AN UNMANNED AIR VEHICLE SIMULATOR IN C#

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Sudesh Mukhami

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

February 2013

Fargo, North Dakota

North Dakota State University
Graduate School

Title

AN UNMANNED AIR VEHICLE SIMULATOR IN C#

By

Sudesh Mukhami

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall Nygard
Chair

Simone Ludwig

Chau You

Approved:

02/26/2013	Dr. Brian Slator
Date	Department Chair

ABSTRACT

Simian is an agent based legacy simulator developed in Java used to develop scenarios involving unmanned air vehicles. The goal of this paper is to host the Simian simulator on a C# platform and also to extend the currently existing functionality using C# by adding more behaviors to the UAVs. The Java code is converted into C# code in order to run the simulator on a visual studio framework, but the structure and interfaces defined in the Java version have been retained in the new framework.

The scenarios added to the UAV include: 1. the ability to exhibit a swarm type behavior while performing a sweep search, 2. to detect a target in the hostile environment and rotate over the target, and 3. to search through the hostile environment and destroy a target when detected. All the behaviors have been demonstrated using multiple scenarios in the Simian simulator.

ACKNOWLEDGEMENTS

I would like to thank Dr. Kendall E. Nygard for his continued support, help and direction. Also, I would like to acknowledge the guidance given to me by others on Dr. Nygard's research team. My sincere thanks to Dr. Kendall Nygard, Simone Ludwig and Chau You for serving on my committee.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

An unmanned air vehicle is an aircraft that can be flown without a pilot on board. These vehicles are controlled/monitored by a pilot at a ground control station. These aircrafts travel into an environment with prior knowledge of the area and also with a specific set of tasks to perform while flying in the area of interest. The action of performing the given set of tasks is described as a mission, and the entire mission is described as a scenario where a UAV exhibits certain behaviors to perform a given task. In the initial days of UAV evolution, the pilot at the ground control station would control the mission with UAV sensor information being his main source of information.

The growing research in this area of interest gave way to new breed of UAVs which were more autonomous because they can be programmed to perform more complex and dynamic missions. There is a lot of flexibility to the mission when the vehicles are autonomous and discrete. UAVs can exhibit very interesting and complex behaviors when they are autonomous and have the freedom to perform dynamic tasks.

1.1. Objectives

1.1.1. Objective 1

The UAV simulator “Simian” has some basic behaviors required to perform simple, autonomous mission-based scenarios; this simulator is developed using Java. To add new, complex behaviors such as the Low Cost Autonomous Attack System (LOCAAS) requires a more flexible design which could be accomplished by hosting the simulators on a C# code base. This paper re-hosts the simulator on a C# code base by keeping the layer structure of the Java-based simulator. The new design makes the simulator more efficient with increased performance metrics and also makes it dynamic by eliminating the XML input system used by the Java version. The user interface (UI) enhancement using C#’s windows presentation foundation (WPF) technology would be an added advantage to demonstrate the new behaviors in a more robust fashion.

1.1.2. Objective 2

Existing behaviors are tested for their functionality on the enhanced version of the UAV simulator. Waypoint following and inter-agent communication are tested by adding those behaviors to create new scenarios.

1.1.3. Objective 3

This paper focuses on developing behaviors which can be easily extended or can be re-used in different scenarios. These behaviors give a new dimension to develop real-time scenarios which can be successfully implemented in real-time situations. For example, LOCAAS is the type of UAV designed to attack a target. These UAVs can self-destruct themselves on a target. To achieve the self-destruct mechanism, they are built light and have a specific set of features that help in achieving the mission of destroying the target [1]. Adding new behaviors to a UAV would be a great asset to any UAV simulator. Behaviors then can be added to the simulator:

1) Sweep search
2) Swarm formation
3) Collision detection
4) Target detection
5) Target destruction
6) Rotation motion
7) Communication while in rotation motion

2. THE UAV SIMULATOR

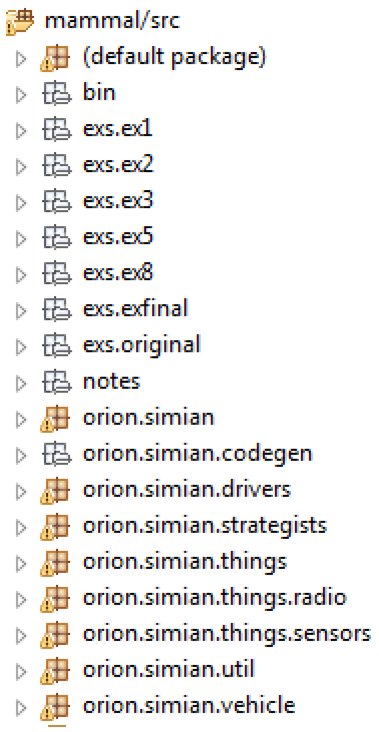
A UAV simulator is a tool on which real-time scenarios can be developed involving UAVs and target sets in an environment. Simian is such simulator where agents, such as UAVs, are given a set of behaviors which they exhibit while performing a task in a real-time scenario. The Simian simulator was built using an open-agent architecture in Java. The simulator is divided into layers which are standalone and layers which have the ability to communicate with each other. The focus is to develop scenarios involving low-cost, effective missions using unmanned air vehicles. The layered design supports the capability of inter-agent communication and also provides the flexibility of adding new behaviors to an agent [2].

2.1. Simian in Java

The Simian simulator was developed in Java using socket and thread programming when a post listener is required in order to run the application. A port number is entered as an argument to run the VIS2D class which communicates with the port and hosts the simulator successfully if it listens to the socket. After setting up the port, the DXML class is passed with an XML file that contains the metadata about the scenario that needs to be run. Each XML represents a unique scenario which exhibits a behavior (functionality) of a UAV. These behavioral classes are defined as abstract classes which can be used in subsequent subclasses which can inherit the properties of these classes, thereby extending the current functionality. The layers defined in the Java version of the Simian simulator cannot communicate complex commands like swarm behavior (defined in later chapters of this paper) which are required to have extended functionality added to the UAV's behaviors [3].

The Simian simulator exhibits the flight functionality of a UAV by allowing the UAV to fly from one coordinate to another; this behavior can be achieved using the UAV's behavior of motion. The coordinate and the path to be followed are communicated to the UAV by the environment classes. The UAV's physical model is developed using transmitters and sensors along with the basic UAV functionalities. The transmitters are used to communicate with the environment, and the sensors are used to scan for the targets in the search area defined by the environment. The Java framework uses threads to run multiple methods concurrently because there are different behaviors of a particular UAV at a given instance of time and as some scenarios require multiple UAVs flying at a single instance of time.

The class structure is encapsulated into packages, and each package is defined by the type of classes that are present. Figure1 is the package structure of Simian under the mammal/src.package.



```
mammal/src
  (default package)
  bin
  exs.ex1
  exs.ex2
  exs.ex3
  exs.ex5
  exs.ex8
  exs.exfinal
  exs.original
  notes
  orion.simian
  orion.simian.codegen
  orion.simian.drivers
  orion.simian.strategists
  orion.simian.things
  orion.simian.things.radio
  orion.simian.things.sensors
  orion.simian.util
  orion.simian.vehicle
```

Figure 1. Package structure of Simian in eclipse environment.

2.2. Simian Architecture

With the Simian simulator, the UAVs and targets are classified as agents. Both agents have a separate set of behaviors which results in different functional characteristics. The agents use the environment as a common communication medium. The Simian simulator uses open-agent architecture to establish communication between the UAVs and the environment, and also to allow integration between the agents. The environment is just one layer which facilitates communication between agents. Simian consists of four layers: strategist, autopilot, physical and the environment. UAV motion is encapsulated at a higher level which can be accessed by other layers through method calling or message passing. Figure 2 displays the top down layer structure of the Simian simulator and also how the communication happens between the layers.

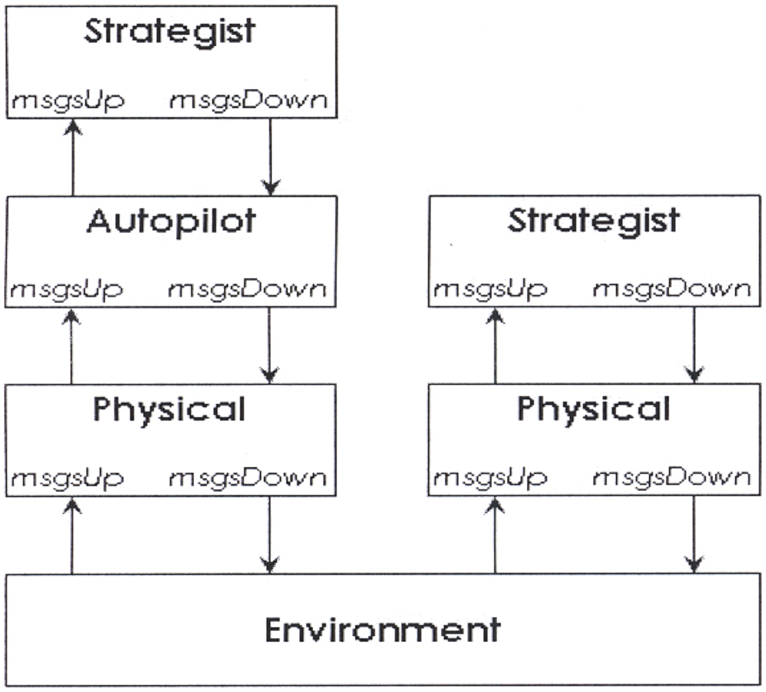


Figure 2. The layer structure of the Simian simulator [3].

The physical layer is responsible for the UAV shape, the sensors and the flight functionality. It also determines the target shape and functionality. The target can be static or dynamic. When the dynamic behavior is activated, the physical layer provides the target with motion functionality. If not, the target is static at a specific coordinate. The physical layer differentiates the hardware capabilities of the agents; for example, the UAV has a flying capability which is not available for the target.

The autopilot layer helps the UAV achieve flight motion with finesse because it assists the UAV with the direction and also with the other factors required for the flight. This layer makes sure that the UAV is flying at the correct altitude, at the correct speed and in the correct path, which are all vital for the mission to be successful.

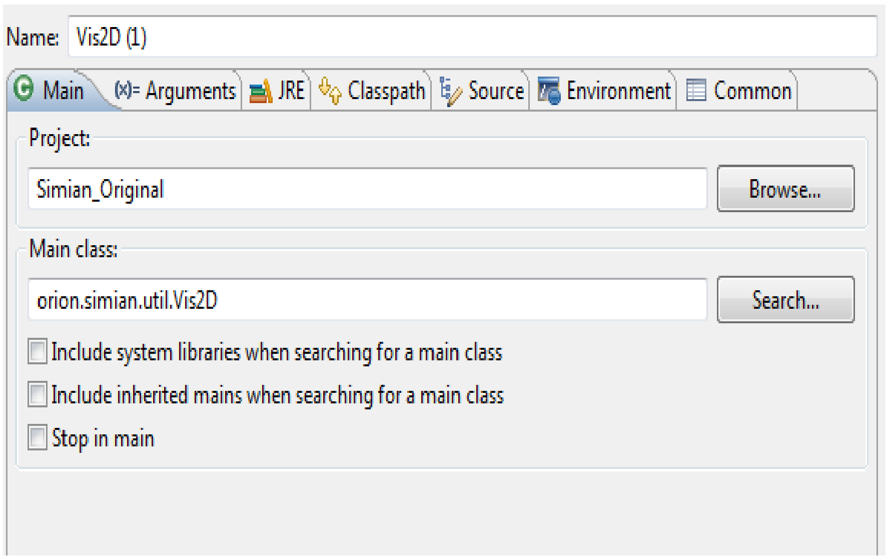
The deciding factors for the mission, such as the UAV position at a given instance of time, the speed and also the success of the mission, are evaluated by the strategist. The strategist layer has all the logic behind a mission which, in turn, communicates with the autopilot to start the mission. The autopilot gets the required physical attributes based on the mission provided by the strategist. The environment helps perform all the communication between the agents and layers.

