APPLICATION OF ARTIFICIAL INTELLIGENCE BASED SIMULATIONS FOR MILITARY LOGISTICS SUPPLY NETWORK DECISION MAKING

Christopher J. Bevelle
The University of Manchester
Alliance Manchester Business School
Booth Str W, Manchester, M15 6PB
UNITED KINGDOM

christopher.bevelle@postgrad.manchester.ac.uk

Richard Allmendinger
The University of Manchester
Alliance Manchester Business School
Booth Str W, Manchester M15 6PB
UNITED KINGDOM

richard.allmendinger@manchester.ac.uk

Darminder Ghataoura
Fujitsu
Basingstoke, Hampshire, RG22 4BY
UNITED KINGDOM

Darminder.Ghataoura@uk.fujitsu.com

ABSTRACT

This paper discusses the application of Artificial Intelligence-based simulations currently used by the video gaming industry to evaluate decisions common to a military logistics supply network (MLSN). MLSNs have an array of challenges, some which are comparable to commercial logistic networks such as balancing resources and time-constrained decision making. The two greatly differ in the degree and frequency of dynamic shifts that may occur mission to mission. This causes a high level of uncertainty to an already complex field of study. Additionally, these decision makers operate under duress or crisis while bearing the fact that the decisions, if made incorrectly, can cost lives.

The case study in this paper provides computer-generated solutions to select MLSN missions. By employing current gaming technologies, decision science, and simulation techniques we show in great detail how the computer solves the mission, action by action; why it eliminates scenarios and what options are feasible. In doing so, unit commanders can utilize the concept of reinforcement learning in day-to-day missions or carryout training objectives with potential coalition partners providing visualization, decision support, what-if analysis, or problem simplification by eliminating scenarios that do not achieve the overall mission goals or violate constraints.

1.0 INTRODUCTION

Today, the military utilizes technology within gaming and design engines to create extreme realism and immersion into synthetic environments. A synthetic environment can be defined as the artificial representation of the behaviour external to some real system of interest (i.e. it forms the systems environment) [21,16]. Within this information rich environment is the chance to enhance our capability to plan, decide, act, and react during daily challenges. This research leverages recent tools in Unity's Gaming Engine for design, artificial intelligence (AI) behaviour planning, and data visualization to prototype a logistics synthetic environment for the purpose of enabling enhanced decision-making process for

logisticians seeking new strategies for their missions. Military decision makers must continuously adapt to today's increasingly fast paced environment to handle big data and leverage new capabilities from the commercial sector. The UK's Ministry of Defence seeks to employ the concept of *Human-Machine Teams* which exploits the capabilities of people and technologies to outperform opponents [5]. Synthetic environments can support these human-machine teams, informing and enhancing decision making by providing a digital twin of the world to process data, uncover opportunity, and visualize solutions. The system, when fully developed is not a closed simulation model, but can constantly react to events from multiple live data sources and machines, inputs from human response cells, or user initiated events to continuously generate an effect within the environment, such as a response mission or contingency plan to interact with or study [21].

The Unity Gaming Engine offers several frameworks to program AI behaviours as well as extravagant design tools needed to develop large worlds and models of these immersive systems. Here we orchestrate complex system behaviours through intelligent agents who use reinforcement learning principles to represent cooperative or autonomous planning that perform missions. Our missions may involve disruptions or specific constraints that can require adaptive AI for the agent to re-plan and respond optimally based on available actions and decision maker's preference in strategy.

2.0 REVIEW OF CONCEPTS AND RELATED WORK

Non-Player Characters (NPCs) are the represented agents that make up our simulation programmed with AI behaviours to act intelligently, emulating behaviours such as follow, flee, navigate, pickup which are applied to individual agents or groups [20]. NPC AI behaviour is commonly implemented re-actively by the use of a tool to assist in visualizing and controlling AI behaviours enabling complex story-lines, and scenarios [20,4]. Common tools are Finite State Machine and Behaviour Trees [20,4,2]. Goal Oriented Action Planning is used by game developers to achieve more realism [4].

