Applying Deductive Techniques to the Creation of Realistic Historical 3D Spatiotemporal Visualisations from Natural Language Narratives

Amanda Oddie
Liverpool Hope University
Hope Park,
Liverpool L16 9JD
oddiea@hope.ac.uk

Paul Hazlewood
Liverpool Hope University
Hope Park,
Liverpool L16 9JD
hazlewp@hope.ac.uk

Brian Farrimond
Liverpool Hope University
Hope Park,
Liverpool L16 9JD
farrimb@hope.ac.uk

Steve Presland
Liverpool Hope University
Hope Park,
Liverpool L16 9JD
preslas@hope.ac.uk

This paper builds on previous research into the development of the TMap3D system which allows for the creation of historical 3D spatiotemporal visualisations from natural language narratives [1]. This paper addresses the use of deductive 3D modelling techniques to generate the realism of the object behaviours, the environmental conditions and their interactions within the visualisation. The authors define deductive 3D modelling techniques as follows:

(i) Given an object and its specification the Vist3D system (formally TMap3D) can deduce the object's visualised behaviour in response to a given "real world" instruction from the natural language narrative.

(ii) The Vist3D system can deduce visualised environmental conditions from the natural language narrative.

(iii) The Vist3D system can deduce the interactions between the objects (e.g. ship) and the environmental conditions (e.g. rough sea) to generate, for example, a ship rolling in rough seas.

Deductive modelling. Vist3D. TMap3D. Spatiotemporal. 3D Modelling. Deductive visualisations. History.

1. INTRODUCTION

The creation of animated visualisations of real and imagined events has significant advantages in many fields. These include the study of historical events and the demonstration of the working of complex historical devices and operations. There is much interest in these areas as is evidenced by the popularity of social and economic history, military history and industrial heritage and archaeology. A significant contribution to the UK economy is made by the heritage industry. A fundamental need when exhibiting these areas in books and museums is trying to explain what was actually going on. Visualisations have a major role to play in meeting this need. Many museums, for example, use re-enactors and recreate bus and train timetables using vintage buses and trains to provide such visualisation.

Visualisations deepen understanding of what is going on especially for the majority of people who powers of mental visualisation when reading textual descriptions are quite limited. For example, the description of a battle can soon become confusing especially when several things are happening at the same time. Some of these problems can be overcome by the aid of photographs, movies and diagrams but these often do little to help beyond giving an impression of what is happening. For example, movies of battles show guns firing etc can make a strong impression but give little feeling for the context in which the images are embedded, for example what are the guns firing at and why, and what else is going on at the same time. Similarly the functioning of a complex operation such as a nineteenth century harbour is inadequately described in this way. Strong impressions are given by surviving photographs of the harbour but the sense of how the harbour operated and so the purpose of much of what is seen in the images is missing.

Visualisations have been achieved in a number of ways:

- Recreating or restoring parts of the real thing. Examples include heritage railways as at the Keighley and Worth Valley Railway. These rely on systematic support by volunteers who regard it as a labour of love.
- Creating physical animated models such as is done by railway modellers and naval modellers.
- Creating computer models (Goskar 2007)

The current problem with visualisations by recreation/restoration or by physical modelling is that they are expensive and time consuming to make and often only give a sample of the real thing. For example, heritage railways and model railway layouts focus on a handful of stations with usually a limited number of trains in operation. Heritage railways are also very expensive to maintain. The locomotives used in model layouts require considerable skills to make or are expensive to buy.

There are visualisations where environmental conditions are portrayed. Often these:

(i) Are limited to pre rendered video clips or animated fly through (Goskar 2007) which involve significant expertise in high end visualisation packages and data capture facilities.
(ii) Utilise game engines such as CryEngine 3 (CryEngine3 2011) which allow for realistic environmental conditions such as fire, water and smoke which require specific expertise in game engines and programming and the use of high powered computers [3].

The problem with computer visualisations is that they are also expensive to make since they require considerable modelling skills and take a great deal of time to produce (Slusallek 2010**,** Bentkowska-Kafel 2007). These constraints make it very difficult to create fully functional and true to historical period visualisations that can be observed from multiple viewpoints over periods of time.