2.3. Running the Simulator in Java

Simian uses XML files as input files for initialization to achieve this design goal. The four XML files used in Simian are situation, policy, and strategy and global. The Vis2d.Java classes in orion.Simian.util acts as the server. The DXML.Java clas in orion.Simian.drivers acts as the client. Running Simian on Eclipse requires the following steps.

2.3.1. Running the Server

To run the server, the arguments must be set for the Vis2D classes; the port number, 9902, is passed in the arguments tab of the open run dialog, and the server can be started by clicking the run button. Figures 3 and 4 are the screenshots for the main classes and to set the port number in the arguments.



Name: Vis2D (1)

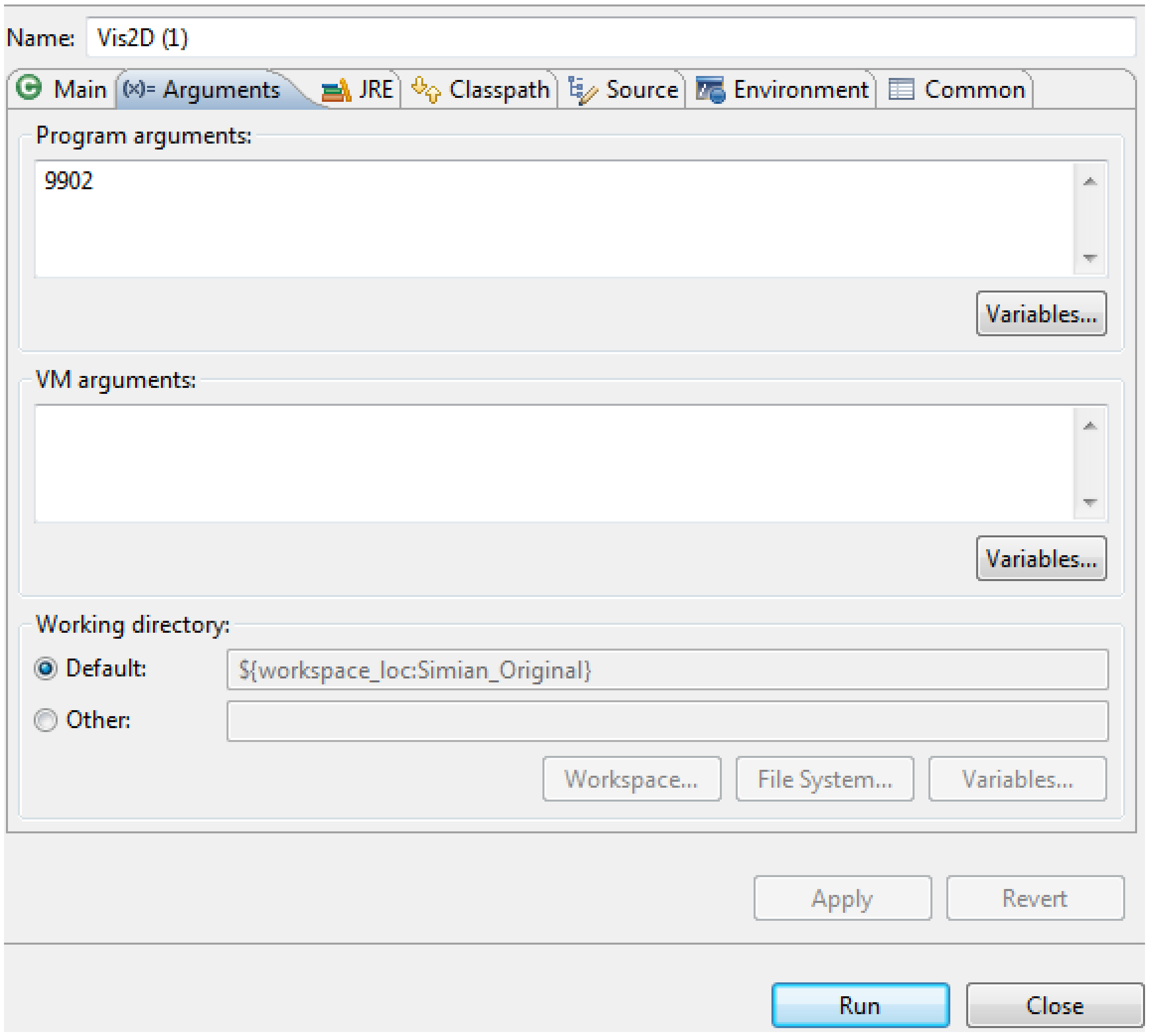
Project: Simian_Original

Main class: orion.simian.util.Vis2D

☐ Include system libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main

Figure 3. Vis2D is the main class.

Figure 5 displays the confirmation message returned by the server that the specified 9902 port is being used and also that the listener is initiated. This message confirms that the server has been started successfully.


Name: Vis2D (1)

Program arguments:

9902

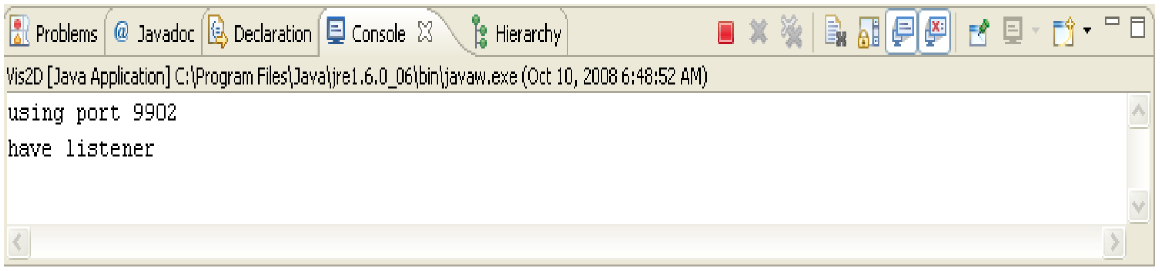
VM arguments:

Working directory:

Default: ${workspace_loc:Simian_Original}

Other:

Figure 4. Setting up the port number, 9902, to initiate a listener.



Vis2D [Java Application] C:\Program Files\Java\jre1.6.0_06\bin\javaw.exe (Oct 10, 2008 6:48:52 AM)

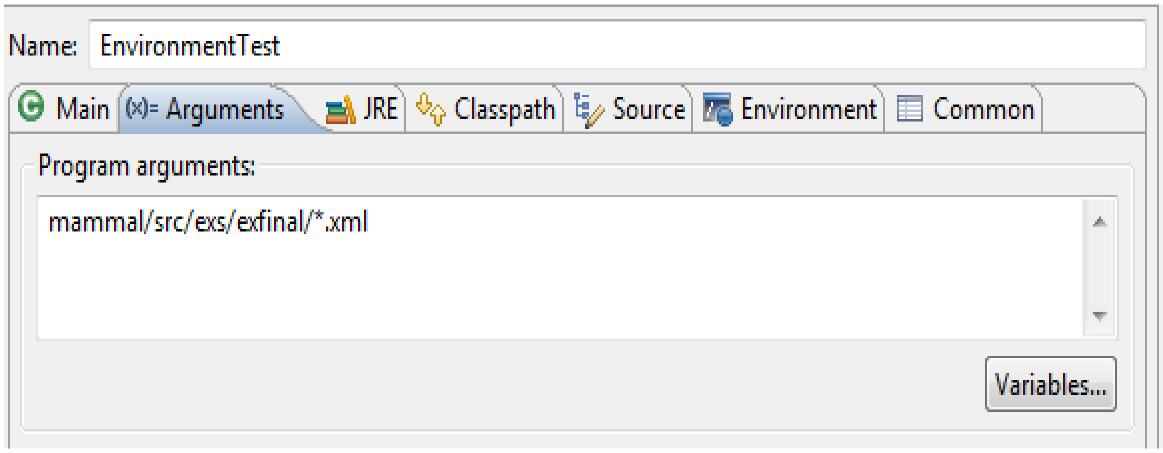
```
using port 9902
have listener
```

Figure 5. Server using port 9902.

2.3.2. Running the Client

The XML file needed to be run is passed in an argument to the DXML classes which acts as a client for the simulator. The run button is clicked to kick start the client. Figure 6 illustrates how to run the client by passing the required arguments.

.


Name: EnvironmentTest

Program arguments:

```
mammal/src/exs/exfinal/*.xml
```

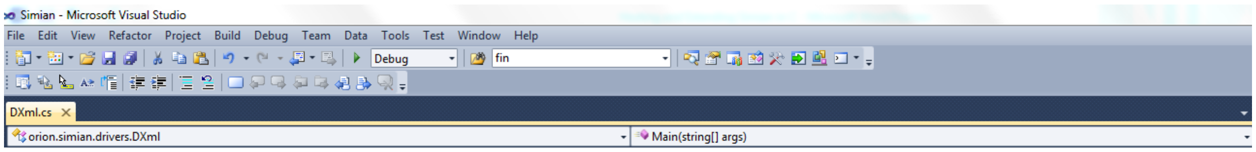
Figure 6. Passing the XML name along with its location as an argument.

2.4. Running the Simulator in C#

Unlike the Java version, the server which the simulator needs is already set up and ready to run. This process eliminates the extra step of setting up the server every time the simulator needs to be run.

2.4.1. Running the Client

The simulator can be started by either clicking the start debug button or pressing the F5 key. See Figure 7.


Simian - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Team Data Tools Test Window Help
Debug
fin
DXml.cs
orion.simian.drivers.DXml
Main(string[] args)
Figure 7. Running the client in C#.

The server port is set as soon as the simulator starts its run, and after the port is set, the interactive simulator is displayed. The interactive user interface is run as a part of running the client. Figure 8 is the new start screen.

