AR-009-634

DSTO-GD-0092

DSTO

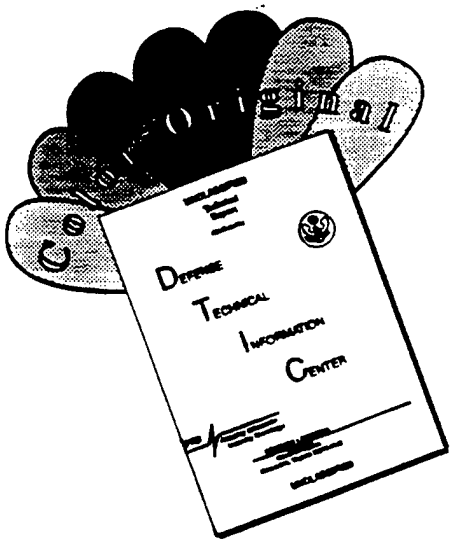
Description and Worked Example of STAGE

Sabrina Sestito
and Jodie Doman

DEPARTMENT ◆ OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

Description and Worked Example of STAGE

Sabrina Sestito and Jodie Doman

Air Operations Division
Aeronautical and Maritime Research Laboratory

DSTO-GD-0092

ABSTRACT

This paper describes the key features of the Scenario Toolkit And Generation Environment (STAGE) software. STAGE provides an environment for the development of real-time tactical situations. It uses menus and additional pulldown menus for easy entry of data and provides a powerful display capability including a dynamic link to the positioning of entities on the display. STAGE supports user written scripts, which are associated with individual platforms, to dictate the behaviour of each entity. User written code, known as user modules, allow the user to extend the capability provided by STAGE (by expanding) the scripting mechanisms, replacing existing models and providing access to STAGE's internal data structure. Use of STAGE and a course of action for developing scenarios is described in this paper, along with a worked example giving detailed listings of all data, profiles and scripts required to run a simple scenario.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Victoria 3001

Telephone: (03) 9626 8111
Fax: (03) 9626 8999

AR No. AR-009-[illegible]634
April 1996

APPROVED FOR PUBLIC RELEASE

Description and Worked Example of STAGE

Executive Summary

This paper describes the key features of the Scenario Toolkit And Generation Environment (STAGE) software. STAGE provides an environment for the development of real-time tactical situations. It uses menus and additional pulldown menus for easy entry of data and provides a powerful display capability including a dynamic link to the positioning of entities on the display. STAGE supports user written scripts, which are associated with individual platforms, to dictate the behaviour of each entity. User written code, known as user modules, allows the user to extend the capability provided by STAGE by expanding the scripting mechanisms, replacing existing models and providing access to STAGE's internal data structure. Use of STAGE and a course of action for developing scenarios is described in this paper, along with a worked example giving detailed listings of all data, profiles and scripts required to run a simple scenario.

STAGE has several applications. It would be useful for simulating real combat scenarios. In addition, it would be useful for evaluating tactics and weapons. Due to its real-time interactivity and flexibility, STAGE would be very useful for training; this is because a trainer could instantaneously and interactively change a situation and see the effect this has on the trainee(s). Finally, STAGE could also be used to control several simulators and other hardware; this is possible through its distributed interactive simulation communication protocol.

Authors

Sabrina Sestito
Air Operations Division

Dr Sestito is a Research Scientist at the Aeronautical and Maritime Research Laboratory (AMRL), Department of Defence, in Melbourne, Australia. She completed her B.Sc. (Hons) in Meteorology and Oceanography at the Flinders University of South Australia in 1983 and a Postgraduate Diploma in Computer Science at the University of Adelaide in 1984. She started working for AMRL, in Melbourne in 1985 in the Operational Research Group. In 1988 she commenced her Ph.D. in Computer Science at LaTrobe University, which she completed in 1991. Her Ph.D. was expanded and published as a book titled "Automated Knowledge Aquisition". She is currently working in the Air Operations Simulation Centre at AMRL. Her interests include the use of Computer Graphics and Virtual Environments for Simulation, and Artificial Intelligence. She is a member of the IEEE Computer Society.

Jodie Doman
Air Operations Division

Ms. Jodie Doman is an Engineer at the Aeronautical and Maritime Research Laboratory (AMRL), Department of Defence, in Melbourne, Australia. Jodie started work at AMRL, Melbourne for 11 months in 1991 as an industry based learning student. She recommenced her employment there in the Air Operations Simulation Centre at the beginning of 1995. She completed a B.App.Sc. in Computing and Scientific Instrumentation at Swinburne University in 1992 and is currently completing a Master of Applied Science at the same university. Her interests include the use of artificial neural networks for problem solving and amateur theatre.

Contents

List of Acronyms

AAA	Anti-Aircraft Artillery
dalt	altitude variation
DE	Database Editor
dhdg	heading variation
DI	Development Interface
DSTO	Defence Science Technology Organisation
dv	velocity variation
ESM	Electronic Support Measures
ETA	Estimated Time of Arrival
LST	Landing Ship Tank
MCMV	Mine Counter Measure Vessel
PK	Probability of Kill
roc	rate of climb
rod	rate of dive
rot	rate of turn
RPM	Revolutions Per Minute
SAD	Situation Awareness Display
SAM	Surface-Air-Missile
SIM	Simulation Engine
STAGE	Scenario Toolkit And Generation Environment
vert spd	vertical speed

1. Introduction

This paper describes the key features of the Scenario Toolkit and Generation Environment modeling tool known as STAGE. This tool provides an environment for the development of real-time tactical situations. It uses menus with additional pulldown menus for easy entry of data. Platforms and weapon vectors are the entities which inhabit the synthetic environment. STAGE provides a powerful display capability, known as the Situation Awareness Display (SAD), to show the evolution of the scenario both during design and runtime. A dynamic link for the positioning of entities on the display is also provided. Scripts, associated with individual platforms, are user written and dictate the behaviour of each entity. User written code, known as user modules, allow the user to extend the capability provided by STAGE by allowing the user to expand the scripting mechanism, replace existing models and by providing access to STAGE's internal data structure. The flexibility of this tool is thus apparent.

STAGE has several applications. It would be useful for simulating real combat scenarios. In addition, it would be useful for evaluating tactics and weapons. Due to its real-time interactivity and flexibility, STAGE would be very useful for training; this is because a trainer could instantaneously and interactively change a situation and see the effect this has on the trainee(s). Finally, STAGE could also be used to control several simulators and other hardware; this is possible through its distributed interactive simulation communication protocol.

This paper is broken up into the following sections.

- Section 1 contains this brief introduction.
- An overview of STAGE is given in Section 2.
- Section 3 describes the main components comprising STAGE.
- The basis of STAGE's scenario organisation is described in Section 4. Within this, detailed descriptions of the platform and weapon vectors which inhabit the environment are presented. Various necessary profiles, such as sensor and weapon profiles, are also described.
- User written scripts and user modules are described in sections 5 and 6, respectively. These mechanisms give STAGE a powerful means for a user to interact and control every aspect of a scenario.
- Using STAGE is then discussed in section 7. The main options available to STAGE through the initial window and a plan of action when developing a new scenario are discussed.
- The models provided by STAGE are briefly described in Section 8.
- Section 9 presents a worked example. Instances of profiles and scripts are provided.
- Section 10 provides a summary of the main features of STAGE. The potential uses of STAGE are then also briefly discussed.

2. Overview

The Scenario Toolkit and Generation Environment (STAGE), which was developed by Virtual Prototype Incorporated (VPI), is a software package supporting the definition and real-time execution of synthetic tactical environments. STAGE provides the user with a graphical user interface to enter information into a tactical database. It does this by providing a set of menus and windows to facilitate the creation of the tactical database. The tactical environment is composed of both entities and environmental influences. These components are all defined by various parameters. The tactical database is used to generate a dynamic, interactive, complex and real-time tactical environment. STAGE also provides a set of menus and windows to control and view the data generated during the runtime simulation.

STAGE simulates the synthetic environment by using moving entities (such as planes and missiles) which interact by tactical means (such as detection and engagement). The moving entities, referred to as platforms, along with weapon vectors, inhabit the synthetic environment. The type of platform (such as a F15) are defined by a platform profile consisting of the type of platform, the physical, dynamic and acoustic profile of the platform, the attached sensors and weapons, and its combat effectiveness. Some of the types of platforms supported are fixed-wing aircraft, helicopters and submarines. Weapon vectors are missiles and torpedos which inhabit the environment. A script is attached to individual platforms and weapon vectors, it consists of a procedural description of the actions to be carried out based on the tactical environment status. Thus, it defines the behaviour of the various entities and actually extends the capability of the platform beyond the input data entered. User modules provide the user with the capability to integrate user-written simulation models into the simulation environment, allowing the user to extend the capability of STAGE. Therefore, through the use of scripts and user modules, the user is able to control many aspects of the synthetic environment.

STAGE is a fully interactive, real-time tactical simulator. A useful aspect of this interaction is available during program execution, where the scenario can be stopped at any time and changes made. It can either be run with detailed models or another model can be run and linked into STAGE. These and other features allow STAGE to operate as a totally stand-alone environment generator or as a fully integrated simulator for other applications.

