**USING STK TO VISUALIZE DEVS SIMULATIONS**

**Use case: An Air Defence System Scenario**

|  |
| --- |
| Françoise Blanc |
| Department of Systems and Computer Engineering  Carleton University |
| 1125 Colonel By Drive  Ottawa, ON, K1S 5B6, Canada |
| francoiseblanc@cmail.carleton.ca |
|  |
|  |

# ABSTRACT

Cadmium is a simulator for discrete-event modelling and simulation using discrete-event systems specification (DEVS) formalism. Even though Cadmium simulation results can be visualized using a tool called DEVS WebViewer, other existing visualization tools may provide more insights into the simulation results. This paper presents the findings of an explorative study on the use of Systems Tool Kit (STK), a commercial off the shelf software developed by Analytical Graphic's Inc., to visualize Cadmium DEVS simulations. A simplified DEVS model representing an air defence system was developed to explore and demonstrate the use of STK. This study revealed that STK can effectively visualize Cadmium DEVS outputs by converting the simulation outputs into a set of STK Connect commands autogenerating an STK scenario.

**Keywords:** Discrete-event system, Visualization, Systems Tool Kit, STK, Defence

# INTRODUCTION

Discrete-event systems specification (DEVS) formalism (Zeigler et al. 2000) provides a common basis for discrete-event modelling and simulation. Several DEVS simulators have been developed to provide an environment to define and execute DEVS models. One of these simulators is Cadmium (Belloli et al. 2019), a tool developed at the Advanced Real-Time Simulation Laboratory at Carleton University, Ottawa. To visualize DEVS simulation results from Cadmium, a visualization tool was also developed: DEVS WebViewer. DEVS WebViewer provides an animated visualization of the simulation outputs generated by Cadmium. For DEVS models, this viewer uses a Scalable Vector Graphics file representing the model's structural elements (i.e., atomic and coupled models, ports, and couplings). The DEVS WebViewer displays the simulation results by highlighting the active model's elements in every simulation time step. This viewer gives an accurate representation of the model's behaviour; however, for DEVS models using geographic locations, other existing viewers may better represent the simulation. An existing tool that can be explored for analyzing DEVS outputs is Systems Tool Kit (STK) (Analytical Graphics Inc. 2020).

This paper presents the main elements of an explorative study evaluating STK's ability to visualize DEVS simulation outputs. This demonstration is supported by the use of a basic DEVS model representing an air defence system. Section 2 contains background information on STK. Section 3 provides detailed information about the DEVS model used for the demonstration of STK. Section 4 details the implementation activities. Section 5 provides the STK results for a specific experiment using the air defence system DEVS model. Section 6 concludes this paper with the main findings and proposed future work.

# BACKGROUND

## Systems Tool Kit

Systems Tool Kit (STK), formerly known as Satellite Tool Kit, is a 2D and 3D modelling environment developed by Analytical Graphics, Inc. STK was designed initially for orbit analysis and access calculations between sensors. STK has now been expanded to perform complex analysis in multiple domains, including ground, sea, air, space and cyber. Typical STK scenarios include intelligence, surveillance and reconnaissance (ISR), missile defence, space systems, and electronic warfare.

STK requires a license to either access STK Pro or STK Cloud. STK Pro is the standard product from which specialized modules can be added. STK Cloud allows access to STK's capabilities in a browser. For this study, STK Pro without specialized modules was used.

Building a simple STK scenario can be accomplished following a few steps. The user must first create a new scenario in STK. Then, the user inserts STK objects to the scenario, as shown in Figure 1. These STK objects provide a general template that can be configured as desired. Once an STK object is inserted into a scenario, the user can attach other STK objects to it. For example, a user can insert an *Aircraft* and then attached to it an *Antenna* and a *Radar*. One of STK's key features is its *Compute Access* function, which allows determining the times one object can access another object. In other words, this feature calculates object-to-object visibility based on the STK objects' properties and constraints. STK provides several different types of constraints to model real-world limitations. Such limitations include, for example, basic constraints (e.g., range, altitude and angular limitations) and temporal constraints.

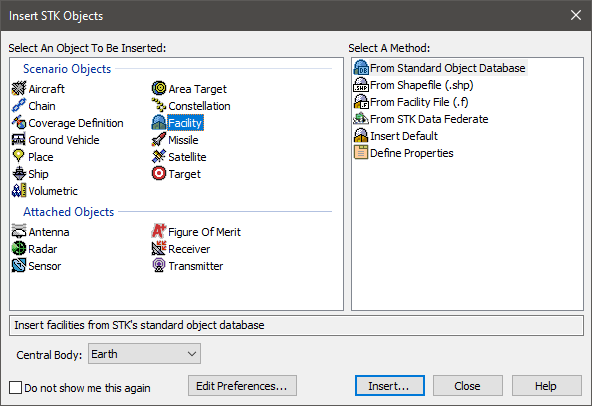


Figure 1: STK objects and attached objects.

# USE CASE: AN AIR DEFENCE SYSTEM SCENARIO

A simplified air defence system model was designed to demonstrate STK's potential in visualizing Cadmium DEVS outputs. As STK is often used for defence applications, this scenario provides a realistic demonstration of how STK could be integrated for DEVS modelling and simulation. This section provides background information regarding the chosen scenario and also describes the DEVS model's elements.

## Context

Military aircraft deployed in theatre can encounter various threats. As new threats and new technologies are being developed, countries need to address these challenges to ensure military aircraft protection readiness. One way to assess aircraft protection is through the use of modelling and simulation.

One major threat against military aircraft in conflicts involving non-state armed groups (NSAG) is man-portable air defence systems (MANPADS). MANPADS, such as the one shown in Figure 2, are short-range surface-to-air missile systems intended for low-flying aircraft. These systems are usually designed to be shoulder-launched by one single person. MANPADS are often the weapons of choice to NSAG as they are widely proliferated, relatively cheap, and easy to transport and use. Cueing devices, such as radar providing target position well beyond the visual acquisition range, are now being incorporated into MANPADS and used by NSAG. DEVS modelling can represent such air defence systems using radar-augmented MANPADS to answer specific questions. For example, this model could be used to measure the effectiveness of an air defence system and to assess and develop tactics on either the enemy or the friendly side.



Figure 2: Radar-augmented MANPADS (Rheinmetall Defence 2020).

## Model Overview

Figure 3 depicts the conceptual model developed to represent an air defence system using radar-augmented MANPADS. Note that this model was not designed to provide an accurate representation of such a system. A high-fidelity model would involve extensive modelling, which is not the project's purpose. The model consists of two main components: an *Aircraft* and an *Air Defence System*. The *Air Defence System* component is further divided into four atomic models: radar, command and control (C2), MANPADS gunner and missile.

The simulation starts once an aircraft approaches the air defence system. Once the aircraft is within the radar's detection range, the C2 starts analyzing the situation and then coordinates missile launches with the gunners. These components have been developed using Cadmium. Each atomic and coupled model has been tested individually to ensure correct model's behaviour.