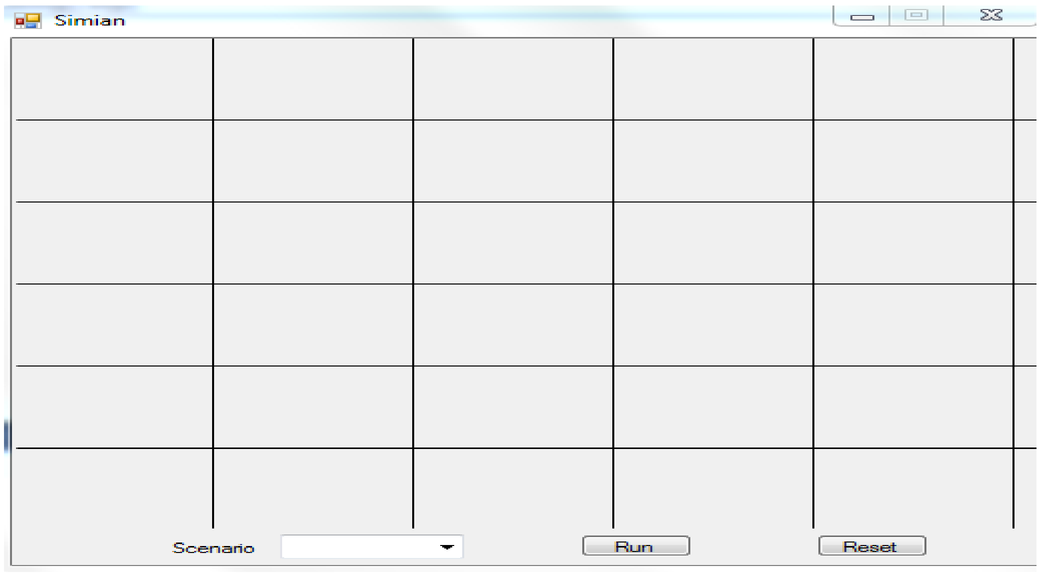


Figure 8. Start screen for the Simian simulator in C#.

3. HOSTING IN C#

C# is also an object-oriented programming language like Java, but C# is more developer friendly and is commonly known as a sugar-coated language. C# allows you to extend existing classes, even if they are declared final, without actually extending them. The rapid development environment provided by the visual studio cuts down the development time, helping to focus more time on designing the code. Operator overloading is more flexible when compared to Java which is used extensively in the Simian simulator.

3.1. Why C#

The list functionalities are more flexible and refined in C# when compared to Java. Because we define our agents and store them in a list, a much more flexible way of implementation was provided when migrating to C#. The messages between the agents are also exchanged using XML list which is well defined in C#. The XML list also prevents data leaking, which was found a lot while debugging the Java application. Also, the scenarios are run based on a dynamic structure where the C# functionality of supporting no-reflection dynamic invocation is very useful. By using the dynamic type, we can resolve the issues faced while accessing a member dynamically at run time.

As the layer structure from Java is still being honored even when hosting it on C#, the type-casting instance of dynamic types was handy because lot of methods and properties were invoked directly without having the trouble of using the reflection API. C# uses the lightweight thread mechanism which helps maintain a robust, concurrent simulator without a lot of thread overhead which can be developed from running multiple threads which access the same resources at a given instance of time. The Common Language Runtime (CLR) used by C# applications introduces the additional concept of “Application Domain,” allowing multiple programs to execute in a single hardware address space, but that does not affect how your program uses threads [4].

The user interface developed using C# programming is more robust and developer friendly. The flexible design functionality helps the developer achieve a given design in a shorter amount of time with more user friendliness. WPF and windows forms are two different approaches for developing the user interface using C# programming on a visual studio framework. The Simian simulator on C# uses the windows presentation foundation to develop the user interface. WPF also helps retain the XML-based scenarios which were developed in the Java version of the simulator [5].

The strong data binding provided by the WPF application would solidify communication between the strategist and the environment because it keeps the business logic and user interface separate. WPF enhances the UI for the Simian simulator with its set of unique and enhanced tools. The tools are also flexible, and it is easy to extend a given application without worrying about the infrastructure implementation. Modifying existing UI or shapes is flexible with WPF because no external editor is required and because all the components can be controlled by the code or data bounded so that the desired functionality can be depicted. The dynamic nature of the user interface is useful in Simian-like simulators because the scenarios are real-time executions using dynamic parameters [6].

An application developed using WPF does not need to be confined to a particular set of device metrics, for example, the application can be run on any screen resolution. Expression blend can be used to enhance the user interface of the application, and it can be stand alone. Changes to the UI will not affect the business logic code. Using WPF, we can modularize the interface very easily; the screen can be broken down to independent controls, thereby removing the code interdependency [7].

3.2. Structure in C#

The namespaces in the Simian architecture of the C# version are based on the package structure of Java version. The folder structure is divided to adhere to the layer structure of the Simian simulator. All the input XML files which contain the autopilot and scenario information are stored in an EXS folder. Different scenarios have different input parameters, so each scenario is defined by the XML file being passed as parameter to run the simulator.

The main method is declared in the DXML classes which is declared under the driver's namespace. The dxml classes takes the input parameters from the XML files and initializes the environment based on the parameters passed. The environment classes is declared under the top-tier structure of orion.Simian which takes the messages passed from the DXML classes. The messages are passed to the flight plan classes under the strategist orion.simain.strategist namespace which controls the scenario to be run. Figure 9 represents the namespace structure of a Simian project in C#.

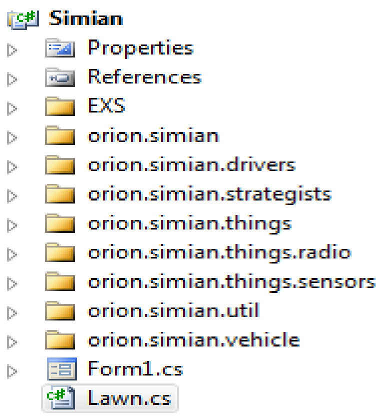


Figure 9. Namespace structure in C#.

The physical layer is stored under the orion.Simian.physical name space which contains the vehicle classes. The vehicle classes has the physical attributes to build the required agent so that behaviors can be added to perform the actions. Based on the values passed to the physical layer, the agent is built and added to the list of agents.

The autopilot layer contains all the sensors and other behaviors which are required to accomplish the required scenario. The behaviors are added to the physical attributes created for an agent in the physical layer. The sensors which pick up the targets and also communicate with the environment, are added functional classes in the autopilot layer. The altitude, speed and other factors which are passed via the XML file are initialized in this layer, and the autopilot helps the UAV with the flight path and other factors such as altitude adjustment.

The util namespace under orion.Simian.util contains the loislane classes which take the XML elements and hosts the port server. The pseudo code for the hosting the server from the loislane classes is shown in Figure 10.

The port shutdown method is declared in the loislane classes which is invoked after the scenario has been completed. The message passed from the upper layers is parsed as an XML element. Then, the UAV is visualized by the server, and is the UAV motion is stored in the list. Once the scenario ends, the message is picked by the report method, and the port server is shut down until further activation is requested. See Figure 11.

```
                    int port = 9902;
                    InetAddress host = InetAddress.LocalHost();
                    string hostStr = el.GetAttribute("host");
                    string portStr = el.GetAttribute("port");
                    if (null != hololstStr || null != portStr)
                    {
                        if (null == hostStr || "".Equals(hostStr))
                        {
                            hostStr = "127.0.0.1";
                        }
                        Console.WriteLine("LoisLane initializing from xml");
                        if (null != hostStr && !("".Equals(hostStr)))
                        {
                            host = InetAddress.getByName(hostStr);
                            Console.WriteLine("LoisLane: host=" + hostStr);
                        }
                        if (null != portStr && !("".Equals(portStr)))
                        {
                            port = Convert.ToInt32(portStr);
                            Console.WriteLine("LoisLane: port=" + port);
                        }

                        Console.WriteLine("socketry");
                        sock = new TcpListener(host.addr, port);
                        Console.WriteLine("have socketed");
          NetworkStream temp = new NetworkStream(sock.Server);
                        fout = new StreamWriter (temp);

                        feedback = Xml.getBooleanAttribute(el, "feedback", false);
                        Console.WriteLine("LL: feedback=" + feedback);
                        if (feedback)
                        {
                            fin = new NetworkStream(sock.Server);
                        } // feedback
```
Figure 10. This piece of code is to host the 9902 port server.

```
else if (msg is DeathMsg)
    {
        visualize("DEAD " + ((DeathMsg)msg).deceased);
    }
```
Figure 11. This piece of code closes the server after the scenario ends.

4. AGENT BEHAVIORS

Unmanned air vehicles have been a large area of research with growing interest in that area of expertise. The factors for the boom in unmanned air vehicles have been the cost, accuracy and ease of use. More intelligence, surveillance, and reconnaissance’s (ISR) missions have been used more successfully because of UAVs. The surveillance capacity of a UAV can be stretched to use based on the mission that is being executed. With the help of intelligence and reconnaissance, any UAV mission can be described as an ISR mission.

The flight capacity of the UAV can be enhanced based on the terrain or hostile environment where the mission is being performed. Repetitive surveillance missions are very useful in collecting important daily-basis data intercepts. The environment can be any extent of a dirty area a UAV can be configured for these areas, and because the building costs are very affordable, the base UAV models can be designed for the mission requirements. UAVs can be differentiated based on their capabilities, functionalities, size, etc. Different tires of UAVs can be short- range UAVs, tactical UAVs, medium-altitude lone endurance (MALE) UAVs, Etc. [8].

Endurance is one of the biggest factors of a UAV making a lot of difficult possible. Supporting the endurance is the fact that UAVs do not need to have a pilot running them; the autopilot feature can help achieve many missions which are impossible with a human pilot. The flight duration of a UAV is very big when compared to other combat non-auto pilot vehicles. The durations depend on the fuel burned with total weight and flight terrain; the endurance of the internal combustion engine also depends on this factor. Various ways to increase endurance are to experiment with the sensor, engine and material types.

The factors swinging in the favor of a UAV for 3D-type missions are autopilot and cost. The cost of a UAV mission is measured based on various factors such as the self-destruction behavior. A UAV does not need to carry weapons on board if there is a target that has to be destroyed, but the limitation here would be that the size of the target should be proportional to the UAV size. Precision strikes are used to hit the required targets for self-destruct missions. The complexity of the mission increases based on the target and the environment type. Striking a static target is less complex when compared to a dynamic target. The environment/terrain also plays an important role in the strikes because even if a target is static, the terrain should not hinder the UAV from striking.

Strategic UAVs can also be used for a search mission, where the UAV is sent into a hostile environment to conduct a sweep/search mission. The mission can be conducted with one or more sets of UAV's; when there is more than one UAV, then the UAVs are equipped with communication behavior. Establishing the communication is very important while doing a multi-agent mission because the UAVs have to maintain a certain formation type throughout the flight. Strategic UAVs have artificial intelligence equipped in their system for dynamic reactions, but they also can be controlled to a certain extent, by the control station [9].

Control stations are usually stationed near the base from where the UAVs are controlled. They can autopilot the UAV into the hostile environment to perform the mission; a typical control station has two consoles. These consoles are used to operate the aircraft and also to be a payload operator. The data are collected at the station and the success of the mission is monitored.

If the UAV is equipped with cameras when it is on a search mission the control station can also be equipped with relay transmission devices for real-time data observation. The UAVs have their mission plans configured before the start of the mission which requires less or no assistance from the ground-monitoring station. However when these missions are performed in real time, in some rare cases, the pilot monitoring the mission can steer the UAV to its preconfigured route map. There are missions where the payload from the acquired data needs special monitoring apart from operating the aircraft. The intelligence from the ISR sensors or other radars is handled by the payload operator, whereas the aircraft operator, commonly known because the pilot handles the flight plan. Any person with training in air-traffic control operations can function as a pilot [10].

Strategic missions, which include the terrain and type of task required to be completed, depend on the behaviors of the UAV. The type of UAV also depends on the behaviors which are needed for the mission. The basic behavior is to transit, to be able to move from one co-ordinate to another in the given environment. Communicating with the base station and also with other agents is another important behavior, along with the ability to perform any given set of operations.