Finite State Machines (FSMs) manage states and transitions between states of the game, an NPC, or the player playing the game [12]. FSMs are useful for cycling characters through series of states that collectively form AI behaviours and are notably used to control animations within Unity's animator [20]. An FSM, for instance, can make a character transition from walking to jogging, running, and then back to walking based on commands or conditions being met [20]. As the complexity of the agent behaviours grow though, the Finite State Machine will become more difficult to work with [4]. In 2006, Hassaine et al [11] explore concepts of a synthetic environment for Maritime-Air Tactical Experiments. In their study, an entity can be controlled by a behaviour model implemented in a Finite State Machine, and entities evolve in a synthetic environment simulating real world terrain, oceans, and weather conditions that affect the entities' behaviours and their overall capabilities (e.g. line of sight, speed, sensors performance, probability of kill, etc.).

Behaviour Trees (BT) can offer a structure to organize complex behaviours but may require more understanding of coding and scripting than FSMs. In 2019, Evensen et al [2] investigated the BTs to Model Battle Drills for Computer-Generated Forces. The authors point out that, from a development perspective, important advantages of BTs include it being highly composable, highly modular, reactive, human readable, and suitable for automatic generation (e.g. machine learning). Mentioned limitations include BTs' poor nature at modelling uncertainty when there are multiple options available, tedious to represent typical state-based behaviour, and that the BTs will cause performance issues when scaled since the whole tree is executed from beginning for each simulation step. The work mentions the popularity of BTs to develop AI in games. Examples include Halo 2 being one of the first in 2004 as mentioned in this paper.

Goal-Oriented Action Planning (GOAP) is a more advanced concept that allows planning by agents who can communicate with a representation of the world state in order to accomplish goals from a set of currently available actions at the agent's current state [4]. In planning, there may be implied task to accomplish a goal,

such as a fuel needed to perform a vehicle routing operation. GOAP enables the agent to understand its available actions from its current state and determine a means to get to a goal through search [4]. The actions that make up a goal can be organized and then a heuristic is required to properly map the necessary decisions. Such a system requires a planning domain to orchestrate behaviours systematically. The game F.E.A.R. is a popular example for its advanced AI planning which uses GOAP [4].

The computer-controlled characters may use complex AI to handle strategy, navigation, or animation [4]. Notably, the methods go through a series of decision-making processes that can be classified as reactive AI or deliberate AI [4]. In reactive AI, the developer must explain exactly how the agent will reach its goal. In deliberate AI, the AI method can evaluate the best path towards a goal [4]. This concept is the difference between telling the NPC the shortest path and providing it the means to evaluate the shortest path to a goal.

Reinforcement Learning (RL) principles are applied within this research to either create complex agent behaviour to plan in cycles or towards goals that can model either a command and control (C2) relationships (such as a coalition cooperative behaviour) or the represented physical behaviours and traits of certain types of vehicles that perform various tasks [4]. The practical application of RL with planning models enables a decision maker the opportunity to efficiently search for what can or should be done under various constraints with implied tasks and predicates [10].

3.0 METHODOLOGY

The examples that will be explored focuses on missions from military logistic supply networks that involve uncertainty, changing constraints, and goals with moving planning periods. NPCs will be used to execute a response plan to some disruption that may have occurred, requiring a quick solution. The concept is designed for synthetic environments therefore the aim is to complement realistic models. Several assets and tools were tested to form the means to develop the prototype. Many of the resources require more computing resources than were available for this research.

3.1 Terrain Tools

Terrain tools were explored for importing models from satellite data. Figure 1 shows a location from Manchester, UK using CityGen3D. The building models can be replaced and all objects imported are distinct with building labels. World Composer (see Figure 2) imports maps and works with other design tools. Both tools are capable of generating height maps for the terrain.

