Keg Master: a Graph-Aware Visual Editor for 3D Graphs

Hannah Slay, *Matthew Phillips, Bruce Thomas, and *Rudi Vernik
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,Bruce.Thomas}@unisa.edu.au
{Rudi.Vernik,Matthew.Phillips}@dsto.defence.gov.au

Abstract

This paper presents Keg Master, a graph-aware visual editor for 3D graphs. Keg Master is novel in that it works with a graph description language as opposed to a scene graph description language. Keg Master supports a range of 3D graph operations. Graphs may be stored to and retrieved from files. A range of objects may be added to a graph, cone, cube, sphere, and edge. Attributes may be specified to the nodes and edges. These properties of edges may interactively changed to allow users to better understand the graphic structure.

Keywords: Information Visualisation, 3D graph, direct manipulation.

1 Introduction

We have been investigating the design and implementation of a visual editor for a 3D representation of a graph. Existing visual editors for 3D graphs are not graph-aware, and therefore cannot manipulate or save a representation of the underlying graph data structure (as opposed to the 3D graph) for use or analysis by other tools. Our current research into the use of augmented reality (AR) as an intuitive means for interacting with 3D information structures (Slay 2001 and Slay 2002) required a tool to support the importation of both the graph data structure and its presentation into our existing multi display environment. An extensive search of both the literature and existing off-the-shelf products was performed and many applications were tested, but the applications we found did not provide all the tools we considered essential in 3D graph creation software. The decision was then made to investigate and create our own application.

The criterion for our target application is a visualisation environment that provides 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 the Future Operations Centre Analysis Laboratory (FOCAL). The application allows users to automatically arrange the elements of their graph, but also allows them to manually move elements to new locations (to prevent occlusion of essential information, etc). Most importantly, when an element is moved to a new location, all of its connections need to be dynamically resized and appear to move with the element.

As mentioned previously, InVision and FOCAL represent two key technologies in the Command & Control Division at the Defence Science and Technology Organisation's (DSTO) multi-display framework. InVision has traditionally supported 2D models and views of data. We have extended InVision to allow users to view data with an AR 3D interface (Slay 2001 and Slay 2002), which allow users to display and interact with 3D models. The primary motivation for creating Keg Master was therefore to provide a mechanism for bridging the gap between the traditional 2D views supported by InVision and the 3D display capabilities provided by the AR features and FOCAL.

Section 2 of this paper presents related work to our field of study, followed by an overview of InVision in Section 3. Section 4 provides an overview of the FOCAL. Section 5 gives an overview of the Keg Master system. Finally some concluding remarks are presented.

2 Related Work

Existing research into 3D graphing methods may be split into two categories: automatic generation and automatic layout of graphs. Automatic generation of graphs involves the program parsing a form of input (whether it is a table of data, a module of code, or a text document) and forming a 3D representation of the information contained in the graph. Feijs and De Jong's (1998) system 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. Wesson and Warren's (2001) StudentView allows the user to specify information from a database to be combined and displayed on an XYZ scatter chart. The scatter chart is expressed in VRML, again allowing users to move the camera viewpoint, but not to manipulate the nodes. A final example is Diva [Schönhage, van Ballegooij and Eliëns (2000)] that was used by Gak NL to visualise business processes. Diva allows users to manipulate an existing graph using drill down techniques, but not to graphically create a new graph.

Automatic layout of graphs involves creating algorithms to automatically layout data, so as to enhance the user's ability to perceive and comprehend the visualised data. 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*. This algorithm calculates the ideal position for each node by modelling forces of attraction and repulsion between the nodes. The final position of each node is determined when the graph reaches a stable state. Neville Churcher (2001) uses a modification to this algorithm, the big-bang modification to also calculate positions for nodes.

3 InVision

InVision is a research initiative at DSTO into tools and techniques for the rapid assembly and deployment of information visualisation solutions (Goodburn, Vernik et al. 1999; Pattison, Vernik et al. 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 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.

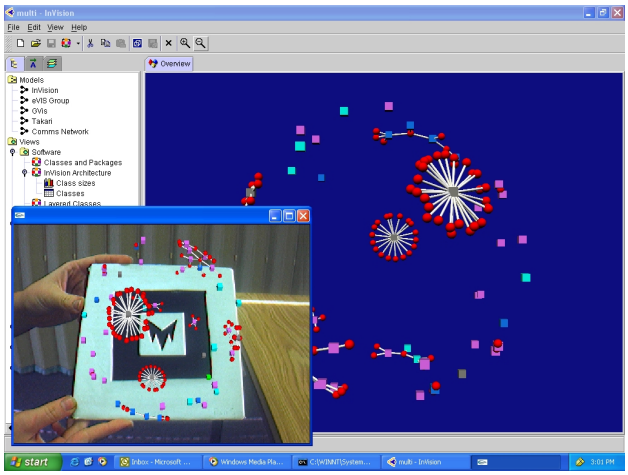

Figure 1: InVision Workspace

A variety of visual forms may 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 et al. 2002). AR refers to the process of overlaying computer generated images co-located with objects in the physical world. By using a see through head mounted display, 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 tracked wands and pinch gloves. ARVIS supports the use of fiducial markers as its tangible user interaction device. A fiducial marker is a black and white asymmetric unique pattern mounted on a sturdy piece of cardboard. The lower left window in Figure 1 shows a model superimposed on such a marker.

