3D SCENE CREATION USING STORY-BASED DESCRIPTIONS

Xin Zeng, Q. H. Mehdi and N. E. Gough
Games Simulation and AI (GSAI) Centre,
Research Institute of Advanced Technologies (RIATec)
University of Wolverhampton, Wolverhampton, WV1 1EQ, UK
E-Mail: x.zeng@wlv.ac.uk

KEYWORDS

Story visualization, natural language processing, graphic representation, scene graph.

ABSTRACT

Developments in the creation of 3D virtual environments have made this a popular approach in a variety of applications. However, creating 3D scenes still requires professionals to spend an enormous amount of effort and time on it. In this paper, we present a fully implemented prototype system called 3DSV (3D Story Visualiser) that generates a virtual scene by using simplified story-based descriptions. This prototype system is based on the integration of advances in computer graphics and text-to-visualization technology to generate 3D virtual environment. This makes it easier and faster for non-professionals to create 3D environments compared to using traditional 3D design packages.

INTRODUCTION

Developments in the creation of 3D virtual environments have made this a popular approach in a variety of applications. In particular, it now appears as a powerful new medium for storytelling. The combination of realistic images and interaction give participants more freedom to interact with the story, but how to build this kind of virtual environment still lies in the domain of highly skilled professionals. Currently, human computer interface (HCI) design has now widely stabilized following the so-called WIMP model (*i.e.* windows, icons, menus, and pointers). Also, several menu-based interface design tools (*e.g.* Maya, 3DMax, SoftImage, *etc*) appear as mainstream for designing and building simulated virtual worlds. However, as real-time graphics applications become more complex, the demands of controlling and communicating with them can limit their benefits. This is particularly true of applications where the user manipulates and places objects in 3D scenes, as is the case with scene builders for animation, CAD, architectural design, virtual reality, and visual simulation (Clay & Wilhelms 1996). There is now a growing awareness of the potential for applying natural language (NL) in the computing domain by incorporating knowledge of human-to-human interaction. Research on automatic construction of visual scenes based on NL input dates back to the early 1970s and the SHRDLU system (Winograd 1972). SPRINT focused on spatial relationships to reconstruct 2D models of the world from language input and output the corresponding image on the graphic display (Yamada *et al* 1992). Put is language-based system that focuses on spatial relationships, such as *in*, *on*, and *at*, parameterized by a limited number of environmental variables for object manipulation (Clay & Wilhelms 1996). CarSim implemented language descriptions to generate animation scenes and replay accidents (Tabordet, *et al*). Mukerjee *et al* (2000) used multi-dimensional fuzzy functions to match the linguistic description to present scene reconstruction from conceptions of 2D urban parks. More recently, Coyne & Sproat (2001) presented a system called WordsEye for automatically converting text into representative still images by using linguistic analysis and depiction techniques. This system relies on a large database of 3D models and poses to depict still scenes. These efforts either focus on implementation of natural language in object spatial relationships or represent the still 2D images and scenes without creating real-time interactive 3D virtual environments.

In this paper, we present an approach that enables non-professionals to generate a 3D virtual environment (3DVE) by using simplified natural language input. This work has been most influenced by Yamada *et al.* (1992), Clay & Wilhelms (1996) and Coyne & Sproat (2001) and we also draw on spatial theory from Talmy (1983), Herskovits (1986) and Jackendoff (1993). In contrast to other authors, the originality of the approach described here lies in the generation of a 3D virtual environment by integrating Java, VRML and XML technologies, which enables the user to manipulate the visual features of virtual scene in real-time. The main challenges of this work are to encapsulate the natural language understanding, incorporate real world knowledge, and generate appropriate graphical output. We first provide a system overview and present an example. The subsequent sections then describe the implementation of the prototype system with evaluation and discussion. The final section covers conclusions and directions for future work.