3. Components

In this section, the main components of STAGE are described. Included in this discussion, is a description of the Situation Awareness Display (SAD) which provides a crucial link between the scenario and the user.

There are three interacting components in STAGE. These are :

- database editor (DE);
- simulation engine (SIM); and
- development interface (DI).

The database editor (DE) features a structured user interface (with windows, icons, menus and pointers) to manipulate entities and create scenarios. This database editor is used to build and assemble the components of a scenario during the design phase. Menus and pulldown displays provide an easy method for entering the components, which inhabit the environment, into a tactical database. The simulation engine (SIM) uses the tactical database to produce and simulate the synthetic environment. The scenario is thus prepared in game preparation mode and downloaded to the simulation engine for running. Game control and monitoring takes place using the runtime model of the database editor; here, the scenario can be run, stopped and returned to the initial conditions, frozen mid-scenario or resumed.

During runtime, the DE is used to visualise the evolution of the scenario's entities within the synthetic environment. The simulation engine communicates with the database editor in order to:

- get descriptions of the scenario and its entities;
- return data about the current state or the evolution of the scenario; and
- respond to the simulation flow control commands.

The SIM makes available, via a shared area, key descriptive and runtime state information. This enables STAGE to operate as the central simulation in a set of co-operating simulation processes.

The focal point of the Database Editor is the Situation Awareness Display. SAD provides a god's eye view of the scenario and consists of a map of the gaming area overlayed with symbology representing the scenario's entities and their trajectories. A powerful feature of this display is that entities can be moved graphically. Thus, during the execution of a scenario the location of any entity can be modified and its affect gauged. For instance, any platform or weapon vector can be hooked and displayed in a separate window and this hooked entity deleted, repositioned and commanded to change speed, altitude and heading in real-time. It is also possible to dynamically instantiate an entity in the scenario. This highlights the flexibility of this display. Zooming and decluttering functions of SAD are also available to allow the user to focus on particular areas of interest. Group editing capabilities are also provided by STAGE to make multiple copies of individual entities, and to cut, copy and paste groups of entities.

The Development Interface (DI) allows one to extend the capabilities of STAGE beyond those that are already present in the SIM, DE and tactical database. For example, the user can :

- add functionality to the SIM;
- replace simulation modules;
- extend the functionality of the scripting mechanism via user written code; and
- add user-written communication modules to the SIM to link it with local or remote simulation processes.

The DI can also add new fields to the existing profiles, add new profiles and add new relationships between profiles.

4. Scenario Organisation

The scenario in STAGE is composed of entities and environmental components, each of which are parameterized. Simulation models are attached to the entities and components generate the behaviour. The entities and environmental components are assembled hierarchically to form a complete scenario. A scenario is the aggregate unit which is simulated to generate a synthetic environment. This modular approach supports the independent, incremental and re-useable creation of the scenario's building blocks. The following class organisation characterises the STAGE generated synthetic environment. Figure 1, reproduced from the STAGE User Manual, lists most of the important components. Figure 2 arranges these components hierarchically.

Default settings will be used for absent gaming area, and atmospheric and oceanographic conditions. The atmospheric and oceanographic profiles are affected at two levels. Firstly, the global level allows an overall atmospheric attenuation factor to be specified for a given scenario. This attenuation factor acts upon all simulated sensors by linearly degrading their detection capability. Secondly, or at a local level, the user is allowed to specify an attenuation factor for each type of active sensor.

4.1 Global Level

The geographical location and extent of the scenario is known as the gaming area. The atmospheric and oceanographic profiles specify the characteristics of the environmental volume above and below the sea level, respectively. The effects of these profiles is specified at a global level. An overall attenuation factor can be specified for a given scenario - this factor acts upon all simulated sensors by linearly degrading their detection capability.

A SCENARIO consists of:
- a GAMING AREA
- an associated ATMOSPHERIC and OCEANOGRAPHIC PROFILE
- a table of ENTITIES

an ENTITY can be either:
- a PLATFORM INSTANCE or
- a VECTOR WEAPON INSTANCE

a PLATFORM INSTANCE consists of:
- a name
- an associated PLATFORM PROFILE
- INITIAL CONDITIONS
- an associated BEHAVIOUR SCRIPT
- FORMATION data
- a BASIC TRAJECTORY for navigation

a WEAPON VECTOR INSTANCE consists of:
- a name
- an associated WEAPON PROFILE
- targeting information
- an associated BEHAVIOUR SCRIPT

a PLATFORM PROFILE consists of:
- a name
- a PHYSICAL PROFILE
- a DYNAMIC PROFILE
- an ACOUSTIC PROFILE
- a table of ATTACHED SENSORS
- a table of ATTACHED WEAPONS
- COUNTER-MEASURES against VECTOR WEAPONS
- DAMAGE IMPACT results

an ATTACHED SENSOR consists of:
- a name
- an associated SENSOR PROFILE

an ATTACHED WEAPON consists of:
- a name
- an associated WEAPON PROFILE
- a count of the number of weapons available.

Figure 1. A list of the components in STAGE.

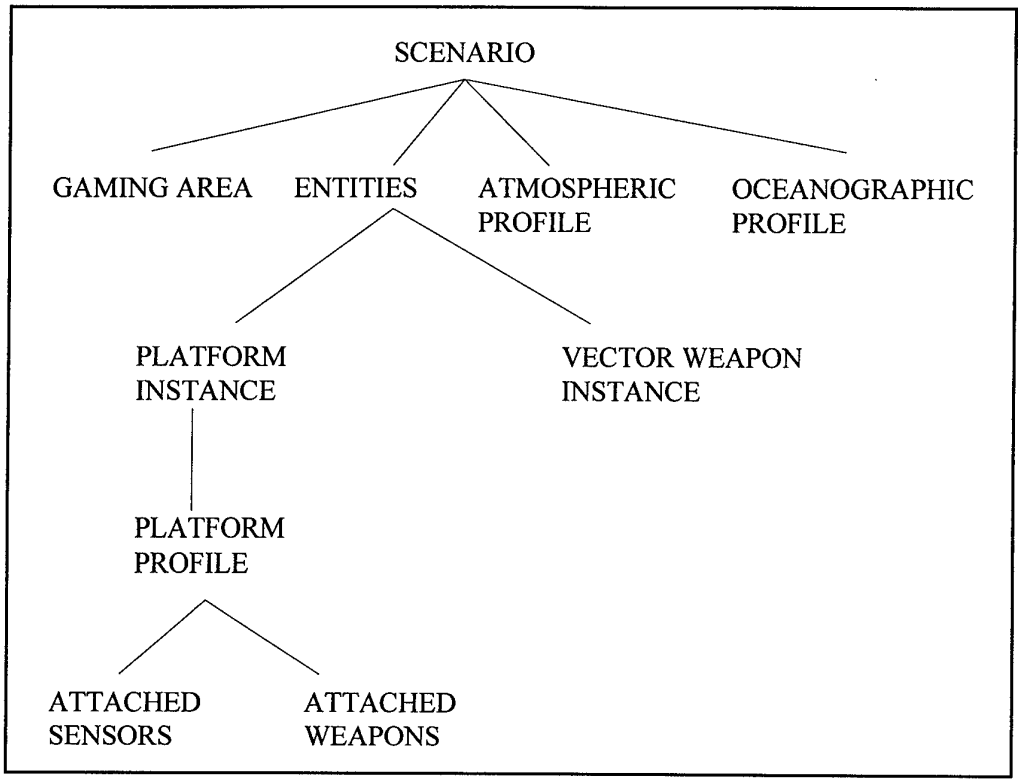


Figure 2: A hierarchy of the components in STAGE.

4.2 Sensor and Weapon Profiles

Sensor and weapon profiles define the characteristics of the sensors and weapons possessed by an entity. They are linked to an entity through the profile name. These profiles are defined separately from the entities because more than one entity can have the same sensors and/or weapons.

The sensor profiles define the capabilities of the sensors. The following characteristics can be defined:

- the type of sensor - eg radar, sonar, passive sonar, infrared, visual, or ESM;
- sensor characteristics - such as minimum and maximum sensor elevation, maximum power and scan rate; and
- sensor curve which specifies, for a given target cross-section factor, the sensor's probability of detection versus the target range. This is entered graphically.

The performance of all sensors is linearly degraded via the atmospheric and oceanographic attenuation factors specified at the global level

There are two types of weapons supported by STAGE; point and vector weapons. A point weapon is simulated statistically, whereas a weapon vector inhabits the simulated environment, along with platforms. A gun is an example of a point weapon.

This weapon is characterised by a firing rate with its lethality expressed as a probability of kill versus launch (firing) ranges to target. A successful hit is determined by whether the volumes of the weapon and target have intersected. Weapon vectors, on the other hand, are items such as torpedos and missiles. The motion of these weapons through the scenario is defined through dynamic models.