The ARVIS component is based on the public domain AR technology, ARToolkit Version 2.52 (Kato and Billinghurst 1999). 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. Employing a traditional tab interaction device, the user may also display the

selected view in AR mode relative to a particular fiducial marker pattern. The user can then inspect the model from different viewpoints by physically picking up and rotating the fiducial marker. This form of interaction is particularly useful for 3D 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.

4 FOCAL

DSTO established the FOCAL to 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.

KegMaster was designed in part to be used in this environment, maki

5 Keg Master

As previously mentioned, a search was completed for a suitable 3D graph manipulation editor and found an interesting tool Wilma (Dwyer and Eckersley 2001). Wilma is a 3D UML viewer that incorporated many features we desired. We extracted the design of a number of key features from Wilma and used these as a design 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 VRML is predominately a scene graph description language, as opposed to what we require - a graph description language. The difference between these language types may on first glance seem insignificant, so the two terms shall be defined.

A scene graph description language defines all elements of a graphical scene to be rendered: the lighting, the fog, animations, behaviours, and finally graphical objects. Contrastingly, a graph description language is only concerned with the elements of a graph, and the connections between these elements.

Our graph description language KEG is an XML-based language. 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 et al. 2001). A second incentive is a graph description language is a higher-level language than a scene graph description languages. A 3D graph description of a model can be created from a graph description, but not vice versa. This allowed a graph to be translated into multiple different formats. 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 graph. The user can dynamically alter the Java 3D 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 3D 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 et al. 2001).

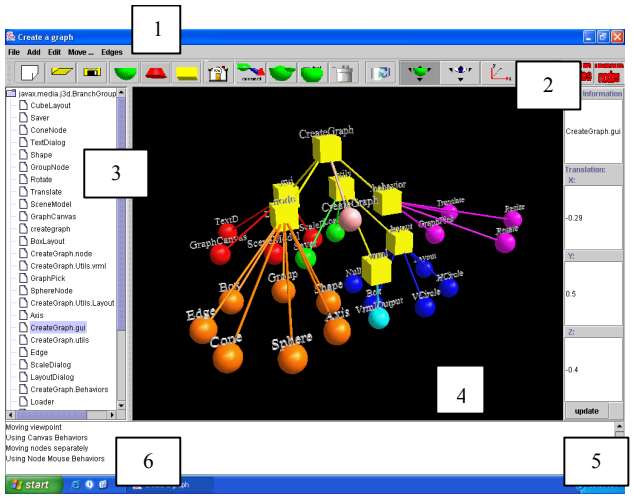

Figure 2 Keg Master screen shot

As Figure 2 shows, Keg Master is made of six sets of 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 3D 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. 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. The attributes of the object that can be changed are as follows: label, position, rotation, colour, and size. An object position can also be changed through a direct manipulation drag and drop operation. The default plane of movement is set to the X-Y plane. The plane of movement may be change to X-Z plane via the tool bar item labelled "X-Z". To rotate a selected object, the control key and the left mouse button are be pressed simultaneously, and then the mouse should be dragged in the direction the object should rotate.

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.

Keg Master supports grouping and arranges selected nodes in one of following patterns: circle on X -Y plane, circle on X - Z plane, cube, or square. The user may also ungroup collections of nodes.

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. KegMaster allows the following edge operations: show or hide all edges, show or hide selected edges, change edge colour to colour of start node or colour of end node or user define colour, and show or hide direction of edges.

6 Conclusion

Keg Master is presented in this paper, a graph-aware visual editor for 3D graphs. Keg Master works with graph description languages as oppose to a scene graph description language. A scene graph description language defines all elements of a graphical scene to be rendered: the lighting, the fog, animations, behaviours, and finally graphical objects. Contrastingly, a graph description language is only concerned with the elements of a graph, and the connections between these elements.

Keg Master supports a range of 3D graph operations. Graphs may be stored to and retrieved from files. A range of objects may be added to a graph, cone, cube, sphere, and edge. Attributes may be specified to the nodes and edges. These properties of edges may interactively changed to allow users to better understand the graphic structure.

