# A 3D Visual Analysis Tool in Support of the SANDF's Growing Ground Based Air Defence Simulation Capability

Bernardt Duvenhage*
South African Council for
Scientific and Industrial Research

JP Delport†
South African Council for
Scientific and Industrial Research

Anita Louis‡
South African Council for
Scientific and Industrial Research

## Abstract

A 3D visual analysis tool has been developed to add value to the SANDF's growing Ground Based Air Defence (GBAD) System of Systems simulation capability. A time based XML interface between the simulation and analysis tool, via a TCP connection or a log file, allows individual simulation objects to be wholly updated or partially modified. Live pause and review of the simulation action is supported by employing data key frames and compressed XML for enhanced performance. An innovative configurable filter tree allows visual clutter to be reduced as required and an open source scene graph (OpenSceneGraph) manages the 3D scene representation and rendering.

A visualisation capability is developed for the effective presentation of the dynamic air defence system behaviour, system state transitions and inter-system communication. The visual analysis tool has successfully been applied in support of system performance experiments, tactical doctrine development and simulation support during training and live field exercises. The 3D visualisation resulted in improved situational awareness during experiment analysis, in increased involvement of the SANDF in experiment analysis and in improved credibility of analysis results presented during live or after action visual feedback sessions.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; J.7 [Computers in Other Systems]—Command and Control J.7 [Computers in Other Systems]—Military

**Keywords:** Command and Control Visualisation, Battlefield Visualisation, Analysis

## 1 Introduction

This paper describes the development of a 3D visualisation capability driven by unique requirements and presents the application of graphics technology within the African defence domain. The introduction provides a brief overview of Ground Based Air Defence (GBAD) and the system of systems level simulation capability to be visualised. The motivations behind doing 3D instead of a 2D visualisation (for example a plan view) will also be presented.

The SANDF use their need for decision support during the different phases of a GBAD System (GBADS) acquisition program to grow an indigenous system of systems modelling and simulation support capability within the acquisition environment[Pretorius 2003][Baird and Nel 2005]. The broad requirement of the capability is to simulate a GBAD battery of existing and still to be acquired (possibly still under development) ground based gun, missile and sensor systems, together with their related human operators, at a system of systems level. A GBADS deployment, shown in Figure 1, usually consists of a layered air defence. The outer layer typically consists of eight very short range missile systems (VSHORADS), each having a virtual operator and an accompanying buddy with a pair of wide angle binoculars. The second layer has four gun systems (CIWS), each consisting of two guns, a tracking radar, a designation radar, a fire control system and at least three operators to operate the guns. The inner layer of defence usually has two short range missile systems (SHORADS), each consisting of a ground based launcher, a designation sensor, a fire control system and a few virtual operators. The deployment would defend some vulnerable point (VP) against an airborne threat scenario. A typical threat scenario consists of one to many incoming attack aircraft which are the 'targets' to be engaged by the air defence system before they reach the Weapon Release Line (WRL).

The operators follow standard operating procedures based on the principles of warfare that capture the doctrinal procedures and instructions on troop and system deployment, battle handling and roles and responsibilities. The modelling of this *Tactical Doctrine*, sometimes referred to as the Air Defence Control (ADC) Model, must include such aspects as operator reaction delay and the effects of the tactical communication network.

The modellers are interested in the emergent behaviour of the interacting air defence systems and human operators and possibly the current state of each system. The system specification hierarchy, formalised by Zeigler et al. [2000] and shown in Table 1, is often used to create a common understanding, among the parties involved, of the detail of the modelling of a system required. The required detail level in this case is at a state transition system specification level. At this system specification level a system is still seen as a black box in the sense that only the aggregated perceived system state is known. It therefore makes no sense to attempt to visualise such a system at the higher *structure* system specification level as the structural information is simply not available or adds no value. The visualisation may however be augmented with *realness* aspects such as high detail textured or animated 3D objects for demonstration value.

A number of techniques to visualise simulation results in a 2D fashion already exist, but from experience in using some of these it was often found that not enough information is presented to the user in order to form a clear idea of what is happening in the simulation. Uncertainty seems to exist because it is often difficult for a simulation analyst or a client to construct a unique 3D mental model from multiple 2D

*e-mail:bduvenhage@csir.co.za
†e-mail:jpdelport@csir.co.za
‡e-mail:alouis@csir.co.za

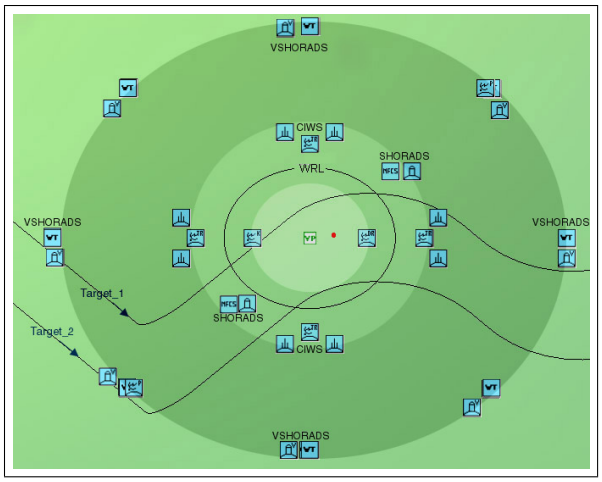


Figure 1: A Typical GBADS Deployment

projections and without this 3D interpretation of the scene it becomes difficult to visualise the geometry of situations such as:

- a radar designation and lock-on process or why radar lock-on is lost,
- whether a target is within a sensor's detection volume (envelope) or a weapon's launch envelope and
- locations of objects within an operator, sensor or weapon field of view.