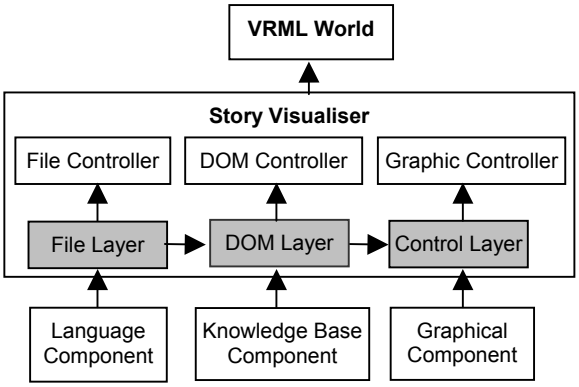

Figure 1. Architecture of the 3DSV

SYSTEM OVERVIEW

To develop a system that could be used to generate a 3D virtual environment (3DVE) and to visualize all of the elements of the story *(i.e.* environment, character and events) is a long-term goal of this research. To limit the scope of the current inquiry, this study focuses on reconstructing the environment of the story, it presents the information of where the story happens, and it is the setting and stage of the story. We argue that it is unnecessary to involve a complicated description to reconstruct the 3D scene. For creating a 3DVE by story-based NL input, we should focus mainly on the direct use of visual features of the environment, such as colour, size, material, direction and spatial features, *etc*. Hence we may start with children's picture books to understand the structure of the story (Zeng 2002). However, it should include constraints and context to help avoid ambiguity. This research is mainly concerned with representing the visual features of the environment by using a simplified story based NL input, which includes attributes of depicting concrete objects and their spatial relations. Furthermore, the system provides an interface that enables the users to manipulate the visual features of the virtual environment in real time. There are three main parts in the system shown in Fig. 1: Language Component, Knowledge Base Component and Graphical Component. There are three main processing layers associated with these three components, namely File Layer, DOM Layer and Control Layer, and each layer comprises a specific controller module to control the sub-modules. The system is highly pipelined and incremental and the input text is processed through these layers before finally being visualized. The system works as follows: Once the text has been entered, the language component analyzes it to extract the information and output a formatted semantic representation as XML; then the output is parameterized by the knowledge base component and is converted into a low-level data structure using a process termed 'visual semantic representation'; and finally the graphic component uses the information to visualize the scene.

Consider a storyline that includes the following sentences to describe the environment:

Part 1. Generate the virtual scene from descriptions.
A wall is left of a room. A cloudy picture is on the wall.
A lake rug is in the room. A wooden table is on the rug.
A green vase is on the table. A gold bowl is next to the vase.
A glass box is right of the table.

Part 2. Manipulate the virtual objects in real time.
Change wall_01 to black. Change box_01 to red.
Move box_01 in front of wall_01.

Figure 2 shows the interface of the prototype 3DSV system and the 3D scene that was generated automatically from the text descriptions. This prototype system current consists of two parts. The first part is a Java application that processes natural language analysis, and the other contains a VRML world and a Java Applet interface to perform real time interaction using language instructions. As seen in Fig. 2, the Java Applet interface comprises three areas that are designated: (a) instructions input field, (b) virtual objects listing area, and (c) information display area.