For a reconnaissance mission, a UAV needs the flight, communication and also the sensor support behavior. Sensors play an important role in target identification and have to be accurate in their findings. A sensor also acts as transducer to collect all the data and to transmit them to the base station. The sensors can either have a camera or can detect the infrared, thermal sensors for identifying the set of targets. Sensors which depend on signals emitted by the target or on the target motion have a strategy plan embedded at the start of the mission. The strategist transmits the data collected by the sensor in real time. UAV design is also based on the kind of sensors embedded; for example, some UAVs have sensors throughout their frame, a sensor range fitted at the start of the frame or a camera system attached at the bottom.

Communication by the UAVs is one of the important mechanisms required for dynamic data-exchange missions. The data captured by the sensors need to be communicated to the base station or among the UAVs. The process of communication is strategized based on different types of algorithms, such as the Orthogonal Frequency Division Multiplexing (OFDM) technique, which can be used to readily achieve variable data rates and also to provide a multipath-resistant solution. The complexity and accuracy of the communication depends on how extendable the UAV can be. If the UAV is low cost, then it can accommodate a fairly simple communication mechanism [11].

The frequency at which the communication can happen and the maximum communication distance also depend on the UAV design. A lot of factors are considered while designing the communication for a UAV the first being the distance, a UAV has the flight plan set to a specific area of interest because it can only communicate to a certain distance. The other factor is the communication frequency; the UAV can only communicate in the given frequency range, and it should be able to communicate at the frequency throughout the mission. Connection with the base or between the UAVs should be maintained until the mission ends; this factor is very important for the success of any mission.

Combat missions can be conducted by the UAVs where they have the capability to destroy a target. These missions are carried out when the UAVs have weapons equipped so that they can drop them at a given coordinate or target. The costs increase when an external weapon system is involved because the strategy is precise and complex with these types of UAVs. These are famously called “drones” where there is no pilot on board and artillery are equipped.

Cost effectiveness depends a lot on the combat capacity of a vehicle. The higher the weight, the more fuel is burned to travel a given distance. There exists missions which require low costs and are lenient on the risk factor involved. Self-destructing UAVs can be used for such missions. Combat missions involve heavy planning prior to the start of the mission; some factors that impact the plan are the target size, amount of destruction required and the location of target. Based on these factors the, UAV’s physical design is constructed; fuel capacity at the impact is a major factor for the self-destructing combat missions. Fuel is the weapon in these low-cost missions because a UAV with the required amount of fuel can become a missile itself. The disadvantages of these missions is that the vehicle is destroyed, meaning that these vehicles should be made with the least-cost material and should not have data storage mechanisms.

Self-destructing, or killer, UAVs usually work in pairs with a reconnaissance-type UAV commonly known as hunters which provide the killer with the target location. Not only does the hunter search and relay the target location, it also makes sure that the mission is success. Once the strike happens, which is confirmed by the hunter using its sensors, it also scans the area of impact for destroyed targets. Because these UAVs are autonomous and self-destructing they would not leave any information about the striker. The strategy for the killer is to do the pre-scripted navigation and to strike as soon as it reaches the target location relayed by the hunter.

Autonomy is another behavior which should be depicted by a UAV while performing missions involving combat or group communication. The ground-control station usually acts because a communication medium and also a flight-path control system. If the mission involves more than one UAV, then the ground-control station acts as a communication medium. The problem with this type of relay system is that the UAVs have to depend on the station for almost every part of the mission, causing delays in relaying the message and also causing bandwidth limitations. If the agents are programmed to be autonomous, then they do not have to depend on the control station to perform their set of tasks. The mission would be more dynamic if the agents had control over their behaviors. Agents are identified by their unique identification; this information is controlled by the control station and also programmed into other UAVs when conducting a group mission [12].

The unique identification helps UAVs achieve better communication, and also, autonomy can be easily exercised. If the mission involves a formation type throughout the flight, then each UAV will identify itself and try to maintain the formation. Swarm missions, which involve having a swarm structure and a head of the pack is a mission type with relies heavily on autonomy and uniqueness of a UAV. If a UAV goes down during a mission, then in order to keep the formation intact, the rest of the agents react dynamically and change their position to avoid the gap left by the UAV which went down.

5. UAV SCENARIOS

The Simian simulator was developed keeping in mind that the existing UAV scenarios can be extended or that new scenarios can be developed. In order to either extend or develop a new scenario, the existing behaviors have to be extended, or new behaviors should be added. As discussed in Chapter 3, new behaviors are added based on the scenario being developed. The Java version of the Simian simulator had the basic behaviors tested by developing sample test scenarios. The new Simian version tests the old behaviors and also the new behaviors that have been added. New scenarios are created using the new behaviors, these behaviors can also be extended or used in with other scenarios.

Existing behaviors were the flight, search, way point following and communication. Flight is the ability to fly in a given environment. To search for targets in a given environment, way point following: way point following is path planning strategy used by UAVs to design a flight path in the environment, communication: The ability to send/receive data or signals to/from the control station.

5.1. Scenario 1: Sweep Search Using Swarm Behavior

Search behavior by a UAV can be represented in a scenario where the UAV is placed in an area of interest where it scans for targets. The scenario has to deal with the finer details of path planning, flight motion, target identification and message relaying. The simplest case would be if a UAV has to travel from point A to point B (assuming it is a straight path), it scans the area under its sensor range. During the flight duration, if a target falls under its range, then its location and identification details are communicated back to the control station. In the Simian simulator, a target is identified based on its shape; the UAV is given the shape metrics of a target, and a target is identified if it falls under the sensor range. The mission termination factor can be 1) once the UAV reaches point B from point A or 2) if a target is located.

Sweep search is one the cooperative search techniques carried out by UAVs using local communication and way point following techniques. A UAV examines the entire search area by incorporating a sweep motion. The UAV starts from a set of coordinates to reach the end of the path; as soon as it reaches the end coordinates, it makes a turn in the southward direction of the search area. The end points of the turn now become the starting coordinates of the flight path. The motion is continued until the final end coordinates are reached [13]. Figure 12 displays the turn behavior.

.

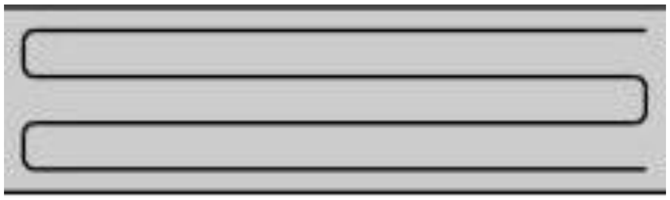


Figure 12. Turning behavior while performing a sweep search [13].

The turn angle at the end of the route is the important aspect to be considered while designing a sweep search. Because the UAV sensor range is limited, the turn should not result in missing any area of interest. The turn can be made outside the environment so that starting point of the new path is under the sensors range. The turn angle should be calculated based on the next starting coordinate of the search area and also the turning radius possible for the UAV. The vehicle estimates the distance to the end point where the turn has to be made and adjusts its speed to make a successful turn outside the environment.

A single UAV scenario would be pretty straightforward where the UAV travels in the set flight path until it reaches the end point; a target is detected as soon as it falls under the sensor's radar. The sweep search becomes a cooperative control mission when it involves a group of UAVs. The inter-UAV communication aspect becomes more vital in cases involving more than one UAV [14].

UAVs have to maintain a specific formation while performing the search. If any UAV falls, then the rest of the UAVs communicate and get back into their formation. Different formations can be used by the UAVs based on the size, sensor range or the type of the environment in which they are performing the search. The UAVs are lined up in the formation prior to start of the mission or entering the environment.

Swarm formation or behavior is a formation type used by bees or a set of birds where they travel in a triangular formation. Similarly, UAVs can exhibit swarm behavior when they are in triangular formation throughout the mission. They keep a specific set distance from each other in order to avoid a collision. To avoid colliding the UAVs must exhibit collision-detection behavior where they try to maintain a safe distance from each other; they have the ability to communicate with the other UAVs, so if there arise a situation of collision, the UAV can communicate to the colliding UAV with an alarm. There is always a leader of the pack when they are exhibiting the swarm behavior; the leader is situated at the top of the triangular formation, making the rest of the UAVs follow it [15].

5.1.1. Scenario Description

The sweep search using swarm behavior has been implemented because one of the two scenarios developed in the C# version of Simian. There are three UAVs that form a triangular shape with a head of the pack aligned in the front. These three UAVs perform a sweep search of the entire environment with swarm behavior. Because the environment is divided into grid cells, the three UAVs travel from one grid cell to another, and their formation helps in sweeping entire grid area. When the sweep is finished, the entire environment has been searched for targets.

The UAVs start from the top-right corner of the environment, traveling in a straight-line path and covering the grid cell with their sensor range. If a UAV is detected its coordinates its color turns red thereby marking the target as already searched. The UAVs travel in the same path until they reach the end coordinate of the current grid cell. They make a turn after reaching the coordinates. The turn is made outside the environment so that they can start the sweep right at the start of the second grid cell; by doing this there are no grey spots left in the environment.

Targets are deployed randomly in the environment so that the UAVs do not have prior knowledge of the target location. The targets are static, so their location is fixed from the start to the end of the mission. They turn red when they are detected, a spotting mechanism used by the UAVs for detected targets. The behaviors exhibited by this scenario are as follows:

1) Way point following
2) Collision detection
3) Inter agent communication
4) Sweep search
5) Target detection
6) Swarm formation

All the above-listed behaviors are designed in such a way that they can be extended or can be used in different scenarios along with the basic-agent behaviors such as flight, strategy, etc.

The scenario-chooser option provides a dropdown way of selecting which scenario needs to be run. This method is a different way of selecting the scenarios compared to the Java version because the Java version requires a hard-coded way of selecting the scenario by passing in the XML file to choose.

5.1.2. Scenario Illustration

Section 5.1.2 illustrates the UAV behaviors along with the step-by-step execution of scenario 1.

5.1.2.1. Screen 1: Start Screen

The sweep scenario is selected from the drop down list. The run button is clicked to start the scenario. The run button is disabled while the scenario is in the running state. Figure 13 displays the scenario selection.

