REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggesstions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA, 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any oenalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
09-06-2021	Final Report	1-Nov-2020 - 30-Apr-2021

4. TITLE AND SUBTITLE
Final Report: ISimA: Intelligent Simulation for Testing and Training of Autonomous Single- and Multi-Vehicle Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER
W911NF-21-P-0004

5c. PROGRAM ELEMENT NUMBER
665502

6. AUTHORS

5d. PROJECT NUMBER
665502

5e. TASK NUMBER
665502

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
RobotWits, LLC
1100 Crescent Pl

Pittsburgh, PA 15217 -1111

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
77853-MA-ST1.1

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jonathan Butzke
UU	UU	UU	UU		19b. TELEPHONE NUMBER
					860-984-8766

RPPR Final Report
as of 10-Jun-2021

Agency Code: 21XD

Proposal Number: 77853MAST1 **Agreement Number: W911NF-21-P-0004**

INVESTIGATOR(S):

Name: Jonathan Butzke
Email: jon@robotwits.com
Phone Number: 8609848766
Principal: Y

Organization: **RobotWits, LLC**
Address: 1100 Crescent Pl, Pittsburgh, PA 152171111
Country: USA

DUNS Number: 079139167 EIN:

Report Date: 30-Jun-2021 Date Received: 09-Jun-2021

Final Report for Period Beginning 01-Nov-2020 and Ending 30-Apr-2021

Title: ISimA: Intelligent Simulation for Testing and Training of Autonomous Single- and Multi-Vehicle Systems

Begin Performance Period: 01-Nov-2020 **End Performance Period:** 30-Apr-2021

Report Term: 0-Other

Submitted By: Jonathan Butzke Email: jon@robotwits.com
Phone: (860) 984-8766

Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 0 **STEM Participants:** 1

Major Goals: The ISimA project was designed to develop a state-of-the-art software simulation system leveraging commercial off-the-shelf simulation engines to allow for the testing, training, and evaluation of autonomous single- and multi-vehicle systems. In Phase I of the project, we focused on the following three questions in order to prove the feasibility of the overall architecture:

1. The first question is whether it is feasible to build a single simulation tool that will provide adequate fidelity of simulation for testing and training autonomous single- and multi-vehicle systems operating off-road. In particular, the simulator will need to provide proper simulation of sensing (both LIDAR and RGB camera), vehicle dynamics, vehicle-to-terrain interaction, limited communication bandwidth between multiple vehicles, and simulation of the behavior of the virtual world.
2. The second question is whether it is feasible to develop an intelligence engine for generating test scenarios and dynamic control of dynamic entities within ISimA to explicitly optimize the level of confidence in the ability of autonomous vehicles to achieve their designated tasks.
3. The third and the final question is whether ISimA can be built on top of open-source software under proper licenses that will allow for unrestricted use of and the ability to extend ISimA by others.

To support those major goals, we have completed the following tasks:

1. Evaluation of existing simulator systems;
2. Designed the ISimA framework;
3. Designed the Intelligent Scenario Generation framework;
4. Designed a methodology to support custom vehicle-terrain interfaces;
5. Implemented and demonstrated the feasibility of key elements of the ISimA and Intelligent Scenario Generation frameworks.

Accomplishments: Completed all tasks as part of the Phase I project.

1. Evaluation of existing simulator systems - complete Dec 20;
2. Designed the ISimA framework - complete Apr 21;
3. Designed the Intelligent Scenario Generation framework - completed Apr 21;
4. Designed a methodology to support custom vehicle-terrain interfaces - completed Apr 21;
5. Implemented and demonstrated the feasibility of key elements of the ISimA and Intelligent Scenario Generation frameworks - completed Apr 21.

See uploaded PDF for more detail.

RPPR Final Report
as of 10-Jun-2021

Training Opportunities: Nothing to Report

Results Dissemination: Held discussions with Department of Energy, the Army Cold Regions Research Laboratory, Ground Vehicle Systems Center, and other potential future users to discuss their needs regarding a simulation framework. We have incorporated their feedback into our design as well as in our follow-on Phase II proposal.