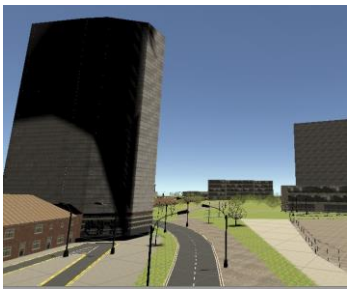

Figure 1: Using CityGen3D

Figure 2: Screenshots from World Composer's website

3.1.1 Unity's AI Planner for Behaviour Planning

The intelligent agents in this research are programmed with a new (at the time of this article) planning framework or behaviour planning tool in Unity called the AI planner. Examples from Unity involve the

intelligent agents demonstrating the optimal manner to maximize the score on the level of a game or to solve an intricate escape room. The AI planner utilizes the Trial Based Heuristic Tree Search which is a state-of-the-art algorithm that generalizes the Monte Carlo Tree Search, useful for programming AI for decision making and game development [4,15]. The AI planner was selected because it follows principles in explainable AI which is an underlying goal within this research to ensure the concepts are repeatable and can be verified. For example it uses a semantics based approach for solving a Markov Decision Process (MDP) but notably has a very effective state representation that uses trait-based logic – each object of the environment can have traits and each trait can have multiple properties, enumerated states, or custom fields that can be used to define the characteristics of each objects state representation. The use of the trait-based system enables any object with a trait to be involved or excluded by trait within actions to affect the environment or the agent decisions. For validation, all decisions and effects by the agents can be viewed in the plan visualizer which shows the tree search used by each of the agents to optimize their rewards towards an overall goal (See Scenarios 1 and 2 in Section 4).

Additionally, any animated behaviours of actions to represent an effect may be used when actions are executed to convey the solution to the desired level of realism since the solution exists within a system capable of film production or video game level design. Alternatively, the AI planner can be used abstractly to develop a plan without animation for faster responses as will be shown in Section 4. This work enables a contingency or response plan to be programmed for use in a short notice situation requiring the need for an automated response. Animation is not the focus of this research outside of the movement, but this can be added as necessary within the steps of the created plans.

3.1.2 EasyRoads3D

This asset is a paid tool that was used to develop roads. While there is a free version, the paid version was best for complex road systems. The result enabled the ability to connect many miles of virtual roads of various types with ease and great visuals. The tool is capable of creating bridges and more complex roads.

3.1.3 Navigation

For navigation, we employ a waypoint system (see Figures 3 and 4) which is an advanced technique in AI behaviour planning to provide agents a graph network composed of nodes and links [20]. Links may be uni-directional or bidirectional which was used to enable vehicles to drive on the correct side of the road. With this approach, the A* Algorithm is used to compute the shortest path. A waypoint manager is developed and used by NPCs to navigate the terrain. The waypoint manager references its graph and the waypoint array that is used for designating the path to travel.

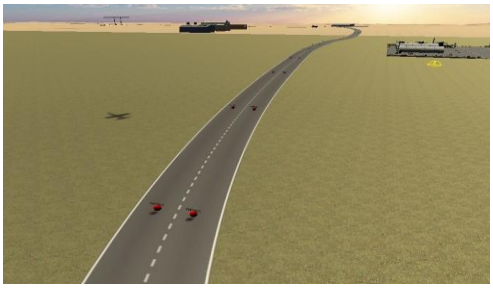

Figure 3: Waypoint system

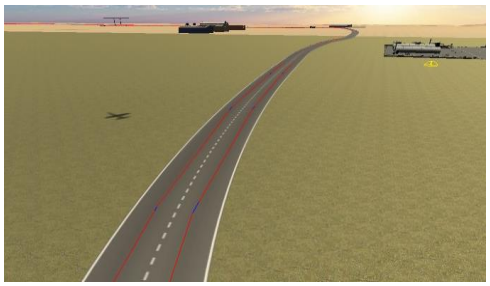

Figure 4: Graphing road network from waypoints

3.1.4 Driving Behaviours

For movement, the speed, forward direction, and look angle are scripted behaviours that are controlled by referencing the objects *transform* [20]. The transform contains the 3D references of any objects position, rotation, and size. To create the movement, the vehicles travel towards closest waypoint in the designated path from the method using the A* algorithm. Once the vehicle reaches each waypoint, the vehicle continues to move to the next waypoint in the path. Additionally the vehicles use some physics such as gravity, wheel and box colliders, but this wasn't the main focus of the research.