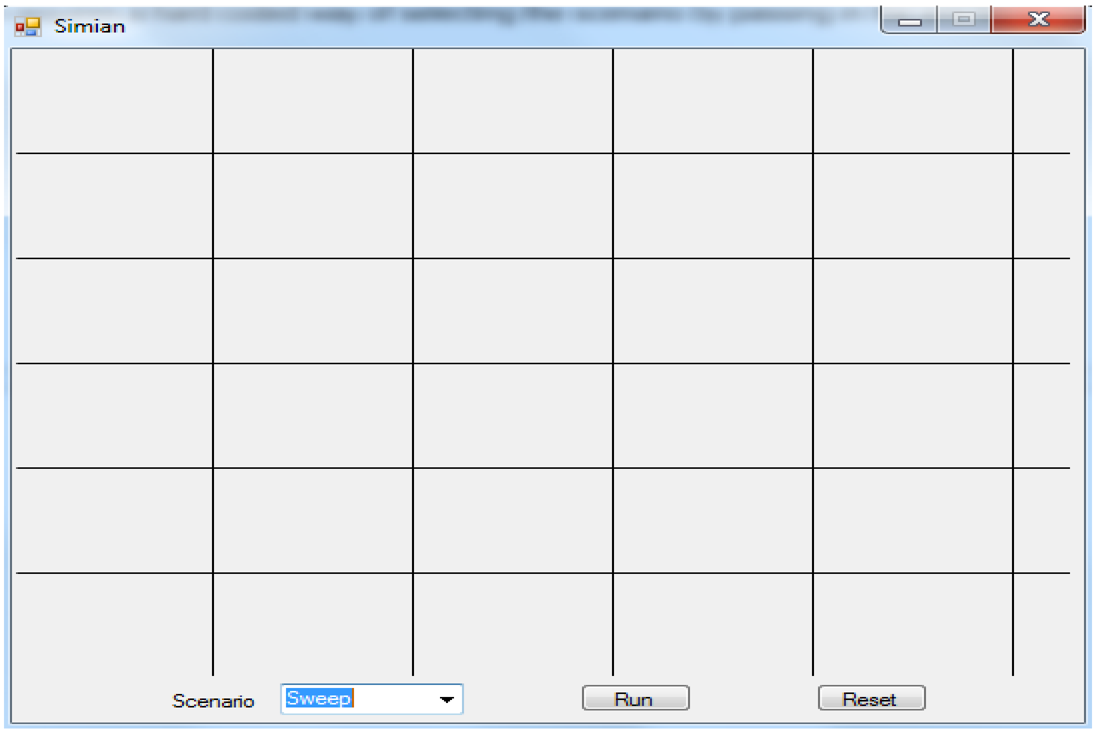


Figure 13. The start screen is displayed when the simulator is run.

5.1.2.2. Screen 2: Target Deployment

The four static targets are deployed in the environment. The location of the targets, as explained in the scenario description, is random. Every time the scenario is run, the targets get a new set of coordinates. By having the targets deployed at random coordinates, the UAVs do not have a prior knowledge about the target location, making the search scenario challenging in a hostile environment. Figure 14 four targets are deployed at different locations, but they can be deployed in the same grid cells as displayed.

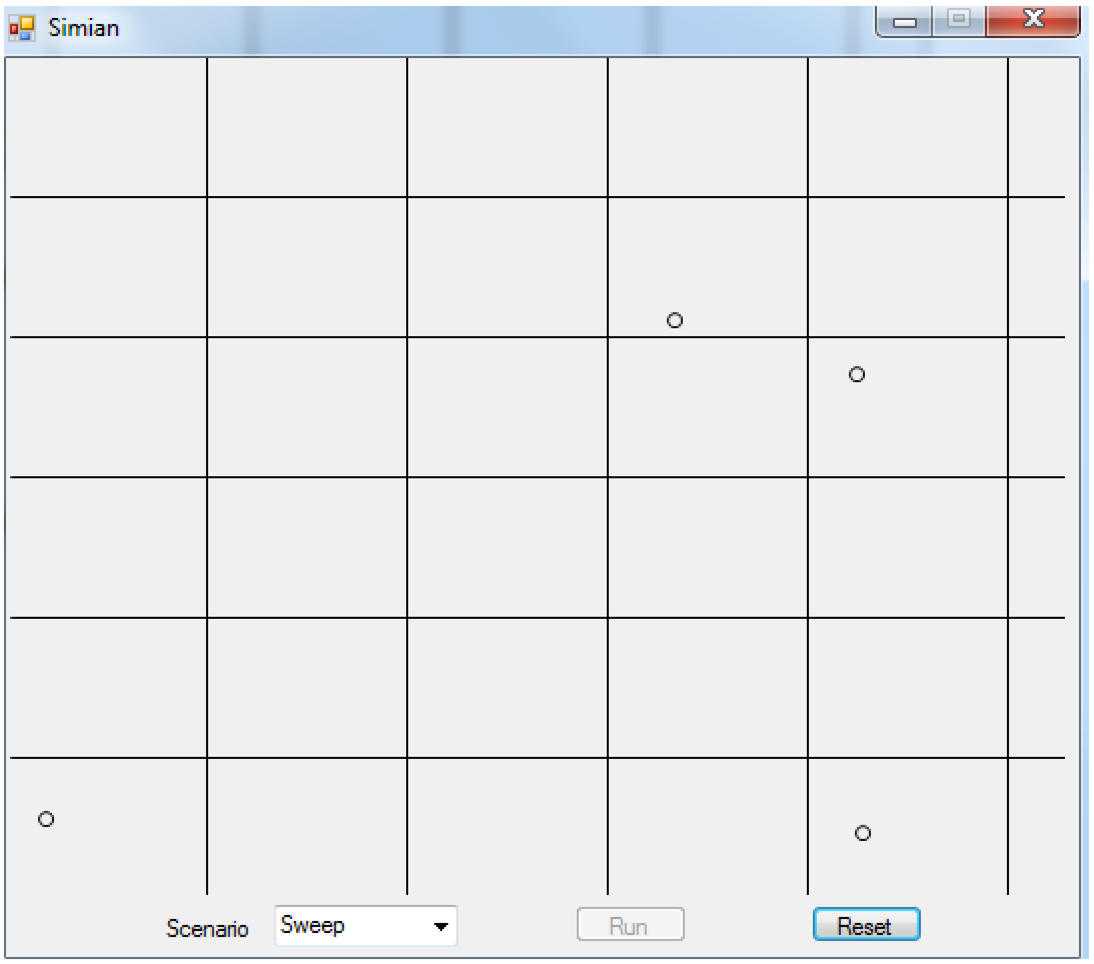


Figure 14. Static targets are deployed at random co-ordinates in the environment.

5.1.2.3. Screen 3: Swarm Motion

The UAVs enter the search environment forming a triangular formation exhibiting the swarm behavior. The distance between three UAVs is constant, and the collision detection is always monitored by the three of them. The search involves moving in a designated area and gathering information. At the lowest level, the UAVs are moving from point to point and gather target information. Basic search behavior is more visible than any other behavior. A first look at the simulation depicts a group of UAVs moving in an area and gathering information. The search commences when each individual UAV is started and concludes when the final destination is reached. Figure 15 illustrates the three UAVs entering the environment exhibiting swam behavior

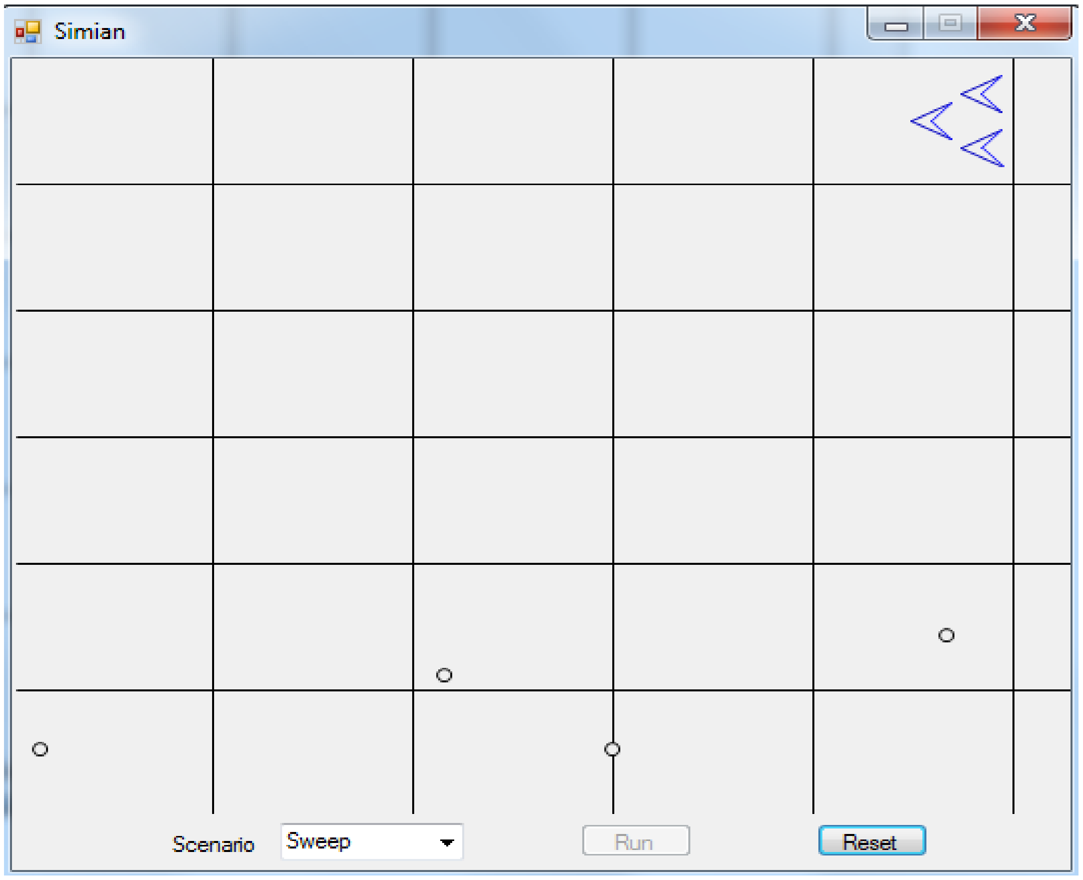


Figure 15. The three UAVs entering the environment exhibiting swam behavior.

5.1.2.4. Screen 4: Target Detection

A target is detected if it falls under the sensor range of any one of the UAVs sweeping the search environment in the swarm motion. Figure 16 demonstrates the process of a target being detected by simultaneously while doing a search. If the search area is broken down into points along the X and Y axis, each UAV can "see" targets 15 points ahead in its flight path, on either axis. When a target falls into the UAV range, it sends out a detected signal, and the UAV gets the location of the target. In the sweep scenario, UAVs identify the target. Visually, it is represented by changing the color of the target from transparent to red. In an attack mission, The UAV moves into "circle the target" behavior as soon as target identification is done.

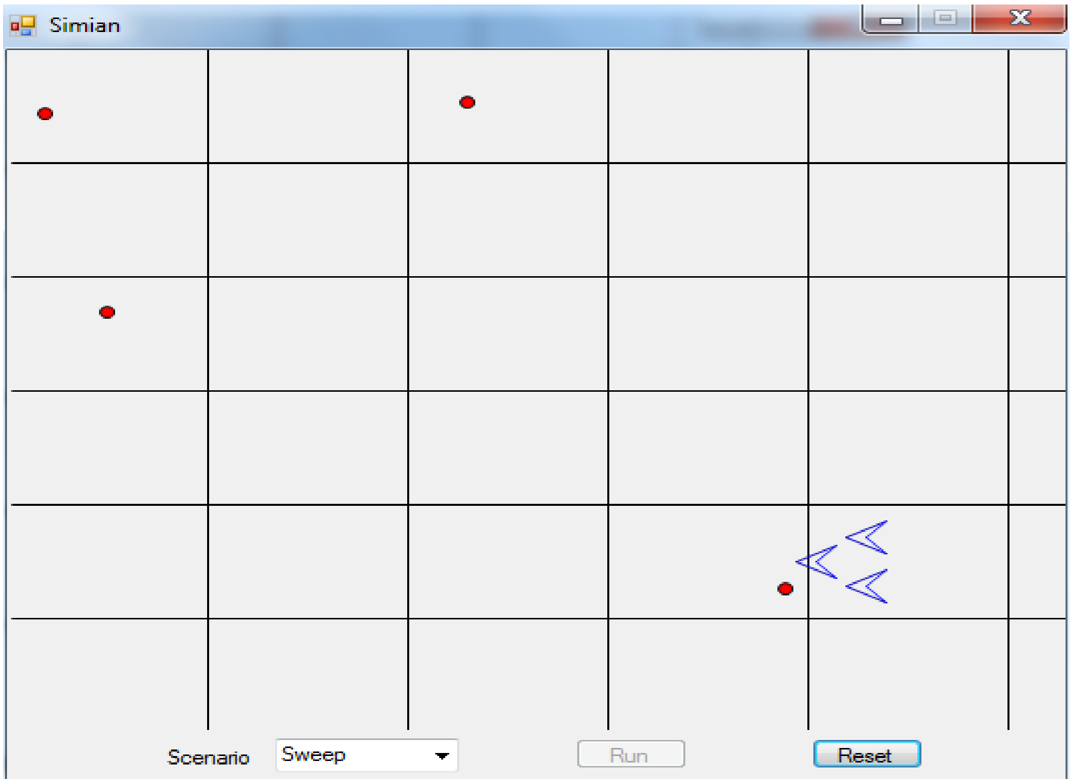


Figure 16. Targets have already been marked red aka as detected.