The weapon profile lists the parameters describing a weapon. The following characteristics can be defined:

- type - eg torpedo, tracking missile, cruise missile or gun;
- a bitmap for displaying the item on the Situation Awareness Display;
- a profile for torpedo and missiles which includes the following:
    - dimension (width, length, height, weight);
    - cross-sector factor in the 4 sensor spectrums (of radar, sonar, infrared and visual);
    - dynamic maximums, such as maximum turn rate and acceleration; and
    - dynamic responses such as time interval necessary to reach maximum acceleration and time interval necessary to reach maximum turn rate.
- firing rate for a gun; and
- probability of kill which specifies the probability of kill versus the range between the weapon and its target at launch time (entered graphically).

4.3 Platforms

Platforms, along with weapon vectors, are the entities that inhabit the synthetic environment. In this section, the characteristics for defining a platform are discussed.

4.3.1 Platform Instances

Platform instances define the individual characteristics of a particular platform. These characteristics are described below.

- Initial conditions - this include aspects such as colour identification and initial speed, heading and altitude.
- Behaviour script - described in the scripting language and consists of procedural description of actions to be carried out by the platform based on the status of the tactical environment. This script consists of the ability to change the platform's speed, heading, altitude and turn on/off any of the platform's systems (eg sensors and counter-measures). A script can also scan the list of detected entities and incoming weapons and select a prime opponent.
- Formation data - this indicates whether this platform is a master or slave.
- Trajectory - a series of waypoints that the platform follows. When the platform reaches the end, the platform may stop, deactivate, become invisible, destroy itself, continue, loop back or repeat a portion of trajectory.

4.3.2 Platform Profile

Platform profile specifies the characteristics and capabilities of the particular type of platform. These components contain definitions of:

- type and sub-type;
- physical, dynamic, acoustic profiles;
- counter measures;
- damage impact; and
- attached sensors and weapons.

Types

The platform types supported in STAGE are:

- fixed wing - fighter, marine patrol, bomber, transport, propeller;
- helicopter - attack and search;
- submarine - diesel and nuclear;
- surface - patrol, aircraft carrier, frigate, merchant, destroyer, fishing, LST (landing ship tank), MCMV (mine counter measure vessel), command, cruiser;
- land - tank, truck, SAM (surface-air-missile), AAA (anti-aircraft artillery), walking personnel, bicycle, motorcycle, automobile; and
- site - radar, headquarters, fire unit.

The type of the platform is used by the simulation model to control certain aspects of the platform's simulated behaviour. For example, ships and planes do not bank in the same direction when they turn.

Physical

This profile defines how the platform is 'seen' by other entities for sensor detection, collision detection and weapon scoring. This is broken up into 2 parts :

- physical dimension of the platform; and
- platform's dimension in the visual, radar, sonar and infra-red spectrum.

Dynamic

The dynamic profile is simulated via a 5 degree of freedom model. Changes to current speed, heading and altitude are made through scripts or user modules.

Acoustic

Surface and submarine platforms can have these acoustic profiles; they define the acoustic emissions of the platform.

Counter-Measures

An entity's combat effectiveness is defined in terms of the platform's overall ability to defend itself against incoming weapon vectors. It is specified as a probability of defence versus launch range of the incoming weapon function. There are two

functions of this type available, one for missiles and another for torpedoes. A platform's counter measures must be activated via a script or a user module. Entities sustain damage and reduce their combat effectiveness as a result of a weapon impact. The following levels are possible:

- no damage;
- loss of systems;
- loss of mobility; and
- destruction.

Platforms also have an overall cross-section factor (similar to all sensor spectrums) that linearly affects their detectability by sensors.

Note that no counter measures of the jammer, flare or chaff type are supported. Indirect effect can be simulated using the combat effectiveness mechanisms of the platforms.

Damage Impact

As an entity is damaged in a simulation, the degree of damage impact to the entity can be controlled. This can include, loss of weapons, loss of sensors, loss of mobility and complete destruction occurring when the entity's damage reaches a particular level.

Attached Sensors

Each attached sensor has a name and an associated sensor profile. All sensor profiles are defined elsewhere and include aspects such as type, sensor elevation and maximum power. These profiles are linked to a particular platform through the profile name. Both active and passive sensors are supported:

- active - visual, radar, sonar and infrared; and
- passive - ESM and passive radar.

Attached Weapons

Each attached weapon has a name, an associated weapon profile and a count of the number of the weapons available. Each weapon profile (as the sensor profile) is defined elsewhere and is linked here by the type of weapon chosen. This profile includes type of weapon (eg torpedo, gun) and dynamic models.

4.4 Weapon Vector Instance

Weapon vectors inhabit the simulated environment, along with platforms. A weapon vector instance consists of a name, an associated weapon profile and targeting information. Weapon profiles are defined elsewhere and are linked here by type.

5. Scripts

The scripting mechanism provided by STAGE is a powerful method of controlling and dictating the behaviour of the entities within the tactical environment. This mechanism enables the user to attach scripts to individual platforms and weapon vectors in a scenario. For instance, a platform's behaviour can be scripted allowing the platform to react to environmental and tactical conditions by performing navigation, engagement or other actions. The core functionality of scripts consists of the ability to:

- change an entity's speed, heading and altitude;
- turn on/off any of the entity's systems (ie sensors and countermeasures);
- scan the list of detected platforms and incoming weapons, and examine selected information about them; and
- select a prime opponent, obtain additional information about it and engage it by firing any of the entity's weapons at it.

5.1 Scripting Language

Each script is expressed in a scripting language which consists of a procedural description of the actions to be carried out based on the tactical environment status. The platform's script is a text file, entered by the user which contains 3 broad categories of statements:

- conditions - these statements evaluate the characteristics of the tactical environment;
- actions - these statements specify the desired changes in the platform's behaviour; and
- control - these statements bind conditions and actions together by allowing the user to:
  - trigger actions based on conditions;
  - select among alternatives;
  - repeat specified statements for a fixed or variable number of times; and
  - provide temporary storage for some results.

The scripting mechanism is build around a set of pre-defined constants, functions and objects. Some of the options available are:

- type declarations, such as **int** (integer), **float**;
- mathematical operators such as **+**, **-**, and ***;**
- logical operators such as **AND** and **OR**;
- mathematical functions such as **sin, cos**;
- formatted output functions, such as **report_flo** (print a float);
- conversion factors, such as **RAD_TO_DEG** (radians to degrees); and
- control structures which are similar to C syntax, such as **if-endif** and **while-do** statements.

An interesting construct available in the script language is a time statement to execute a particular statement at a specified time.

Each script requires the following three keywords:

- INITIALIZATION-SECTION.
- REACTION-CONTROL-SECTION.
- END-SCRIPT.

The statements listed after the initialization are executed once before the execution of the scenario. The statements listed after the control section are executed periodically. Section 4 of the STAGE User manual describe the Scripting language in detail.

5.2 Pre-Defined Script Objects

In the scripting language, objects are used to bind or consider the tactical simulation. These objects are paralleled within the tactical scenario. For instance, there are objects relating to platforms and weapons. These objects can have three types of components or members :

- variables which can be read - read-only object members;
- variables which can be set - read-write object members; and
- actions which can be performed - object member functions.

There are seven (7) built-in objects provided by STAGE. These provide access to the information of the platform owning the script, plus any other information it can have access to. Each object has data members and functions associated with them. These objects are briefly described below.

Entity - this object allows direct access to the information of the platform owning the script. The identification and dynamic status of the platform can be accessed. The entity can be instructed to change its navigation parameters. This object has:

- 17 members or parameters (eg name, actual-x-position, etc); and
- 1 function which forces the platform to change its trajectory.

Systems - this object allows access to the classes of systems (eg sensors) on board the platform owning the script. There are:

- 8 members (eg radars-active, pas-sonar-active, etc); and
- 2 functions for activating and deactivating systems.

Track - this object allows access to other entities within the scenario. However, to be accessible, the entity must be detected by the platform owning the script, or be a missile or torpedo whose target is the platform owning the script. Through this object, track id and dynamic status can be accessed. This object has:

- 15 members - (eg count, range); and
- 2 functions which allows the script to cycle through the tracks which are visible.

Opponent - selecting an entity to be the prime opponent of the platform owning the script (need not to be detected beforehand) is the purpose of this object. It has:

- 14 members - (eg name, range); and
- 1 function which assigns an entity to be an opponent.

Weapon - this fifth object allows the platform owning the script to fire one of its gun or launch one of its missiles or torpedo at its opponent. It has:

- no members; and
- 3 functions for determining the number of weapons available, firing named weapon for specified number of seconds and launching a missile/torpedo, respectively.

Script - this object allows the platform owning the script to access elapsed time of its script becoming active, to deactivate its own script and to activate and deactivate another platform's script. It has:

- 1 member which accesses elapsed time; and
- 2 functions for activating and deactivating a named platform's script.

Data-link - this final predefined object allows multiple scripts to share data in a named networks. Each platform, however, can be a member of only one network at a time. It has:

- 1 member which states the name of the currently joined network; and
- 8 functions for creating a network, joining a network, assigning integer, float and strings to appropriate buffers in the named network, and retrieving integer, float and string from the named network.

