white, asymmetric and unique pattern mounted on a sturdy piece of cardboard. The lower left window in figure 1 shows a model superimposed on such a device.

The AR view capability was added by incorporating the ARToolkit Version 2.52 (Kato and Billinghurst 1999) as a component into the InVision framework. Figure 1 shows a screen shot of InVision with the AR plugin. The main window shows a model displayed in a traditional InVision view. The window in the lower left corner of the image shows an AR view of this model. Right clicking on a traditional view's tab allows the user to also display the selected view in AR mode relative to a particular fiducial marker pattern. The user can then inspect the model by rotating it, changing the azimuth etc. This is particularly useful for three-dimensional models. By rotating the marker, the user can examine the model from all viewpoints. By bringing the marker closer or further from the user, their viewpoint can be zoomed in or out. This provides an interaction that can be closely mapped to the natural interactions between humans and the objects that they want to analyse.

3 FOCAL

DSTO has established the Future Operations Centre Analysis Laboratory (FOCAL), which will enable research into the effectiveness of advanced visualisation technologies in the Australian Defence Force's (ADF) situation awareness, mission planning and decision making. The centre is currently equipped with a 12 foot (3.5m) radius, spherical section screen, illuminated by 6 projectors and driven by a 3-pipe, 8 CPU Onyx 3400, supplied by SGI and Trimension Systems(Lambert 2001).

While responsibility for decision-making rests with the commander, typically many others will contribute information and advice. A key factor in choosing this display technology is its ability to support collaborative decision-making by allowing up to 10 people to share an immersive experience.

The facility breaks new ground by using Liquid Crystal Display technology projectors. These projectors are extremely bright and allow high contrast to be achieved with reasonable levels of ambient light, enhancing the collaborative workspace.

4 Keg Master

While undertaking our search for alternate applications, we discovered a tool called Wilma(Dwyer and Eckersley 2001), a 3 dimensional UML viewer that came close to what we wanted. We studied and extracted key features from Wilma and used these as a base for Keg Master. In particular these features include the mechanism for attaching nodes to a canvas, and the means by which the identity of a picked shape is found.

As well as providing the criteria mentioned in the introductory section, Keg Master allows the user to load and save models to a number of different formats. It was first intended that the primary language used in Keg Master would be VRML. However, later it was discovered that VRML is predominately a scene graph description language, as opposed to what we wanted - a

graph description language. The difference between these language types may on first glance seem insignificant, so the two terms shall be defined. We understand a scene graph description language to include all elements of a scene: the lighting, the fog, animations, behaviours, and finally the elements of the graph. Contrastingly, a graph description language is only concerned with the elements of a graph, and the connections between these elements. Because it is an XML-based language, it exhibits behaviour synonymous with XML. As stated by Blais, "XML describes 3D scenes as structured data, which can be processed without paying attention to how the data should be presented"(Blais, Brutzman, Horner and Nicklaus 2001). Another incentive for using graph description language is that it is a higher-level language than scene graph description languages. This means that a scene graph description of a model can be created from a graph description, but not vice versa. KEG, a new format was therefore created to model our graphs. KEG can be seen as an extension of the format supported by InVision, and can even be loaded by InVision components.

Once resolved by the XML parser, the KEG model is translated into a Java 3D scene graph. The user can dynamically alter the Java 3D scene graph by dragging and dropping primitive shapes onto the canvas (see Figure 2 for a screen shot of Keg Master). Using Extensible Stylesheet Language (XSL) the scene graph can be parsed and output using numerous different formats (including KEG and VRML).

As well as saving models to the KEG and VRML formats, Keg Master also has the ability to save its models to another new format, ARVIS. ARVIS is a format supported by the AR component of InVision that provides users with the ability to view models using augmented reality (Slay, Thomas, Vernik and Phillips 2001).

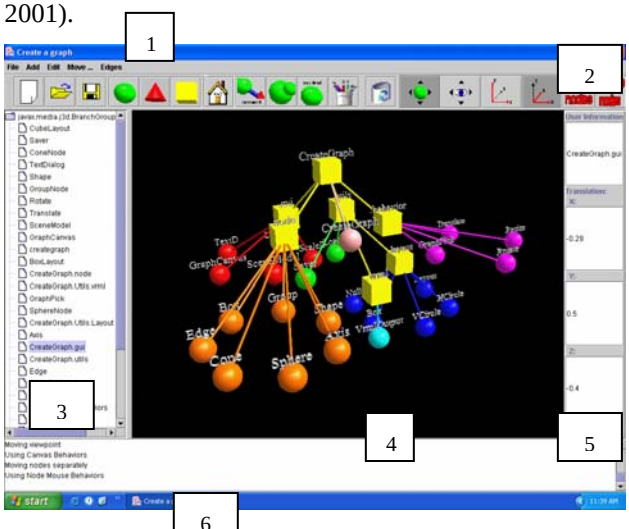

Figure 2 Keg Master screen shot

As figure 2 shows, Keg Master is made of six components, labelled 1 to 6, which will now be discussed. Component 1 refers to the menu bar at the top of the window. Component 2 shows the toolbar underneath the menu bar. These two interfaces are the primary tools for driving the application. All buttons on the toolbar are also mapped to menu option in the menu bar. Component 3 indicates the tree on the left hand side of the screen. This tree contains a list of all elements of the scene graphs. Elements can be selected by either clicking on the label associated with the element in this tree or on the graphical representation of the object as seen in the centre of the screen. Component 4 shows the canvas for the application. The canvas provides a graphical representation of the model as specified by the data inside the tree (component 3).