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: Nothing to Report

PARTICIPANTS:

Participant Type: PD/PI
Participant: Jonathan Michael Butzke
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Other Professional
Participant: Dan DeLano
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Faculty
Participant: Maxim Likhachev
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

Participant Type: Graduate Student (research assistant)
Participant: Manash Pratim Das
Person Months Worked: 6.00 **Funding Support:**
Project Contribution:
National Academy Member: N

RPPR Final Report
as of 10-Jun-2021

Partners
Carnegie Mellon University
Pittsburgh, PA USA
4
CMU and RobotWits collaborated throughout this project including on conducting detailed analysis of existing simulation systems, generation of the simulation system framework, design of the vehicle-ground interaction interface and conducting the demonstration and assessment of the prototype simulation framework.

I certify that the information in the report is complete and accurate:
Signature: Jonathan Michael Butzke
Signature Date: 6/9/21 7:01PM

1. Contract and Proposal Number: W911NF21P0004 A20B-T006-0268

2. Contractor's name and address: Dr. Jonathan Butzke, 5001 Baum Blvd STE 434, Pittsburgh, PA 15213

3. Title of the project: ISimA: Intelligent Simulation for Testing and Training of Autonomous Single- and Multi-Vehicle Systems

4. Contract performance period: 01 Nov 2020 – 30 Apr 2021

5. Total contract amount: $166,478.30

6. Amount of funds paid by DFAS to date: $138,731.92

7. Total amount expended/invoiced to date: $166,478.30

8. Number of employees working on the project: 2 + 2 subcontractors (CMU)

9. Number of new employees placed on contract this month: 0

STTR Topic: Intelligent Simulation for Testing and Training Autonomous Teams (ISimA)

1 Overview and Project Objectives

The ISimA project was designed to develop a state-of-the-art software simulation system leveraging commercial off-the-shelf simulation engines to allow for the testing, training, and evaluation of autonomous single- and multi-vehicle systems. In Phase I of the project, we focused on the following three questions in order to prove the feasibility of the overall architecture:

1. The first question is whether it is feasible to build a *single* simulation tool that will provide adequate fidelity of simulation for testing and training autonomous single- and multi-vehicle systems operating off-road. In particular, the simulator will need to provide proper simulation of sensing (both LIDAR and RGB camera), vehicle dynamics, vehicle-to-terrain interaction, limited communication bandwidth between multiple vehicles, and simulation of the behavior of the virtual world.
2. The second question is whether it is feasible to develop an intelligence engine for generating test scenarios and dynamic control of dynamic entities within ISimA to explicitly optimize the level of confidence in the ability of autonomous vehicles to achieve their designated tasks.
3. The third and the final question is whether ISimA can be built on top of open-source software under proper licenses that will allow for unrestricted use of and the ability to extend ISimA by others.

To answer these questions, we have designed a simulation architecture called ISimA and tested critical technologies as a proof-of-concept. There are three main characteristics that set this software apart from existing simulation tools. First, ISimA provide a suite of simulation capabilities directed specifically towards the operation of single- and multi-vehicle systems performing missions in unstructured environments and over a wide range of terrains. Second, ISimA is integrated with an intelligent scenario generation capability for the automated creation of test scenes and supporting independent autonomous control of dynamic entities present in the virtual world in order to dynamically provide test conditions that build up confidence in the ability of autonomous vehicles to accomplish their designated tasks. Finally, ISimA is designed on top of existing open-source software and allows for extension of the baseline system by other parties.

1.1 Summary of Phase I Work Performed

In phase I we have completed five primary tasks supporting our overall objective.