## Model Assumptions

As previously mentioned, this model does not provide an accurate representation of such air defence systems. This model was designed as a proof of concept to demonstrate STK's ability to display DEVS outputs. The following assumptions were made:

* The model consists of one aircraft, one radar, one C2, three MANPADS gunners and three associated missile models;
* The mission area is a flat terrain (no obstruction) and sky clear;
* The aircraft maintains a predefined straight path at constant altitude and speed without evasive manoeuvre;
* The radar detection range is not target-dependent;
* The maximum missile range is set to 6 km; and
* The probability of hit only depends on the distance between the missile and the aircraft.

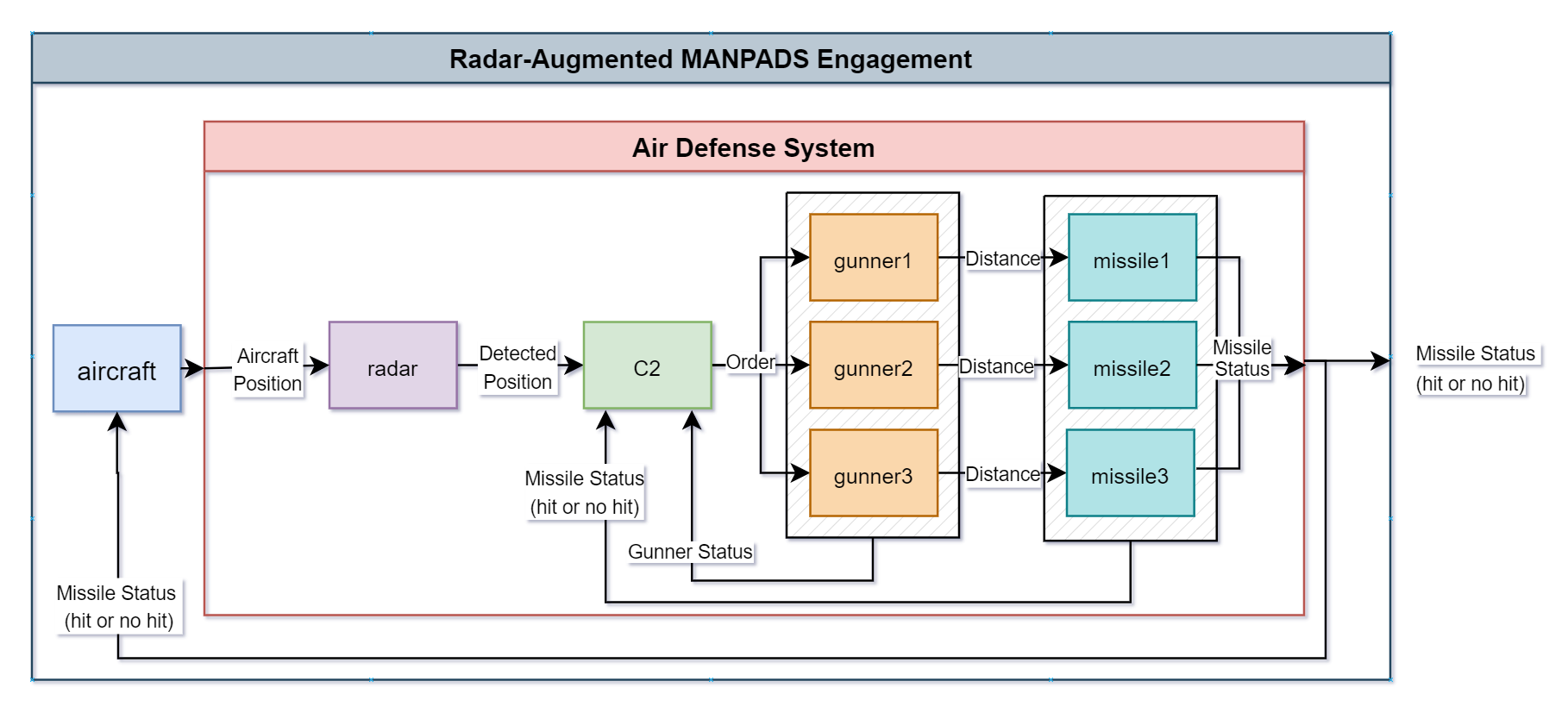


Figure 3: Air defence system conceptual model.

## Atomic Models

### Aircraft Atomic Model

The aircraft atomic model has three phases: *FLYING*, *MISSION COMPLETED*, or *HIT*. The aircraft model generates an updated aircraft position every second based on its initial position, final position, speed and altitude. The model generates an updated aircraft position until the aircraft reaches the end of its flying path or until the aircraft receives an input indicating a missile hit. Figure 4 provides the aircraft atomic model formal specification and pseudo-code.

|  |  |
| --- | --- |
| **Aircraft Atomic Model = <X, Y, S, δext, δint, λ, ta>** | |
| **State Variables** | **Formal Specification** |
| Phase = FLYING  Sigma = 0  Aircraft Initial Position  Aircraft Final Position  Aircraft Speed  Aircraft Altitude  Updated Aircraft Position  Total Flight Path Distance  Distance Flown  Aircraft Bearing  Missile Status | X = {Missile Status}  Y = {Aircraft Position}  S = {Phase, Sigma, Aircraft Initial Position, Aircraft Final Position, Aircraft Speed, Aircraft Altitude, Updated Aircraft Position, Total Flight Path Distance, Distance Flown, Aircraft Bearing, Missile Status} |
| **Functions** | |
| **δext (s, e, x = Missile Status)** {  if (Distance Flown < Total Flight Path Distance) {  if (Missile Status = hit) {  Phase = HIT  Sigma = ∞  } else {  Phase = FLYING  Sigma = 1 second  }  }  }  **δint (s)** {  Calculate updated Aircraft Position  Calculate Distance Flown  if (Distance Flown > Total Flight Path Distance) {  Phase = MISSION COMPLETED  Sigma = ∞  } else {  Phase = FLYING  Sigma = 1 second  }  }  **λ(s)** {  if (Phase = FLYING) {  Output Updated Aircraft Position  }  } | |

Figure 4: Aircraft atomic model formal specification and pseudo-code.

### Radar Atomic Model

The radar atomic model has two phases: *SEARCHING* and *TRACKING*. Once the radar detects an aircraft within a predefined range, the radar model starts outputting an updated aircraft position every second. The radar model reverts to *SEARCHING* once the aircraft is no longer within the radar range. Figure 5 provides the radar atomic model formal specification and pseudo-code.

|  |  |
| --- | --- |
| **Radar Atomic Model = <X, Y, S, δext, δint, λ, ta>** | |
| **State Variables** | **Formal Specification** |
| Phase = SEARCHING  Sigma = ∞  Aircraft Position  Detected Position  Radar Position  Radar Range  Distance Radar-Aircraft | X = {Aircraft Position}  Y = {Detected Position}  S = {Sigma, Phase, Aircraft Position, Detected Position, Radar Position, Radar Range, Distance Radar-Aircraft} |
| **Functions** | |
| **δext (s, e, x = Aircraft Position)** {  Calculate Distance Radar Aircraft  if (Distance Radar Aircraft ≤ 20 km) {  Phase = TRACKING  Detected Position = Aircraft Position  Sigma = 1 seconds  } else {  Phase =SEARCHING  Sigma = ∞  }  }  **δint (s)** {  case Phase  DETECTED:  Phase = SEARCHING  Sigma = ∞  }  **λ(s)** {  if (Phase = TRACKING) {  Output Detected Position  }  } | |

Figure 5: Radar atomic model formal specification and pseudo-code.