Elements can be added to the canvas by selecting the object from the tool bar or by selecting the object type from the *Add* menu and clicking on the desired location of the object in the canvas. An object is shown to be selected when its axes are visible and its colour has been inverted (note, the lower cone in figure 1, labelled "another cone", has been selected). Once an object has been selected, its attributes can be changed by clicking on the corresponding button in the tool bar, or by selecting the corresponding menu option in the menu bar. Also, when an object is selected, its position can be changed by clicking and dragging with the right mouse button, until the element reaches its desired position. The default plane of movement is set to the X-Y plane. To move in the X-Z plane, the button in the tool bar labelled X-Z should be pressed (and vice-versa to re-select the X-Y plane).

To rotate the selected object, the control key and the left mouse button should be pressed simultaneously, and then the mouse should be dragged in the direction the object should be turned. Component 5 denotes the panel on the right hand side of the screen. This panel provides four text areas that specify features of the selected object. The first area, labelled *User Information*, allows the user to store extra information about the element. This information is used as a reference to the elements in the tree referred to by component 3. The next three areas allow the user to manually specify the X, Y and Z positions of the selected object. Finally, component 6 provides system messages to the user. Any major event or error is reported to the user in this area.

As well as allowing users to create graphs, Keg Master also allows them to view and manipulate the displayed attributes of the model.

When a large graph is viewed, it can become complex for the user to understand through the entangled connections. We have resolved this problem by allowing the user to decide which connections to show. By selecting nodes singularly or as a group, the user can choose to show connections associated with each node. Figure 3 shows an example of this. In this image, the node *CreateGraph.node* has been selected, and all of its connections are displayed. This provides the user with a simplified view of the graph (the whole graph can be seen in Figure 2), which can aid to ease in the understanding for the user.