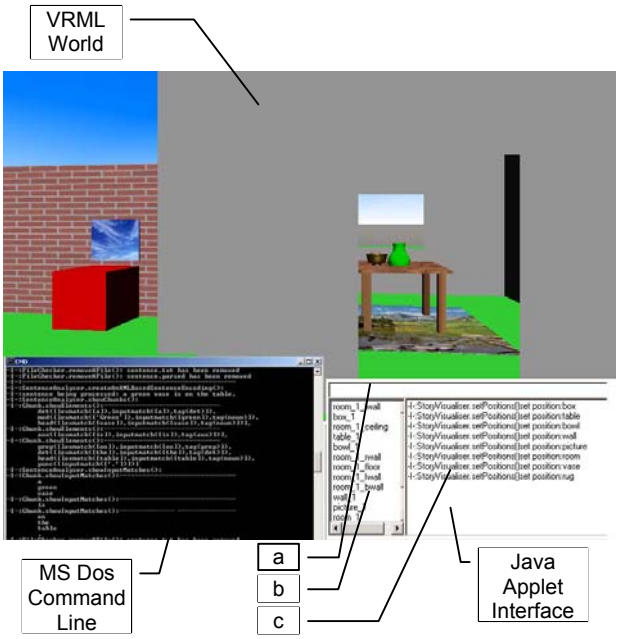

Figure 2. Final Virtual Scene Generated from the Descriptions

The system is written entirely in Java. XML is used as the primary data structure and it provides a link between these components. VRML was used as the graphic presentation engine. The system comprises a natural language analyser in XML and Java, the VRML world, and a Java applet to form the input interface and finally create links between natural language and the VRML world. The combination of Java, XML, and VRML provides cross-platform

capability and makes multi-user interaction possible on the Internet; it also offers real time interactivity and extensibility for further development (Zeng 2004). The NLS Java natural language processing tool developed by the National Library of Medicine is used as the main language parser in this work to handle natural language processing tasks. The language component of this system produces a dual representation during the language parsing process, with both conceptual and visual levels. One is to convert the syntactic structure of sentence into a semantic representation to present the meanings and relations inherent from the input sentences; the other is to allow the graphic component to begin to assemble the original virtual scene once the 3D objects (entities) associated noun phrases have been extracted. To illustrate the entire process of the language analysis, consider the sentence *A gold ball is on a blue box*, the data flow diagram of the language component associated file layer is shown in Fig. 3.

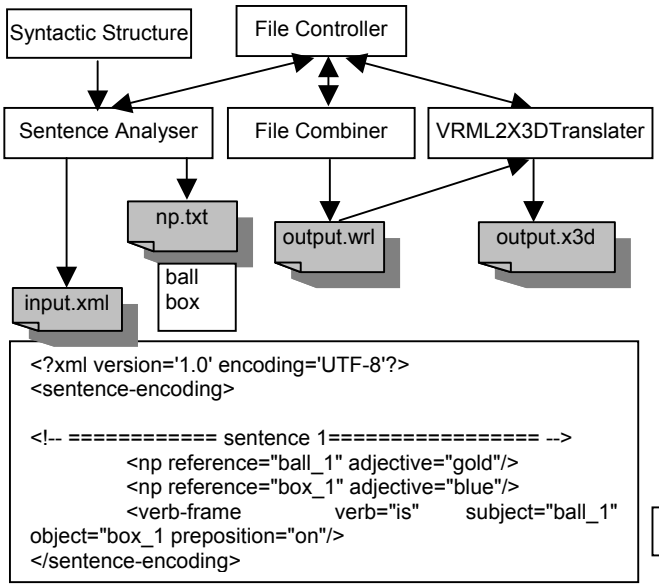

Figure 3. Data Flow Diagram of File Layer (Language Component)

There are two outputs after the sentence analyzer. One of the files is called *input.xml* which contains the *np* elements, with reference and adjectives attributes, the value of the attributes associate with 3D objects and their properties; it also includes a verb-frame element which indicates the spatial relations of the identifiable 3D objects of the virtual scene. It acts as the input source for the graphic component to reconstruct a virtual scene. Another output is *np.txt*, which helps the file combiner to create a new VRML file (*i.e. output.wrl*) from the VRML object database. This new VRML file also has two kinds of usages: one is to enable the system to generate the original virtual scene; the second is to convert it into an X3D file (*i.e. output.x3d*) for further operations.