1. Evaluation of existing simulator systems
2. Designed the ISimA framework
3. Designed the Intelligent Scenario Generation framework
4. Designed a methodology to support custom vehicle-terrain interfaces
5. Implemented and demonstrated the feasibility of key elements of the ISimA and Intelligent Scenario Generation frameworks

2 Evaluation of Existing Simulator Systems

Before we started designing the ISimA system, we first evaluated the existing open-source simulation software. We looked at all of the features required to adequately support our goal of a system capable of training, testing, and evaluating autonomous single- and multi-vehicle systems. In particular, we evaluated each candidate under the following criteria:

1. Physics Engine – fidelity and accuracy
2. Terramechanics – modeling of tire-ground interface, physics support for deformable surfaces, off-road capabilities, difficulty in adding new tire-ground interface models
3. Customization – ability to generate custom maps and environments, modeling scenarios
4. Open-source/licensing – what are the restrictions on licensing
5. ROS support – capabilities to integrate with ROS or other middleware
6. Photo-realistic rendering – ability to support high-definition camera models for accurate simulation of camera-based sensors
7. GPU support – ability to speed up processing by using the GPU
8. Built-in sensor support – which sensors are available off-the-shelf such as LIDAR, RADAR, GPS, IMU, etc., how difficult to add custom sensors
9. Scenario details – pedestrians, weather conditions, different/custom vehicles and their associated intelligence capability/interface
10. Plug-in support – ability to automate aspects of the simulation system or tune environmental parameters via plug-in API
11. Operating System support
12. Communications modeling – inter-vehicle communications modeling with the capability to insert delays, simulate packet loss, or model occlusions/dead zones

Based on these 12 criteria we examined 13 candidate simulation systems, both open- and closed-source. While none of the candidates were able to perform all of the required elements, two broad categories met the majority of requirements and also had a significant installed user base developing new capabilities. These two groups were the Unreal Engine based simulators, to include the two derivatives: CARLA (self-driving car focused), and AirSim (autonomous aerial vehicle focused); and the Unity simulator system. The full comparison and evaluation details can be found in Appendix 1.

Ultimately, we selected the CARLA simulator running on Unreal Engine as our baseline simulator for three primary reasons:

1. Large installed user base meant large amounts of support and expertise exists for making modifications and implementing new features
2. Well defined capability to create custom scenarios easily as well as stable API for interacting with simulator system
3. Built in autonomous vehicle capabilities allow for jumpstarting new project development

3 FRAMEWORK FOR ISIMA

Overall, the ISimA framework was responsible for supporting the simulation of off-road navigation tasks for single- and multi-vehicle test scenarios. These scenarios could encompass a variety of terrain types, weather conditions, communication environments, and vehicle types. ISimA was designed to support this variety through a combination of purpose designed components coupled with existing open-source simulation engines along with a scheme to allow for user created or specified plug-ins to augment this baseline system as required.

The ISimA framework is built around the Unreal Engine 4 and CARLA functionality to provide the additional flexibility and functionality required to support off-road simulation of multiple vehicles. The primary ISimA components are built on the Unreal Engine foundation with custom CARLA plug-ins to extend the functionality to the CARLA specific capabilities. By using this setup, we can reuse existing capability from both CARLA and Unreal Engine, while allowing for new functionality across both. The overall system is shown in Figure 1.

Plugins provide the implementation of objects such as Autonomous Agents, Weather Engine, Communications Emulator, and Sensors. To enable any of these objects in the simulator, they must be spawned into the Virtual World via the Runtime Environment Builder or Autonomous Agent Handler(s).

Autonomous Agent Handler(s) deals with spawning / despawning only autonomous agents (each agent can have its own handler), while Runtime Environment Builder deals with spawning / despawning anything other than autonomous agents, such as landscape, map objects, world sensors, weather engine components, etc.

Once all the necessary objects are spawned into the Virtual World, all the runtime data such as controlling the agents, reading scenario performance, world state, ground-truth navigation maps etc., can be accessed from the objects in the world via the APIs as Runtime World Data.

Thus, ISimA has only three Data Interfaces: two of them only deal with spawning / despawning (Environment Description and Autonomous Agent Description), and the last one deals with run-time data.