3D visualisation is particularly helpful for simulation analysts and clients that are new to the game, so to speak, and not yet experienced in combining multiple 2D pictures and other information into a higher dimensional mental picture.

The remainder of the paper is structured as follows. The next section discusses the visualisation requirements within the applicable visualisation domains: system performance analysis, tactical doctrine development and then finally training and live field exercises. The following sections then present the different aspects of the analysis tool's design and the resulting GBADS visualisation software to satisfy the requirements. The paper concludes with remarks on the added value of the visualisation capability and possible future work.

## 2 Visualisation Requirements

The earlier phases of the acquisition life cycle and simulation support capability established certain modelling and simulation requirements. During the concept and definition phases of the acquisition life cycle [Naidoo and Nel 2006] the GBAD simulation was applied for as fast as possible system performance analysis. Entering the development phase of the acquisition life cycle however, the development of tactical doctrine became a priority which required human operators to replace selected virtual ones. Human operators are included in the simulation through mock-up equipment consoles which require real-time simulation execution for realistic human-to-simulation interaction. The visual analysis tool therefore has to support both as fast as possible and real-time simulation visualisation. It is also required to visualise both live and recorded data for on-line and off-line simulation analysis respectively. The general aspects to be visualised are:

Table 1: System Specification Hierarchy

| Level | Specification Name | What is known at this level |
|---|---|---|
| 4 | Coupled component | Components and how they are coupled together. The components can be specified at lower levels or can even be structure systems themselves - leading to hierarchical structure. |
| 3 | State transition | How states are affected by inputs; given a state and an input what is the state after the input stimulus is over; what output event is generated by a state. |
| 2 | I/O function | Knowledge of initial state; given an initial state, every input stimulus produces a unique output. |
| 1 | I/O behaviour | Time-indexed data collected from a source system; consists of input/output pairs. |
| 0 | Observation frame | How to stimulate the system with inputs; what variables to measure and how to observe them over a time base. |

- the system and target physical positions and/or motion, their orientations and other states,
- virtual spatial envelopes of the weapon and sensor systems,
- tactical messages and operator states,
- spatial and temporal relationships between the above and
- the virtual environment.

As special cases of system performance analysis and tactical doctrine development respectively, the simulation and visualisation capability is also applied to training exercise and live field exercise integration and support. The following subsections introduce the three visualisation domains and the visualisation requirements unique to each.

### 2.1 System Performance Analysis

The majority of experiments analyse the air defence timeline with outcome measures such as number of threats successfully engaged or percentage weapon misses. The objective of the analysis being to come up with proposals on how to optimise the system deployment within existing/simplified tactical doctrine. The focus is therefore on visualising the application of tactical doctrine within a specific instance or multiple instances of an air defence system deployment and threat, and finally visualising the resulting outcomes.

### 2.2 Tactical Doctrine Development

Tactical doctrine development requires the visualisation of Air Defence Control events/messages and the states of the air defence systems in order to capture the essence of the doctrine as applied in the simulation. Tactical doctrine development also requires realistic human-simulation interaction which happens through mock-up Operator In the Loop (OIL) interfaces that are being developed as part of the simulation capability. The objective of this being the optimisation of tactical doctrine within a specific instance or

multiple instances of an air defence system deployment and threat. Aspects to be visualised is much the same as in system performance analysis, but usually focus on the threats and equipment immediately related to the operator in front of the OIL console.

### 2.3 Training and Live Field Exercise Support

As mentioned, training and live field exercises are special cases of system performance analysis and tactical doctrine development. Simulation and visualisation have been used to aid in training exercises such as deployment planning. During such a deployment planning exercise, students are presented with a potential threat to defend an asset against and each student must then plan their air defence deployment accordingly. Simulation is then used to evaluate the effectiveness of each student's deployment. The effectiveness and reasons for failure may then be visualised/analysed using the 3D analysis tool during student feedback sessions.

During live field exercises, real equipment is often replaced with virtual versions to save cost (augmented/mixed reality). The virtual versions still have a role to play in the exercise though, and their virtual presence must be visualised in a suitable fashion. The threat may even be virtual, making 3D visualisation of particular importance for partaking operators.

## 3 Analysis Tool Design

The analysis tool has followed a design process of evolutionary increments which was driven by the simulation interface requirements and the characteristics of the data to be visualised. The design of the simulation interface, how the simulation data is managed and the aspects surrounding the 3D scene management, navigation and interaction are discussed.

### 3.1 Simulation Interface

The simulation interface (also referred to as a protocol) to the Analysis Tool is XML based. Two versions of the protocol exist and both require initialisation information and static object data to be transferred followed by time stamped simulation *updates*. Each update contains uniquely identifiable state descriptions of all simulation objects modified within that simulation update frame. Only at this level can the two versions of the protocol be differentiated.