### C2 Atomic Model

The C2 atomic model serves as a central point of coordination of the air defence system. It receives information from radar, gunner and missile atomic models. The C2 model has four phases: *WAITING*, *DECISION MAKING*, *ORDERING*, and *STANDING DOWN*. The C2 remains in the *WAITING* state until it receives an aircraft position from a radar. The C2 is now deciding on whether to order one of the gunners to launch a missile. This decision is made based on the received gunners' status. Figure 6 provides the C2 atomic model formal specification and pseudo-code.

|  |  |
| --- | --- |
| **C2 Atomic Model = <X, Y, S, δext, δint, λ, ta>** | |
| **State Variables** | **Formal Specification** |
| Phase = WAITING  Sigma = ∞  Aircraft Position  Detected Position  Gunner Position  Gunner ID  Gunner Range  C2 Position  Distance Gunner-Aircraft | X = {Detected Position, Missile Status, Gunner Status}  Y = {Order}  S = {Phase, Sigma, Aircraft Position, Detected Position, Gunner Position, Gunner ID, Gunner Range, C2 Position, Distance Gunner-Aircraft} |
| **Functions** | |
| **δext (s, e, x = Detected Position, Missile Status, Gunner Status)** {  if (x = Gunner Status) {  if (Phase = DECISION MAKING) {  get gunner ID and gunner position  get distance between gunner and aircraft  if (distance gunner-aircraft < gunner range) {  Phase = ORDERING  Sigma = 0  }  }  }  if (x = Missile Status) {  if (Missile Status is no hit) {  Phase = WAITING  Sigma = ∞  } else {  Phase = STANDING DOWN  Sigma = ∞  }  }  if (x = Detected Position) {  get aircraft position  if (Phase = WAITING) {  Phase = DECISION MAKING  Sigma = ∞  }  }  }  **δint (s)** {  case Phase  ORDERING:  Phase = WAITING  Sigma = ∞  }  **λ(s)** {  if (Phase = ORDERING) {  Output order to gunner  }  } | |

Figure 6: C2 atomic model formal specification and pseudo-code.

### MANPADS Gunner Atomic Model

The gunner atomic model consists of five phases: *READY, REPORTING, LAUNCHING, LOADING,* and *UNAVAILABLE*. The gunner outputs a status to the C2 model every 5 seconds, reporting its availability on launching a missile. When the gunner receives an order from C2, it switches to *LAUNCHING* and then generates an output to the missile model. The gunner then starts *LOADING* another missile for 40 seconds. Once the gunner has finished loading a missile, it switches to *READY* and starts *REPORTING* its status to the C2 model every 5 seconds. The gunner has a limited number of missiles and becomes *UNAVAILABLE* once all missiles have been launched. Figure 7 provides the MANPADS gunner atomic model formal specification and pseudo-code.

|  |  |
| --- | --- |
| **MANPADS Gunner Atomic Model = <X, Y, S, δext, δint, λ, ta>** | |
| **State Variables** | **Formal Specification** |
| Phase = READY  Sigma = 5 seconds  Gunner ID  Gunner Position  Gunner Range  Gunner Missile Number  Missile Reload Time = 40 seconds  Distance Gunner Aircraft | X = {Order}  Y = {Gunner Status, Distance Gunner-Aircraft}  S = {Phase, Sigma, Gunner ID, Gunner Position, Gunner Range, Gunner Missile Number, Missile Reload Time, Distance Gunner-Aircraft} |
| **Functions** | |
| **δext (s, e, x = Order)** {  if (Phase = READY or Phase = REPORTING) and (GunnerID = Order Gunner ID) {  Phase = LAUNCHING  Missile Number = Missile Number – 1  Sigma = 0  }  **δint (s)** {  case Phase  LAUNCHING:  if (Missile Number > 0) {  Phase = LOADING  Sigma = Missile Reload Time  } else {  Phase = UNAVAILABLE  Sigma = ∞  LOADING:  Phase = READY  Sigma = 5 seconds  READY:  Phase = REPORTING  Sigma = 0  REPORTING:  Phase = READY  Sigma = 5 seconds  }  **λ(s)** {  if Phase = LAUNCHING {  Output Distance Gunner-Aircraft  }  If Phase = REPORTING {  Output Gunner Status  }  } | |

Figure 7: MANPADS gunner atomic model formal specification and pseudo-code.

### Missile Atomic Model

The missile atomic model has two phases: *WAITING* and *LAUNCHED*. The missile receives the distance between the gunner and the aircraft from the MANPADS gunner's output and determines the missile status (i.e., hit or miss). The probability of hit depends on the distance between the missile and the aircraft at the missile launch. The missile status is relayed to the aircraft and C2 models. Figure 8 provides the missile atomic model formal specification and pseudo-code.

|  |  |
| --- | --- |
| **Missile Atomic Model = <X, Y, S, δext, δint, λ, ta>** | |
| **State Variables** | **Formal Specification** |
| Phase = WAITING  Sigma = ∞  Missile Position  Missile Status  Flying Time  Probability of Hit (POH) by missile range  Gunner ID  Missile Range  Distance Missile-Aircraft  Distance Gunner-Aircraft | X = {Distance Gunner-Aircraft}  Y = {Missile Status}  S = {Phase, Sigma, Missile Position, Missile Status, Flying Time, Probability of Hit, Gunner ID, Missile Range, Distance Gunner-Aircraft, Distance Missile-Aircraft} |
| **Functions** | |
| **δext (s, e, x = Distance Gunner-Aircraft)** {  get GunnerID  get Missile Position  if (Phase = WAITING) {  get Distance Gunner Aircraft  if (Distance < Missile Range) {  Phase = LAUNCHED  if (Distance ≥ 4 km and ≤ 6 km) {  Missile Status = hit or not hit based on POH  Missile Flying Time = 7 seconds  Sigma = Missile Flying Time  } else if (Distance ≥ 1 km and < 4 km) {  Missile Status = hit or not hit based on POH  Missile Flying Time = 5 seconds  Sigma = Missile Flying Time  } else {  Missile Status = hit or not hit based on POH  Missile Flying Time = 5 seconds  Sigma = Missile Flying Time  } else {  Phase = WAITING;  Sigma = ∞;  } else {  Sigma = Missile Flying Time – e  Missile Flying Time = Sigma  }  }  **δint (s)** {  case Phase  LAUNCHED:  Phase = WAITING  Sigma = ∞  }  **λ(s)** {  if Phase = LAUNCHED {  Output Missile Status  }  } | |

Figure 8: Missile atomic model formal specification and pseudo-code.