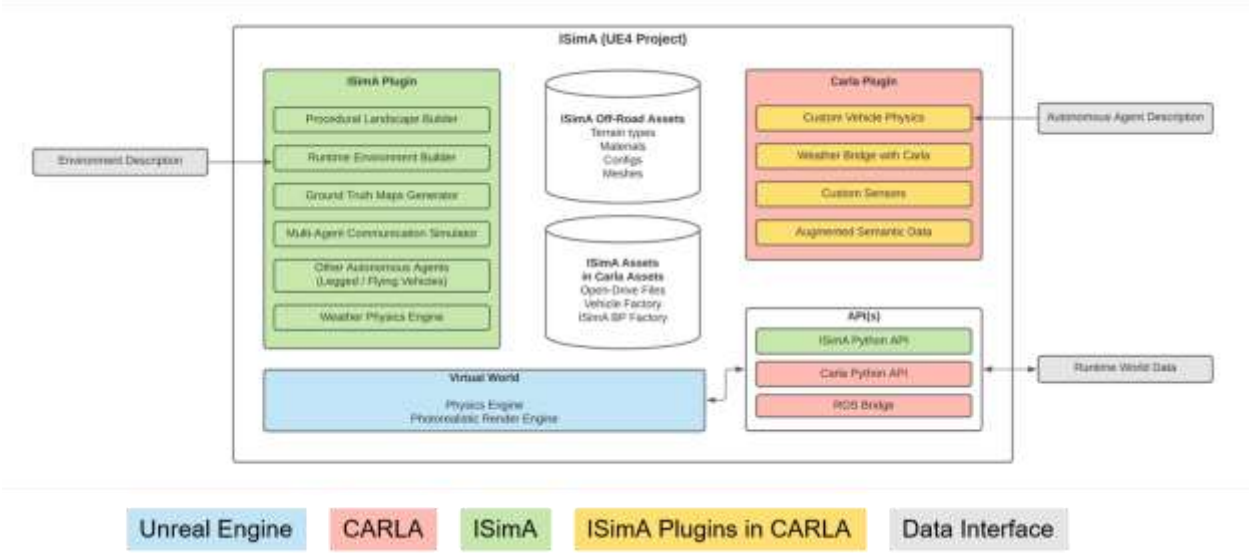

Figure 1: ISimA Overview

The Unreal Engine based components are:

1. Runtime Environment Builder – creates the simulator environment for a given test run based on the Environment Description provided. This also allows other objects to be spawned in the simulation.
2. Procedural Landscape Builder – constructs the physical landscape in the simulator based on directions from the Runtime Environment Builder
3. Ground Truth Map Generator – creates the maps used by various other components based on the environment and virtual world generated by the Runtime Environment Builder
4. Multi-agent Communication Simulator – provides a simulation pathway for vehicle-to-vehicle and vehicle-to-infrastructure communications
5. Autonomous Agents – extends the existing Unreal Engine framework for wheeled vehicles to allow for additional motion modalities such as tracked, legged, or aerial vehicles. Additionally, allows for custom physics for wheeled vehicles.
6. Weather Physics Engine – Allows weather effects to modify terrain characteristics and the underlying vehicle physics simulation

Within CARLA there are four planned plug-ins to provide additional capabilities:

1. Custom Vehicle Physics – allows for replacement of the standard NVidia PhysX engine for wheeled vehicles with a custom physics simulation for special case scenarios and to allow for weather-based physics
2. Weather Bridge – ties CARLA weather phenomena to the Weather Physics Engine
3. Custom Sensors – allows for additional sensor modalities as well as weather impacts to sensors
4. Augmented Semantic Data – Provides higher level labeling of elements of the Virtual World to allow for training, testing, and specialized physics

3.1 Unreal Engine-based Components Overview

3.1.1 Runtime Environment Builder

The Runtime Environment Builder is the principal component for establishing and defining the Virtual World within the simulation system and is shown in Figure 2. It is responsible for the creation and destruction of elements within the simulator Virtual World (called, respectively, spawning and despawning) such as landscape, map objects, world sensors, weather engine components etc. The one exception is all controlled vehicles[1] are spawned / despawned by their respective Autonomous Agent. To accomplish this, the Runtime Environment Builder has access to a library of objects such as 3D meshes and weather engine objects, which can be spawned and initialized in the Virtual World.

The Runtime Environment Builder internally calls the Procedural Landscape Builder module to generate 3-D off-road environments without manual 3D modelling. Additionally, it can populate objects for later use by CARLA into the appropriate library. This module can also change the environment during runtime providing for quick scenario changes without requiring time consuming simulator resets. It can also generate and track numerous sub-worlds inside a single Virtual World.