5.1.2.5. Screen 5: Sweep Search

Figure 17 demonstrates the flag end of the scenario where the group of UAVs have finished sweep searching the entire environment, detecting UAVs that fall under their sensor range. The scenario ends as soon as the UAVs leave the search environment. The sweep scenario has three UAVs aligned as an equilateral triangle. The idea behind the specific alignment is to cover the maximum possible area in a grid cells with smallest number of UAVs and to provide the ability of functioning effectively if one of the UAVs goes down. The alignment is preserved throughout the sweep search.

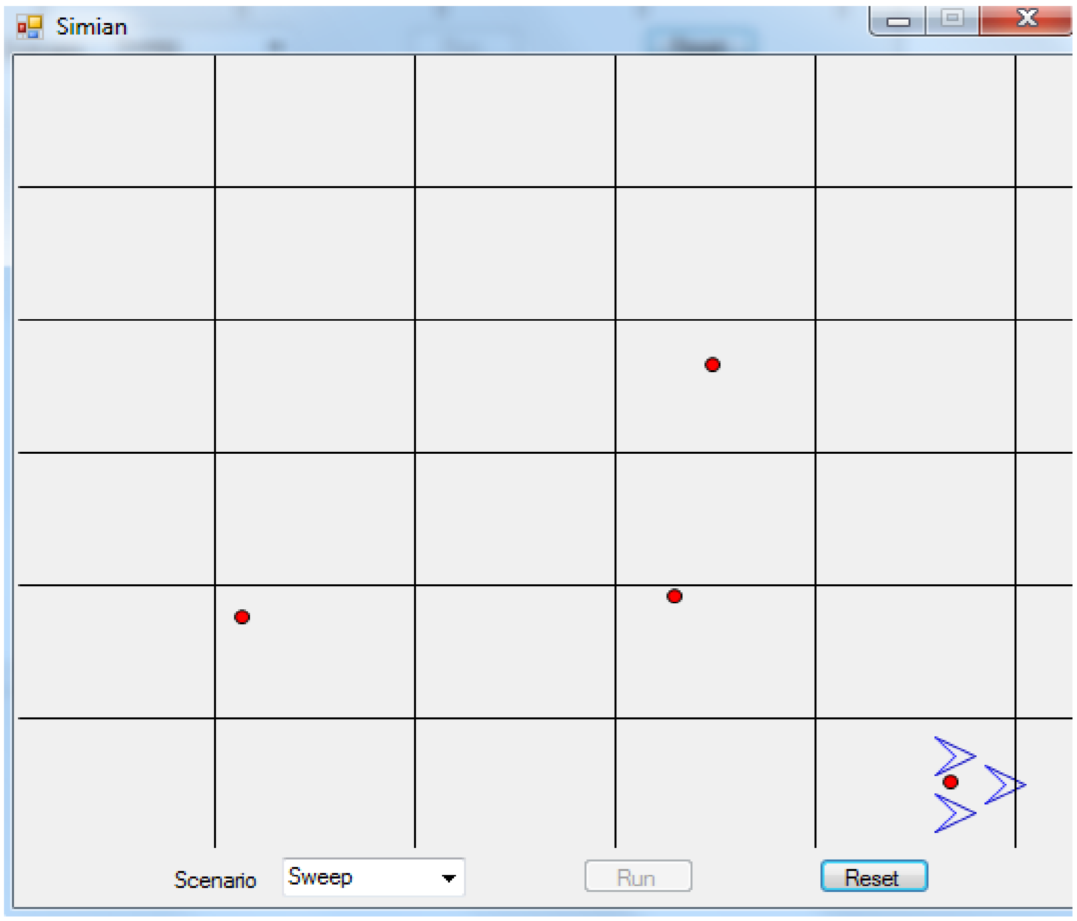


Figure 17. The UAVs exiting the environment. Their alignment is not disturbed throughout the mission.

5.1.2.6. Screen 6: End of Scenario

The scenario is complete when the three UAVs complete the sweep search of the entire environment and exit the environment. The targets marked in red at the end of the scenario are the targets that were identified by search mission. The reset button can be pressed to reset the environment to an empty state with no target deployed. At this state we can either hit the run button to continue with the same scenario or choose a different scenario from the dropdown list.

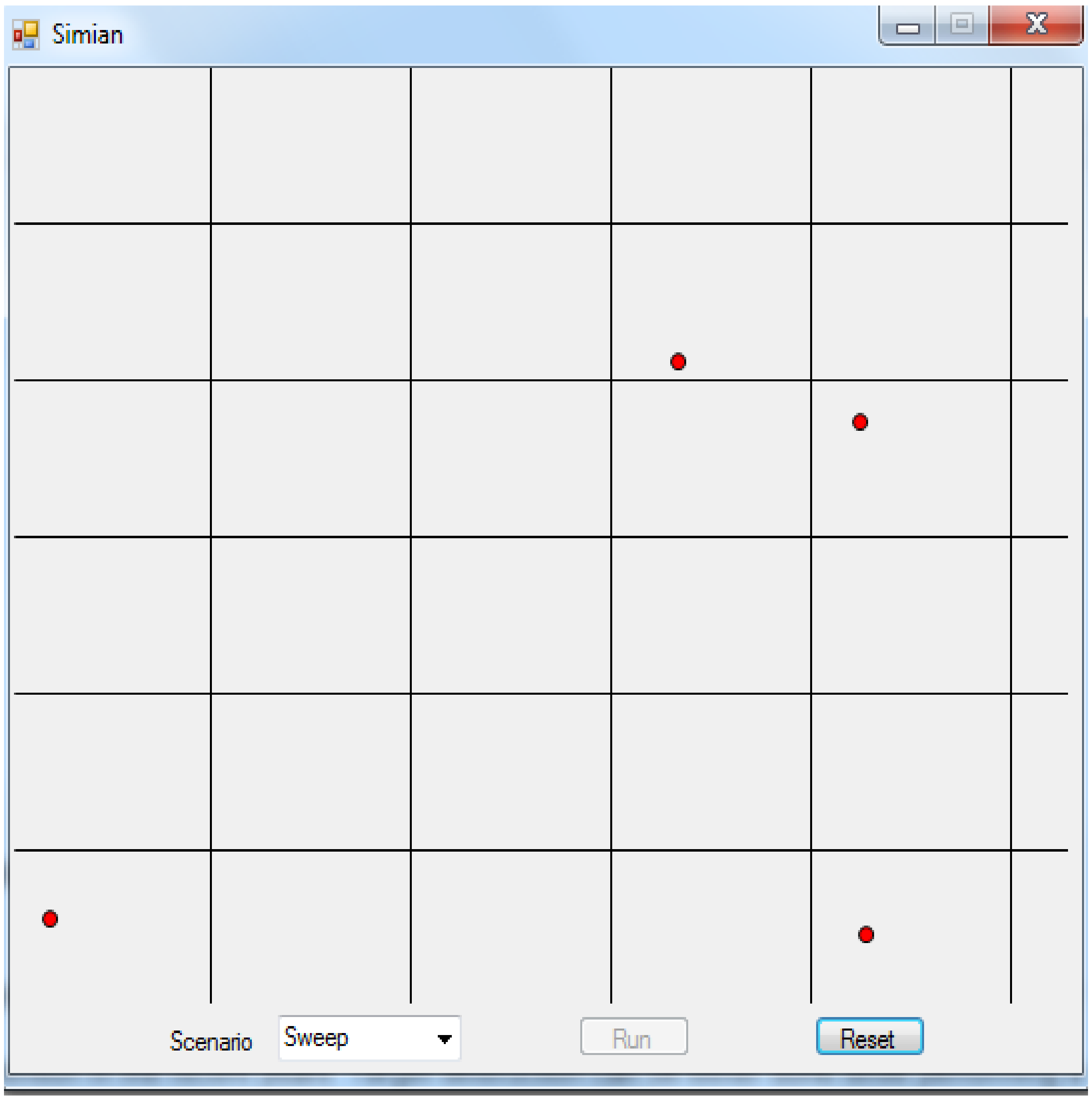


Figure 18. End of the scenario.

5.2. Scenario 2: Target Attack

Target identification and destruction are the UAV behaviors which have attracted research attention in recent years. Target destruction can either be done while performing a sweep search, which is destroying targets dynamically as they are spotted in the environment or the target can be detected first, and then its coordinates communicated to a different UAV which performs the destruction part. The type of target destruction used is based on the scenario being developed and also the behaviors that can be exhibited by the UAV being used.

Scenarios involving identification and detection as two different behaviors utilize two or more UAVs to perform the scenario. The hunter UAV performs the search operation in the given environment; as soon as target is detected, its location is communicated a killer UAV which is situated at the control station. The killer UAV is given a flight path to reach the target location; the target is then destroyed by the UAV. The destruction can be achieved by either dropping an explosive device such as a bomb on the target, or the UAV can self-destruct itself on the target.

Rotating at the same coordinate for a specific time span is a behavior exhibited by UAVs while performing a survey an environment. One type of rotation behavior is where the UAV rotates in a circle with a set radius where the target is based at the center of the rotation circle. This type of rotation requires more fuel and also different turn angles fed into the UAV strategy, increasing the UAV cost. When using a self-destructing behavior, the UAV cost should always be kept at a minimum because we destroy the vehicle so we do not want to destroy a high-end UAV [16].

The rotating behavior developed in the new version of Simian incorporates a different style of rotation where the tip of the UAV is locked onto the target location and the tail rotates in a circular motion. The advantages for this type of rotation motion is that the target location are always locked, that the amount fuel burned is less and that there are not complex turn angles that need to be strategized by the UAV.

5.2.1. Scenario Description

The scenario starts with a single UAV sweeping the search environment in the first grid cell. The search path is continued until 1) it reaches the end of the grid or 2) a target is detected. If there is no target in the first grid cell, then the UAV turns after reaching the first grid cell to start its search in the second grid cell. The UAV stops its path as soon as it detects a target. The target color is turned to red, indicating detection. Then, the UAV demonstrates its rotation behavior over the target; by doing so, the UAV has eyes locked on the target and also buys time to communicate with the control station about its further task.