# IMPLEMENTATION

The main design criteria for implementing a process using STK to visualize DEVS outputs are limited user inputs and reduced dependence on other third-party software. These design criteria are compatible with the general intent to increase DEVS modelling and simulation to a broader public. This section describes the main implementation activities leading to STK's use as a practical option to visualize DEVS outputs.

## STK Connect Module

STK offers a multitude of technologies to control and automate STK from external applications. Without using STK's graphical unit interface, the STK programming interface provides the tools to create a scenario, run simulations and perform analytical analysis. One of these options to communicate with STK is through the STK Connect module, which provides an easy way to connect with STK. This module contains an extensive library allowing the user to define and send commands to STK. Considering the design criteria, the chosen environment to communicate with STK via the Connect module is the use of a simple program called AgIPCExp. This program, which comes installed with STK, communicates with STK via a TCP socket. When STK runs, STK opens a socket connection at port 5001 by default, which "listens" for Connect commands. AgIPCExp allows a user to send individual commands to STK, as shown in Figure 9. In this example, the user provided one command to create a new STK scenario named "Demo" and one command to create an STK *Aircraft* object named "Aircraft1". This utility also allows executing multiple Connect commands that are stored in a text file, as shown in Figure 10.

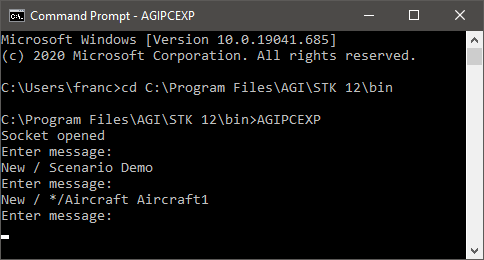


Figure 9: AgIPCExp using individual Connect commands.

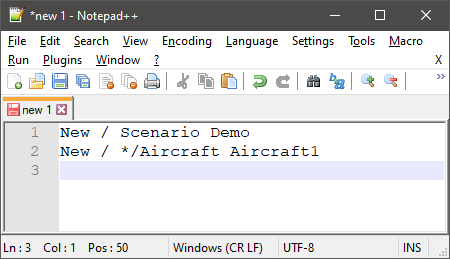


Figure 10: Text file example with one Connect command per line.

Prior to building a series of Connect commands to create an STK scenario based on the Cadmium simulation results, each DEVS model's component and output must be defined with corresponding STK objects or features. Table 1 provides the selected corresponding STK objects and features for the air defence system DEVS model.

Table 1: Corresponding STK objects and features for each DEVS model component.

|  |  |  |
| --- | --- | --- |
| **DEVS Model Component** | **Use Case** | **STK Objects / Features Correspondence** |
| Atomic model (stationary) | Radar  C2  Gunner1 / Gunner 2 / Gunner 3  Missile1 / Missile2 / Missile3 | One scenario object *Facility*  per atomic model instances |
| Atomic model  (moving) | Aircraft | One scenario object *Aircraft*  per atomic model instances |
| Range limitation | Radar  Gunner1 / Gunner 2 / Gunner 3 | One attached object *Sensor*  per atomic model instances |
| Input ports | All atomic models | One attached object *Receiver*  per atomic model instances |
| Output ports | Aircraft – outAircraftPosition  Radar – outDetectedPosition  C2 – outOrder  Gunner1 – outDistanceGunnerAircraft  Gunner1 – outGunnerStatus  Gunner2 – outDistanceGunnerAircraft  Gunner2 – outGunnerStatus  Gunner3 – outDistanceGunnerAircraft  Gunner3 – outGunnerStatus  Missile1 – outMissileStatus  Missile2 – outMissileStatus  Missile3 – outMissileStatus | One attached object *Transmitter*  per output port of the atomic model instances |
| Couplings | All connections between input and output ports shown in Figure 3. | STK *Compute Access* features |

Based on this correspondence table, a series of Connect commands were defined and tested individually into STK to verify their applicability to the DEVS model. The main STK Connect commands required to visualize the DEVS model outputs into STK are described in Figure 11. These commands allow generating all atomic models, input ports, output ports, and couplings. It also includes commands for scenario setup and visualization.

A critical feature of STK to visualize the DEVS outputs is the *Compute Access* feature. *Compute Access*, as previously explained, calculates the time intervals that an STK object can "see" another STK object based on the objects' constraints. This feature is used to visualize the DEVS outputs by computing access between *Transmitters* (output ports) and *Receivers* (input ports) based on temporal constraints. The temporal constraints are defined as time intervals corresponding to the simulation time of each DEVS output of a specific output port. Access between an output port and the corresponding input port is only possible at the time of a DEVS output for the output port. Line 19 of Figure 11 provides an example of a Connect command allowing such calculation for a specific coupling defined in the DEVS model.

|  |
| --- |
| **Scenario Setup**  Creation of a new scenario:   1. New / Scenario Demo   Setting up the scenario epoch:   1. SetEpoch \* "15 Dec 2020 00:00:00.00"   Setting up the scenario analysis time period:   1. SetAnalysisTimePeriod \* "15 Dec 2020 00:00:00.00" "15 Dec 2020 00:14:00.000"   Setting up the animation start time and time step:   1. SetAnimation \* StartAndCurrentTime UseAnalysisStartTime TimeStep 1   **Creation of an STK object representing a moving atomic model instance (e.g., aircraft atomic model)**  Insertion of a new STK object *Aircraft*  representing an aircraft atomic model instance (e.g., *aircraft*):   1. New / \*/Aircraft aircraft   Selection of the desired 3D model to represent the aircraft:   1. VO \*/Aircraft/aircraft Model File "helicopter.mdl"   Insertion of an STK attached object *Receiver*  named "in\_port" to the *Aircraft* object to represent input ports:   1. New / \*/Aircraft/aircraft/Receiver in\_port   Creation of a waypoint each time the aircraft atomic model generates an output:   1. AddWaypoint \*/Aircraft/aircraft DetVelFromTime -31.30851058 136.44974 300 "15 Dec 2020 00:00:01.000"   **Creation of an STK object representing a stationary atomic model instance (e.g., gunner atomic model)**  Insertion of a new STK object *Facility*  representing a gunner atomic model instance (e.g., *gunner1*):   1. New / \*/Facility gunner1   Insertion of an STK attached object *Receiver*  named "in\_port" to the gunner object to represent input ports:   1. New / \*/Facility/gunner1/Receiver in\_port   Setting off the line-of-sight parameter of the STK attached object *Receiver* acting as input ports to ensure that couplings between input ports and output ports will not be affected by STK terrain data (i.e., terrain obstruction):   1. SetConstraint \*/Facility/gunner1/Receiver/in\_port LineOfSight Off   Setting the position of the gunner and setting the gunner's altitude at ground level using STK terrain data:   1. SetPosition \*/Facility/gunner1 Geodetic -29.9181 135.907 Terrain   Insertion of an STK attached object *Transmitter*  to the gunner STK object to represent a specific output port (e.g. *outDistanceGunnerAircraft*):   1. New / \*/Facility/gunner1/Transmitter outDistanceGunnerAircraft   Setting off the line-of-sight parameter of the STK attached object *Receiver* acting as input ports to ensure that couplings between input ports and output ports will not be affected by STK terrain data (i.e., terrain obstruction):   1. SetConstraint \*/Facility/gunner1/Transmitter/outDistanceGunnerAircraft LineOfSight Off   Insertion of a time interval as a temporal constraint for the *Transmitter* acting as an output port. Each time there is an associated DEVS output for the *Transmitter*, a time interval is added:   1. SetConstraint \*/Facility/gunner1/Transmitter/outDistanceGunnerAircraft Intervals Include SetIntervals Add 1 "19 Nov 2020 00:00:05.000" "19 Nov 2020 00:00:05.999"   To visualize the maximum range of an atomic model instance, an attached object *Sensor*  must be added and configure as a simple cone of 90 degrees with a maximum range using these commands:   1. New / \*/Facility/gunner1/Sensor range 2. Define \*/Facility/gunner1/Sensor/range SimpleCone 90.0 3. SetConstraint \*/Facility/gunner1/Sensor/range Range Max 6000   **Compute access for one DEVS coupling (between a *Transmitter* (output port) and a *Receiver* (input port))**   1. Access \*/Facility/c2/Transmitter/outOrder \*/Facility/gunner1/Receiver/in\_port AutoAddTimeline on |