6. User Modules

Much of the behaviour of an entity can be controlled through the scripts associated with the platforms. However, STAGE has an extra capability which highlights its true flexibility and versatility. STAGE provides a Developer Interface (DI) which allows a user to extend the capability of STAGE beyond those already built in.

With this development interface, a user can modify the baseline configuration of STAGE to incorporate new models and functions. The DI allows the user, through user written user modules, to:

- extend the simulation - by adding simulation models or interfaces to external software;
- extend the script language - achieved by adding new constants, functions and objects (defined by members and functions); and
- accessing simulation data - by allowing access to internal data structures within the simulation engine.

Thus, these user modules can be used to customise the behaviour of any platform and weapon in the scenario. Besides adding functionality, user modules can be used to add to or modify the functionality of single or multiple entities. Selected models can be overridden by user module code. To achieve this, user modules install replacement code for the key built in simulation functions for the selected entities. Note that, as expected, if one replaces a built in model, the new one must generate, at the minimum, the same data that the old one did.

The key built-in simulation functions which can be overridden for platforms and weapons are, the calculation of an entity's:

- local atmospheric conditions;
- local oceanographic conditions;
- dynamics;
- navigation;
- position keeping;
- relative position keeping information;
- acoustic generation (platform only);
- sensor simulation (platform only);
- weapon scoring;
- collision detection; and
- gun firing simulation (platform only).

User modules must contain the following three sections:

- INIT - the statements within this section are executed at scenario startup;
- CONTROL - these statements describe the flow of control of this entity. Describing how the entity should work when the simulation is run, stopped, or frozen; and
- STEADY-STATE - these statements or code are carried out in steady-state mode of execution. They are invoked periodically based on the execution rate.

DI allows the end user to integrate external models and functions as external processes communicating with STAGE through the standard UNIX shared arena. These external modules can represent existing simulation systems, data analysis functions or gateways to other application software.

7. Using STAGE

When developing a scenario in STAGE, it is useful to follow the procedure listed in Figure 3. This course of action is top-down in that generic information, ie profiles, are developed first.

Consider the overall scenario to be modelled in terms of interactions and moving entities. From this, determine :

- the sensors which will be in the scenario ==> fill out a SENSOR PROFILE for each sensor;
- the weapons which will be in the scenario ==> fill out a WEAPON PROFILE for each weapon;
- the generic entities (eg aircraft, landsites) which will be in the scenario ==> fill out a PLATFORM PROFILE for each type of generic entity;
- the generic tactics, if any, which need to be applied to the entities ==> describe in words, the appropriate TACTICS (ie procedures) and the conditions of when these tactics should be employed. (For example, after aircraft A and B fire at each other do they continue to the next waypoint or do they re-evaluate their fuel condition, for instance, and possibly head home);
- the specific intances of each entity (which will include initial heading, speed, location, etc.) which inhabit the scenario ==> fill out the PLATFORM information for each entity in the scenario; and
- if STAGE does not provide the required information, consider developing specific User Modules.

Figure 3: Course of action for developing STAGE scenarios.

The main options which allow the user to perform everything that is required to create a scenario in STAGE are listed below:

- **atmospheric, oceanographic, acoustic** - brings up a menu for entering various parameters relating to the chosen option;
- **platform** - type, physical, dynamic, defence missile, defence torpedo, damage impact, sensor, weapon and acoustic information can be entered under this option. The defence missile and torpedo have a probability of defence against missile/torpedo window. A platform's probability of successful defence against a missile/torpedo versus ranges at which the missile/torpedo was launched is specified; this can also be entered as a graph. A maximum of 5 sensors (with unique names) and associated profiles are allowed. For the weapons, a maximum of 8 unique names and associated profiles are allowed;
- **scenario** - this option allows general parameters in the gaming area to be specified as well as the creation of platform instances within it. Saving and retrieving profile

data from disks along with the ability to add, delete or view platform instances is also allowed;

- **sensor** - the characteristics defining the various sensors are defined here. This includes the type of sensor (eg radar, infrared) and power settings. These profiles (like the weapon profile) are defined separately from the entities inhabiting the environment, because more than one entity can have the same sensor (or weapon); and
- **weapon** - besides a weapon profile, the PK can be entered under this option. The PK specifies the probability of kill versus ranges between a weapon and its target at launch time. This can be entered as a graph. The weapon profile includes specifying the type of weapon (eg torpedoes, guns) and dynamic models. Note above comment about the positioning of this option.

8. Built-in Models

Several built in models are provided in STAGE. As stated above, these models can be replaced and new models added. A brief description of the function of these models follows:

- evaluates the local atmospheric and oceanographic conditions surrounding a platform and /or a weapon, respectively;
- calculates the range of each platform to the nearest coast and/or obstacle and detects collision;
- calculates the relative elevation, range, bearing and absolute bearing between each platform;
- compares the position of each platform;
- calculates the requested speed, altitude and ground track of a platform in order to follow its trajectory or manoeuvre;
- calculates actual position, speed, altitude, heading, etc. of a platform in order to follow the dynamic envelope of a fixed-wing, helicopter, ship or submarine;
- evaluates the probability of defence of a platform in the occurrence of a missile or torpedo hit;
- evaluates whether a platform has been destroyed after a missile or torpedo hit;
- evaluates the probability of detection of:
  - emitting sensors (radar, sonar, or visual);
  - a platform above sea level (radar, infrared); and
  - underwater platform.;
- activates a gun for a specified time against a target;
- calculates the amount of bullets fired by an active gun and evaluates if its target has been destroyed;
- activates the launch of a missile/torpedo against a target;

- calculates the requested speed, altitude and ground track of a missile/torpedo in order to guide it toward its assigned target;
- calculates the actual position, speed, altitude, heading etc of a missile/torpedo in order to follow its dynamic envelope;
- compares the position of a missile/torpedo against the position of its target; and
- evaluates if a weapon has been destroyed after a missile or torpedo hit.

9. Worked Example of a STAGE Scenario

This section describes a worked example of a scenario constructed in STAGE. All profiles, instances and scripts are described. Basically, the aim of the scenario was for two aircraft to attack two land sites. Please note that this example is to demonstrate STAGE's capabilities and that the scenario, profiles and scripts are not meant to be representative of a real situation.

9.1 Scenario Details

This scenario consisted of four entities of two fixed wing aircraft (r_fighter_01 and r_fighter_02) and two land sites (b_land_01 and b_land_02). Each of the fighters were equipped with four cruise missiles and radar sensors. Each of the land sites were equipped with six surface-to-air tracking missiles, as well as radar and visual sensors. The trajectory of each aircraft consisted of waypoints which enabled the aircraft to fly directly over the two land sites. The weapons launch of each aircraft in the scenario was controlled by user-written scripts.

In order to appreciate the scenario's real-time progression, Appendix A contains Figures A1 to A5 which show the scenario at different times. These figures, which are a screen dump of the Situation Awareness Display (SAD), are:

- Figure A1 : At time 0:00 - initial placement of entities;
- Figure A2 : At time 1:00 - both aircraft r_fighter_01and r_fighter_02 have fired missiles at the land base b_land_02. The land base, b_land_0,2 has fired two missiles, one at each aircraft. Note in this figure that the missiles are now entities in the scenario, having their own names;
- Figure A3 : At time 2:00 - land base b_land_02 is destroyed by the cruise missile (cruise_mis);
- Figure A4: At time 3:30 - both aircraft have fired missiles at the land base b_land_03. Land base b_land_03 has fired two missiles, one at each aircraft; and
- Figure A5 : At time 4:00 - final state of the scenario. The land base b_land_03 has been destroyed and the two aircraft have escaped from the missiles that were following.

The profiles, instances and scripts for this scenario are described in the following sections.

9.2 Sensor Profiles

Each sensor attached to an entity (either platform or weapon) is described in a sensor profile. This profile contains the generic information pertaining to the sensor, as provided by STAGE; note that these profiles are not realistic and are only meant for demonstration. In this scenario, there was a need for four sensor profiles. The first three are associated with the aircraft, while the fourth one is associated with the land site. The sensor profiles used in this scenario are:

- demo_radar as listed in Table 1;
- demo_visual as listed in Table 2;
- demo_infrared as listed in Table 3; and
- demo_sam as listed in Table 4.

Table 1: The demo_radar sensor profile in the scenario

Type	Sensor_Radar		
Categories Detected	All	Earth Radius Factor	1.000
Attributes Detected	All	emission power	500
Detection Envelope			
min elevation	-89.954 °	scanning rate	6 RPM
max elevation	89.954 °	scanning # of hits	3
min azimuth	-180 °		
max azimuth	180 °		

Table 2: The demo_visual sensor profile.

Type	Sensor_Visual
Earth Radius Factor	1.000
Detection	Envelope
min elevation	-89.954 °
max elevation	89.954 °
min azimuth	-180 °
max azimuth	180 °

Table 3 : The demo_infrared sensor profile.