KNOWLEDGE BASE DEFINITION

As natural languages often describe visual scenes at a high level, these leave out many of the details that have to be specified and translated into a program language that the computer can understand. Most graphics community research in language-based control has involved associating some appropriate words with program functions Zeltzer (1991). The meaning of the words is considered as a primitive resource for visualization, and this direct association word-concept-visual is used for knowledge representations. A representation formalism based on word-concept-visual information has been proposed by linking between the levels of meaning and visualization. Hence, an extendable dictionary called a "**Descriptionary**" was created in the knowledge base component by using XML-based word frames to parameterize a few "visual or describable words" into low-level data in order to bridge the gap between natural language and graphic component. In the current stage of development, the vocabulary has been semantically parameterized and classified into three XML files: Virtual Object Definition (VOD) describes nouns; Virtual Object Attribute (VOA) defines adjectives, and Virtual Object Spatial-relation (VOS) associates prepositions. The basic idea of the design was to incorporate real world knowledge and the principals of linguistic and cognitive theories proposed by Talmy (1983), Herskovits (1986) and Jackendoff (1993), and also to expand the applicability and specifications of VRML in order to suit the task of scene generation (Zeng, *et al* 2005).

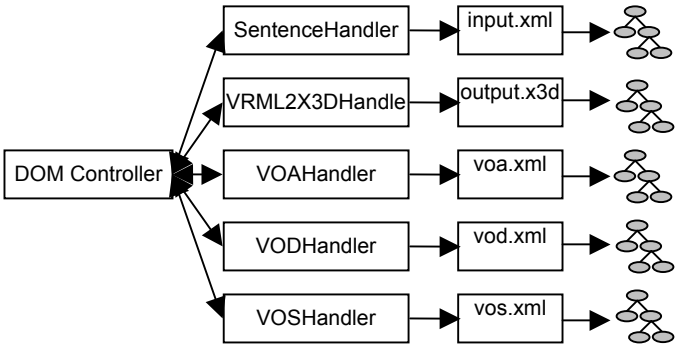

Figure 4. Data Diagram of DOM Layer

In order to describe the parsing process, Fig. 4 illustrates the data flow of the DOM Controller Module. During the initialization phase of a simulation, the DOM controller module handles all the XML files and exchanges the data between the different components. Three DOM handler classes have been created to parse the VOD, VOA and VOS associated XML files respectively. Each DOM handler reads the file and produces a DOM tree; then the trees are stored in the memory, and they can be accessed by the graphic component to set the attributes of the object or change the spatial relations of the objects in the virtual

scene. The remaining DOM handlers are designed for handling output from semantic representation and X3D files respectively.

GRAPHIC REPRESENTATION

The graphic generation stage follows the visual semantic parameterization in our visualization process. Parameterized data is specified and translated into arguments of Java classes to call corresponding objects or manipulate them by sending events into an existing VRML scene to construct a VE accordingly. An overview of the structure of the graphic component is shown in Fig. 5. It comprises three parts: a Control Layer that forms an input interface; the VRML world that is linked through the EAI (External Authoring Interface); and the VRML Object Database that stores the 3D models.

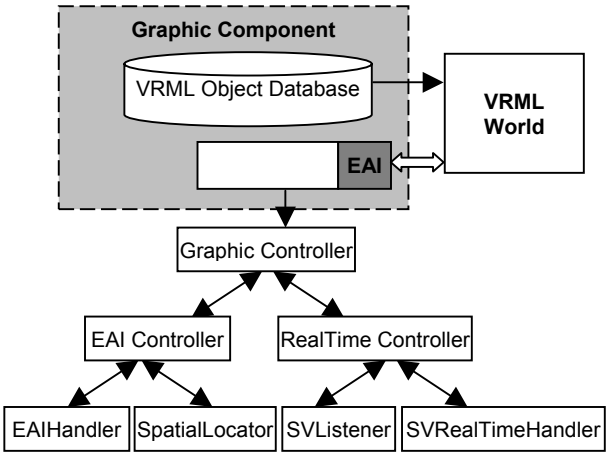

Figure 5. Overview of Graphic Component

The Graphic Controller consists of two main controllers, each controller having several different sub-modules to handle the different tasks (*i.e.* modify the attributes of the objects, placement of the objects, *etc.*) and determine the final visual representation. There are three main implementation modules of our graphic component, which include an EAIHandler module for handling the visual attributes of the object, a SpatialLocator module for operating the spatial relations of the object and a RealtimHandler module for manipulating the virtual scene in real time. Each module is designed to address different goals, but they all share information through the control layer of the StoryVisualiser module.