Version 1.0 provides the complete state of each simulation object while version 2.0 needs only to convey a partial object state consisting of the object's unique ID and its modified attributes. The generation of the version 2.0 protocol is however more complex as it attempts to view XML elements as sets of sets and require the definition of XML unions and differences. A partial update in protocol version 2.0 is a three step process:

- First a *union* is built up from all the simulation frames up to just before the current frame which is $update_t$,
- then the *partial update* is computed by the XML set difference $update_t - union$, and
- the updated *union*, up to time $t$, may now be computed by, $union+ = update_t$.

Version 2.0 requires a significantly reduced interface bandwidth (measured to be in the order of approximately 10 times smaller) and produces less XML text to parse and internalise by the viewer compared to the less efficient version 1.0.

### 3.2 Managing the Simulation Data

The simulation data received over the simulation interface needs to be managed in a time and space efficient manner for both quick data retrieval and judicious use of storage resources. The components for managing the simulation data are the GUI, live pause and review functionality, key frames within the protocol and the filter tree.

#### 3.2.1 The Application GUIs

The GUI has three separate floating parts, namely the *Main* window, the filter tree GUI and the window containing the 3D view.

The main GUI provides access to file and TCP connection management. It additionally has a *Review* and a *Play/Pause* button along with a time slider that may be used to select a simulation time to jump to or review from (additional info on the review functionality is provided in the next subsection).

The *Filter Tree GUI* and the *3D View Window* is discussed in subsection 3.2.4 and subsection 3.4 respectively.

#### 3.2.2 Live Pause and Review of Simulation Data

The time indexed simulation data is cached to system memory as it is read from TCP or when the log file is initially opened. This is done in support of an online review capability, referred to as *Live Pause*. It allows the user to jump to any simulation time and to optionally then start playing from that location. This may happen independently of the live data coming in on the TCP connection.

#### 3.2.3 Adding Key Frames to the Protocol

As mentioned, the simulation protocol conveys updates to the 3D analysis tool at an object level and additionally at an object attribute level when Version 2.0 of the protocol is used. Due to the possibility of partial updates the simulation state updates must all be internalised by the analysis tool and applied to keep the current simulation state up to date. If the user jumps forward in time, assuming that there is data up to that time in the cache, then the new simulation state is built by quickly processing all updates between the current time and the new time.

If, however, the user wants to rewind the process, unapplying updates from the current time to the earlier time is not straight forward and the alternative is to rebuild the simulation state from time 0. This is unfortunately extremely inefficient when the simulation has been running for some time. To alleviate this inefficiency the *union* element used to generate protocol version 2.0 is added as a *Key Frame* approximately every 100th simulation frame. Whenever the simulation state has to be rebuilt, the analysis tool need only rebuild the state from a key frame earlier than the required time. The addition of key frames has a measured bandwidth overhead of approximately 10% above that of version 2.0 without keyframes, but it improves the performance to allow interactive random access even in very large log files.

#### 3.2.4 The Filter Tree

The visual representation of a system of systems simulation with many objects, events and visual aids can become very cluttered and important aspects of a particular experiment or exercise might not be visible as shown in Figure 2.

The use of filters enables the user to highlight certain objects and events. A user can also quickly switch between various filters according to the visualisation need. Because the filters are applied within the viewer, they do not affect the recording of the simulation data and allow a user to view a recorded simulation multiple times and analyse different aspects by configuring and using different filters.

