2020 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
MODELING & SIMULATION & SOFTWARE (MS2) TECHNICAL SESSION
AUGUST 11-13, 2020 - NOVI, MICHIGAN

Using A Gaming Engine for Autonomous Vehicle Modeling and Simulation

John Brabbs[1], Benjamin Haynes[1], Thomas Stanko[1]

[1]US Army CCDC-GVSC, Warren, MI

ABSTRACT

Autonomous vehicles provide a unique challenge for simulation to effectively and performantly model due to their system level complexity and the inclusion of autonomy software. This environment is made even more challenging when looking at the interactions of humans in-the-loop with the vehicles and autonomy software and also how to include more simulation in the testing process for Autonomous Vehicles. With the use of a software framework built from a Commercial off the Shelf (COTS) game engine the Ground Vehicle Systems Laboratory demonstrated the feasibility of real-time human, software and hardware in the loop testing of autonomous systems. This approach facilitated the execution of two major events which are described herein.

Citation: John Brabbs, Benjamin Haynes, Thomas Stanko, "Using A Gaming Engine for Autonomous Vehicle Modeling and Simulation", In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 11-13, 2020.

1. INTRODUCTION

Research in autonomous military operational environments is hindered by the logistical difficulties of testing and evaluating systems in the physical world. Additionally, the effort to engineer and integrate a small number of vehicles can be cost and safety prohibitive, while certain behaviors and test results only present themselves when multiple platforms are operating together. Similar problems occur in urban driving, when training machine learning algorithms [1]. An alternative to physical testing is evaluation in simulation. Simulation is a broad term, and encompasses many forms of computer-based simulacrums of real-world phenomenon, but in this case is the presentation of a virtual environment to either humans or agents in-the-loop to evaluate their performance, integration, and usage. The Ground Vehicle Systems Center's (GVSC) Immersive Simulation's three capabilities Warfighter Experimentation (looking at Tactics, Techniques & Procedures); crew station design and development; and Autonomy (Autonomy-in-the-loop) will be used in the research and development of autonomous vehicles.

2. BACKGROUND

GVSC's Immersive Simulation's three capabilities of Warfighter Experimentation, crew

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300

station and Autonomy previous to 2020, had been using multiple simulation engines to support these three capabilities. Two different capability (crew station, Autonomy) efforts were completed at the GVSC's Ground Vehicle Simulation Laboratory (GVSL) in 2019, with one of the goals to use the same COTS game engine. These efforts sought to evaluate the human and autonomy algorithm performance in the context of military operational environments. The research objectives of each study below are unique. The crew station capability effort seeks to capture baseline data related to Manned-Unmanned Teaming (MUM-T) performance to support development and integrations of intelligent aids/crew enabling technologies [2]. The Autonomy capability effort looks at the testing of autonomous leader follower (LF) software in a simulated environment.

2.1. Human in the Loop

The goals of the Crew Optimization and Augmentation Technologies (COAT) research study were as follows; Conduct missions with active duty Soldier operators to obtain feedback on current system capabilities and performance, Capture baseline data related to Manned-Unmanned Teaming (MUM-T) performance to support development and integration of intelligent aids/crew enabling technologies, and to inform the process used to collect and analyze MUM-T research data, specifically team performance measurement(s); ideally to refine a set of metrics and procedures to be used for future testing [2]. These goals required a detailed, operationally relevant virtual environment to adequately focus the presence of the participants in the scenario [3]. To provide more realism the COAT experiment used a six degree of freedom motion platform, Crew Station/Turret Motion Based Simulator (CS/TMBS), shown in Figure 1.

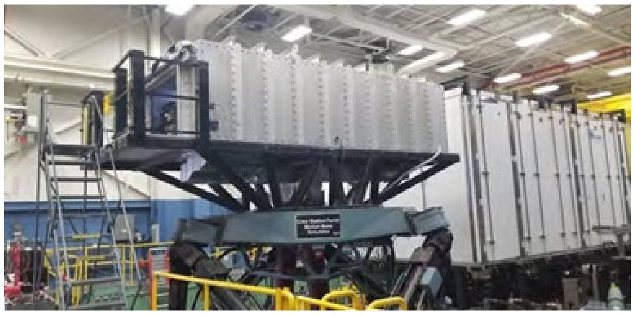
Figure 1 - CS/TMBS with Cab

Three major components make up the user experience when interacting with a MUM-T system. The controller, the algorithms, and the environment.