In contrast using deductive 3D modelling the Vist3D system can generate high quality interactive visualisations without the need for in depth expertise or an understanding of 3D modelling techniques and operates on a wide range of computer specifications.

The TMap3D project addresses the goal of creating realistic visualisations that explain what is going on in complex events and operations at considerably less expense. It aims to achieve this by combining parameterised modelling with deductive visualisation to create sets of tools that enable enthusiastic communities to build such visualisations without needing skills in 3D modelling on computers

This paper specifically examines the use of deductive 3D modelling using two scenarios relating to the Battle of the River Plate (1939) and the reconstruction of Cape Town Harbour during the 1890s. The scenarios illustrate how the deductive modelling process is triggered from the natural language narrative, as illustrated in the figures 4 and 7. The paper then goes on to explain how the system then deduces behaviours and events.

2. RELATED WORK

There are a range of approaches to modelling and visualisation. The main difficulty with 3D development and visualisation is that it requires a good knowledge of the application. There are a range of approaches to the visualisation of 3D scenes.

- Use of modelling software and use of constraints and constraints solvers
- Declarative 3D modelling
- Text-to-scene techniques

The more traditional approach makes use of editing software for 3D modelling and CAD work. This approach relies of knowledge of the application and modelling expertise either to be able to visualise the scene, work form reference images or make use of data input. Any movement or animation is done by hand with the aid of application functions such as modifiers, constraint solvers and physics engines which, in themselves, require understanding or a scientific approach. Constraints describe the relationship between objects that need to be maintained when a function, such as rotation, is applied, for example movement in an arm. In this example all segment of the arm move relative to each other when a constraint solver is applied (in this case Inverse Kinematics, IK).

Using such functions the user can define the relationships between object and physically build and co-ordinate the scene graph i.e. set up constraints and apply constraint solvers in the form of, for example, modifier functions and physics.

Declarative modelling enables designers to concentrate on describing what they want to create instead of how they should model it. Declarative modelling and text-to-scene approaches, discussed later, endeavour to mitigate the requirement for expertise in different ways. Declarative systems try to tackle the issues by proposing mark up languages that allow a user to describe their ideas for models or 3D scene textually. The language

and their underlying framework will try to resolve issues, build object relationships, constraints and visualise the objects. Use and style of declarative techniques vary. Mark up languages such as VRML, its descendent X3D (Web3D Consortium, 2011) and declarative X3D in HTML 5 (Beher & Jung 2010) can be used to define visualisations. However these require a level of expertise with the syntax and a technical understanding of the framework. Some approaches use a hybrid approach, i.e. using natural language with modelling techniques, to capture the users ideas and hide issues to do with modelling and visualisation. Donikian *et al*. (1993) with the aid of architectural primitives allow a user to make statements about the objects' size, position, angle of viewing etc. Smelika *et al.* (2011) combine textual with procedural visualisation techniques to achieve 3D visualisations. It is important to note that declarative approaches do not normally have a spatiotemporal component.

Text-to scene research illustrates efforts into applying a natural language approach to creating 3D visualisations. Ideas are presented in the form of a textual description and the system processes this into a 3D scene. Text-to-scene conversion involves a computer analysing a textual narrative and interpreting the spatiotemporal data held within it and then displaying that data in the form of a 2D or 3D visualisation.

Natural language input to 3D systems has been investigated in a number of systems. An early system by Adorni *et al*. (1984) led the way. This system and others, e.g. (Coyne & Sproat 2001, Arens *et al*. 2002) were limited to very simple narratives and tended to focus on spatial relations and ignore the temporal aspects. In addition, most of these were limited to generating static scenes. The Vist3D system currently utilises a simple natural language parser based on regular expressions and so is similar to these, but does include specific identification of temporal aspects by recognising timings of events explicitly.

The Carsim system (Dupuy *et al*. 2001, Johansson *et al*. 2004) is a text-to-scene converter that analyses real traffic accident reports and generates a 3D animation from it, allowing the accident to be visualised in three dimensions. Like the TMap3D system, CarSim objects are placed in both time and space. It is restricted to the car accident domain. The system had a similar aim to that of the TMap3D system in that its developers recognised that some people have difficulties in imagining situations and that they may need visual aids developed by professional analysts. Their system, like the TMap3D system, uses both a natural language processing component and a visualiser in order to generate a 3D scene. One of its significant features of Carsim is that it recognises time and temporal relations between events and uses this information to produce the animated synthesised scene.