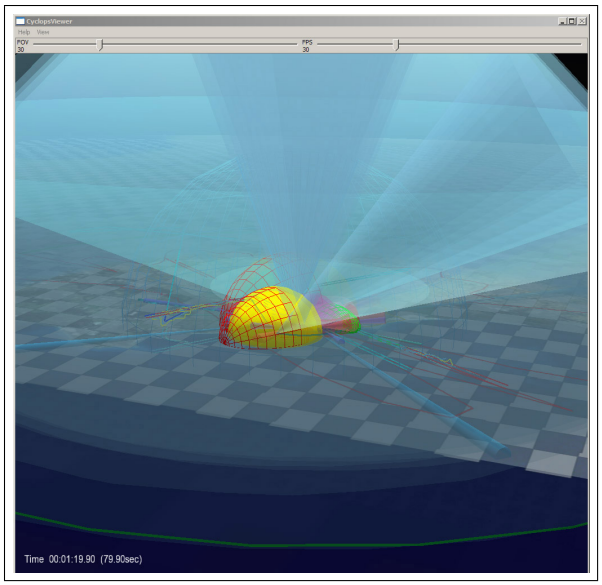

Figure 2: Cluttered View of GBADS Deployment

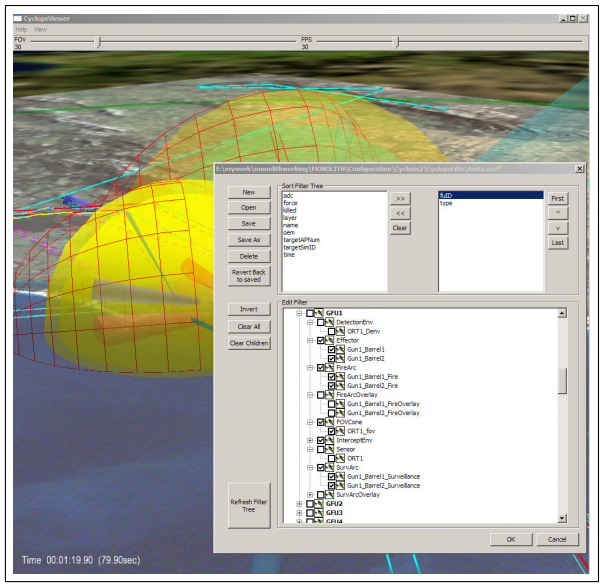

Figure 3: Cluttered View of GBADS Deployment

A dynamic tree structure was chosen for management of the filters as it enables hierarchical grouping of objects by their attributes. The tree structure is also advantageous as opposed to for example a linear structure, as it organises the elements in a flexible manner and allows the user to easily navigate the tree and locate particular elements and groups of interest.

Figure 3 shows the list of available attributes in the list to the left and the attributes chosen for categorisation in the *hierarchy list* to the right. Attributes are moved across from right to left and vice versa using the ⟨⟨ and ⟩⟩ buttons respectively. The order of the attributes in the *hierarchy list* can be changed using the *first*, *up*, *down* and *last* buttons on the right. The resulting tree created from the attributes *fuID* and *type* shows each of the fire units at the topmost level, with IDs: GFU1, GFU2, GFU3 and GFU4. GFU1 has been expanded to show the elements that belong to GFU1 as categorised by their types. Visibility of each object, at any level in the tree, can be switched on or off by changing the status of the tick-box associated with it or its parents. Standard functionality to Create new, Open, Save and rename filters is provided.

## 3.3 3D Scene Representation Library

The purpose of the 3D scene representation library is to create a 3D view into the state of the simulation world. The 3D scene is updated as simulation data, possibly filtered, are received.

The 3D visualisation was decoupled from the rest of the GUI logic and placed into a separate library by defining a clear interface between the two parts. The split allows the Analysis Tool to focus on the domain specific simulation state, while the 3D library only concerns itself with 3D representation and drawing of generic objects. The Analysis Tool communicates with the 3D library using standard C library calls.

The library allows for limited user interaction with the 3D scene. The user can move around the 3D world using a variety of camera manipulators, toggle the visibility of larger groups of objects (for example all text objects), set some view parameters (for example the camera field of view and the target frame rate) and perform simple distance and angle measurements.

The following sections discuss some of the more interesting design aspects of the the 3D library.

### 3.3.1 External Dependencies

As a first step, a choice had to be made between the two most popular 3D graphics APIs, namely DirectX and OpenGL. This choice also influenced further decisions regarding other supporting libraries. A clear need for cross-platform[1] availability of the 3D library resulted in the decision to use the OpenGL API.

A GUI toolkit library was needed for the user interface of the library. The wxWidgets library was found to be suitable as it supported the cross-platform creation of a variety of UI elements as well as an OpenGL rendering context.

Finally, the use of a higher level library that could assist with interfacing to OpenGL was investigated. The following requirements strongly influenced the decision:

- Support for OpenGL, especially newly added API extensions.
- Cross-platform availability.
- Loading of 3D models in a variety of formats.
- Easy creation of text and labels.
- Built-in support for reading and rendering large sections of terrain.
- Rendering speed, for example support for culling.
- Source code availability.
- Large user base.

The OpenSceneGraph library [www.openscenegraph.com] was found to satisfy the stated requirements and was subsequently chosen. The library was also found to provide

[1] Mainly Windows and Linux operating systems.