Type	Sensor_Infrared
Earth Radius Factor	1.000
Detection	Envelope
min elevation	-89.954 °
max elevation	89.954 °
min azimuth	-180 °
max azimuth	180 °

Table 4 : The sam_sensor sensor profile.

Type	Sensor_Radar		
Categories Detected	All	Earth Radius Factor	1.000
Attributes Detected	All	emission power	500
Detection Envelope			
min elevation	0 °	scanning rate	35 RPM
max elevation	90 °	scanning # of hits	1
min azimuth	-180 °		
max azimuth	180 °		

9.3 Weapons Profiles

In this scenario, there was a need for two weapon profiles. One is associated with the aircraft and the other is associated with the landsite. Recall once again that the numbers in these profiles are not meant to be real. These profiles are:

- GF_MSL_CRU as listed in Table 5; and
- demo_missile as listed in Table 6.

9.4 Platform Profiles

In this scenario, only two platform profiles were required. These relate a generic fighter and the demo-sam profiles, which relate to the aircraft and land sites, respectively. The numbers within the profiles are meant to be representative only; they are not real numbers. The fighter profile is listed in Table 7, while Table 8 contains the demo-sam profile. Note that the sensors and weapons defined above are linked into the entities through these profiles; both attached sensors and weapons are associated via the associated profiles.

Table 5 : The GF_MSL_CRU weapons profile STAGE.

Type	Vector Weapon	Type	Cruising_Weapon
max range	100 km		
	Physical Characteristics		
Dimensions		Cross Section Factor	
length	20 m	radar	10
height	1 m	sonar	10
width	1 m	visual	10
mass	52 kg	infrared	10
	Dynamic Maximums		
Maximums		Responses	
min speed	0 m/s	dv to max acc	25 m/s
max speed	500 m/s	time to max acc	5 s
acceleration	20 m/s^2	dhdg to max rot	1.146 °
deceleration	20 m/s^2	time to max rot	0.5 s
turn rate	44.691 °/s	max pitch rate	44.691 °/s
roll	89.954 °	max roll rate	17.762 °/s
altitude	10000 m	dalt to max rot	500 m
climb rate	200 m/s	dalt to max rod	500 m
dive rate	200 m/s	dalt to max vert spd	0 m

Table 6 : The demo-missile weapons profile

Type	Vector Weapon	Type	Tracking_Weapon
max range	35 km		
	Physical Characteristics		
Dimensions		Cross Section Factor	
length	20 m	radar	10
height	1 m	sonar	10
width	1 m	visual	10
mass	52 kg	infrared	10
	Dynamic	Maximums	
Maximums		Responses	
min speed	0 m/s	dv to max acc	25 m/s
max speed	500 m/s	time to max acc	5 s
acceleration	20 m/s^2	dhdg to max rot	1.146 °
deceleration	20 m/s^2	time to max rot	0.5 s
turn rate	44.691 °/s	max pitch rate	44.691 °/s
roll	89.954 °	max roll rate	17.762 °/s
altitude	10000 m	dalt to max rot	500 m
climb rate	200 m/s	dalt to max rod	500 m
dive rate	200 m/s	dalt to max vert spd	0 m

Sensor Name	Sensor Profile
ir	demo_infra_red
sam_sen	sam_sensor

Table 7 : The fighter platform profile.

Type	wing	Sub type	fighter
	Physical Characteristics		
Dimensions		Cross Section Factor	
length	17 m	radar	125
height	6 m	sonar	0
width	11 m	visual	125
mass	5277 kg	infrared	0
	Dynamic	Maximums	
Maximums		Responses	
min speed	0 m/s	dv to max acc	25 m/s
max speed	1000 m/s	time to max acc	5 s
acceleration	20 m/s^2	dhdg to max rot	40.001 °
deceleration	20 m/s^2	time to max rot	2 s
turn rate	20 °/s	max pitch rate	20 °/s
roll	89.954 °	max roll rate	20 °/s
altitude	40000 m	dalt to max rot	1000 m
climb rate	200 m/s	dalt to max rod	1000 m
dive rate	200 m/s	dalt to max vert spd	0 m
Weapon Name	Weapon Profile	Sensor Name	Sensor Profile
crusie_mis	GF_MSL_CRU (x4)	radar	demo_radar

Table 8 : The demo-sam platform profile.

Type	land	Sub type	SAM
Physical Characteristics			
Dimensions		Cross Section Factor	
length	20 m	radar	100
height	10 m	sonar	100
width	10 m	visual	100
mass	5277 kg	infrared	100
Dynamic Maximums			
Maximums		Responses	
min speed	0 m/s	dv to max acc	25 m/s
max speed	15.433 m/s	time to max acc	5 s
acceleration	20 m/s^2	dhdg to max rot	179.909 °
deceleration	20 m/s^2	time to max rot	2 s
turn rate	44.691 °/s	max pitch rate	0 °/s
roll	89.954 °	max roll rate	0 °/s
altitude	10000 m	dalt to max rot	0 m
climb rate	0 m/s	dalt to max rod	0 m
dive rate	0 m/s	dalt to max vert spd	0 m
Weapon Name	Weapon Profile	Sensor Name	Sensor Profile
sam_missile	demo_missile (x 6)	sam_radar	demo_radar
		sam_visual	demo_visual

9.5 Scripts

The actions of each of the entities in this scenario, were controlled by written scripts. These scripts activated the platform's sensors, detected other platforms in the gaming area, and launched weapons at opponent platforms when specific conditions were met. The two fighter aircraft were both controlled by the fighter script and the two land sites were both controlled by the Surface to Air Missile (SAM) script.

The fighter script (listed in Figure 9) controls the actions of the two fighter aircraft in the scenario. The initialisation section of the script activates the radar sensors on the platform. The reaction control section of the script detects other platforms in the gaming area that are seen by the sensors of the this platform and determines what action to take. If the detect platform is hostile and close enough for the weapons to be effective (in this case 50 km), the platform is targeted and a missile is launched against this target. Once the platform has launched missiles at both land sites, the

script changes the platforms heading and speed to remove it from the gaming area (0° heading, speed of 250 m/s).

The SAM script, listed in Figure 10, controls the actions of the two land sites in the scenario. The initialization section of the script activates the radar and visual sensors on the platform. The reaction control section of the script detects other platforms in the gaming area and determines action to be taken against them. If the detected platform is a fixed wing entity and hostile and within effective range of the missiles (35 km), the platform is targeted and a missile launched against it. Only one missile is launched at each target.

```
INITIALIZATION_SECTION

systems.activate(radar_sensors); # activate the radar sensors
int fired_at_track = -1;         # initialization of variables
int fired_at_target_1 = -1;
int fired_at_target_2 = -1;

REACTION_CONTROL_SECTION

fired_at_track = FALSE;
track.cycle_on(detected_platforms);
while (track.next( ) > 0) do
  if ((track.range <= 50000) and (track.ident = hostile)) then
    if (fired_at_target_1 = -1) then
      fired_at_target_1 = track.index;
      fired_at_track = TRUE;
    endif;
    if ((fired_at_target_1 != track.index) and
        (fired_at_target_2 = -1)) then
      fired_at_target_2 = track.index;
      fired_at_track = TRUE;
    endif;
    if (fired_at_track = TRUE) then
      opponent.assign(track.index);
      weapon.launch(cruising_missile,"crusie_mis");
    endif;
  endif;
endwhile;

if ((fired_at_target_2 != -1) and
    (entity.actual_ground_track != 0)) then
  entity.requested_ground_track = 0;
  entity.requested_speed =250;
endif;
END_SCRIPT
```

Figure 9 : Fighter script used in scenario

```
INITIALIZATION_SECTION
```

```
systems.activate(radar_sensors);  # activate radar sensors on
                                  # the platform
systems.activate(visual_sensors); # activate visual sensors on
                                  # the platform
int fired_at_track = -1;
int fired_at_target_1 = -1;
int fired_at_target_2 = -1;

REACTION_CONTROL_SECTION

fired_at_track = FALSE;
track.cycle_on(detected_platforms);
while (track.next( ) > 0) do
  if ((track.range <= 35000) and (track.ident = hostile) and
      (track.type = fixed_wing)) then
    # target found
    if (fired_at_target_1 = -1) then
      fired_at_target_1 = track.index;
      fired_at_track = TRUE;
    endif;
    if (fired_at_target_1 != track.index) and
       (fired_at_target_2 = -1) then
      fired_at_target_2 = track.index;
      fired_at_track = TRUE;
    endif;
    if (fired_at_track = TRUE) then
      opponent.assign(track.index);
      weapon.launch(tracking_missile,"sam_missile");
    endif;
  endif;
endwhile;
END_SCRIPT
```

Figure 10: The SAM script used in the scenario.

9.6 Platform Instances

Each entity in a scenario is an instance of a platform profile. In this demonstration scenario, there are four entities, two instances of the fighter platform profile, and two instances of the demo_sam platform profile. Each instance of a profile has the characteristics inherent in that profile: physical and dynamic characteristics, sensor cross sections, sensors and weapons. However, the force colour (blue for friendly, red for hostile and white for neutral), initial conditions, scripts and movement paths are all unique to an instance. Listed in Tables 9 to 12 is the platform instance information for each entity in the scenario; this information contains the initial conditions of the 4 entities.