7 References

Blais, C., D. Brutzman, et al. (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.

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

Goodburn, D. P. J., R. J. Vernik, et al. (1999). Integrated Visualisation and Description of Complex Systems. Salisbury, DSTO.

Kato, H. and M. Billinghurst (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., R. J. Vernik, et al. (2001). Rapid Assembly and Deployment of Domain Visualisation Solutions. Salisbury, DSTO.

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

Slay, H., B. Thomas, et al. (2002). Tangible User Interaction using Augmented Reality. Australasian User Interface Conference, Melbourne, Australia.

Wesson, J. L. and Warren, P. R. (2001). Interactive Visualisation of Large Multivariate Datasets on the

World-Wide Web. Australian Conference on Information Visualisation, Sydney, Australia, Australian Computer Society.

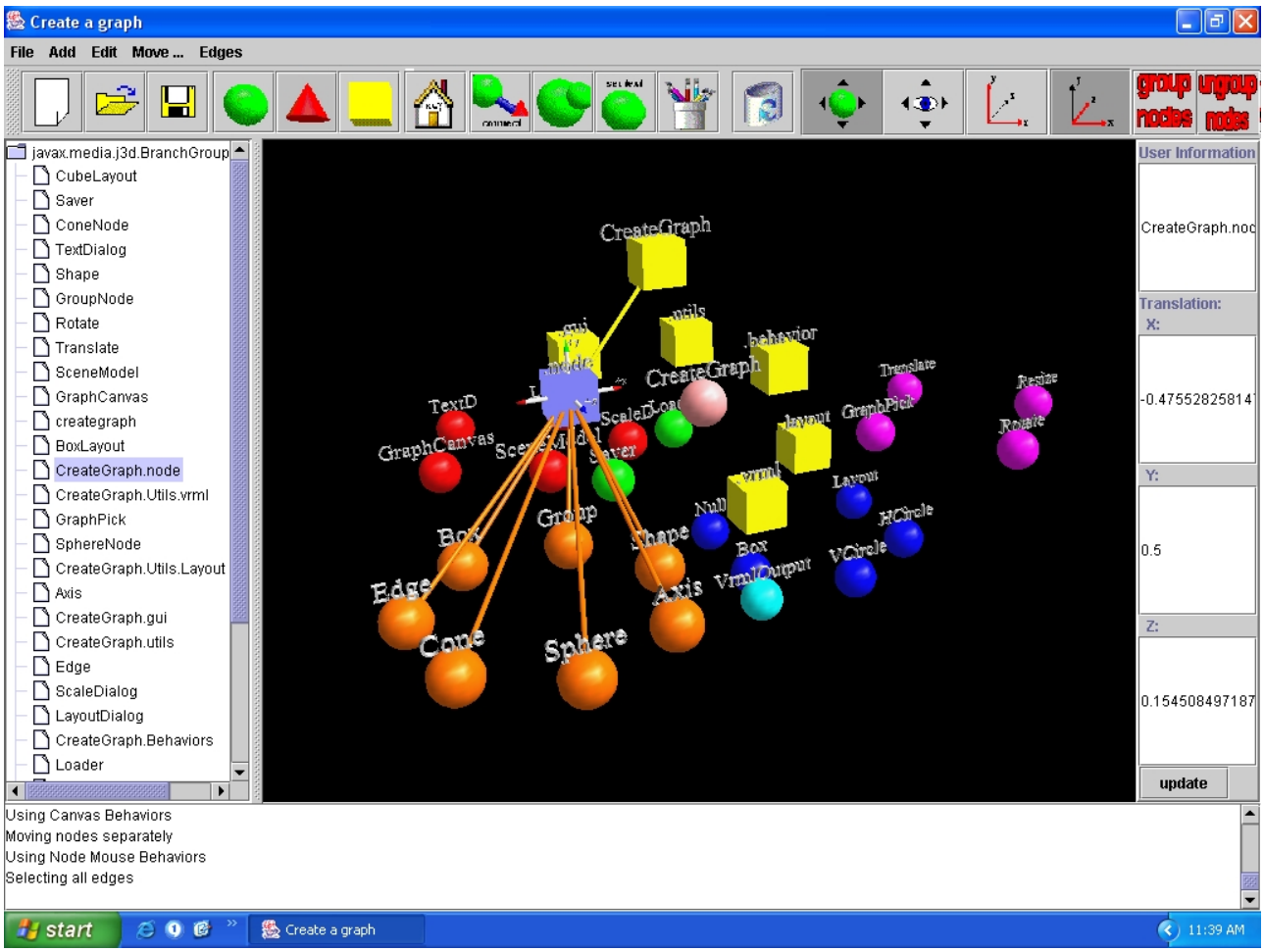

Figure 3 Inspection of Model