The language processor in Carsim uses an information extraction strategy that is significantly more complicated than that currently used in the Vist3D system. It includes machine learning methods to solve coreference, classify predicate/argument structures, and to order events temporally. The developers recognised that real natural language texts suffer from under-specification and, in their case, rarely contain sufficient geometric descriptions of accidents to enable the automatic conversion of narratives into images. To overcome this, Carsim infers implicit information about both the environment and the entities involved from key phrases in the narrative text, knowledge about typical traffic situations and the properties of the entities involved. The TMap3D system also resolves under-specification using similar deductive techniques, particularly using knowledge about typical object behaviours and the properties of the entities involved. Carsim uses a visualisation planner to apply spatial and temporal reasoning to "imagine" the entities and actions described in the text and that tries to find the simplest configuration that fits the description. Similarly, the TMap3D system uses the Vist3D component to generate keyfames for the animation based on information from the scenario, common resources, visualisation specification, model files and terrain files

AVis (O'Kane *et al*. 2004) is another automatic text-to-scene conversion system similar to CarSim. Its developers also recognised that it can be difficult for non-expert readers to trace increasingly complex arguments that are typically presented in the form of accident reports. It uses GIS information to generate scene layouts and Case-Based Reasoning techniques to augment information extraction. So if, for example, the extracted information is insufficient to construct a scene, then a corpus of accidents is consulted to fill in any gaps in the output file using information from similar accidents.

The Vist3D system utilises similar features to those found in the text-to-scene applications by allowing the user to generate spatiotemporal visualisations by simply entering a historical narrative and textual descriptions. The Vist3D system adopts the basic idea of constraints and builds up the relationships between objects (either separate ones within a scene or complex objects e.g. a ship and its gun turrets) by parsing the narrative and looking for appropriate characteristics. The simulation file and the associated scene graph act as a constraints solver. State changes in the scene graph carry out

movements and changes in the objects such as turrets elevating.

This paper examines how the system makes use of these approaches to deduce the behaviours simulated in the visualisation.

3. DEDUCTIVE MODELLING AND TECHNICAL OVERVIEW OF THE VIST3D SYSTEM

3. 1 Achieving Deductive Modelling

The TMap3D project is developing tools to enable the rapid development of classes of objects such as ships, harbours, steam engines. These tools enable reasonable 3D models to be created in a very short period of time. The models can include components that can be moved realistically e.g. the turrets and guns on a ship, the yards and sails on a sailing ship, the valve gear in a steam engine.

Deductive visualisation tools enable visualisations to be generated by deducing the animation of objects within the visualisation from textual descriptions of their operation and specifications of how the objects themselves operate

Using the Vist3D system visualisation is achieved using scene graphs in vector based graphics. Deductive visualisation is achieved by identifying the corresponding state changes over time to the scenario scene graph from a textual description of a scenario and textual operational specifications of the objects involved in the scenario. The scene graph state changes implement the movement of the objects and the operation of their component parts.

In the TMap3D project a scenario is a textual description in near natural language that is parsed and stored in a temporal database. Implementation of a visualisation of a scenario consists of:

- creating the models of objects and terrain to be used in the visualisation. The models are held in a model library
- generating a simulation file containing the scene graph specification and the keyframe and action information to be used to animate the scene graph.
- converting the simulation file into a serialised object file or a VRML file to be loaded and displayed by a scenario viewer.

The project currently generates simulation files that can be viewed in VRML browsers or in Panda3D. Other scene graph viewers using formats such as Open Scene Graph and X3D may be created.

3.2 Vist3D System Overview

Vist3D represents the further development of the TMap3D system (Presland *et al*. 2010). It generates interactive 3D spatiotemporal visualisations of places and events and currently allows these to be visualised in either Panda3D or VRML. The Vist3D system architecture is illustrated in Figure 1. The system comprises of three main sections; the Narrative Component, the Analyser and the Viewer.