Table 9: Instance Information for aircraft entity "r_fighter_01".

Platform Name	r_fighter_01		
Profile	fighter		
Color	red		
Activate Time	00:00:00		
Script	fighter_script		
Script Active	yes		
Initial			
Position: Latitude	N49:46:34.5		
Position: Longitude	W001:00:19.0		
Speed	200 m/s		
Heading	180°		
Altitude	500 m		
Trajectory	1	2	3
ETA	00:03:00	00:04:00	00:05:00
Position: Latitude	N48:50:11.4	N48:40:00.0	N49:10:31.0
Position: Longitude	W001:00:04.1	W001:20:00.0	W001:36:25.9
Altitude	1000 m	1000 m	1000 m
Wait Time	00:00:00	00:00:00	00:00:00

Table 10: Instance information for aircraft entity "r_fighter_02".

Platform Name	r_fighter_02	
Profile	fighter	
Color	red	
Activate Time	00:00:00	
Script	fighter_script	
Script Active	yes	
Initial		
Position: Latitude	N49:49:45.2	
Position: Longitude	W001:00:00.9	
Speed	200 m/s	
Heading	-170°	
Altitude	500 m	
Trajectory	1	2
ETA	00:02:30	00:04:00
Position: Latitude	N49:02:57.4	N48:39:56.8
Position: Longitude	W001:04:21.1	W001:20:12.7
Altitude	1000 m	1000 m
Wait Time	00:00:00	00:00:00

Table 11: Instance Information for land base entity "b_land_02".

Platform Name	b_land_02
Profile	demo_sam
Color	blue
Activate Time	00:00:00
Script	SAM_script
Script Active	yes
Initial	
Position: Latitude	N48:58:39.1
Position: Longitude	W001:03:19.3
Speed	0 m/s
Heading	0°
Altitude	63.00 m

Table 12: Instance information for land base entity "b_land_03".

Platform Name	b_land_03
Profile	demo_sam
Color	blue
Activate Time	00:00:00
Script	SAM_script
Script Active	yes
Initial	
Position: Latitude	N48:40:00.0
Position: Longitude	W001:20:11.3
Speed	0 m/s
Heading	0 °
Altitude	71.00 m

10. Recapitulation

This paper has described the key features of the modeling tool known as STAGE. STAGE provides an environment for the development of real-time tactical situations. It uses menus and pulldown menus for easy entry of data. It has a powerful display (SAD) capability which has a dynamic link to the positioning of entities on the display. Hooked entities can be repositioned and instantiated during the running of a simulation. Platforms (eg aircraft, ships and land vehicles) and weapon vectors (missiles and torpedos) are the entities which inhabit the synthetic environment. These entities were described in Section 4 with their main features highlighted. STAGE provides a scripting mechanism which associates a script with individual entities (platforms or weapons). This script is used to control the behaviour of the entity (Section 5). User modules, described in Section 6, give the user the capacity to extend the capability provided by STAGE by allowing the user to extend the scripting mechanism, replace existing models and providing access to STAGE's internal data structures. The flexibility of this tool is thus apparent. Using STAGE was described in Section 7, while Section 8 briefly described the built in models provided by STAGE. A worked example of STAGE was provided in Section 9, where examples of profiles, instances and scripts were given.