The UAV does a three-time circular rotation with the tip locked on the target; this delay time is also termed as relay time where the task of communicating is performed. Once the time expires, the UAV starts its decent to destroy the target; the explosion happens as soon as both agents collide. The explosion is demonstrated by a big red circle around the target area. One UAV destroys only the detected target; the explosion does not affect any target in the vicinity of impact.

UAV communication can be the factor determining the success of the mission. If there is no communication from the UAV after collision, then the mission is termed a success. However, in this scenario, the mission is declared successful as soon as the explosion happens.

To implement the circling behavior, we used the following algorithm to calculate the next point on the circle, where UAV.X and UAV.Y are the X and Y coordinates of the points on the UAV and Target.X and Target.Y are the X and Y coordinates of the target. Circle at the angle = 30; PI = 22 / 7; angle (in radians) = (angle / 180) * PI; Next point X coordinate = Cos (angle) *

(UAV.X - Target.X) - Sin (angle) * (UAV.Y - Target.Y) + Target.X); Next point Y coordinate = Sin (angle) * (UAV.X - Target.X) + Cos (angle) * (UAV.Y –Target.X). The behaviors exhibited by this scenario are as follows:

1) Way point following
2) Sweep search
3) Target detection
4) Rotation motion
5) Communication while in rotation motion
6) Target destruction

There are three new behaviors exhibited while performing the mission along with using the already established behaviors. This mission also successfully demonstrates the ability to extend an existing behavior, such as the communication behavior, which is extended to communication while in rotation motion.

The behaviors developed for the attack scenario can be extended to perform a forward air control (FAC) mission. In FAC mission, there are two UAVs performing the attack scenario. The two hunters perform the sweep search in the environment. Once a target is detected one hunter performs communication with the control station, and the other hunter rotates over the target thereby locking down the target location. The killer is given a flight path to the target location; it comes and destroys the target. The hunters verify if the target is destroyed, and if yes, then they continue sweeping the rest of the environment.

All the required behaviors to perform the above-described forward air-control mission are embedded in a single UAV. If required, the behaviors can be extended to different UAVs, displays the flexible behavior extension property of the latest Simian simulator.

5.2.2. Target Attack Illustration

Section 5.2.3 illustrates the UAV behaviors along with step by step execution of the scenario2.

5.2.2.1. Screen 1: Start Screen

To run scenario 2, the "Attack" scenario, select it from the dropdown selection. Click the run button to start the scenario. Figure 19 illustrates the dropdown selection.

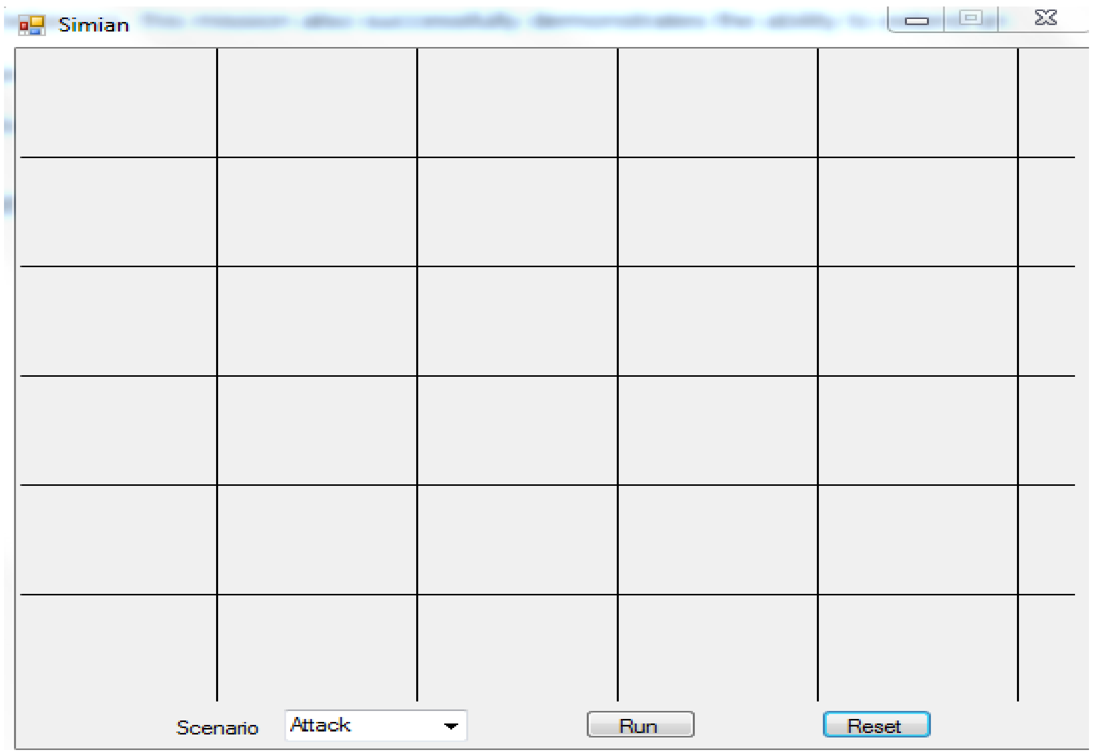


Figure 19. The attack scenario is selected from the drop down.

5.2.2.2. Screen 2: Sweep Search

Because this task is a single UAV mission, the one UAV sweeping the area for targets enters the environment from the top-let corner. This marks the start of the mission by sweeping cell by cell, moving down to the end of the environment. Motion is the physical movement of the UAV from one point to next. The motion is defined by the speed and direction of the UAV. The search area is mapped over the X and Y axes, and divided into grid cell. The UAVs move horizontally along the X axis to cover each and every grid cell on the map. The speed is uniform through the search. Figure 20 illustrates the single UAV motion.

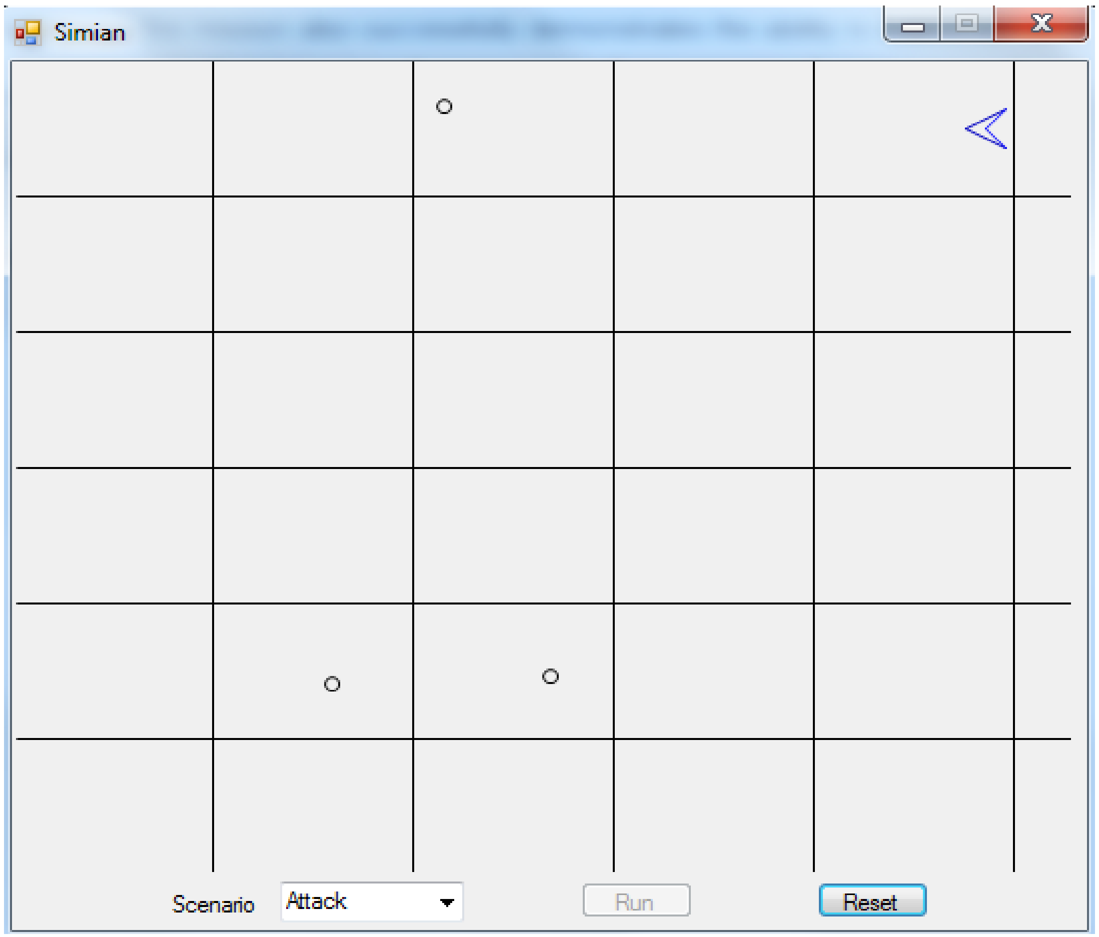


Figure 20. The single UAV motion.

5.2.2.3. Screen 3: Rotation Behavior

Once the target is detected, the UAV starts its rotation motion over the target. This relay time is used by the UAV to communicate with the control station about the target location and also the current state of the mission. When the target is detected in the attack mission. The UAV circles around the target three times before destroying it. The circling behavior is implemented to obtain further details about the target and to pinpoint its location. Figure 21 illustrates the rotation motion.