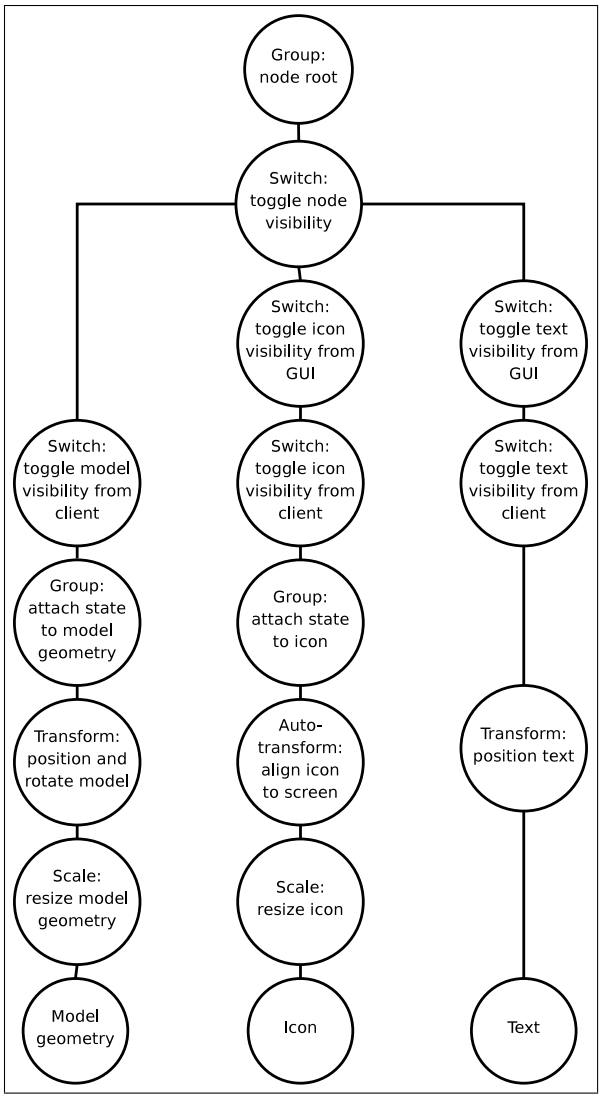


Figure 4: Scene Graph Layout of a Single Object.

functions not envisioned at the start of the project, with support for icons, overlays on terrain and coordinate transformations the most notable.

### 3.3.2 Coordinate System Considerations

GBADS simulations are performed in a spherical earth environment. Object positions in the simulation are described by latitude, longitude and altitude parameters, whereas orientations are described using heading, pitch and roll values. In order to accurately render the positions and orientations of objects, these spherical coordinates had to be converted to a local Cartesian coordinate system that could be used by the OpenGL API.

A local coordinate system convention was defined as follows: For a specific reference position on the earth, let the X-axis point in a northerly direction, the Y-Axis east and the Z-axis into the earth[2]. For objects, let the X-axis point forward of the object, the Y-axis to the right and the Z-axis down. Using the convention, an object positioned and oriented using all zero values would end up at the reference position on the earth looking in a northerly direction.

[2]This is sometimes referred to as a north-east-down or NED coordinate system.

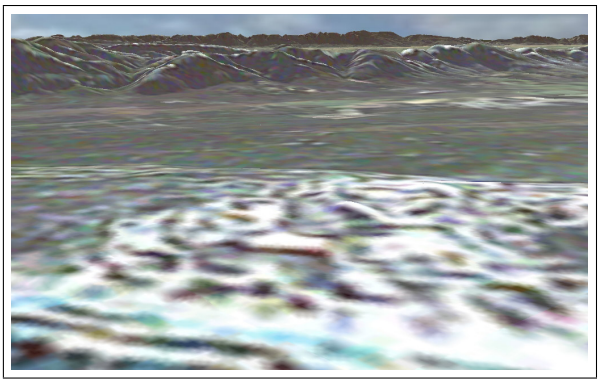

Figure 5: Virtual Environment with Satellite Terrain Map

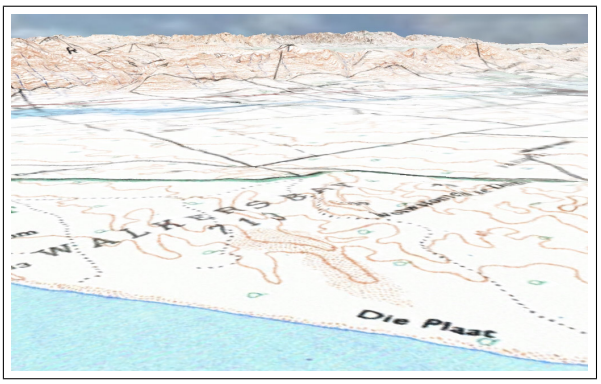


Figure 6: Virtual Environment with 1:50000 Contour Terrain Map

The origin of the local coordinate system is communicated to the 3D library at the creation of a 3D scenario. The user of the library can therefore pick an appropriate reference point that would be used by the 3D library. The choice of reference point has an impact on the accuracy in terms of object placement that can be achieved by the 3D library. Most OpenGL implementations, for instance, used 32-bit floating point values to represent world coordinates. Accuracy can therefore be improved if the reference position is picked as close as possible to the objects in the scenario.

Conversions from the spherical earth coordinates to the local coordinates are performed by partly using built-in OpenSceneGraph functions (for position) and also using additional custom rotations for object orientation. The 3D library also support picking of objects in the scene and in this case the local picked coordinates are converted back to spherical earth coordinates for display to the user.

### 3.3.3 Concurrency

As mentioned in the overview of Section 3.3 a decision was made to split the Visualisation and the 3D scene representation library. A consequence of the split is that the protocol and visualisation logic, and 3D library executes in separate threads. The 3D library contains its own message loop to allow the user to interact with the scene even when no updates are received from the visualisation side.