3.1.5 Zones

The abstract concept of zones and regions apply to military planning are created by a method that detects any waypoints in a zone. The zone may represent a threat area that may be avoided. Please refer to Scenario 3 of the next section for an outlined example.

4.0 CASE STUDY

A logistic supply network for a group of Forward Operating Bases (FOBs) receives supplies from the regional hubs supporting the area. There are five bases and two hubs as shown in Figure 5. The scenarios below evaluate modelling NPCs that carry out missions within the modelled environment.

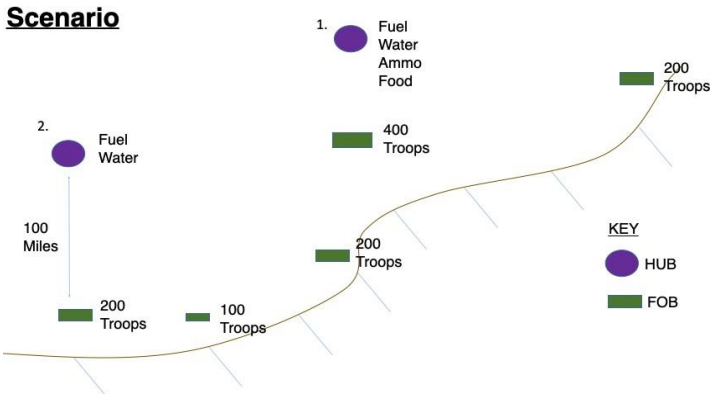

Figure 5: Base scenario for models

4.1 Scenario 1: Vehicle Breakdown Recovery Operation

A military logistics operation supporting a forward-based operation must maneuver assets to react to a vehicle breakdown. The decision to use a nearby asset or to send a vehicle from Headquarters will depend on the current situation. The breakdown event may occur at any specified location by the user to create the realism of the scenario. There are two phases of the operation, normal operations and the recovery plan. Available actions from the normal operations are different than behaviours during the recovery operations and must be distinguishable to agents in the system.

Figure 6: Staging vehicle breakdown with interactive "Collider" to initiate recovery plan

4.1.1 Modelling Results

In Unity, any object may have a physical boundary in space called a **collider**, which can be used to trigger events when that boundary is crossed. The cube shown in Figure 6 has an attached script that will cause an accident if any vehicle touches the cube it triggers the execution the recovery operation. There are multiple ways to create this event, but the chosen way allows for a planner to use this event trigger during a live model. For example, if the model is running the event does not occur automatically until the object is touched and the object can be moved by the user with their mouse during run-time into the path of a vehicle to start the response plan. Alternatively, logic can be scripted if a vehicle had a smart sensor to initiate the plan based on some conditions.

A **queue** system is built to create the current situation where vehicles are working on their designated sequence of tasks. While this may sound complex, we have structured the available tasks that each vehicle works on to be assigned by vehicle ID and rewarded in order of preference such as priority. Once a task is complete, the work is destroyed, and the agent can consider other things. In unity, an *empty object* can be used to enable this. An empty object can be used to store in this case the presence of a unique work task or work items, where each task has a trait and properties that the AI planner uses within the agent's environment. Only one action is developed within the plan to establish this queue-like system since the AI planner understands that work will be destroyed after each completion. This represents actions from normal operations, but the vehicles while working, may experience a breakdown. This is uncertain, but a possibility being continuously planned for by the agent. We create another action called, *MyCheck* to continuously expect a breakdown to occur. If the breakdown occurs, a method from our response plan, called *Scan Distances*, is used to measure the path-distance in meters from the break down event to other vehicles. The scripted output of this function is shown in Figure 7. This output is then stored in the property of each vehicle so that the best vehicle can be measured the agent's state representation at that moment.