The controller constitutes the system with which an operator interacts with a robotic asset. In this case, it was through the use of physical controls, and touch screen displays, running a human-machine-interface software application.

The software within the vehicle, the crew stations, shown in Figure 2, and the associated command and control (C2) systems consists of many individual algorithms. The unique system composed of a particular set of these algorithms is the system under test for this experiment.

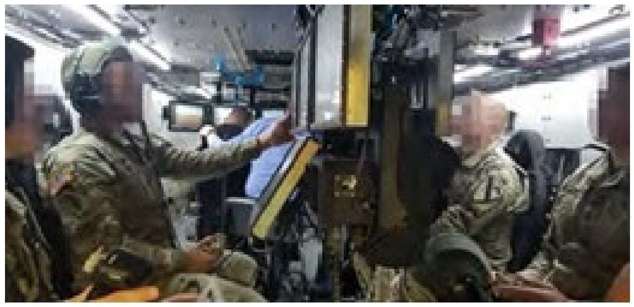
Figure 2 - Robotic Combat Vehicle (RCV) Crew station

The virtual environment, shown in Figure 3, must be capable of providing coherent stimuli to the participants of the study, and the system under test that allow tasks and responses to be reflective of military operational tasks and responses. This is a combined goal with that of presence [3] of the participants, to ensure that environment is effective.

In particular, the environment must provide synchronized sensor feeds which share a cohesive state. Each participant must be able to complete individual tasks that enable the unit-under-test's warfighting functions. The individual members of the section must be able to accomplish tasks that enable movement and maneuver, intelligence, fires, sustainment, and protection[4].

Figure 3 - RCV Conducting an area defense during COAT Experiment

2.2. Autonomy in the Loop

The Program Executive Office Combat Support and Combat Service Support (PEO CS&CSS) goals for the Continuous Autonomy Simulation Test Laboratory Environment (CASTLE) is to develop a virtual test harness for Autonomy-in-the-loop in order to reduce program risk and augment testing of its emerging programs. PEO CS&CSS, GVSC and the Aberdeen Test & Evaluation Center (ATEC) endeavor to verify and validate elements of CASTLE like the virtual environment, shown in Figure 4, and operational vignettes to establish a level of confidence in the ability of the virtual environment to provide adequate stimulus to the autonomy software in order to measure the performance of the system behaviors.

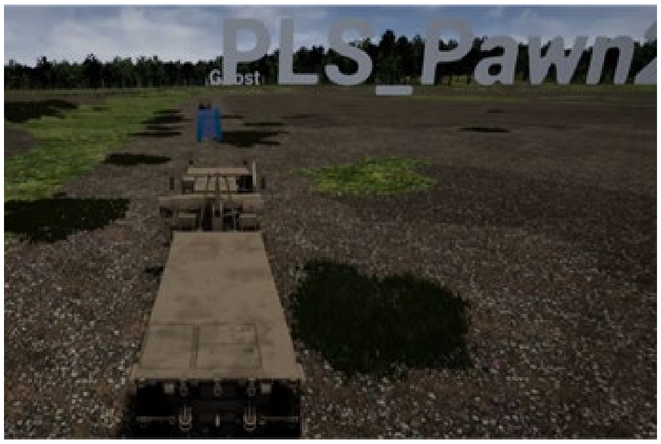

Figure 4 - CASTLE running a playback simulation

We define "Autonomy-in-the-loop", shown in Figure 5, as a system where an autonomous vehicle algorithm is included in a test setup where the inputs are produced in real-time from a simulation framework, and the outputs of the algorithm are fed back to a simulation framework. In this sense, the autonomy algorithm is a 'black-box' within the simulation framework. This is a valuable configuration which allows for testing of autonomy algorithms in a wider variety of operational environments. The virtual environment needs to provide the same stimuli the autonomy system would expect if it is in the real world like data from the vehicle, sensors (e.g. cameras, RADAR, LIDAR …), localization, terrain, environment and communications (e.g. radios). The simulation engine that GVSC has been using for the past ten years was Autonomous Navigation Virtual Environment Laboratory (ANVEL). ANVEL's Application Programmer Interface (API) allowed plugins to be created to support how the autonomy software communicates with the simulation as shown in Figure 5. An example of this is an ANVEL LIDAR plugin was created that provides the Velodyne UDP packets in the sa` me way the real world Velodyne LiDAR's Puck 16 Channel (VLP16) for the autonomy system.