The StoryVisualiser seen in Fig. 1 is a core module of the system that contains the three layers (*i.e.* File Layer, DOM Layer and Control Layer) and acts as an intermediate to link these layers after the text is processed. It also acts as an execution module of the system because it comprises a Java applet and a VRML world. The Java applet is designed for manipulating the elements of the VRML world by implementing EAI.

Dealing with the Objects' Visual Attributes and Spatial Relations

The EAIHandler module contains the methods to handle two tasks. The first is to create a virtual objects container to store the name of the object for further manipulation. The other is to access the data from the VOA tree and set the attributes for the corresponding virtual objects which are obtained from first part of the semantic representation frame. Then StoryVisualiser sets the initial position of the virtual objects with (x, y, z) coordinates by accessing VOD tree, *i.e.* using the base-align mode to position all the objects on the same horizontal level. It also can resolve the ellipsis situation such as *a gold ball* as an uncompleted semantic representation by simply placing the object in its position. The second part of the semantic representation contains a verb frame which, together with a preposition phrase, indicates the spatial relations of the objects in the 3D scene. The SpatialLocator module is requested to locate the objects in the correct positions by accessing both VOD and VOS trees. Finally, an SVListener module is called for listing all the DEFed objects in the GUI in order to allow the SVRealtimeHandler to manipulate them in real time.

Considering the complexities of the task of visualizing a natural language description, there is a need to find a way to restrict both natural language and graphic visualization. Story based natural language has been used as an input source to simplify the task of natural language processing. In addition, we now introduce object-oriented geometric constraints into our prototype system. The essence of our approach is as follows:

- Each object is defined by the real value of its spatial attributes through prescribing the detail of the geometric constraints.
- Meanings of the prepositions are regarded as the constraints among the spatial objects by interpreting these constraints as a set of equations.
- Access the numerical constraints among the parameters and calculate the values.

The SpatialLocator module was created to establish the methods for handling the spatial relations of the virtual scene. The visual semantic definitions of prepositions reflecting our theory for spatial construction are implemented in the VOS component. The SpatialLocator extracts the sequence of the information references from the semantic frame that are used as cues for matching the corresponding parameters. It then extracts spatial constraints about the objects and the preposition phrase by accessing the related nodes from the VOS. Finally, the extracted qualitative constraints are interpreted as the

numerical constraints among the objects. Once the numerical constraints have been extracted, the layout of the virtual scene can be constructed through the Story Visualiser module by updating the position of the objects. However, not all of the prepositions are one-to-one associations and some problems may appear if a preposition contains more than one geometric description. These will cause the system to present the spatial relations incorrectly. Thus, a filter is created in the SpatialLocator module for handling some specific prepositions, such as on, in, under, *etc.* The filter contains different operations corresponding to the different situations. The use of this spatial filter enables the system to cope with both common and specific spatial relations.

Interacting with the VE in Real Time

As in all language-based visualization systems, there does exist a best instantiation that satisfies all the required constraints of the system, from both natural language description and graphic visualization viewpoints. However, there also exist imprecise or unexpected visual scenes that are mainly caused by inherent under specificity or semantic ambiguity in the given text Mukerjee *et al* (2000). In order to get the proper result, the user may be requested to re-phrase the sentence or adjust the descriptions to provide adequate information which the system requires. This prototype offers two types of modification. One allows the user to adjust the descriptions with or without changing previously input text. The system constantly updates the virtual scene by analysing the entire descriptions from start to end. Hence, it does not matter how many descriptions have been given and the system always chooses the last one as final information provided to reconstruct the virtual scene.