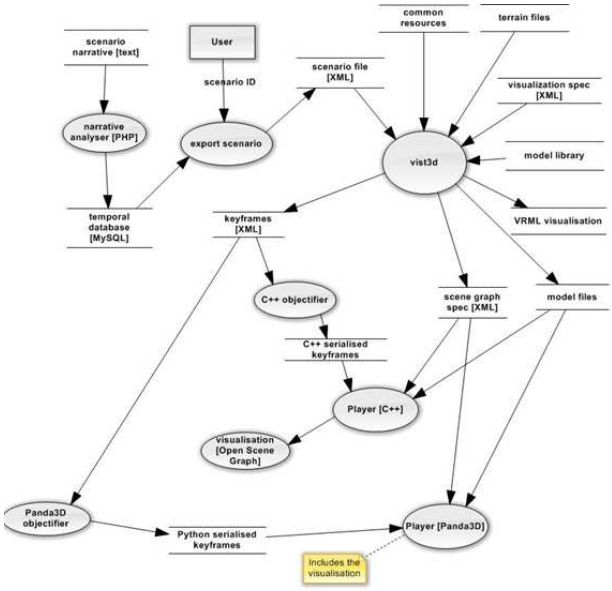

Figure 1: *An Overview of the Vist3D System*

3.2.1 The Narrative Component

The narrative component of the Vist3D system comprises of three main elements the Narrative Editor, the Narrative Parser and the Temporal Database. The Narrative Editor allows a user to supply a textual narrative or to select an existing narrative from the Temporal Database.

The Narrative Parser has been developed in PHP and makes use of regular expressions. This parses any narrative supplied by the user and populates the Temporal Database. The Narrative Parser processes each sentence in turn. Firstly, it identifies any date and time information, recording it as a time stamp. It then removes the date and time information from the sentence. If no date and time information is present, then the previous time stamp is used. If the date and time information is partial (e.g. time of day only) then the missing information is inferred from the previous time stamp. The start time is used to time stamp locations and objects as they are introduced. Any explicit mention of time overrides this initial time. The remainder of the sentence is then searched for particular verbs and verb phrases using the following sentence structure: <subject> <verb> <object>. The Narrative Parser generates records

according to the particular verb or verb phrase found in the sentence. If the <subject> is "It", "He", "She" or "They" then the subject of the previous sentence is assumed. The Narrative Parser finally invokes SQL commands to store the parsed data in the Temporal Database. The class of each kind of object (currently either a place or a unit such as a ship), referred to as a “thing” within the system, is identified from whether it is introduced with the phrase "is located at" or the phrase "is a ... unit". The narrative is written as a sequence of sentences (see Figure 2). Each of which is terminated by a full stop. Notes in square brackets appear as subtitles in the visualisation (Presland *et al*. 2010).

```
"The schooner Compass Rose departs berth
17 at 11:30 am on its way to Rio de
Janeiro.
At 13.00 the Compass Rose turns to heading
080"
```

Figure 2: *Cape Town Harbour Narrative Extract*

A temporal data model has been designed to allow the storage of time-stamped facts. The data model captures the temporal data content from the narrative. The system is designed to be flexible and store any attributes specified in the narrative as it is impossible to predict what attributes the user may include in the narrative.

3.2.2 The Analyser

The Analyser creates the scenario file by extracting the time stamped data from the Temporal Database along with other information such as model files and terrain files to create visualisation files (scene graph and keyframes) to be processed by the Viewer. The Analyser generates its visualisation files in the following way:

- By the creation of models (ships, buildings, structures) and terrain to be used in the visualisation. This is currently done using Vist3D tools such as ship builder. Models are held in a model library.
- The Vist3D Analyser processes the scenario file, terrain, model and other resource files to create a scene graph (the 3D scene). It also generates the keyframes and action information used to animate the scene over time and to provide environmental conditions and other effects. The scene graph and the keyframes data are combined to form the simulation file.

The simulation file can be previewed as a VRML file or run in Panda3D (using the viewer). The simulation file consists of a header, node table, action table, keyframes and linking actions to the scene graph nodes. The simulation file is the key component of the Vist3D system that allows for the deductive modelling.

The Header contains the scenario title, the scenario start and end times and specification of how time is to be displayed. It also specifies how much time changes as the time slider window component is clicked to increment or decrement time. Internally time is measured to the nearest centisecond. The animation viewer will update the visualisation as often as it can as the scenario is run in real time.