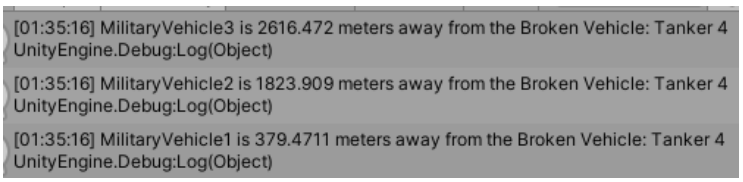

Figure 7: Using path distance to determine best candidate to assist in recovery

Next, the closest vehicle from the position of the breakdown event is selected. The decision space is 100s of miles depending on the situation vehicle availability and location may change the scenario's result. Since the physical road network is developed within the synthetic environment, the path distance instead of the Euclidean distance between vehicles should be measured to weight the decision. The vehicles use the shortest available path to travel between locations, driving on the correct side of the road.

4.1.2 Implied Tasks

To demonstrate the flexibility of the tool, a step is added that requires the chosen repair vehicle to refuel only if it is fuel levels are below a certain amount. Keep in mind, that we have not told the agent explicitly to do this, but it understands that it must do this in order to complete the steps towards its reward. If the vehicle is low on fuel, then the vehicle needs to understand how to refuel before performing the recovery operation.

Figure 8: An implied task to refuel before performing the recovery operation is required.

Using Snaps in Unity, a basic gas station is created to demonstrate the vehicle navigating to this designated location, shown in Figure 8 (right image). Figure 8 (left image) is the associated step being taken by the vehicle. Any vehicle chosen to assist in the vehicle recovery operation must stop what it is doing and navigate to the location of the vehicle that broke down at some location.

Figure 9: The chosen repair vehicle assists the tanker

Figure 10: The tanker has been repaired starts working again

Once repairs are finished (Figure 9), both vehicles should continue working if work exists until all work in its queue is completed. Vehicles of the same type can take on the same work, but the work must be assigned distinctly to ensure the vehicles are not performing the same task. Figure 10 is a snapshot of the vehicle continuing its queue of work after the task is completed.

4.2 Scenario 2: Water Distribution Vehicle Routing Problem

The military faces several logistical nightmares with water shortages. Several problems exist within a logistics supply network to accommodate the demands of water which changes based on each mission location and unique requirements. Research suggests that current demands are unsustainable. During a disruption event, the priorities for balancing supply levels will alter.

Problem Details

- 100 litres of water per soldier per day is needed for the purposes of washing, cooking and general cleaning.

- Large FOB 100 litre x 400 people = 40,000 litres per day
- Medium = 20,000 litre per day
- Small = 10,000 litres per day

- Tanker capacity same is 36,000 litres
  - Tankers can't be interchanged between water and fuel

4.2.1 System Behaviour

The Hubs provide water supply to five FOBs depicted in the scenario diagram. Water tankers have a capacity to hold 36,000 Litres of water for transporting the water from the HUB to the distant FOBs. At any given moment, new demand is created that will imply a schedule based on the state of all resources. The agents must learn how to transport within our system and provide an optimal strategy to meet the water demand while not wasting resources. For each FOB a supply level and a demand is randomly established, but the level can be altered by the user. The demand level represents amount of water needed to satisfy that bases requirement for the day. If a disruption occurs, the user may input the new amount and the system will adjust.