Figure 11: Main used STK Connect commands.

## DEVS Outputs Requirements

Cadmium (Ruiz and Wainer 2020) provides the required services to create logs of the simulation. It generates two logs: one contains the messages generated by each atomic model's output ports, and the other one contains the atomic model's states. For this project, only the messages generated by the output ports are required. Figure 12 shows the main elements of a Cadmium message log, which consists of the simulation time followed by the messages generated by each atomic model on each port at that simulation time. The message defined within the curly brackets is specific to each atomic model. In this example, "Aircraft\_defs::out" represents the output port "out" of the atomic model "Aircraft" and "aircraft" represents an instance of the "Aircraft" atomic model.

|  |
| --- |
| 00:00:00:000  [Aircraft\_defs::out: { *message* }] generated by model aircraft  00:00:01:000  [Aircraft\_defs::out: { *message* }] generated by model aircraft  … |

Figure 12: Cadmium message log example.

The air defence system DEVS model contains a different message type for each atomic model. Limited requirements are necessary for the message structure to ensure that the DEVS outputs will provide sufficient information to STK. For all atomic models, the messages' structure must contain the model instance latitude and longitude (e.g. <-31.0336,136.462>). For atomic models with changing positions, such as the aircraft atomic model, the position must also include the altitude (e.g. <-31.0336,136.462,300>). For atomic models where a range limitation is used, such as the radar and gunner atomic models, the message must contain the range limitation to visualize the range cone in STK. In addition to the position and range information, the simulation time, the output port name and the atomic model instance name are also used to generate the STK Connect commands.

## Converter Application

A converter application was developed to convert the Cadmium DEVS output messages into STK Connect commands. The objective of this converter is to provide all required Connect commands to autogenerate an STK scenario with limited user inputs. The application loops through the text file containing the Cadmium simulation results and generates a Connect command for each DEVS output. If the DEVS output is generated by the aircraft atomic model, a Connect command such as line 8 of Figure 11 will be created to add a waypoint to the STK *Aircraft* object. Otherwise, if the DEVS outputs are generated by either a radar, C2, gunner or missile atomic model instance, a Connect command such as line 15 of Figure 11 will be created to set an access time for the associated output port.

As Cadmium results do not provide information about coupling details between the output ports and input ports, the user must complete the text file containing the STK Connect commands by adding the command lines for each DEVS coupling as shown in line 19 of Figure 11. For any experiments using the same DEVS model, these access command lines remain the same. Therefore, the user can reuse these command lines from one experiment to another. Once executed, the converter application generates a text file with one STK Connect command per line. With STK already running, the user can now execute AgIPCExp with the following command line: AGIPCEXP -t localhost:5001 < absolute\_path\_of\_text\_file. Figure 13 provides an example of Cadmium DEVS outputs for the air defence system model at time 00:01:37.000. At this specific time, the aircraft generated an updated aircraft position, and the radar also generated an output since the aircraft is within the radar's detection range. Using the converter application, these results are converted to the STK Connect commands provided in Figure 14.

|  |
| --- |
| 00:01:37:000  [Aircraft\_defs::outAircraftPosition: {Aircraft position (lat, long, altitude): <-31.25672591,136.44974,300>}] generated by model aircraft  [Radar\_defs::outDetectedPosition: {Radar range: 20000 m | Radar position (lat, long): <-31.0787,136.473> | Detected aircraft position (lat, long): -31.25726533 , 136.44974}] generated by model radar |

Figure 13: Cadmium message log at time 00:01:37:000.

|  |
| --- |
| AddWaypoint \*/Aircraft/aircraft DetVelFromTime -31.25672591 136.44974 300 "15 Dec 2020 00:01:37.000"  SetConstraint \*/Facility/radar/Transmitter/outDetectedPosition Intervals Include SetIntervals Add 1 "15 Dec 2020 00:01:37.000" "15 Dec 2020 00:01:37.999" |

Figure 14: STK Connect commands generated for Cadmium outputs at time 00:01:37.000.

# RESULTS

In this section, an experiment using the air defence system model is first introduced. Then, the experiment's simulation results are used to demonstrate STK's ability to visualize Cadmium DEVS outputs.

## Experiment

Users of the air defence system DEVS model can design a specific experiment by modifying any of the scenario parameters shown in the first column of Table 2. This table also lists the chosen parameter values for the engagement scenario, which will be executed into Cadmium and then visualized into STK. In this scenario, as illustrated in Figure 15, the aircraft flies a predefined straight flight path from an initial position to a final position at a constant speed of 60 m/s and a constant altitude of 300 m. This flight path causes the aircraft to fly through the radar's detection range and each gunner's missile range. The radar detection range and missile range are set to 20 and 6 km, respectively. Each gunner has four missiles in their possession. The probability of a successful hit at any missile launch range is set to 0 to ensure the aircraft will fly its full path without being hit. Note that the gunners' position and their associated missile should be the same; however, to properly visualize the STK results, the user should use coordinates of a few hundreds of meters apart between gunners and their associated missiles. These scenario parameter values were explicitly chosen to demonstrate STK's potential at visualizing DEVS outputs and may not be an accurate representation of a real air defence scenario.