The Node table lists each node in the scene graph, identifying what type of node it is and who its parent is. Types of node include translation nodes, rotation nodes, scaling nodes, loaded model nodes and nodes that are combinations of these states.

The Action table also lists each action in the scenario. The term "action" in this context has a specific meaning. An action is an updating operation on a scene graph node that starts at a particular moment in time and ends at a particular moment. There are different action types. Each type is mapped to a specific function of time that executes the action by determining the translation, rotation and/or scaling at the required moment of time in the action's life. The function can accept parameters that are stored with the action in the simulation file.

The state of the scene graph is recorded at fixed intervals of time. (Currently the time step between one keyframe and the next is one second but future work will aim to optimise performance by adjusting this time). Since the user is able to jump to any particular moment of time in the scenario, the corresponding keyframe must contain sufficient information to enable the scene graph to be put into its correct state. Certain scene graph nodes representing the position and rotation and scaling of objects will have their state updated by interpolation between adjacent keyframes. However, other states of the scene graph are better expressed using the concept of Actions as described above.

The following section explains how actions are linked to scene graph nodes. At present, each node record in a keyframe has an action number field so that the node record can be linked to a particular action associated with that number. Only those keyframes during which the action commences, continues or terminates will use this field to point to the action record in the action table. In other keyframes, the action number field will contain -1 to indicate no action performed on the node at those times. For example, suppose node 10 commences its action 5 when it is 68.4 seconds into the simulation and the action terminates after 70.6 seconds. The corresponding keyframe node records for node 10 will look like this (figure 3).

	Keyframe number	Node number	Action number
67	:		
	10	...	-1
	:		
68	:		
	10	...	5
	:		
69	:		
	10	...	5
	:		
70	:		
	10	...	5
	:		
71	:		
	10	...	-1
	:		
72	:		
	10	...	-1

Figure 3: *Action Table*

Suppose current time is 67.5 seconds into the simulation. The corresponding keyframe shows action number -1 for node 10. This means that no action happens in this keyframe. However, at current time 68.5 seconds we are in a keyframe with action number 5. If the action start time is earlier than current time then the action is processed for this time. This continues until current time goes beyond keyframe 70 after which the action number reverts to -1 so no action is processed.

Actions include rotations about the different axes, translations and combinations of these. These actions are deduced from the activities described in the scenarios. Action functions to produce a uniform rotation are straightforward interpolations. However, there is no limit to the complexity of the functions that can be employed (and look-up tables are also available) so that arbitrarily complicated actions can be recorded by appropriate action functions. A simpler example is the function to handle the recoil of a firing gun. The function uses as parameters the initial backwards velocity of the gun (Vimpulse) and the distance travelled during recoil (dist). The backwards velocity is calculated during the generation of the simulation file from the information about gun and projectile weights and muzzle velocity provided in the ship specification file. The distance travelled is also found in that file. The specification file is created from publicly available information.

3.2.3 The Viewer

The Visualisation viewer displays the scenario in 3D – currently using Panda3D or VRML. Controls display the current scenario time and provide buttons to run the simulation in real time. Buttons may be added to speed up or slow down the simulation. A horizontal scroll bar - the time-slider - enables the user to move quickly to any chosen time. The arrows at the end of the time slider can be used to increment or decrement time by fixed amounts. Viewpoints are added to the individual objects as well as the overall scene. These allow the user to examine the visualisation from different perspectives, for example, from the Compass Rose or from the harbour.

The Vist3D Viewer has been designed for flexibility. The 3D scene is expressed in terms of a keyframes XML file and a scene graph XML file. This means that through the use of Vist3D objectifiers that the visualisation could be targeted to other formats such as 3DsMax, C++ or X3D.

4. SCENARIOS ILLUSTRATING THE DEDUCTIVE TECHNIQUES OF THE VIST3D SYSTEM

This section demonstrates the deductive techniques employed by the Vist3D system. This is illustrated using two scenarios that are currently visualised using the Vist3D system; the Battle of the River Plate and the reconstruction of Cape Town Harbour during the 1890s. The reasons for selecting these scenarios are that the Battle of the River Plate was the first major naval battle in the Second World War and it was a well documented engagement, involving a small number of ships with actions and events that would demonstrate the ability of the deductive modelling techniques for example, gun elevation, ships turning and gunfire. The Cape Town Harbour scenario allows for the illustration of reconstruction over time and the use of sailing ships with different characteristics and properties. This allows for different deductive techniques to be illustrated. This section discusses the implementation of the scenarios and illustrates the results of the visualisation, demonstrating the features of the Vist3D system.