4.2.2 Approach

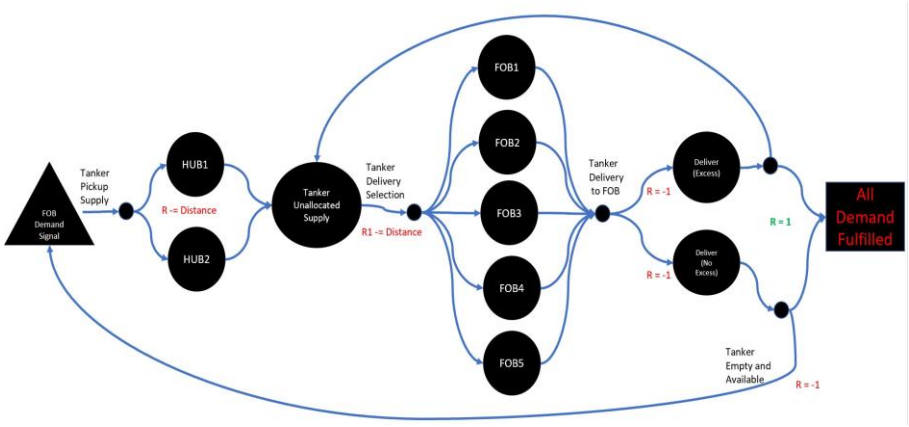

Figure 11: Markov decision process for tanker distribution from Hubs to FOBs

There are different strategies to handle a vehicle routing problem. Figure 11 describes different reward policies used based on the condition. In this case the planner is a system level planner as opposed to each agent having independent brains. For example, the planner may use up to four vehicles, but it chose to only use three vehicles when given negative rewards to use each vehicle. Within the planner, the agents are also able to cycle through states. Before giving the agents a goal, we tested this out by giving rewards at the end of the delivery in two ways. The first policy was to encourage the machine to balance which HUB it pulled water from, but this of course adds more distance. The change of policy was the difference of 15km in total routing. Custom reward systems are designed for each strategy. This reward can be based on the number of miles implied by the choice of action, a base reward, or a combination of factors (e.g. distance and priority).

Table 1: Comparing different strategies for vehicle routing problem

Strategy	Total Distance Travelled	Vehicle Used	Assignments
Shortest Distance	24 KM	3 / 4	11
Highest Demand First	28 KM	3 / 4	11
Highest Demand First	25 KM	2 / 2	11
Random Strategy	39 KM	4 / 4	16
Mixed Strategy	31 KM	3 / 4	12

Table 1 shows the result of testing different reward policies along a certain strategy. Multiple agents are placed along various random locations on the map which had five FOBs and two Hubs. The vehicles must travel to the Hub for pickup and then deliver to the FOB. Figure 12 shows the example output of the assignments chosen by the planner. This is an example of using just the plan and not the animated actions as shown in the first example. For faster answers or for testing strategies, this method provides an immediate output. This method is only appropriate after validation.

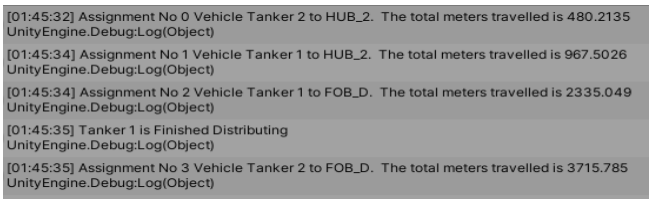

Figure 12: Custom output of AI Planner actions taken

The agent has the ability to predict the best action from its current state if the agent can properly estimate its future rewards. Since the agent's rewards depends on its future location, it is necessary to allow the system to realize that locations change and how. This is similar to game development when we teach a computer to play chess. It has realized the cells that exist in order to plan for future movements from the cells. For instance, the vehicle will move close to the FOB it is travelling to, but not in the same position of the FOB. If you don't use a position, then the vehicle will not be able to estimate its distance from other FOBs. Within the AI planner, the developers have included this logic inherit to enable use position as property. Because of this, we can for instance designate the parking Bay the tanker will be assigned to, providing this transform's position as the future location. We can store this position within the properties of the FOB's traits. Figure 13 shows the plan visualizer during this static plan.