The second modification enables the user to manipulate the scene in real time by combining mouse and language instructions. The entire VRML scene can be driven by a 2D interface Java applet in real time through the EAI. This allows us to take advantage of Java features such as keyboard input, and manipulate the VRML world directly by using language instructions. However, some problems are encountered when using language instructions. The first and also the most difficult problem is how to identify the objects involved in the virtual scene reconstruction. In real life, people tend to use the inherent properties of the objects as the key identification to distinguish other objects, such as colour, shape, *etc.* For computational visualization purposes, we can mimic these methods by adding more constraints to the system, but this approach would be computationally expensive, and clearly this is not a solution for a scene with complicated spatial relations or assembled from a large number of objects. For example, suppose there are two identical tables in the scene, and spatial regions are used such as *the red table or the table on the left/right hand* to identify one specific table. This will present extreme difficulties if this happens on a group of objects and it is unrealistic to ask the user to give instructions such as *the table is all the way on the right hand side below another green table* to select the object before manipulating it. Therefore, we simplify this process here by integrating mouse and language instructions. This also affords us the chance to design the command to be efficient and constrained and keep the manipulations simple and easy to use. All these functions have been defined in the SVRealTimeHandler module, which consists of two kinds of real time instructions: One is for setting the attributes of the object. The second is for changing the positions of the objects. All these commands can be typed in the interface of the StoryVisualiser module.

EVALUATION AND DISCUSSION

The main purpose of the experiments was to evaluate the prototype system in order to determine its capability for use as a system for non-professional to generate a virtual environment. The evaluation also aims to determine the advantages and the weaknesses of this prototype in order to generate suggestions for future development. Therefore we identified the following measures and design criteria: As a system, usability is an important aspect of software products and is about supporting users in achieving their goals in their work Bevan (1997). According to the ISO 9241-11 (1998) definition, usability concerns the use of the system (*i.e.* effectiveness, efficiency and satisfaction in a particular context of use). The value of the prototype as a tool for a non-professional to create virtual environment is measured by Ease of Use and Learnability. The usefulness of the prototype as a means to rapidly generate an interactive virtual environment can be assessed by comparisons with other similar systems.

Guided by above design criteria, we divided the usability evaluation into three main experiments; namely usability test, performance test and database test. The main usability test measures the usefulness, effectiveness, satisfaction and learnability of the prototype system, and is designed to gather data from interviews using the prototype system. The performance test focuses on the responsiveness of the system—the time and resource required for generating different complexity of virtual scenes based on different computer system. The database test concentrates on the database's maintainability and extensibility. Then we present a comparative study and evaluation against other similar systems.

Main Usability Experimental Evaluation

A total of 6 experimental subjects (age between 20-35, one female, five male) were asked to take part in the experiments. They were undergraduate and postgraduate students in the School of Computing and IT at University

of Wolverhampton. The participants first received a brief written instruction describing an introduction to the test and explaining the basic procedure of using the prototype system, followed by 30 to 40 minutes usage of the system. Immediately after completing the experiments, participants were asked to answer a series of questions to rate the overall prototype system. The design of the usability questionnaire is based on IBM Computer System Usability Questionnaire (Lewis 1995). It has 10 usability related assertions, each with a seven point scale ranging from 1 (strongly agree) to 7 (strongly disagree), and open fields for comments. We have extended it with background questions about the participants.

The overall conclusion of the evaluation is that the subjective opinion of users regarding the prototype system is good. Most of participants like the idea of creating a virtual environment by using language descriptions. We find that the results are satisfactory concerning the usefulness of the system and the questionnaire shows that most of the participants say they were very satisfied. This indicates that this prototype system provides the utility and functionality that enables the user to generate various 3D virtual environments. The prototype system has been considered particular easy to deploy, while most participants ranked the ease of use very good. The best score has been given for the learnability of the system. After reading the instructions, participants were able to generate their own virtual environment very efficiently. The overall system satisfaction chosen by the participants is good, implying there is some room for improvement, but overall the result is acceptable because it is affected by the interface. The GUI of the system received the lowest score and the participants stated that they would prefer one world containing all the models rather than separate windows. This is due to the limitation of Java Applet. Based on the observations and participants' comments, several improvements to the prototype system are recommended. An error exception handler is needed to help the users make modifications needed. Currently the prepositions used in the system only handle a single object, and this would be extended to handle two objects, such as *a ball is between a box and a vase*. This can be achieved by defining a relative rule in our knowledge base component, as our language component is robust enough to handle this kind of sentence.