Appendix A
Situation Awareness Display (SAD) Depictions of the Scenario

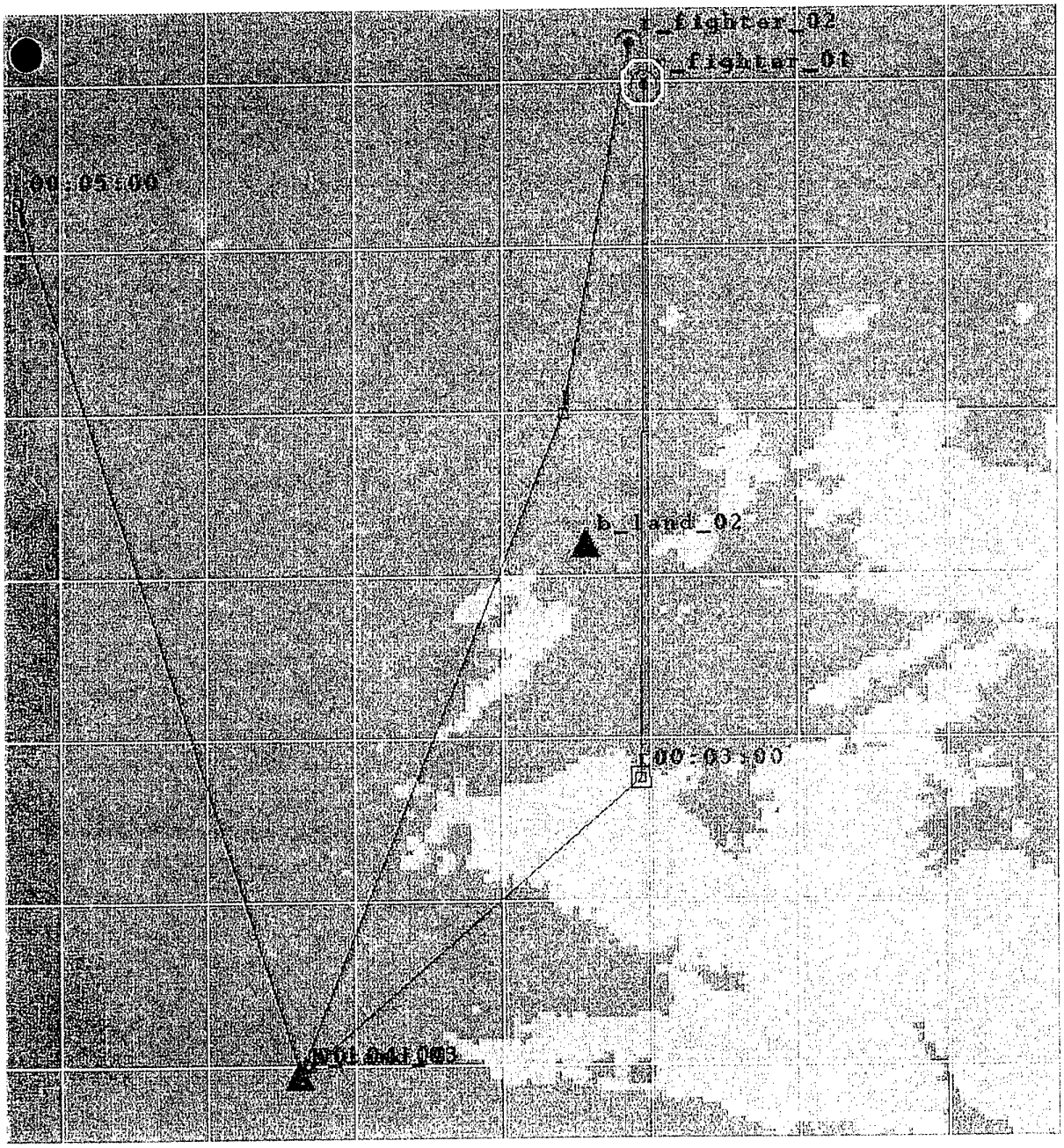


Figure A1. At time 0:00

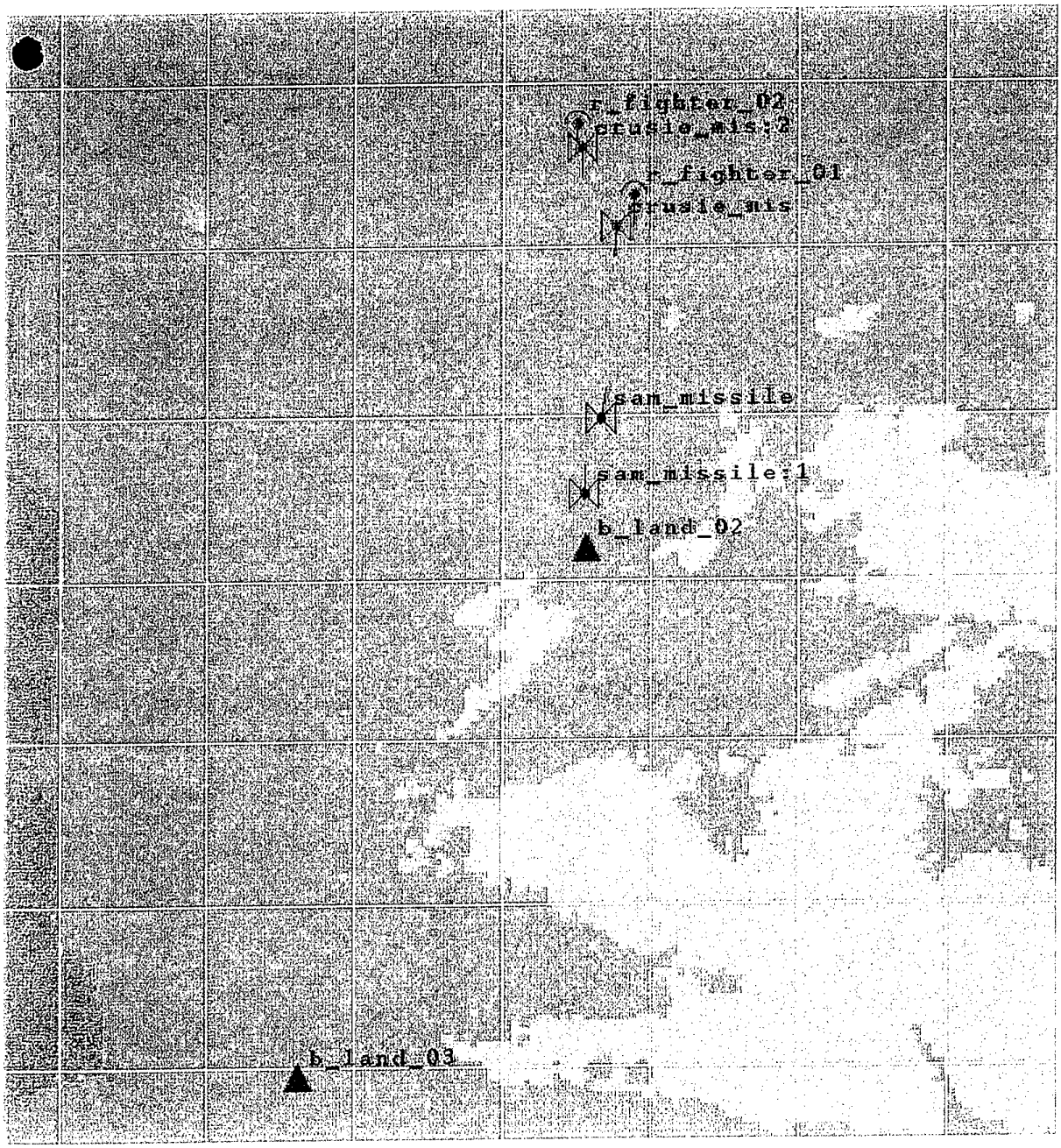


Figure A2. At time 1:00

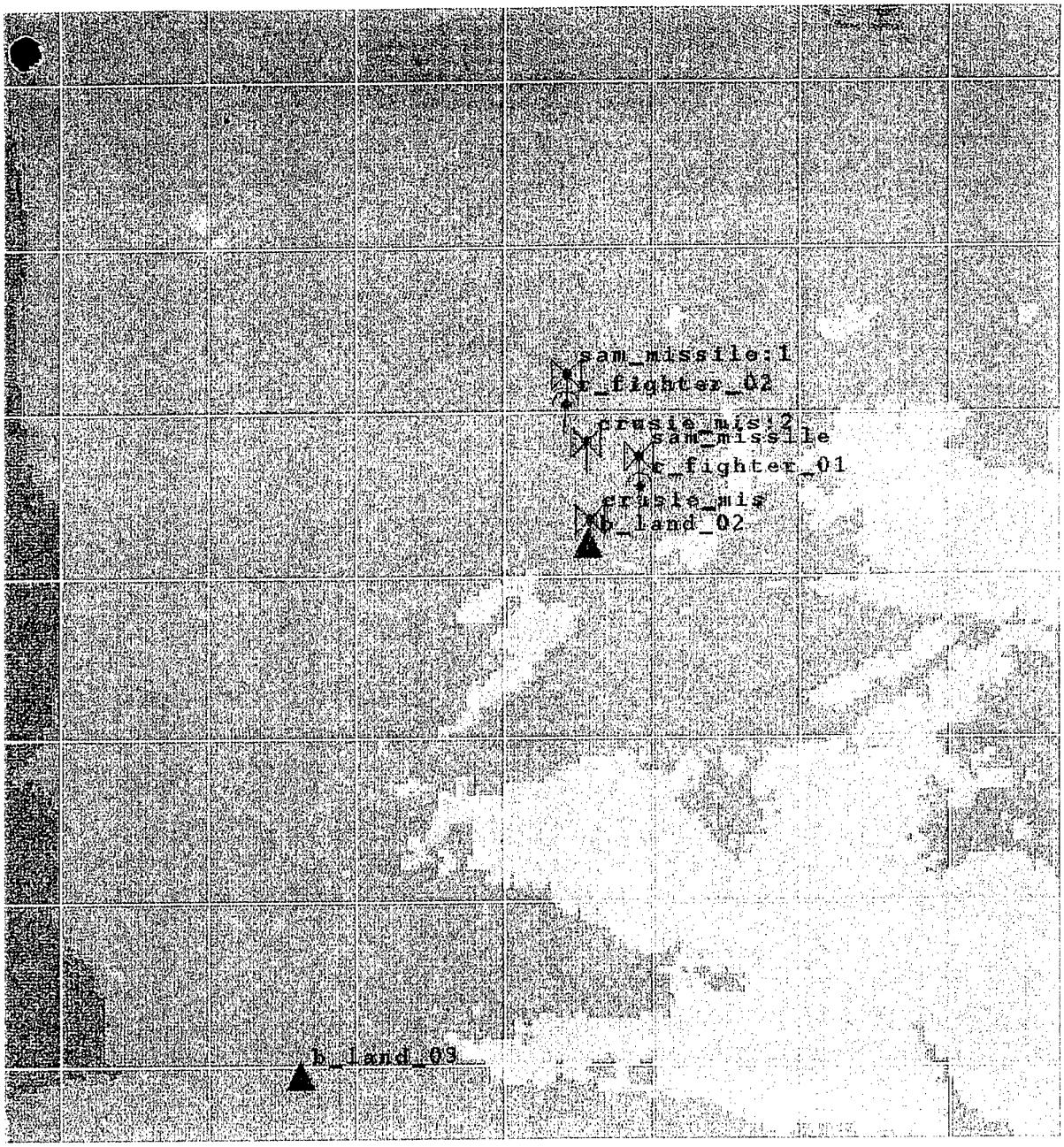


Figure A3. At time 2:00

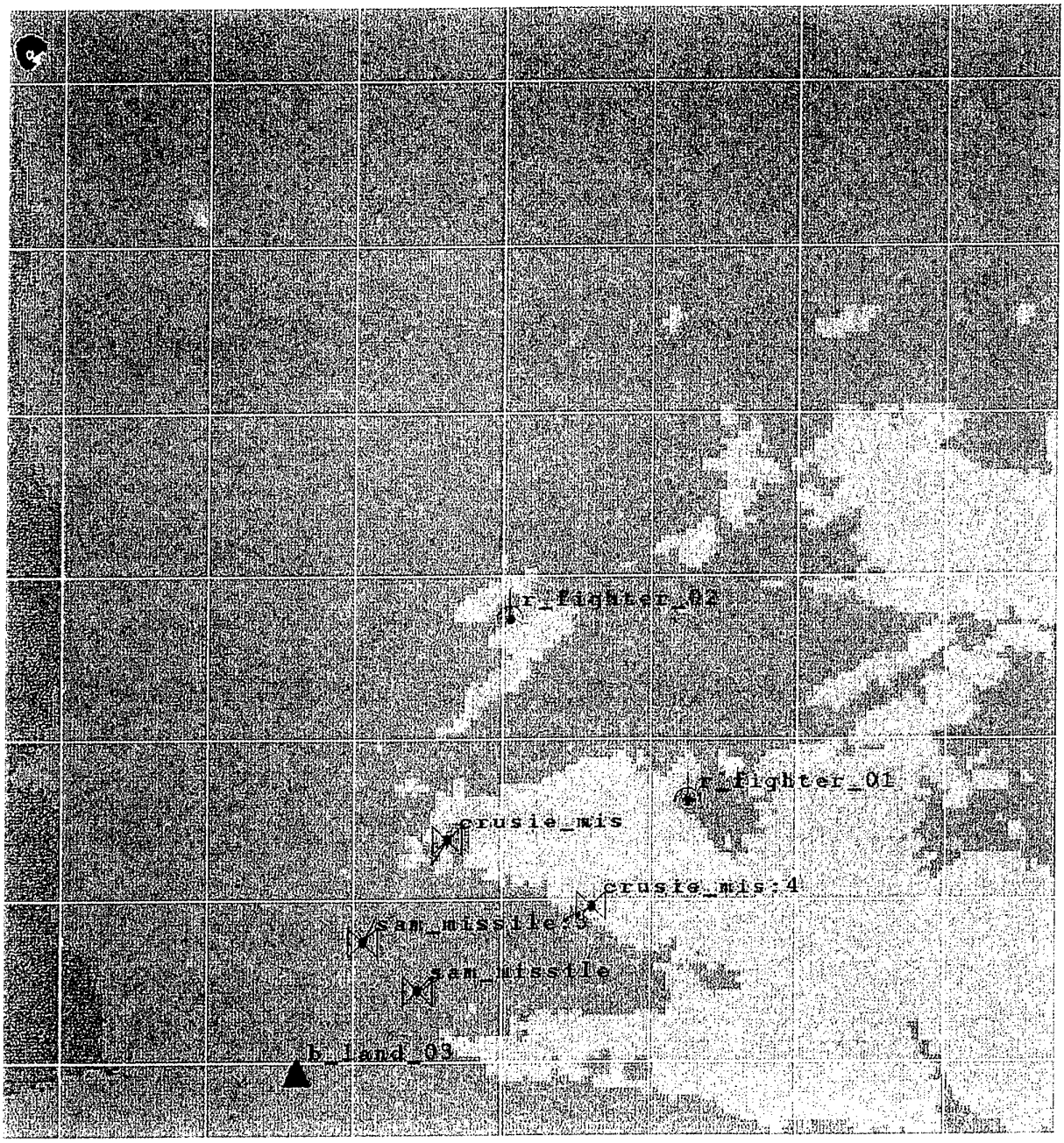


Figure A4. At time 3:30

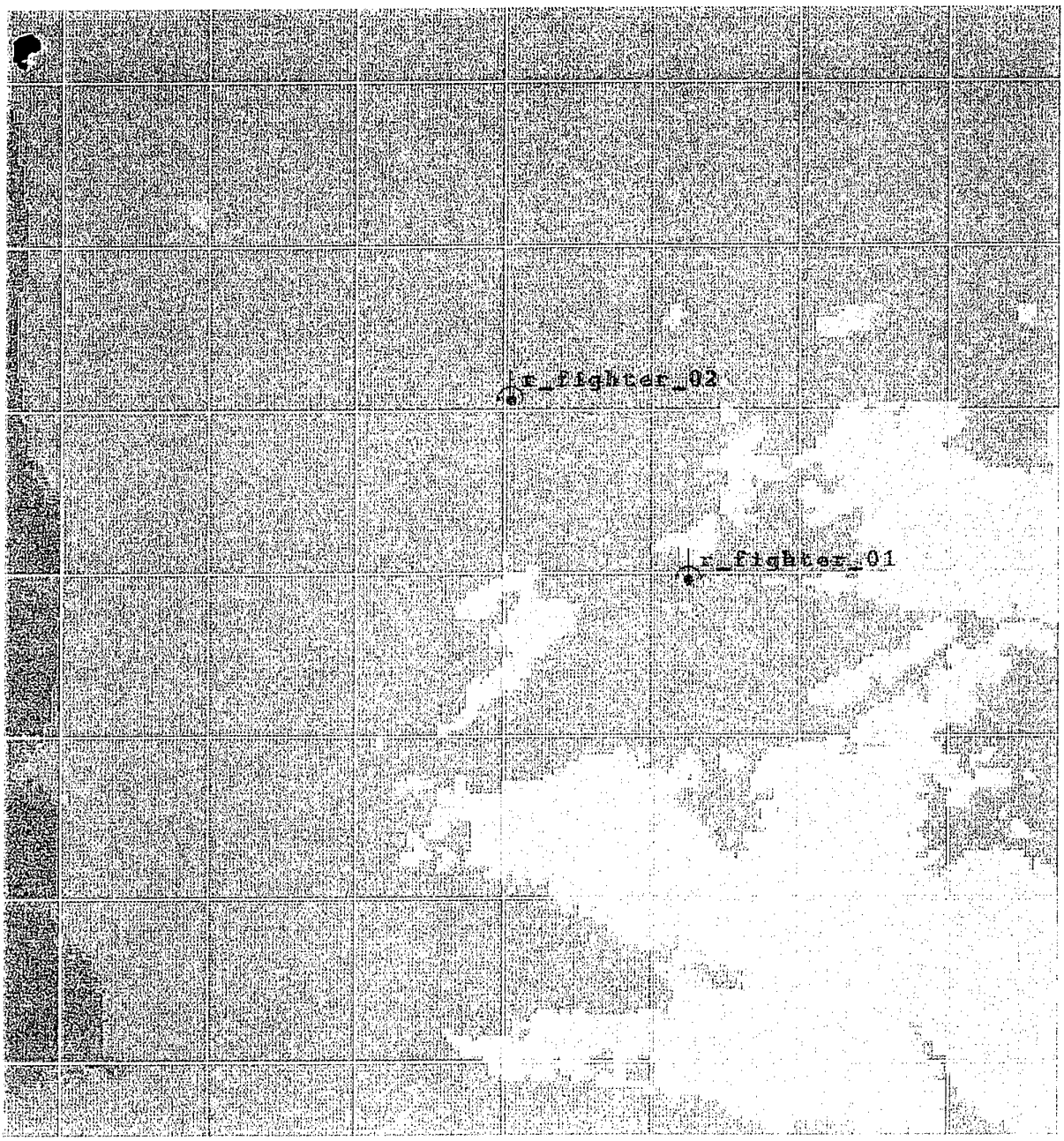


Figure A5. At time 4:00

Description and Worked Example of STAGE

Sabrina Sestito
and
Jodie Doman

AUSTRALIA

DEFENCE ORGANISATION

Defence Science and Technology Organisation

Chief Defence Scientist
FAS Science Policy } shared copy
AS Science Corporate Management
Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington
Scientific Adviser to Thailand MRD (Doc Data Sheet only)
Senior Defence Scientific Adviser/Scientific Adviser Policy and Command (shared copy)
Navy Scientific Adviser (3 copies of Doc Data Sheet and 1 copy of Distribution list)
Scientific Adviser - Army
Air Force Scientific Adviser
Director Trials
Director, Electronics and Surveillance Research Laboratory
Director, Aeronautical and Maritime Research Laboratory

Air Operations Division:

Chief of Division
Research Leader: B. Feik
Task Manager: J. Harvey, M. Mason
Authors: Sabrina Sestito, Jodie Doman

P. Ryan, MOD
D. Fogg, ITD
J. Coleby, LSOD
C. Meline, EWD
G. O'Conner, WSD

G. Lawrie AOD, Salisbury

DSTO Library

Library Fishermens Bend
Library Maribyrnong
Main Library DSTOS (2 copies)
Library, MOD, Pyrmont (2 copies)
Defence Science and Technology Organisation Salisbury, Research Library (6 copies)

Defence Central

OIC TRS, Defence Central Library
Officer in Charge, Document Exchange Centre (DEC), 1 copy

DEC requires the following copies of public release reports to meet exchange agreements under their management:

*US Defence Technical Information Centre, 2 copies
*UK Defence Research Information Centre, 2 copies
*Canada Defence Scientific Information Service
*NZ Defence Information Centre
National Library of Australia
Library, Defence Intelligence Organisation, 1 copy
Library, Defence Signals Directorate (Doc Data Sheet only)

Army

Director General Force Development (Land) (Doc Data Sheet only)
ABCA Office, G-1-34, Russell Offices, Canberra, 4 copies
SO (Science), HQ 1 Division, Milpo, Enoggera, Qld 4057 (Doc Data Sheet)
NAPOC QWG Engineer NBCD c/- DENGRS-A, HQ Engineer Centre Liverpool Military Area, NSW 2174 (Doc Data Sheet)

Navy