Table 2: Engagement scenario parameters.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Scenario Value** | **Units** |
| Aircraft Initial Position | -31.30905, 136.44974 | Decimal degree |
| Aircraft Final Position | -30.85932, 136.44974 | Decimal degree |
| Aircraft Speed | 60 | Meter per second |
| Aircraft Altitude | 300 | Meter |
| Radar Position | -31.07870, 136.47326 | Decimal degree |
| Radar Range | 20000 | Meter |
| C2 Position | -31.05130, 136.46076 | Decimal degree |
| Gunner #1 Position | -31.05007, 136.42800 | Decimal degree |
| Gunner #1 Number of Missiles | 4 | Unit |
| Missile #1 Position | -31.0504, 136.428 | Decimal degree |
| Gunner #2 Position | -31.03852, 136.44373 | Decimal degree |
| Gunner #2 Number of Missiles | 4 | Unit |
| Missile #2 Position | -31.03830, 136.44394 | Decimal degree |
| Gunner #3 Position | -31.03363, 136.46193 | Decimal degree |
| Gunner #3 Number of Missiles | 4 | Unit |
| Missile #3 Position | -31.03361, 136.46159 | Decimal degree |
| Probability of hit for a missile launch range between 4 and 6 km | 0 | Percentage |
| Probability of hit for a missile launch range between 1 and 4 km | 0 | Percentage |
| Probability of hit for a missile launch range less than 1 km | 0 | Percentage |

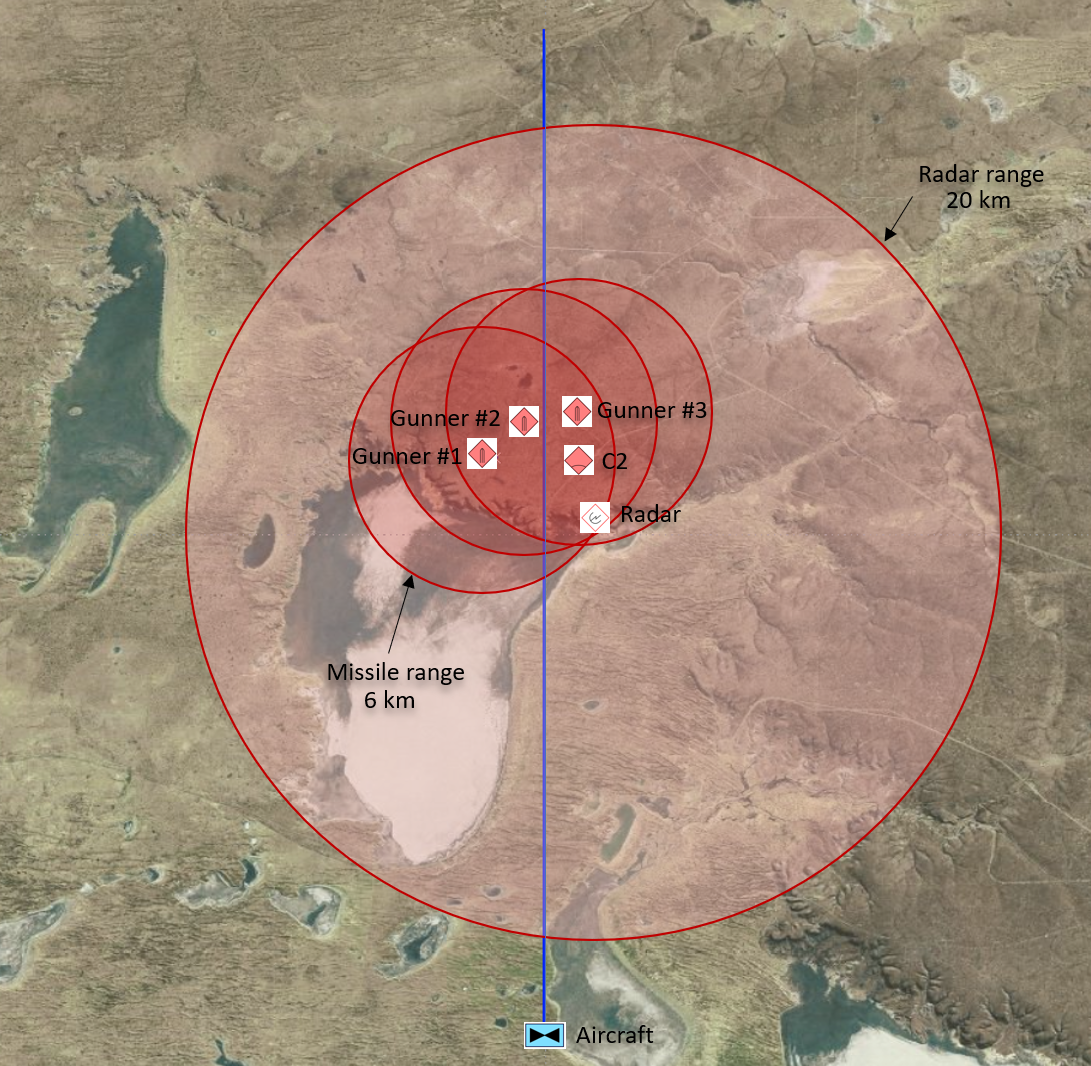


Figure 15: Engagement scenario setup.

## STK Results

After running the simulation for the selected engagement scenario with Cadmium, the converter application was used to convert the Cadmium simulation results into STK Connect commands. With STK running, the AgIPCExp application was then used to execute the Connect commands stored in the text file. The resulting STK scenario is shown in Figure 16. Box 1 lists all expected STK objects and attached objects, as described in Table 1. Box 2 and Box 3 are the default 2D and 3D graphic windows where the user can move around the scenario either by zooming directly to a scenario object or using the mouse controls to zoom and pan around the graphic windows. The timeline view in Box 4 displays the access times for each coupling defined in the DEVS model.