In both cases the Scenarios were entered as natural language narratives using the Narrative Editor of the Vist3D system. Textual descriptions were entered into the Analyser component of the Vist3D system as discussed in section 3. These textual descriptions were then used to generate 3D models which along with the narrative were used to generate the keyframes and Scene graph for use in the visualisation.

Figure 4 shows a fragment of the narrative used in the production of the Battle of the River Plate visualisation. The following extract shows how a fragment of narrative triggers the deductive modeller component of the Vist3D system.

```
At 06:18 Graf Spee engages Exeter.
```

The deductive modeller component of the Vist3D system uses details available from the objects specification (the Graf Spee) and the current relative positions of other objects (here the Exeter, Achilles and the Graf Spee) to calculate the visualisation of the ships' behaviours over time. For the fragment shown above this may involve rotating the turrets of the Graf Spee to the appropriate direction, gun barrel elevation, rate of fire and gun recoil. The deductive modeller also generates the appropriate environmental behaviours such as smoke and flames generated from the gun fire. These behaviours will automatically continue until there is a change in the narrative as shown below.

```
At 06:38 Graf Spee disengages from Exeter.
```

It is important to note that these deductions can be generated for the duration of the visualisation.

The image in figure 5 shows that from the phrase "At 06:21 Achilles engages the Graf Spee" that the Vist3D system can deduce which way the turrets are rotated, the angle of elevation of the guns and the flames of the gun firing. These behaviours can be deduced from details given in the narrative such as the location, speed and headings of the ships and from the properties of the guns used on the ships (muzzle velocity and the weight of the gun and projectile) stored in the model files database in the Analyser component of the Vist3D system.

```
"At 06:20 Graf Spee turns to heading 090.
At 06:20 Exeter engages Graf Spee.
At 06:21 Achilles engages Graf Spee.
At 06:22 Graf Spee turns to heading 000.
At 06:22 Ajax turns to heading 350. It
increases speed to 25 knots.
At 06:23 Ajax engages Graf Spee.
At 06:25 Achilles turns to heading 350. It
increases speed to 25 knots.
At 06:24 Exeter turns to heading 315.
At 06:25 Ajax turns to heading 010.
At 06:26 Graf Spee turns to heading 100.
At 06:26 Exeter is hit by Graf Spee on
turret B.
Turret B is destroyed.
At 06:36 Graf Spee turns to heading 325.
[Note: Graf Spee lays smoke. Ajax
launched her spotter aircraft from its
catapult.]
At 06:38 Exeter turns to heading 080.
[Note: Exeter turned so that she could
fire her port torpedoes, and received two
more direct hits from 11-inch shells. One
hit A-turret and put it out of action,
the other entered the hull and started
fires.]
At 06:38 Exeter disengages from Graf
Spee.
At 06:38 Graf Spee disengages from
Exeter."
```

Figure 4: *Battle of the River Plate Narrative Extract*

The static images in Figures 5 and 6 show a snap shot of the visualisation. The deduced slow rotation of the turrets and elevation of the guns is best demonstrated when the scenario is run in real time.

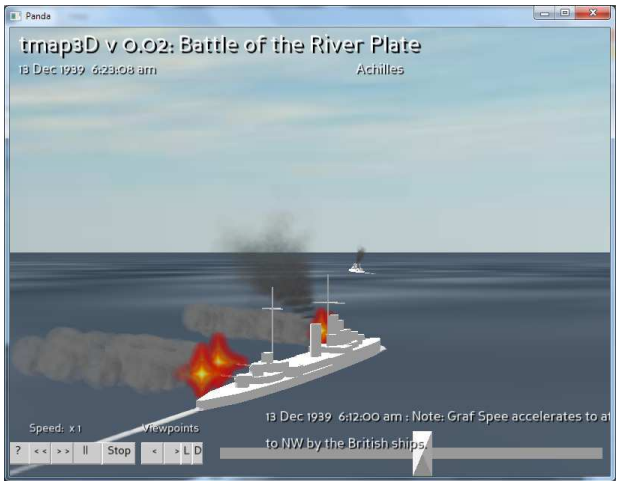

Figure 5: *Battle of the River Plate screenshot showing HMS Achilles opening fire.*