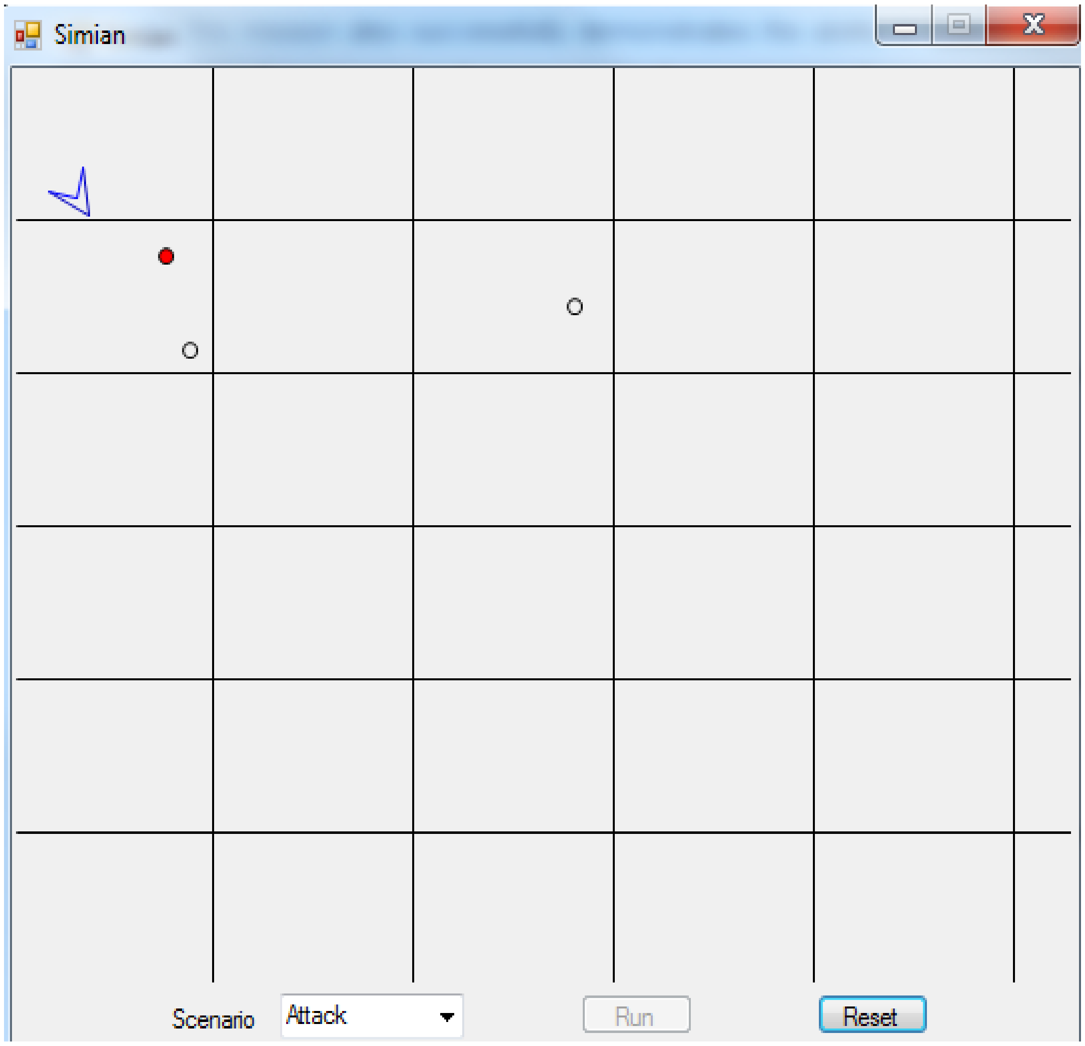


Figure 21. The rotation behavior exhibited as soon as the target is detected.

5.2.2.4. Screen 4: Target Destruction

The target starts it decent onto the target location and self-destructs itself onto the target. The explosion after the contact is displayed with a red circle around the impact area. The explosion marks the end of the mission. The mission ends when the UAV destroys the target, if not, the UAV finishes its sweep mission through the rest of the grid cell as it did for the first grid cell. Target destruction is illustrated in Figure 22.

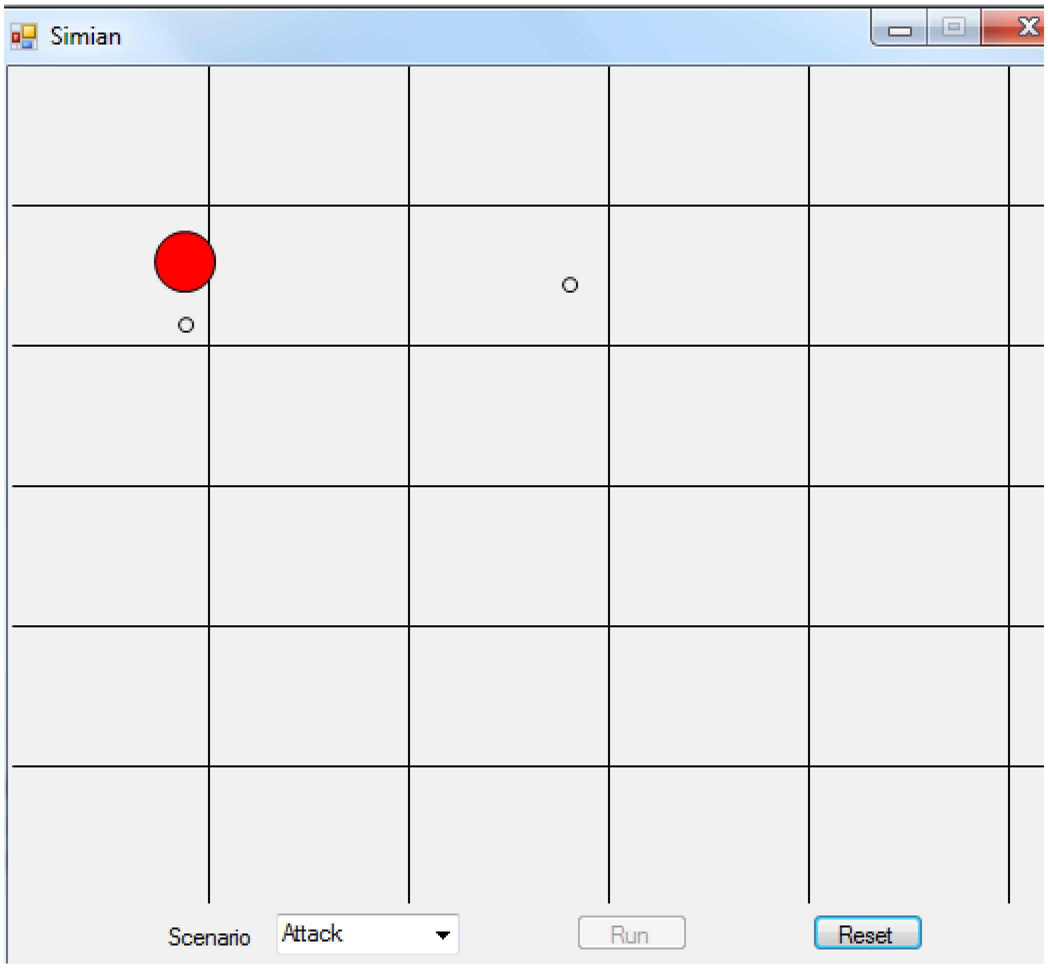


Figure 22. Target is destroyed when the UAV self-destructs itself on the target.

6. CONCLUSION

The three objectives have been met. This paper hosts the Java version of the Simian simulator on a visual studio based C# version (objective1). The reason for hosting on C# is to provide flexibility to the simulator design and to extend the already-existing or new behaviors without encountering the layered complexity of the Java version.

This paper also demonstrates two real-time scenarios involving UAVs which perform 1) a sweep search using a swarm behavior and 2) an attack scenario using a predator-behavior UAV. Both scenarios are run in an environment which is divided into grid cells. The UAVs perform the specific tasks required for the scenario in these grid cells. The new behaviors exhibited by the UAVs which were set as objectives 2 and 3 are achieved through the two scenarios described above:

1) Way-point following
2) Sweep search
3) Swarm formation
4) Collision detection
5) Target detection
6) Target destruction
7) Inter-agent communication
8) Rotation motion
9) Communication while in rotation motion

7. FUTURE WORK

The future work on THE C# version of THE simulator can extend current behaviors or add new ones so that more scenarios can be developed. By making it a flexible system through the current paper, the job of working on the simulator is easier for future developers. Their focus can be more on designing the new scenarios and performing intensive research work because the platform for developing the simulator is already pretty solid.

1) A forward air-control mission can be developed using the current behaviors by having multiple UAVs perform the attack scenario.
2) Steering behaviors can be implemented using the new layered version of the simulator.
3) Extending the current static target behavior to a dynamic target can be a good area of research.

Steering behaviors involve heavy waypoint-following behaviors where the UAVs can roam in an environment based on the behavior embedded into them. The list of steering behaviors can be, for example, wander, obstacle avoidance, wall following, etc. The steering behaviors can also be exhibited by a group of UAVs set in a confined space queuing, flocking, etc.

The scenarios would be more complex when involving a dynamic target where the target gets motion as soon as it is detected by the UAV performing a sweep over the environment. Having dynamic behavior for the target can also extend the possible scenarios that can be added using the dynamic behavior, for example a track scenario where the UAV follows a target for a certain amount of time to study the path strategy adopted by the moving target.

REFERENCES

1. Marlin, Michael. Wide Area Search and Engagement Simulation Validation, March 2007. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA468378
2. Garcia, Richard and Barnes, Laura. Multi-UAV Simulator Utilizing X-Plane, October 2009. URL: http://link.springer.com/content/pdf/10.1007%2Fs10846-009-9372-4.pdf
3. Hennebry , Michael. Quickstart Guide to the Simian Simulator, Department of Computer Science, North Dakota State University, August 2008.
4. Birrell, Andrew. An Introduction to Programming with C# Threads, May 2005. URL: http://research.microsoft.com/pubs/70177/tr-2005-68.pdf
5. Miller, Gerry. .NET vs. J2EE, M June 2003/Vol. 46. URL: http://faculty.washington.edu/mfan/ebiz509/download/readings/DotNetJavaDebate2.pdf
6. Sells, Chris and Griffiths. Ian. Programming WPF, Second Edition, August 2007. URL: http://www.oreillymedia.de/catalog/9780596510374/chapter/ch04.pdf
7. Nishimura, Timossi, Portmann, Pace, Ikami and Beaudrow, Williams. New Epics Display Manager in WPF*. URL:http://accelconf.web.cern.ch/accelconf/icalepcs2009/papers/thp101.pdf
8. Arjomandi, Maziar. Classification of Unmanned Aerial Vehicles, The University of Adelaide, Australia, July 2011. URL:http://personal.mecheng.adelaide.edu.au/maziar.arjomandi/Aeronautical%20Engineering%20Projects/2006/group9.pdf
9. Meighan, Sylvester, Cardenas, Damon, Finch, Mark and Wilson, Laurissa. Research Experience for Undergraduates: Unmanned Aerial Vehicle Design, Phase I, August 2007. URL: http://engineering.utsa.edu/~reu/index_files/Reports/2007_REU_UAV.pdf

10. Nehme, Carl, Crandall, Jacob and Cummings. An Operator Function Taxonomy for Unmanned Aerial Vehicle Missions, June 2008. URL:http://www.mit.edu/~jcrandal/jcrandall/CV_JacobCrandall.pdf
11. Mahmood, Shahid. Unmanned Aerial Vehicle (UAV) Communications, March 2007. URL:http://www.bth.se/fou/cuppsats.nsf/all/e91df00b763f080cc12572990052d819/$file/Masters_Thesis%5B1%5D.pdf
12. Allen, Michael. Guidance and Control of an Autonomous Soaring UAV, February 2007. URL: http://cafefoundation.org/v2/pdf_tech/eCFI.Flight.Deck/NASA.eCFI.UAV.ThermalSoar.pdf
13. Maza, Ivan and Ollero, Anibal. Multiple UAV Cooperative Searching Operation Using Polygon Area Decomposition and Efficient Coverage Algorithms, 2007. URL:http://grvc.us.es/comets/papers/MAZA-DARS-2004.pdf
14. Pajares Martinsanz, Gonzalo. Unmanned Aerial Vehicles (UAVs) Based Remote Sensing, Department of Software Engineering and Artificial Intelligence, June 2012. URL: https://www.mdpi.com/journal/remotesensing/special_issues/uav
15. Parana, Van Dyke, Brueckner, Sven and Odell, James. Swarming Coordination of Multiple UAV's for Collaborative Sensing, September 2003. URL:https://activewiki.net/download/attachments/6258699/AIAA03.pdf
16. Bookstaber, David. Unmanned Combat Aerial Vehicles, What Men Do in Aircraft and Why Machines Can Do It Better, June 2000. URL:http://www.airpower.au.af.mil/airchronicles/cc/ucav.pdf