We therefore have a situation where one thread is responsible for updating data representing the scene and another thread uses a snapshot of this data to create a 3D repres-

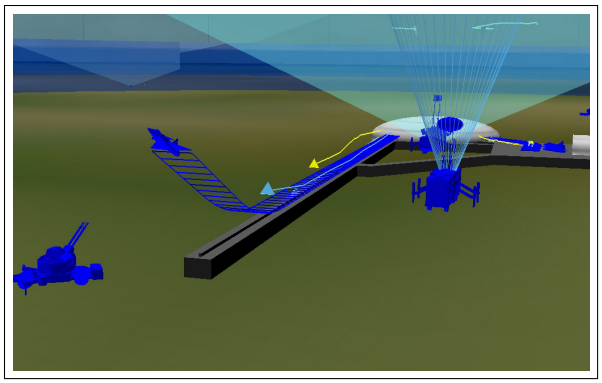

Figure 7: View of Some Simulation Objects Including Radar Tracks

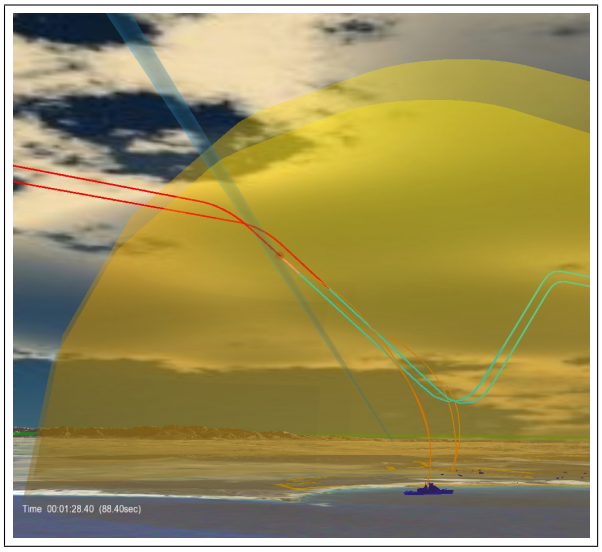


Figure 8: View of Some Simulation Objects Including Radar Domes and Munition Trails

entation. Some sort of synchronisation mechanism had to be found to ensure that the 3D scene presents a consistent view of the simulation state[3].

Two options were considered for dealing with the synchronisation problem, namely double buffering and mutual exclusion (locking).

The idea with double buffering is much the same as that of double buffer rendering to a front and back frame buffers. The 3D library caches all data sent to it and on a single command selects all the cached data to be the new snapshot used for the 3D scene representation. A problem, that increases complexity, with the double buffering approach is that code has to be added to the 3D library for every view related library call[4] or 3D object in order to allow for buffering and swapping. Memory consumption is another drawback as the buffering causes complex objects to be stored using double the amount of memory.

[3]Inconsistency and confusion can arise for example when a 3D scene is constructed using one set of objects at a specific simulation time instance and another set of objects at another simulation time instance.

[4]It can be a call to change geometry of even a call to change state, for example switch from filled to wire frame mode.

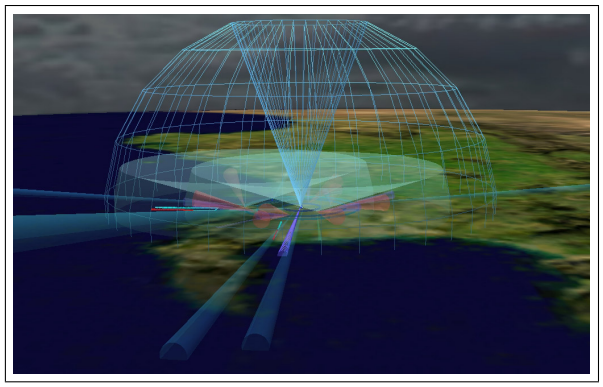

Figure 9: Visualisation of Virtual Cones and Domes on the Battlefield

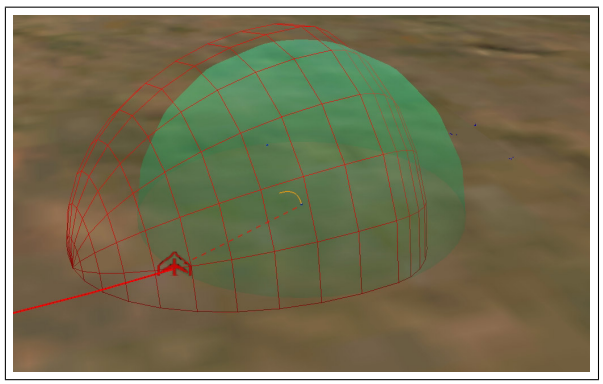

Figure 10: Visualisation of Virtual Intercept- and Launch Envelopes

The second approach, the one being used for new development, is that of mutual exclusion or locking. The 3D library acquires a lock as soon as it starts to update the 3D representation and the visualisation logic is not allowed to send updates during this time. The same lock needs to be acquired by the visualisation logic when it starts sending updates[5] and then the 3D library is not allowed to redraw the 3D representation. The locking approach presents a simple mental model and is easier to program than the buffering approach, but it places more responsibility on the visualisation logic. The visualisation logic has to minimise time spent between acquiring and releasing the lock as the 3D representation is not interactive during this time.

### 3.3.4 3D Object Hierarchy

A single simulation object can be represented by the 3D library using either a 3D model, an icon, text, or combinations of these. The selection of what to view is made by the visualisation logic, but effectively the filter which the user may interact with.

Properties (state) of the 3D model can be changed (for example colour or wire frame mode) and the 3D model can be positioned, rotated and scaled. The icon representing the object can be scaled[6], but it is rotated so that it is aligned

[5]The lock is typically acquired before sending all updates for a specific simulation time instance and released after all updates are completed.