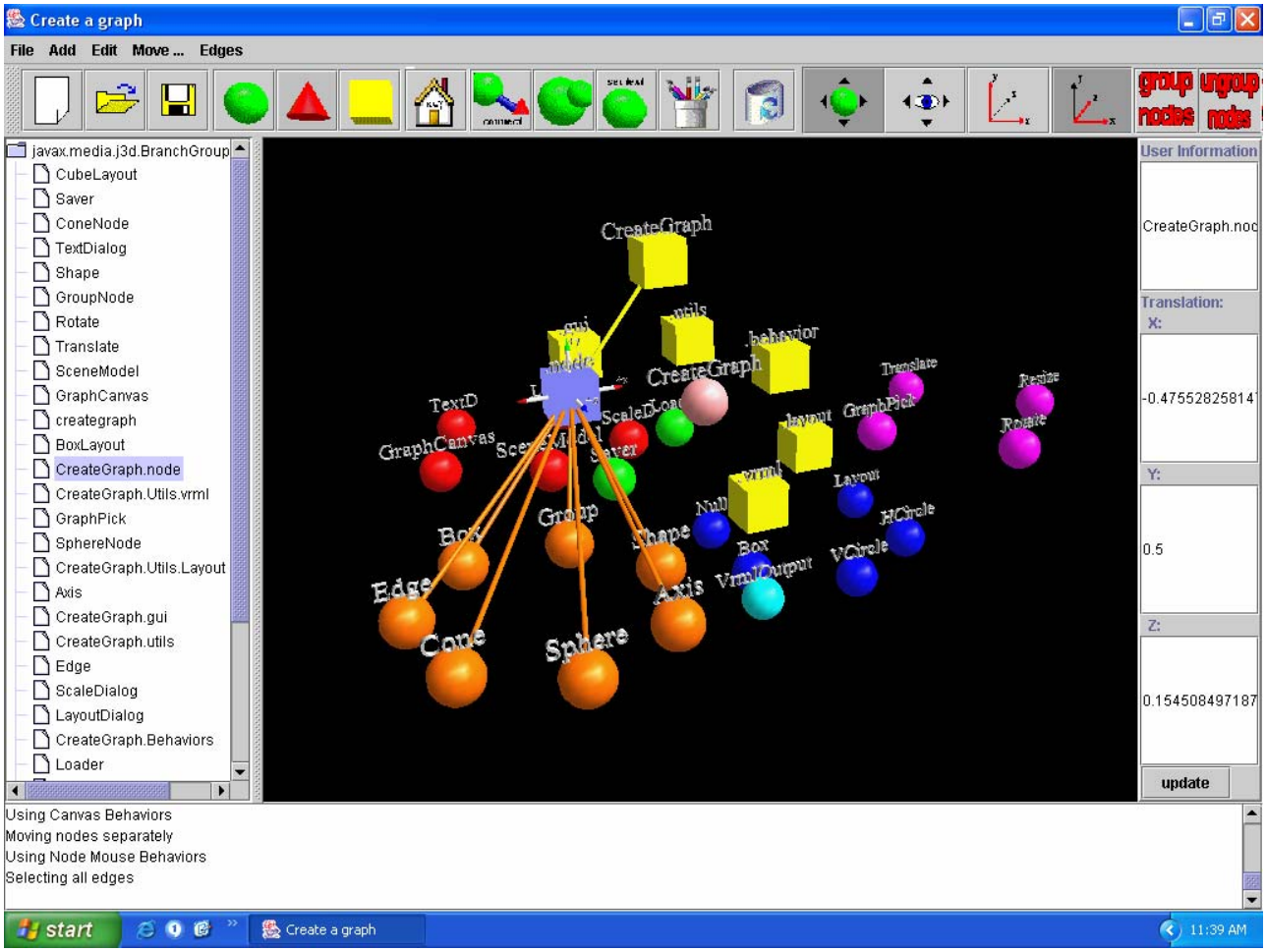

Figure 3 Inspection of Model

5 Related Work

Several organisations have been researching creation of models. However most of these projects are designed to allow users to analyse a graph, but not to create them graphically. The program Feijs and De Jong (Feijs and De Jong 1998) describe in their research accepts as input a module of code and using common objects to represent data types, collections and tables, the program parses the code, locates all "use" statements and creates a graphical representation of the code hierarchy. This representation is expressed in VRML allowing users to analyse it by moving the camera viewpoint, but not to manipulate the nodes. Schönhage (Schönhage, van Ballegooij and Eliëns 2000) for example created a program Diva, which was used by Gak NL to visualise business processes. This program allows users to manipulate an existing graph (drill down techniques) but not to graphically create them.

Several other organisations have been researching methods of automatically laying out nodes in a graph. Tim Dwyer's research (Dwyer 2001) for example allows users to graphically create nodes etc, but not to specify their locations. Their placement is derived using the Force Directed Algorithm. Neville Churcher (Churcher and Creek 2001) uses a modification to this algorithm, the big-bang modification to also calculate positions for nodes.

6 Conclusion

Conclusions go here

7 References

Blais, C., Brutzman, D., Horner, D. and Nicklaus, S. (2001). Web-based 3D Technology For Scenario Authoring and Visualisation: The *Savage* Project. Interservice / Industry Training, Simulation, and Education Conference, Orlando, Florida, USA, Simulation Systems and Applications, Inc.

Churcher, N. and Creek, A. (2001). Building Virtual Worlds with the Big-Bang Model. Australian Conference on Information Visualisation, Sydney, Australia.

Dwyer, T. (2001). Three Dimensional UML Using Force Directed Layout. Australian Conference on Information Visualisation, Sydney, Australia, Australian Computer Society.

Dwyer, T. and Eckersley, P. (2001). WilmaScope, Lesser General Public License.

Feijs, L. and De Jong, R. (1998). "3D Visualization of Software Architectures." Communications of the ACM **41**(12): 73 - 78.

Goodburn, D. P. J., Vernik, R. J., Phillips, M. P. and Sabine, J. J. (1999) "Integrated Visualisation and Description of Complex Systems" DSTO, Salisbury; DSTO-RR-0154

Kato, H. and Billinghurst, M. (1999). Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. 2nd IEEE and ACM International Workshop on Augmented Reality, San Francisco USA.

Lambert, D. (2001). FOCAL. Future Directions for South Australia. Adelaide, Channel 9.

Pattison, T. R., Vernik, R. J., Goodburn, D. B. J. and Phillips, M. P. (2001) "Rapid Assembly and Deployment of Domain Visualisation Solutions" DSTO, Salisbury; DSTO-TR-110

Schönhage, B., van Ballegooij, A. and Eliëns, A. (2000). 3D Gadgets for Business Process Visualization - a Case Study. Virtual Reality Modeling Language Symposium, Monterey, California, USA, ACM Press.

Slay, H., Thomas, B., Vernik, R. and Phillips, M. (2001). Interaction Modes for Augmented Reality Visualisation. Conferences in Research and the Practice in Information Technology, Sydney, Australia.

Slay, H., Thomas, B., Vernik, R. and Phillips, M. (2002). Tangible User Interaction using Augmented Reality. Australasian User Interface Conference, Melbourne, Australia.