Figure 6: *Battle of the River Plate Screenshot showing the Graf Spee turning*

Figure 7 shows a fragment of the narrative used in the reconstruction of the Cape Town Harbour during the 1890s scenario.

```
"The schooner Compass Rose departs berth
17 at 11:30 am on its way to Rio de
Janeiro.
The pilot cutter Bluebell leaves berth 21
at 11:40 am to rendezvous with schooner
Amethyst at 12:30 pm
In 1891 the harbour was extended to
include a new warehouse
The rendezvous point will be deduced from
the movements of the schooner Amethyst."
```

Figure 7: *Cape Town Harbour Narrative Extract*

This scenario shows further evidence of how the Vist3D system can deduce the behaviours of

different objects, in this case sailing ships, but more importantly how the system can be used to reconstruct a place over time, the spatiotemporal, evolutionary aspects of the system.

The image shown in figure 8 illustrates the movement of the Compass Rose after 20 minutes. In this case the deduction consists of: identifying the track followed by the schooner and having the schooner conform to it. The system also deduces what sails would be set as the ship leaves the harbour at this point in the light of wind strength and direction. Note that in the image they are not all yet set. The two smaller boats on the right are pilot vessels which will have their states deduced from the final statement if the narrative shown in Figure 7.

Figure 8: *Compass Rose Leaving Cape Town Harbour*

The static image merely shows a snap shot in time. The deductions that cause changes in the ships and allow the demonstration of the evolutionary aspect of the reconstruction of Cape Town Harbour during the 1890s are best illustrated in the final Vist3D visualisation.

5. DISCUSSION AND CONCLUSIONS

At present, the deductions are carried out by a Java program which represents objects in the scenario as subclasses of an abstract superclass named Thing. Thus the subclass ThingShip has methods which generate actions deduced from scenario actions such as "Graf Spee engages HMS Exeter at 06:25 am" and "The brig Buttercup docked at berth 31 in CapeTown harbour at 11:20 am".

In the case of the Graf Spee, a sequence of actions is generated for each firing of each gun in each turret. This sequence will be repeated as long as the Graf Spee is engaged and as long as the guns remain in action.

Firstly, the deducer identifies which turrets can engage the target ship. It then generates actions to rotate those turrets to train on the expected location of the target when the shells land given the target's current course and speed. The action will involve rotating the turret objects about the vertical axis.

Next, the range to target is calculated and actions generated to elevate the guns to the appropriate angle. Simple projectile mechanics are currently used but there is no reason not to implement a more realistic function when time permits. The action will involve rotating the gun about a horizontal axis through its point of location.

The guns are made to fire. This is visualised by the guns recoiling through another action and smoke and flames emanating from the guns. The action is of the recoil type described above. The shell splashes are controlled by a further action that causes the splashes to rise and fall at the appropriate places. Hits on the target ship are also visualised by actions. After firing, the guns are lowered to their reload angle, pause during reloading and are then elevated to fire again. All these actions are deduced from the fact that one ship has engaged another.

The second example, the brig Buttercup, is implemented by creating a series of actions that move the ship from its current position to the beginning of a route into its destination berth and then follow the route at the appropriate speed. Obstacles such as swing bridges and lock gates are opened and closed as required. If a tug is to be used then the tug boat will also undergo actions as it lines up to tow the brig into harbour. The harbour itself has a number of routes specified to be followed by ships moving into and out of harbour. All these actions are deduced from the fact that a ship has docked at a particular berth.

The Scenarios demonstrate the deductive techniques of the Vist3D system. They illustrate how the system can be utilised by non-technical experts to create interactive 3D spatiotemporal visualisations that can be useful for both historical purposes in terms of recreating battles and for cultural heritage purposes in terms of reconstructing places over time. More importantly the system and scenarios show that this can be achieved from natural language narrative and textual description of objects. The system has the capability to then generate accurate 3D models and top deduce the behaviours of the models and objects in the scene over time.