[6]The scale represents pixel dimensions and after the scale is set it is kept constant in terms of pixels no matter what the distance

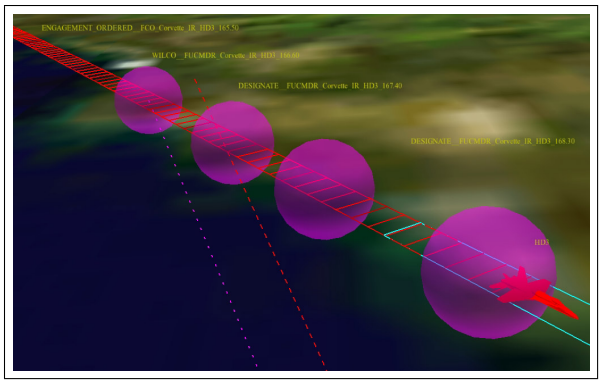

Figure 11: Visualisation of ADC Events and Messages

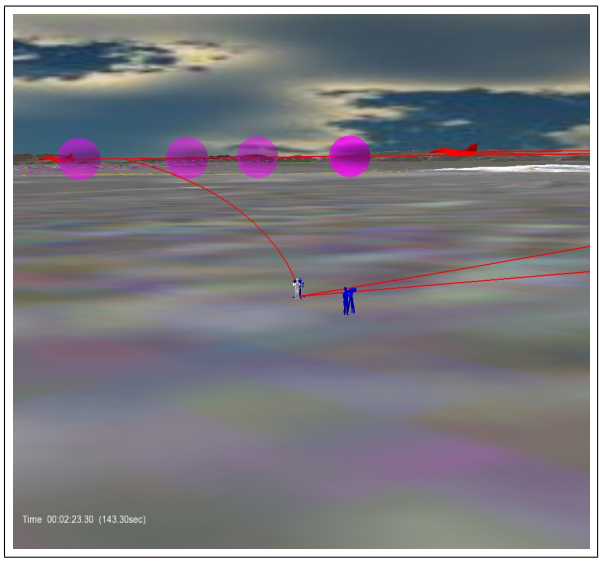

Figure 12: Visualisation of ADC Events and Messages During VSHORADS Launch

towards to view position. The text attached to the object can be positioned relative to the object and is, like the icon, aligned to the screen and the character size is kept constant in terms of screen coordinates.

OpenSceneGraph allows for the creation of a hierarchical scene graph to represent these objects in code. Figure 4 shows the hierarchy of the 3D objects inside the 3D library. The scene graph nodes allow for easily setting of the 3D object properties mentioned earlier.

### 3.4 Scene Navigation and Interaction

The window that contains the 3D view is also the window that is used for navigating the scene as mentioned. The user accesses the list of available camera modes via the 'View' drop-down menu. Whenever the window is active the user may use mouse commands to manipulate the active camera. A slider modifies the field of view of the camera.

The first camera to be discussed is the *Orbit (Trackball)* camera. When this camera mode is chosen the user is prompted to select an object that will be the centre of focus. The camera is then positioned a distance away from the object such that the object fills the view at the current field of view.

between the view position and the object.

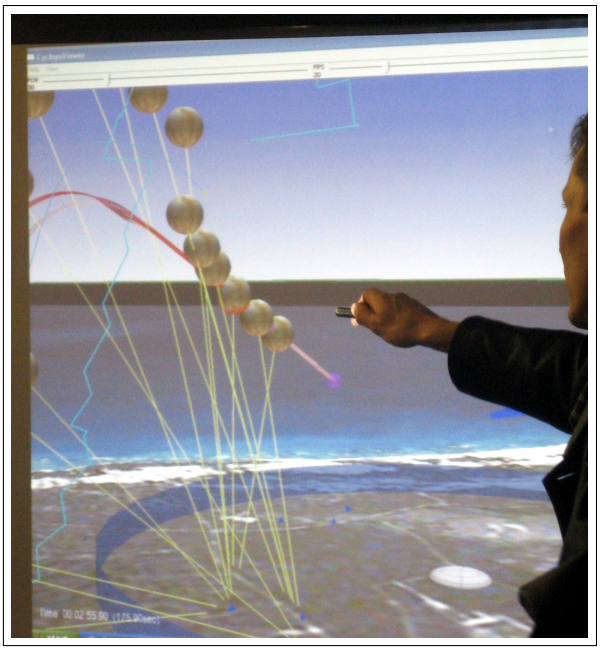

Figure 13: Visualisation During a Presentation Session

Mouse actions allow the user to then zoom, pan and tilt. A slight variation on this mode is the *Orbit (Pitch, Yaw) mode* that constrain the camera motion to the object's local co-ordinate axis.

Another useful camera mode is the *Attach to* camera mode. In this mode the camera's view is aligned with the object in focus' orientation. This allows the user to in effect *see* what a sensor is seeing.

The user may also *pick* a point in the 3D scene by holding down shift while pointing the mouse at something in the scene or a point on the terrain. While holding down shift the distance and angles between the picked coordinate and the coordinate that is the camera's focus is measured and displayed on screen. Additionally if the camera mode is in *Normal Trackball* mode, then clicking while holding down shift changes the focus to the picked coordinate allowing general angle and distance measurements.