The Performance Evaluation

A performance evaluation scheme was designed to analyze the time and computational resources required according to the number of tasks processed. This experiment is thus divided into two subtests. One is to assess the differences for generating different levels of complexity of a virtual environment based on one computer system, and the other is to measure the differences between performing the same tasks on different computer systems.

In the first test, the comparison of how performance is affected by the level of complexity shows that the time increases as the number of objects is added in the virtual scene, since it is required to input more sentences to achieve this goal. However, we can be satisfied with the speed of the natural language component of 3DSV because it can parse more than one hundred lines of text less than one second. The results indicate that a scene containing a greater number of objects take longer to render than a simple scene. Several elements of VRML can influence the rendering speed of a scene and download time. Shapes in a VRML world are made of polygons. The more complex an object, the more polygons are required. The more objects are created in a virtual environment, the browser requires longer to redraw the scene. Since VRML is a real time scene graph, each time a user's viewpoint changes in the VRML world, the browser has to redraw the scene. Therefore, memory requirements depend on the complexity of the scene. As the complexity of the virtual scene increases, the rendering time increases, and the performance of the virtual environment decreases due to the delay of real-time display latency. In order to construct a VRML world efficiently, it is necessary to compromise the complexity of objects to obtain an acceptable navigational speed and performance. In addition, based on the results from the obtained measurements from second test, it was shown that the performance of the 3DSV on the high standard specification computer is better than low standard specification computer, especially in term of the requirements for the time for parsing text and the rendering speed. Therefore, as computer equipment available to most users becomes more powerful and more affordable, it will be feasible to run VRML based virtual environments on general low cost computer systems.

Database Evaluation

Our study examined results from a maintenance period beginning with the original installation of the system. Six users were asked to add several adjectives and prepositions respectively into the databases. These were accessed and used successfully by the prototype system to produce the various virtual environments. However, some aspects need improvement according to the users' comments. For example, it may be more effective to store the content of the New Lexicon in the style of a database table rather than three different files.

Qualitative Comparison with Other Systems

In this comparison, the main features of 3DSV were evaluated by comparing it to four other text to scene systems: SPINT (Yamada *et al* 1992), Put (Clay &

Wilhelms 1996), CarSim (Tabordet *et al* 1999) and WordsEye (Coyne & Sproat 2001). As a result the 3DSV system appears to offer a similar set of features as these systems, regarding semantic representation, knowledge representation and graphic visualization. The 3DSV system in common with SPINT and WordsEye, is a research framework designed to visualize the natural language descriptions. Moreover, it also enables a user to produce a 3D virtual environment rather than 2D image or sketch, and has real time interactive capability that is similar to Put. Although in the current stage of development the prototype system only handles a limited number of the vocabularies, and focuses on creating and manipulating the properties of 3D virtual objects in virtual environment, this can be extended by further development. The 3DSV architecture is built to be modular and user-extensible. The core Knowledge Base component used in the system is encoded in XML. This guarantees ease of implementation and expandability, promoting further growth of the system.