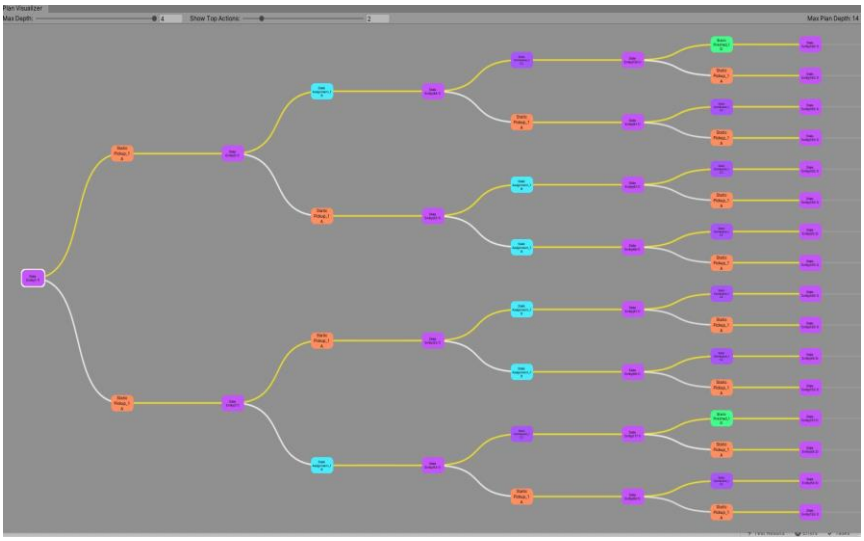

Figure 13: AI Planner's Plan Visualizer showing snapshot of the tanker's process with multiple vehicles

To employ this in a synthetic environment, the demand signal triggers the plan which may be connected to a sensor or database that triggers this condition for real-life applications. Within the model for instance, the vehicles only plan when the demand is given and goal can easily be shifted to service only specific FOBs, a set of FOBs, or other conditions as desired.

4.3 Scenario 3: Human-Machine Teaming Concept

An MDP can be fully observable or partially observable. In the military, information can be generated from various sources which can enable decision making. To demonstrate this, consider the concept of Human-Machine Teaming depicted by a pilot flying a drone. The event is fed into the synthetic environment where the drone has a partially observable environment to observe threats.

4.3.1 Approach

If the drone classifies a threat on the road network, such as an improvised explosive device (IED), that information can be used to update the road network for logistics planning.

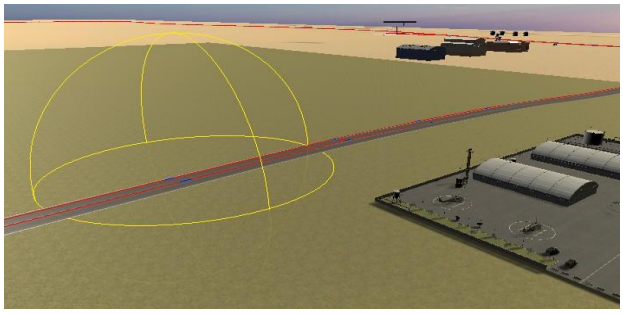

Figure 14: Event classification from represented sensor

A threat event is first created, shown in Figure 14 that includes the event's threat location, and radius. Once the threat is classified, the road network is updated by changing the link cost of the graph network which will cause the A* algorithm to avoid certain routes.

After creating the threat ring, we find the waypoints within the ring. Next, we assign a very large value known as "Big M" to the cost travelling to that area. By adjusting the cost, the area will be avoided when the path is computed.

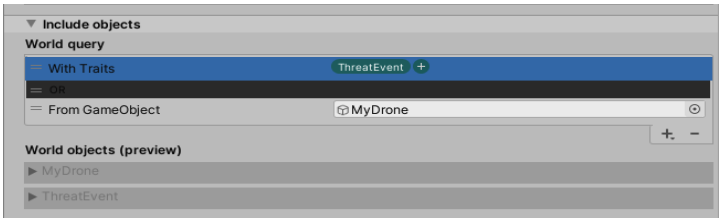

Figure 15: Example of creating Partially Observable environment through traits and objects.

Next the drone is given the ability to only “partially observe" the environment". Figure 15 shows that the represented drone in is only able to work as sensor to our system to report "threat events" and understand its own state. By including this logic in the plan, it allows us to track where the information came from and assign any level of credibility towards how this update affects our decision making. To expound on this, some sources of information may not be trustworthy, but in this case if we know that this drone's assessment is always accurate and should always be followed, then immediate action would be required to maneuver operations away from this threat, or cancel operations. Note that this scenario is only outlined conceptually to show how the AI planner can limit the environment for a human-machine concept of operation.