6. FUTURE WORK

The Vist3D project team will be seeking the help of from maritime historians to determine the kind of schedules the ships and boats kept in the past so that accurate simulations can be produced for the our scenarios. The Vist3D system is still at the proof of concept stage and as such will require improvements to the interface to improve usability for the users. The team are also looking at particle systems, optimisation and generating sea states to improve the realism and efficiency of the visualisations generated by the system.

7. REFERENCES

Adorni, G., Di Manzo, M. and Giunchiglia, F. (1984) Natural Language Driven Image Generation. *Proceedings of COLING*, 84, 495-500, Stanford, California.

Arens, M., Ottlik, A. and Nagel, H. (2002) Natural Language Texts for a Cognitive Vision System. In *Proceedings of the 15th European Conference on Artificial Intelligence (iECAI2002),* Lyon, July 21-26.

Beher, J. and Jung, Y. (2010) X3DOM: Getting declarative (X)3D into HTML.

TPAC 2010 Technical Plenary/Advisory Committee Meetings Week, 1 - 5 November 2010, Lyon, France. http://www.w3.org/2010/11/TPAC/ (10 April 2011)

Bentkowska-Kafel, A. (2007) Needs of the 3D Visualisation Community. JISC 3D Visualisation in the Arts Network (3DVisA), http://3dvisa.cch.kcl.ac.uk/needs/3DVisA%20Report_Needs.doc (4 April 2010).

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

Coyne, B. and Sproat, R. (2001) WordsEye: An Automatic Text-to-Scene Conversion System. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01),* ACM, New York, USA. 487-496.

CryEngine3. (2011) CryEngine Overview. www.crytech.com/cryengine/cryengine3overview (2 January 2011).

Donikian, S. and Hégron, G. (1993) Constraint Management in a Declarative Design Method for 3D Scene Sketch Modeling. Computer Graphics Forum 12, 3, pp. 223–236.

Dupuy, S., Egges, A., Legendre, V. and Nuges,P. (2001), Generating a 3D Simulation of a Car Accident from a Written description in Natural Language. In *Proceedings of the workshop on Temporal and spatial information processing*, 1-8, Association for Computational Linguistics.

Goskar, T. (2007) The Stonehenge Landscape in 3D. Wessex Archaelology. www.wessexarch.co.uk/blogs/computing/2007/11/15/stonehenge-landscape-3d (2 January 2011).

Hagège, C. and Roux, C (2002) A Robust and Flexible Platform for Dependency Extraction. In *Proceedings of LREC02*, 520-523.

Hobona, G., James, P. and Fairbarn, D. (2006) Web-Based Visualization of 3D Geospatial Data Using Java3D. In *IEEE Computer Graphics and Applications*, 26, 4, 28-33.

Johansson, R., Williams, D., Berglund, A. and Nugues. P. (2004) Carsim: A System to Visualize Written Road Accident Reports as Animated 3D Scenes. In *ACL2004: Second Workshop on Text Meaning and Interpretation*, pages 57–64, Barcelona, July 25-26.

O'Kane, M., Carthy, J. and Bertolotto, M. (2004) Text-to-Scene Conversion for Accident Visualization. In *ACM SIGGRAPH 2004 Posters* (Los Angeles, California, August 08-12, 2004) at SIGGRAPH '04. ACM, New York.

Presland, S., Farrimond, B., Hazlewood, P. and Oddie, A. (2010) Creating Complex Interactive 3D Visualisations from Natural Language Narratives. In *Developments in E-systems Engineering (DESE),* September (2010), pp. 113-118.

Slusallek, P., Behr, J. and Sons, K. (2010) Why We Need Declarative 3D. TPAC 2010 Technical Plenary/Advisory Committee Meetings Week, 1 - 5 November 2010, Lyon, France. http://www.w3.org/2010/11/TPAC/ (10 April 2011).

Smelika, R.M., Tutenelb, T., de Krakera, K.J. and Bidarrab, R. (2011) A declarative approach to procedural modelling of virtual worlds. Computers & Graphics, 35, 2, pp 352-363.

Web3D Consortium. (2011) X3D Specifications and VRML97 International Standards. http://web3d.org/x3d/specifications/#x3d-spec (10 April 2011).