Director General Force Development (Sea),
SO (Science), Director of Naval Warfare, Maritime Headquarters Annex, Garden Island, NSW 2000.

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy Library
Senior Librarian, Hargrave Library, Monash University

OTHER ORGANISATIONS

NASA (Canberra)
AGPS

ABSTRACTING AND INFORMATION ORGANISATIONS

INSPEC: Acquisitions Section Institution of Electrical Engineers
Library, Chemical Abstracts Reference Service
Engineering Societies Library, US
American Society for Metals
Documents Librarian, The Center for Research Libraries, US

INFORMATION EXCHANGE AGREEMENT PARTNERS

Acquisitions Unit, Science Reference and Information Service, UK
Library - Exchange Desk, National Institute of Standards and Technology, US

SPARES (10 copies)

Total 73 copies

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Description and Worked Example of STAGE

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Sabrina Sestito & Jodie Doman

5. CORPORATE AUTHOR

Aeronautical and Maritime Research Laboratory
PO Box 4331
Melbourne Vic 3001

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-GD-0092	AR-009-~~692~~ 634	General Document	April 1996

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
M1 /8 /924			39	--

13. DOWNGRADING/DELIMITING INSTRUCTIONS

---

14. RELEASE AUTHORITY

Chief, Air Operations Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Computer Modelling Simulation STAGE Scenario Toolkit And Generation Environment War Games

19. ABSTRACT

This paper describes the key features of the Scenario Toolkit And Generation Environment (STAGE) software. STAGE provides an environment for the development of real-time tactical situations. It uses menus and additional pulldown menus for easy entry of data and provides a powerful display capability including a dynamic link to the positioning of entities on the display. STAGE supports user written scripts, which are associated with individual platforms, to dictate the behaviour of each entity. User written code, known as user modules, allow the user to extend the capability provided by STAGE (by expanding) the scripting mechanisms, replacing existing models and providing access to STAGE's internal data structure. Use of STAGE and a course of action for developing scenarios is described in this paper, along with a worked example giving detailed listings of all data, profiles and scripts required to run a simple scenario.