The Autonomy-in-the-Loop also needs the ability to run 1000's of scenarios using

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300

automation to evaluate edge cases, do regression testing, verify software updates provided via a new feature, address an issue or fix a problem. This can be done by developing a way to automate the tasks that a developer, soldier or tester will need to do for fully exercising the capability and features of the Autonomy System. CASTLE's current configuration supports the Expedient Leader Follower (ExLF) which is a convoy march unit with a lead vehicle driven by a soldier and follower vehicle(s) are driven by the autonomy system. CASTLE has the ability to playback a path (recorded from live or simulation data) for the lead vehicle, so a human is not required to drive vehicle during testing.

In CASTLE there is the need to have a scenario or series of scenarios designed to support debugging/testing a capability or feature for the Autonomy-in-the-loop. The goal is to have the CASTLE user be able to develop scenarios that support the desired configuration to be tested and evaluated. An example of this could be the desire to test how well an ExLF Convoy makes a left turn at different gap distances on different courses. The scenario scripts will allow the CASTLE user to easily change one parameter at a time or change all the parameters to run 100's or 1000's of scenarios testing all the different combinations or only the edge cases. As GVSC developed CASTLE it became apparent that ANVEL would not support the future requirements that GVSC desired in a simulation framework for the Autonomy-in-the-loop. ANVEL had a small user community and any new capabilities desired required the government to provide all the investment. In late 2018 GVSC and PdM ALUGS made the decision to start looking at how to transition CASTLE from using ANVEL to Unreal Engine 4 (UE4). The reason to move to UE4 was a larger user community, free to use, already being used by autonomous vehicle projects (General Motor's Cruise[6], Ford[7], CARLA[1]..), API, documentation and training available for free, supports headless and can run in a container, other Army organizations using.

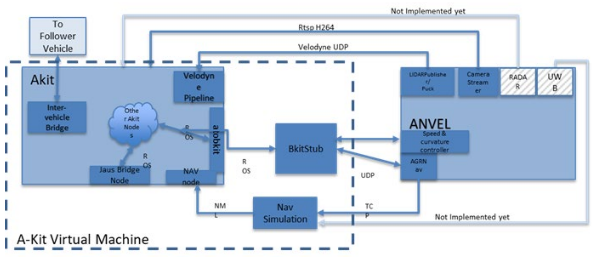

Figure 5 - CASTLE ExLF Follower Configuration

3. SIMULATION FRAMEWORK

In order to simulate the behavior of the vehicles within a scenario, a system of software was developed and integrated (see Figure 6 - MUMTSF Block Diagram). This system constitutes the simulation framework which fed data to the Interfaces and Algorithms under test. The system is based on the open source Unreal Engine 4 maintained by Epic Games [5]. The Unreal Engine provides a sophisticated rendering pipeline which can produce photo-realistic images in real-time. This enables the streaming of game imagery to controllers and algorithms that mimic a variety of sensor systems. In addition, the engine provides a robust networking implementation in the form of a centralized server, and data "Replication". These sub-systems are described in detail below.