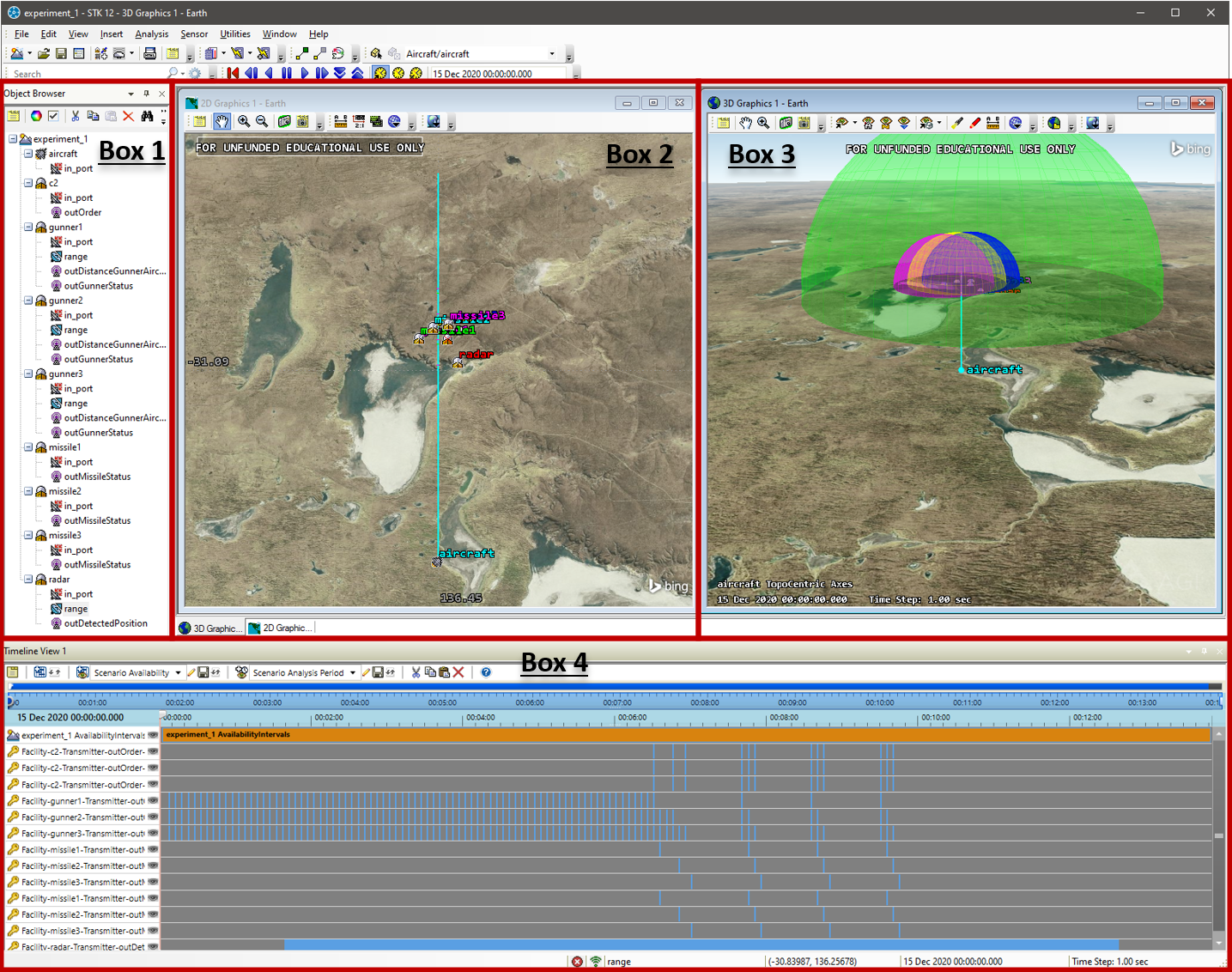


Figure 16: Generated STK scenario.

Figure 17 and Figure 18 are snapshots of the STK 2D and 3D graphic windows at simulation time 00:06:25. At this simulation time, the aircraft is within the radar's detection range but still outside the missile range. In addition to the timeline view, DEVS outputs are also displayed in the 2D and 3D graphic windows by lines between STK objects. The colour of the lines corresponds to the STK object from which the DEVS output is generated. For example, in Figure 17 and Figure 18, we observe a red line between the radar and the C2 objects. This line represents a radar output indicating that the aircraft is within the radar's detection range. The blue, yellow, and pink lines represent the gunners' outputs, providing a status to the C2 atomic model. We can also observe along the aircraft's flight path, in the 2D graphic window, some coloured dashes representing access time between the missiles and the aircraft. During these times, the missile atomic model instances outputted a missile status (i.e., miss) to the aircraft as designed in the DEVS model.

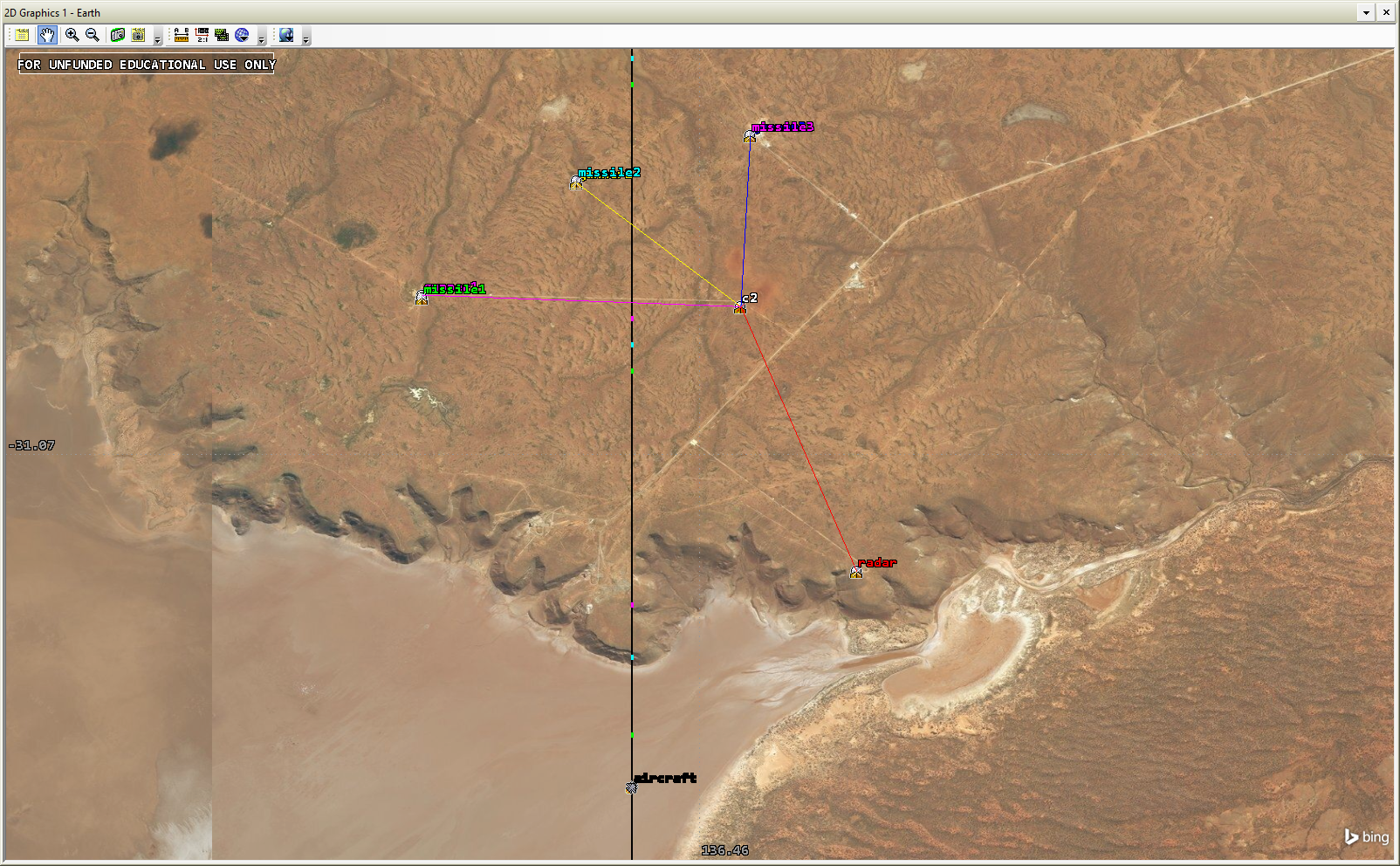


Figure 17: STK 2D graphic window at simulation time 00:06:25.