CONCLUSION

In this paper, we present a fully implemented prototype system called 3DSV (3D Story Visualiser) that generates a virtual scene by using simplified story-based descriptions. We believe that integrating natural language and 3D graphic techniques provides a flexible and easy way for non-professionals to generate a virtual environment as compared to traditional 3D packages. Story is the primary potential source for applying visualization techniques and the scene graph based VRML provides the opportunity to manipulate and modify properties of the environment in real time. Through a mixture of user trials and comparison with other similar systems, it was shown that the method presented in this research works successfully in practice through the experiments. While the main usability evaluation was only assessed by a small group of participants, the evaluation showed that the prototype 3DSV system developed is functionally sound and the greatest advantage of the system lies in its ability to evolve tools that can effectively be used by non-professional to create a real time interactive virtual environment. The development of the 3DSV is only the first step towards our ultimate goal of developing a system for the entire story visualization tasks. Future work will concern improving the language component and expanding the Descriptionary for more complex tasks. Use of X3D, seems to be the most suitable technology, especially in the case of the "semantic-oriented" research. Research will also take place on how to embed a BBAI (Behavioural-Based AI) character and design an event engine to facilitate the generation of more realistic and dynamic story-based 3D virtual environments in the domains of time, space and motion.

REFERENCES

Bevan, N. 1997. "Quality in use: incorporating human factors into the software engineering lifecycle." In *ISESS*, 533-552.

Clay, R. and Wilhelms, J. 1996. "Put: language-based interactive manipulation of objects." *IEEE Computer Graphics and Applications*, 16(2), 31–39, March.

Coyne, B. and Sproat, R. 2001. "WordsEye: An automatic text-to-scene conversion system*."* In *Proceedings of 28th SIGGRAPH Annual Conf. Computer Graphics and Interactive Techniques,* 487-496. Los Angeles. CA.

Herskovits, A. 1996. "Language and Spatial Cognition." *An interdisciplinary Study of the Prepositions in English*. Cambridge University Press, Cambridge, MA.

ISO/IEC. 9241-11. 1998. Ergonomic requirements for office work with visual display terminals (VDT)s - *Part 11 Guidance on usability, International Organization of Standardization*.

Jackendoff, R. 1993. "*Pattern in the Mind: Language and Human Nature".*" Harvester, Wheatsheaf Press.

Lewis, J. R. 1995. "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use." *International Journal of Human-Computer Interaction*, 7(1), 57-78.

Marrin, C. (1997) Proposal for a VRML 2.0 Information Annex. External Authoring Interface Reference. http://www.web3d.org/WorkingGroups/vrml-hisotry/eai_draft.html. Last access: 09/08/2002.

Mukerjee, A., Singh, M and Mishra, N. 2000. "Conceptual Description of Visual Scenes from Linguistic Models." *Journal of Image and Vision Computing. Special Issue on Conceptual Descriptions*. V18.

Tabordet, F., Pied, F and Nugues, P. 1999. "Scene visualization and animation from texts in a virtual environment, CC AI." *The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology*. Special issue on visualization, 15(4): 339-349.

Talmy, L. 1983. "How language structures space." *Spatial Orientation: Theory, Research, and Application."* Herbert, P and Linda, A. (Eds.), 225-282. Plenum Press. New York.

Winograd, T. 1972. "*Understanding Natural Language*." Academic Press.

Yamada, A., Yamamoto, T and Ikeda, H. 1992. "Reconstructing spatial image from natural language texts." In *Proceedings of COLING 92*, 23-28. Nantes.

Zeltzer, D. 1991. *"Task-level Hraphical Simulation: Abstraction, Representation, and Control."* In *Making Them Move*, 3-33. Morgan-Kaufmann. San Francisco.

Zeng, X., Mehdi, Q and Gough, N.E. 2002. "Generation of a 3D virtual story environment based on story description." In *Proceedings of 3rd SCS International* Conference *GAME-ON 2002,* 77-85. London.

Zeng, X., Mehdi, Q and Gough, N.E. 2004. "Implementation of VRML and Java for story visualization tasks." In *Proceedings of 5th SCS International Conference on Intelligent Games and simulation, GAME-ON 2004,* 122-126. Reading.

Zeng, X., Mehdi, Q and Gough, N.E. 2005. "From visual semantic parameterization to graphic visualization." *Proceedings of IEEE 9th International Conference on Information Visualisation,* 133-138. London.