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300

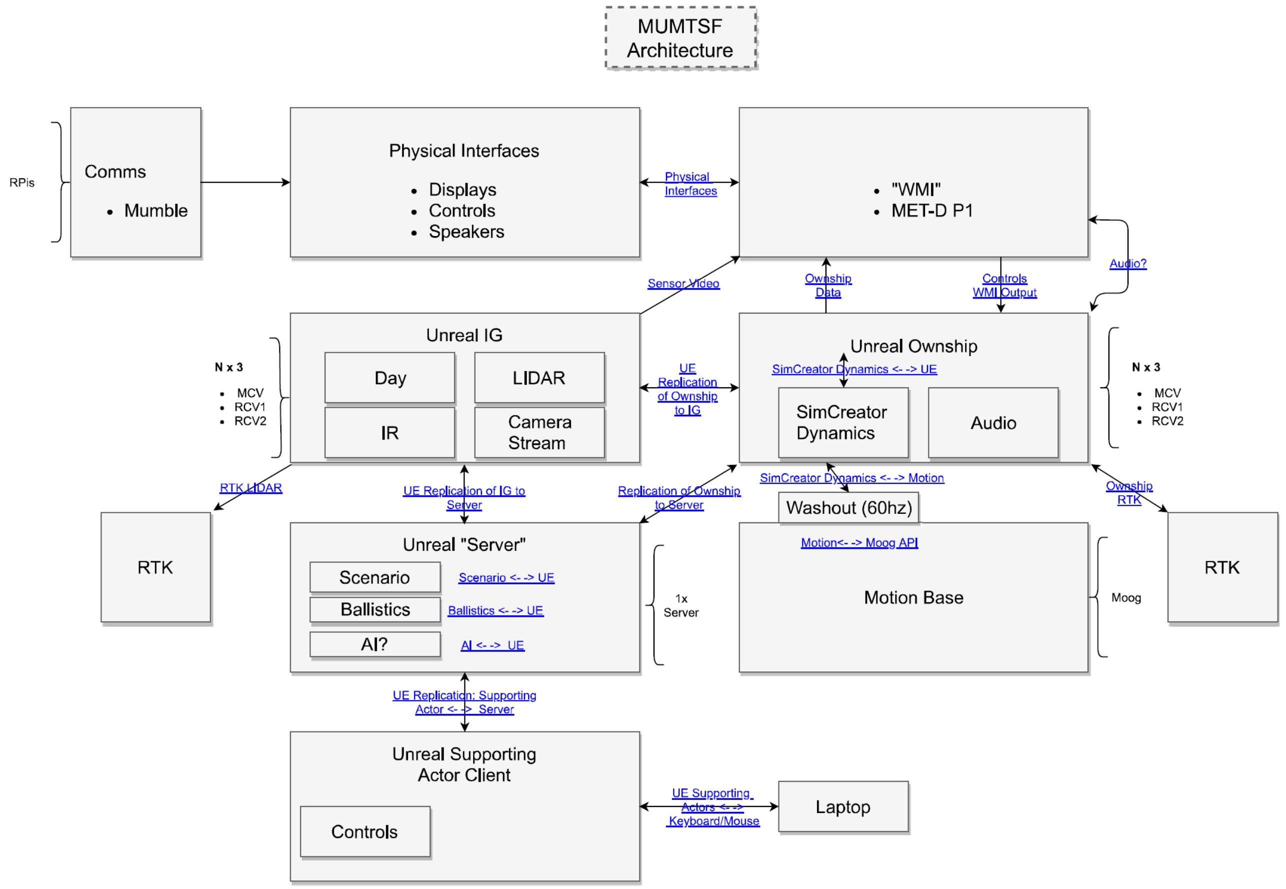

Figure 6 - MUMTSF Block Diagram

3.1. Sensor Simulation

Sensors define how the external systems (humans, software, or hardware) perceive the virtual world. In order to facilitate the objectives several types of system sensors were simulated and passed to the "in-the-loop" systems. These included Electro-Optical/Infra-Red (EO/IR) sensors (in daylight and IR modes), laser range finder (LRF) sensors, RADAR, LIDAR, and Global Positioning System (GPS) /Inertial Measurement Unit (IMU) sensors. Each sensor was implemented to consume data either from the simulation environment or from the vehicle dynamics sub-system. Updates were provided to the external systems every frame (approximately 60Hz). The creation of these sensor feeds is accomplished by sampling of the rendered feeds within the simulation environment, and applying shaders or post process effects, or through custom developed plugins.

Each simulated platform can have a multitude of sensors, and the skeletal meshes of those models are configured with "Sockets" to indicate the locations of sensors. The sensors are configured with basic parameter such as field-of-view (FOV), resolution, and pan-tilt properties as defined by the mission scenario and platform specifications. The simplified models allow for real time per-frame generation of sensor data which can be streamed to external applications to mimic a physical vehicle.

3.2. Physics Simulation

In order to provide appropriate fidelity experiences for agents, the results of the physical models describing vehicles, bipedal characters, and

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300

projectiles must be calculated in real-time and updates across all participating clients in the simulation. Primary information about actors within the environment was contained on the game server. All entities within the simulation registered their position with the game server to be replicated to all other clients.

Ownship vehicles (the vehicles controlled by the physical controls stations) were subject to additional processing for physics calculations. The inputs from the participants were recorded from the physical and touch screen interfaces, and packetized into UDP datagrams. The control messages were sent directly to separate vehicle dynamics processes for each platform. The vehicle dynamics process was an instance of the FAAC Real Time Incorporated (RTI) SimCreator program, configured for either the driving dynamics of a M113 (surrogate RCV vehicle), or a M2A3 (surrogate MCV vehicle). The processes calculated the vehicle dynamics solutions at a high frequency, and packaged up the net movement to the vehicle center of gravity into a UDP datagram which was sent at 60Hz to the central UE4 server. The server then updated the position of the actor representing that vehicle in the course of each frame.

3.3. Networking

It is critical that all connected clients within the simulation experience the same environment, and are able to reason about the virtual "world" with a shared context. Unreal Engine provides a method to execute this model, by declaring a central server as the authoritative source of truth, and providing mechanisms for clients to both submit updates to server data, and to receive changes to that data.