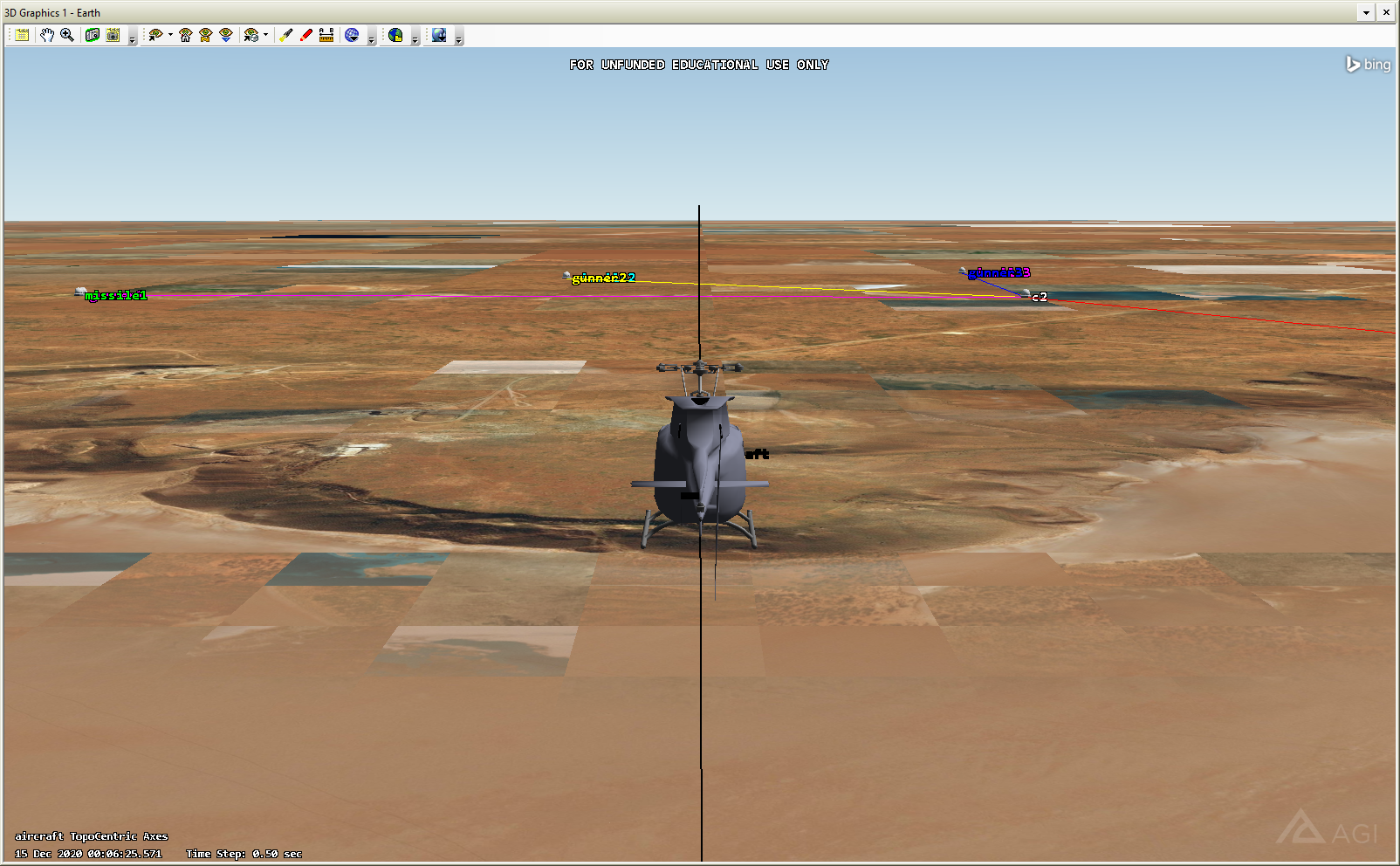


Figure 18: STK 3D graphic window at simulation time 00:06:25.

Once AgIPCExp has finished loading data into the STK scenario, the user can make a few adjustments to ensure proper visualization of the DEVS outputs. The user should consider removing the terrain elevation data in the scenario settings to properly visualize the lines between objects in the 3D graphic window. The user can also adjust any 2D and 3D graphic parameters, such as the objects' colour, to improve the visualization. All these desired adjustments made to the STK scenario can likely be generated automatically by adding Connect commands to the text file generated by the converter application. The Connect library has an extensive list of commands that control an object's 2D and 3D graphic properties.

Overall, the resulting STK scenario shows a promising way to visualize DEVS outputs for complex systems. During the DEVS model's development and testing, the STK timeline view was found particularly efficient at rapidly highlighting errors in the model's behaviour.

# CONCLUSIONS AND FUTURE WORK

This explorative study showed STK's potential at visualizing Cadmium DEVS simulation results. With minimal requirements to the Cadmium output messages defined for each atomic model and a simple converter application, it is possible to build a set of STK Connect commands to autogenerate an STK scenario to visualize DEVS outputs. The STK 2D and 3D graphic windows and the timeline view provide users with various ways to visualize DEVS outputs. The use of STK proved to be useful in verifying that the model behaves as expected and gaining insights into the simulation results. This proof of concept was demonstrated using a simplified air defence system scenario. However, any complex systems using geographic locations modelled with DEVS could be visualized with STK. From a defence perspective, any military scenarios including air, sea, land, space and cyber assets modelled using DEVS formalism could be visualized into STK.

Future work could include further investigation of other means to communicate with STK and other STK objects and features to represent DEVS models' elements. STK is an extensive software, and not all options may have been considered. Some other ways to connect with STK may provide more efficient processing. Other alternatives to STK, which does not require a license, could also be investigated to provide other options to visualize DEVS outputs similarly. Another known software within the defence community that may provide an adequate 3D visualization software is SIMDIS. SIMDIS (SIMDIS 2020) is a 2D and 3D visualization and analysis tool developed by the United States Naval Research Laboratory.

# REFERENCES

Analytical Graphics Inc. 2020. "Systems Tool Kit (STK)." Accessed Dec. 16, 2020. <https://www.agi.com/products/stk>.

Analytical Graphics Inc. 2020. "STK Level 1 and Level 2 Training Manual." STK Version 12.1. Accessed Dec. 16, 2020. <https://help.agi.com/stk/LinkedDocuments/Printed%20Manual%20-%20Comprehensive.pdf>.

Belloli, L., D. Vicino, C. Ruiz-Martin and G. Wainer. 2019. "Building Devs Models with the Cadmium Tool." In *2019 Winter Simulation Conference (WSC)*, National Harbor, MD, USA, 2019, pp. 45-59.

Rheinmetall Defence. 2020. "Stationary Air Defence." Accessed Dec. 16, 2020. <https://www.rheinmetall-defence.com/en/rheinmetall_defence/systems_and_products/air_defence_systems/stationary_air_defence/index.php> .

Ruiz Martin, C., G. Wainer. 2020. *Cadmium – A tool for DEVS Modeling and Simulation – User's Guide*. Accessed Dec. 16, 2020. <http://www.sce.carleton.ca/courses/sysc-5104/lib/exe/fetch.php?media=cadmium.pdf>.

SIMDIS. 2020. "Welcome!" Accessed Dec. 16, 2020. <https://simdis.nrl.navy.mil/>

Zeigler, B.P., H. Praehofer and T. G. Kim. 2000. *Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. San Diego, CA: Academic Press.