5.0 CONCLUSION

Military logistics supply networks face various challenges due to constant disruptions and complexities requiring a response plan and a means to employ it. We investigated the simulation and AI functionalities of Unity to tackle these challenges encapsulated in different scenarios. Within a synthetic environment, information fed from various sensors and sources can be fused to establish an environment for processing and depicting complex scenarios to enhance rapid decision making during disruption events. Enabling an AI to learn your operation's response plans, preferences or possible goals, and causes and effects of actions can enable a decision maker to gain a powerful tool with explainable problem representation to implement strategy.

REFERENCES

[1] US DoD, “Department of Defense Modeling and Simulation Glossary,” 1998. [Online]. [Accessed 01 08 2020].

[2] P. Evensens, H. Stien and D. H. Bentsen, “Using Behaviour Trees to Model Battle Drills for Computer-Generated Forces,” *NATO Science and Technology Organization STO-MP-MSG-171*, p. 01, 2019.

[3] A. Calvin, “Improbable has done £10.4m of work for UK military,” [Online]. [Accessed 01 08 2020].

[4] Unity Labs, “Unite LA 2018. AI for Behavior - Advanced Research for Intelligent Decision Making,” in *Unite LA* , L.A., 2018.

[5] UK MoD, “Joint Concept Note 1/18 Human-Machine Teaming,” 2018.

[6] S. Armstrong, “Game Engine Review,” *NATO Science and Technology Organization STO-EN-MSG-115*, p. 05, 2013.

[7] R. Eglese and S. Zambirinis, “Disruption management in vehicle routing and scheduling for road freight transport: a review,” 2018.

[8] Unity, “Unity powers 3D military simulations,” 2010. [Online]. [Accessed 01 08 2020].

[9] O. Trunda and R. Bartak, “Using Monte Carlo Tree Search to Solve Planning Problems in Transportation Domains,” *MICAI 2013, Part II, LNAI 8266,* pp. 435-449, 2013.

[10] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction 2ed, Cambridge: The MIT Press, 2018.

[11] D. F. Hassaine, D. A. L. Vallerandm and D. P. Hubbard, “Towards a Synthetic Environment for Maritime-Air Tactical Experiments,” *NATO Science and Technology Organization RTO-MP-SCI-180,* p. 08, 2013.

[12] Unity Learn, “Unity Learn Finite State Machines Tutorial,” 2019. [Online]. [Accessed 01 08 2020].

[13] J. Dauble, A. L. Medford and D. J. J. Frey, “Leveraging Commercial Game Engines for Multi-Domain Image Generation,” *MODSIM World 2018,* p. 36, 2018.

[14] T. Keller, “MDP Algorithms: Lecture 2,” in *ICAPS 2018*, 2018.

[15] T. Keller and M. Helmert, “Trial-based heuristic tree search for finite horizon MDPs,” in *23rd International Conference on Automated Planning and Scheduling ICAPS 2013*, 2013.

[16] R. Smelik, F. v. Wermeskerken and R. K. a. F. Kuijper, in *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2018.

[17] D. J. Roy, “Causality - Inferring Causal Effects from Data - 2.2 - Causal graph,” [Online]. [Accessed 01 08 2020].

[18] S. Glen, “"Acyclic Graph and Directed Acyclic Graph: Definition, Examples" From StatisticsHowTo.com: Elementary Statistics for the rest of us!,” [Online]. [Accessed 01 08 2020].

[19] J. Weinberger, “Unity C-Sharp Survival Guide,” 2018. [Online]. [Accessed 01 08 2020].

[20] P. d. Byl, “Unity Artificial Intelligence for Beginners,” [Online]. [Accessed 01 06 2020].

[21] K. Matthews and M. Davies, “C2-led Simulation and Synthetic Environments for Air Operations,” [Online]. Available: http://www.dodccrp.org/events/6th_ICCRTS/Tracks/Papers/Track3/006_tr3.pdf. [Accessed 1 08 2020].