## 4 Results from Visualising GBADS

It has been shown what the visualisation requirements are and how the analysis tool has been equipped for managing the simulation data. What remains is to do the actual visualisation. The visualisations were implemented in an incremental fashion and evolved around useful representations while changing or removing aspects of the visualisation that were not useful. The rest of this section discusses the visualisations along with screen shots where applicable.

The virtual environments, shown in Figure 5 and Figure 6, are realised by a 3D terrain and a sky dome. The user is able to choose the terrain based on the location of the deployment and the type of map required. It is important to note that the actual simulation only uses the height profile of the terrain and the visual aspects of the terrain and sky-dome are to primarily aid in the analysis, for example by addition of the geographical detail by contour or satellite maps. Secondly, the terrain and sky aid in the photo-realism of the visualisation if required for demonstration purposes.

The simulation objects are realised by loading in pre-built

Figure 14: Using the 3D Analysis Tool During Exercise Feedback

3D models and transforming their sizes, positions and orientations to match what is current in the simulation. Some special shapes such as radar tracks, small tetrahedral shapes in Figure 7, flight paths and trails may be added to the visualisation of objects (also shown in Figure 7). An object has a trail flag that, when enabled, causes a line or ribbon, depending on the object's type, to be drawn behind the object. A flightpath, on the other hand, is a static list of timestamped positions that is usually added to the scene at simulation start-up and is erased as the simulation time progresses. The trails and flight paths were added to aid the user in locating objects and at a glance see their past and future position, as well as orientation, which is particularly useful during analysis.

Object types such as sensors and weapons have *Detection Envelope Size*, *Intercept Envelope Size*, and *Field Of View (FOV)* attributes that in reality represent virtual domes and cones, but may in simulation be visualised to aid analysis as shown in Figure 9. These are useful for visualising when a target is within a sensor's detection range or within range of a weapon system. There is a time delay between when a launch command is given to a weapon system and when the weapon intercepts the target due to weapon flight time and system delays in processing the launch command. This delay is critical to the simulation time line and spatially warps the intercept envelope towards an approaching target into the egg shaped envelope shown in Figure 10. This envelope is referred to as the launch envelope. If a launch command is given for a target while outside the launch envelope, the intercept will be unsuccessful. Visualisation of the launch envelope along with location of the target at the time of the command is used to analyse the quality of the launch command. Notice that in the centre of the spherical intercept envelope the missile has already been fired some seconds ago, because the target is within the launch envelope although it is still just outside the intercept envelope.

Figure 11 and Figure 12 show how ADC events and messages are currently visualised. Some events involve the target and the deployment as a whole, indicated by the pink bubbles, and some events involve the target and a specific weapon system, indicated by the pink bubbles with the dashed lines to the weapon system. The line visualised between operator and target, seen in Figure 12, is the VSHORADS dart trail. The spatial and temporal association of the event to the target is indicated, as shown in the figure, by drawing a static event on the target's trail at the time of the event. Doing it in this way compresses the four association dimensions to a drawable three dimensions.

Figure 13 and Figure 14 show the 3D analysis tool being used during presentation and exercise feedback sessions.

## 5 Conclusion

Previously a simulation analyst would start with an event time line or 2D spatial view and then look at incrementally more data dimensions until he or she is able to form a clear mental picture of each event of interest. It does however take time and practice to form an unambiguous mental image from different layers of visual and textual information.

According to the GBADS system simulation analysts, the 3D visualisation resulted in improved situational awareness during experiment analysis, in increased involvement of the SANDF in experiment analysis and in improved credibility of analysis results presented during live or after action visual feedback sessions. The analysis tool has also successfully been used in multiple field exercise integrations and training exercises with positive feedback from the parties involved.

## 6 Future Work

The reported value of the analysis tool is based on the feedback of simulation analysts and the GBADS subject matter experts. Future efforts should be aimed at quantifying the advantage of using the third dimension and visual analysis. This will aid in directing future developments and motivating for funding for the work.

The analysis tool is also finding application within industry (in different departments of Denel Aero Space and within the Optronic Sensor Systems and also the Radar and Electronic Warfare Competency areas of the CSIR) for a large range of conceptual system visualisations. This growth of interest now necessitates the establishment of structures such as an architecture review board and an official user group.

## Acknowledgements

This work was supported by the Council for Scientific and Industrial Research, Defence Peace Safety and Security (CSIR DPSS) and the South African Armaments Corporation (ARMSCOR).

## References

Baird, J., and Nel, C. 2005. The evolution of M&S as part of smart acquisition using the SANDF GBADS programme as an example. In *Proceedings of the 12th European Air Defence Symposium*, vol. 3694, 173–182.

Naidoo, S., and Nel, C. 2006. Modelling and simulation of a ground based air defence system and associated tactical doctrine as part of acquisition support. In *Proceedings of the 2006 Fall Simulation Interoperability Workshop*.

Pretorius, J. L. 2003. *Feasibility Considerations for A Tailored Simulation Based Acquisition (SBA) Approach*. Master's thesis, University of Pretoria.

Zeigler, B. P. 2000. *Theory of Modelling and Simulation*. Academic Press.