4. OPERATIONAL EXPERIMENTATION

The simulation environment was chosen to execute a soldier touch point within the GVSL in November of 2019. 16 participants operated a simulated section (3 vehicles) of MCV and RCV assets over the course of 2 weeks.

4.1. Soldier Touch Point

The goal of the soldier touch point was to provide a set of baseline data of soldier behavior and activity in a virtual environment. The participants were recruited in sections, to crew a single Manned Control Vehicle (MCV), and two Robotic Combat Vehicles (RCV).

4.2. Simulation in Motion

The simulation environment, and integrated applications provided an end-to-end closed loop driving, gunning, and engaging environment for solider participants to execute their operational orders. The use of physics-based vehicle dynamics models provided real-time motion cuing to participants, while visuals on the WMI control stations were driven by Unreal Engine graphics streams over a h264 encoded UDP packet format. The soldiers successfully completed attack, movement to contact, and defense operations with pseudo-simulated opposing forces (OPFOR).

5. CASTLE UE4 DEMONSTRATION

The simulation environment was chosen to demonstrate the feasibility of using UE4 for providing the virtual environment to support Autonomy-in-the-loop as a proof of concept. The goal of the proof of concept was to create a CASTLE UE4 prototype that would demonstrate running the same ExLF four vehicle convoy for a Camp Grayling scenario that was currently running in ANVEL by the end of December 2019, using the knowledge gained during the development of the COAT experiment.

5.1. CASTLE UE4 Prototype

The GVSC MS2 Immersive Simulation software development team's first step was to look at what would be needed to demonstrate the feasibility of using UE4. For creating a CASTLE UE4 prototype the requirements included needing a PLS Vehicle model, Camp Grayling Terrain, LIDAR, Speed &

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300

Curvature controller, ability to communicate with the autonomy kit and navigation, ability to create a path for leader and data collection for pass/fail. One of the first steps was to take the heightmap used for the Camp Grayling terrain in ANVEL and convert that to how UE4 needs heightmaps. Since Camp Grayling terrain is a geo-specific area, this process also required correlation with the real-world location, so leader path data collected from real testing could be replayed to test the autonomy-in-the-loop.

Next step was to create a Palletized Load System (PLS) model which required getting a 3D model of the PLS. A 3D M1075 PLS that included the skeletal/static meshes was available from US Army's MILGAMING website, so that was downloaded and imported into UE4. Since the PLS is five-axle truck, this require using an N-wheeled vehicle movement component which exposes to Unreal the PhysX class for N-wheeled vehicle movement. Once the PLS 3D Model was created, parameters (provided from real world testing) needed to be set and tuned to represent how the PLS physics should perform realistically to how it would work in the real world. A virtual test course was created to test and validate acceleration, braking, turning and climbing grades. The ANVEL Playback plugin was converted to UE4, so that the UE4 Vehicle model could be moved using a path generated from live vehicle testing or generated by driving a PLS vehicle in simulation. The ANVEL Path Painter plugin was converted to UE4, so that CASTLE developers can drive the PLS vehicle in simulation and create a path for testing the UE4 PLS model working with Playback. Playback required testing how well the correlation of a path created from real world ExLF data capture worked, based on differences between ANVEL and UE4.

For communication with the Autonomy-in-the-loop the ANVEL Speed & Curvature and Autonomous Ground Resupply Navigation (AGRNAV) plugins were converted to UE4. The Autonomy runs on a Linux Virtual Machine (VM) that includes Robotic Operating System (ROS) Core, the ExLF Autonomy Software, Navigation Simulation (converts the simulation localization information to format Autonomy needs) and By-Wire Stub (sends/receives messages between autonomy & simulation). In the real ExLF system there is a Navigation system that provides the localization of vehicle to the Autonomy but when running the Autonomy-in-the-loop the simulation provides via the UE4 AGRNAV plugin and the Navigation simulation, which converts into correct format for Autonomy Software. Also on the real ExLF there is a By-Wire system that provides updates, controls moving and stopping the PLS via communication with the Autonomy, in simulation this is done via By-Wire stub communicating with the Speed & Curvature controller and AGRNAV. The LIDAR allows the Autonomy to determine if there are any obstacles in the path of the PLS. A new UE4 LIDAR plugin was created that could support two simulated VLP-16s attached to the 3D PLS vehicle model and send the same Velodyne UDP stream. The CASTLE ANVEL version also had the ability to collect data from both the Autonomy and the simulation to determine if the scenario passed or failed this Sentry Plugin was also converted to UE4 in support of the proof-of-concept.

The ANVEL version of CASTLE had one computer represent one vehicle (which included the Autonomy SW VM, simulation SW) this same configuration was kept for CASTLE UE4 prototype, this included one Lead Vehicle simulation computer and three Follower Vehicle simulation computers. When using ANVEL as the simulation engine, each instance of ANVEL that is used to represent a vehicle uses the Distributed Interactive Simulation (DIS) plugin to communicate with other ANVEL instance the state

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300

of the vehicle. The UE4 version of CASTLE decided to use the built in UE4 replication software for the communication between the UE4 instances representing each vehicle, this provided the additional benefit of communicating everything each UE4 instance is doing in simulation. Once all the UE4 plugins were created and tested for one vehicle, they needed to be integrated for running an ExLF convoy. A decision was made to run with a UE4 Listen Server and each vehicle simulation using the UE4 editor and communicating with the Listen Server as shown in Figure 7. There were some challenges getting the communication correct with the speed & curvature controller, replication and correlation but by Dec 31, 2019 the development team was able to demonstrate a four vehicle ExLF convoy working with Autonomy-in-the-loop.

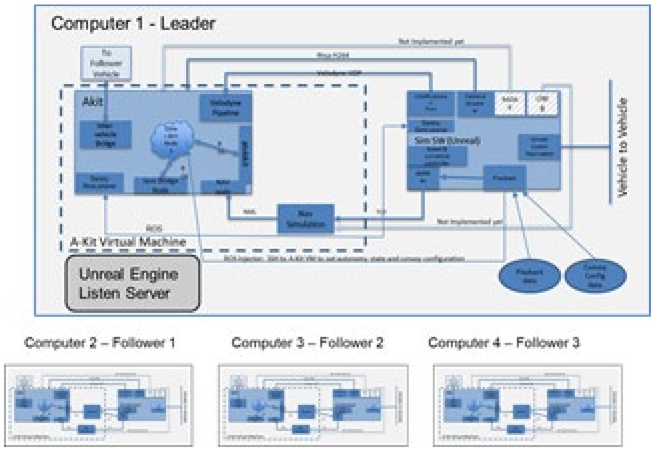

Figure 7 - CASTLE UE4 Prototype

6. CONCLUSIONS AND FUTURE WORK

The GVSC MS2 Immersive Simulation has begun to demonstrate the feasibility of a gaming engine (Unreal Engine 4) to drive the simulation of both human and autonomy in the loop experimentation. This design will allow the US Army to invest in the additional capabilities needed for the simulation framework above what is provided by Unreal Engine, and to take advantage of the features that a gaming company like Epic Games is already creating to keep the gaming engine as modern as possible to support games like Fortnite and the large user community using Unreal Engine.

The GVSC MS2 Immersive Simulation software development team has taken the knowledge gained from COAT experiment and CASTLE UE4 prototype demonstration to develop a core baseline simulation application using UE4 that will be able to support Immersive Simulations three capabilities of Warfighter Experimentation (looking at Tactics, Techniques & Procedures); crew station design and development; and Autonomy (Autonomy-in-the-loop). Follow on experimentation is also scheduled using the methods and software described here.

1. REFERENCES

[1] A. Dosovitskiy, "CARLA: An Open Urban Driving Simulator," p. 16.

[2] Immersive Simulations, "NGCV MUM-T VE COAT-OCT19." Nov. 01, 2019.

[3] B. G. Witmer and M. J. Singer, "Measuring Presence in Virtual Environments: A Presence Questionnaire," *Presence Teleoperators Virtual Environ.*, vol. 7, no. 3, pp. 225–240, Jun. 1998, doi: 10.1162/105474698565686.

[4] U.S. ARMY, *ADP 3-0 OPERATIONS.* Army Publishing Directorate, 2019.

[5] Epic Games, *Unreal Engine.* .

[6] K. Wiggers, "GM's Cruise is preparing for a self-driving future in the cloud," VentureBeat, APR 20, 2019.

[7] A. Jayaraman, A. Micks, and E. Gross, "Creating 3D Virtual Driving Environments for Simulation-Aided Development of Autonomous Driving and Active Safety," WCX™ 17: SAE World Congress Experience, Detroit, MI, April 4-6, 2017

Distribution A. Approved for public release; distribution unlimited. OPSEC Number: OPSEC4300