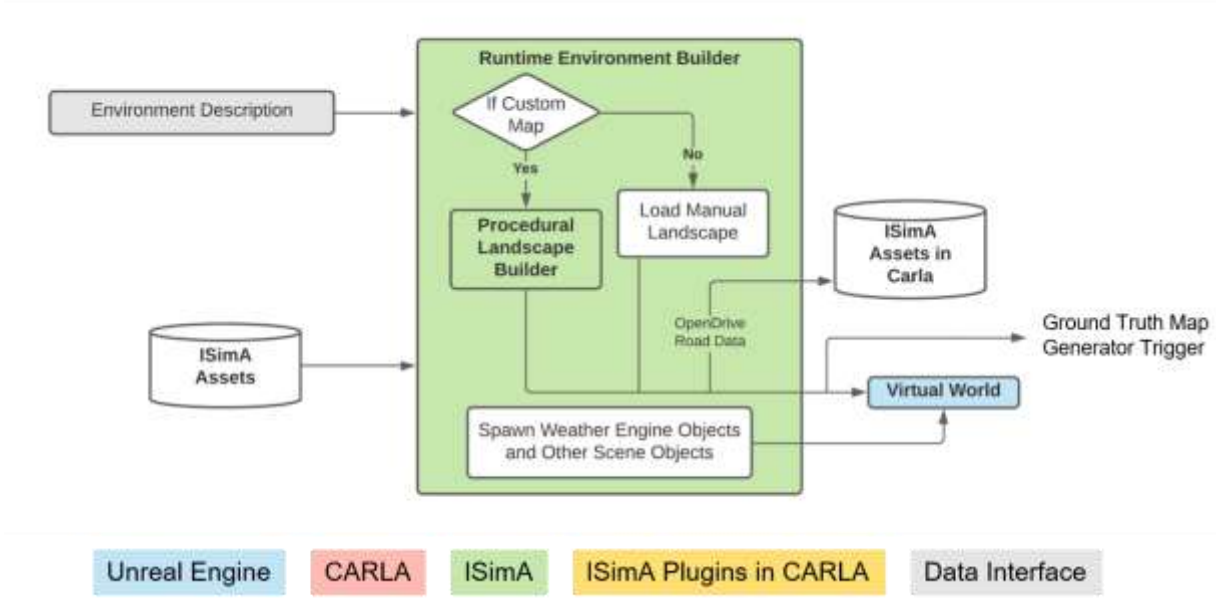

Figure 2: Runtime Environment Builder

3.1.2 Procedural Landscape Builder

This module allows for the generation of 3-D terrains without manual construction based on directions from the Runtime Environment Builder. It can programmatically generate height maps, road networks, and terrain types (such as mud, grass, rock, etc.) and use that information

[1] A controlled vehicle is one that has logic attached to its motion so that it moves about the environment according to some directives. For example, other friendly forces, hostile forces, and moving neutral forces are all controlled vehicles, while a permanently stationary vehicle is considered part of the landscape and is not a controlled vehicle.

for constructing the follow-on OpenDrive formatted map information as well as other required data for the Ground Truth Map Generator. This data is primarily for use by the ISimA system, rather than for the autonomous agents themselves. The Autonomous Agents get their maps from the Ground Truth Map Generator.

3.1.3 Ground Truth Map Generator

Creates the maps used by the Autonomous Agents based on the Virtual World and a trigger from the Runtime Environment Builder. The Ground Truth Maps Generator is depicted in Figure 3. Upon trigger, this module will generate all necessary maps such as Elevation Map, Occupancy Grid, Semantic Segmentation Map, etc. which might be required by the Autonomous Agents for navigation. By separating this functionality from the Procedural Landscape Builder, it becomes possible to introduce mapping errors into the Autonomous Agents' maps while still retaining perfect mapping within the simulator system for other purposes.

It is supported by Augmented Semantic Data module which augments the standard CARLA map data to add semantic labels for objects that can be found in off-road environments.

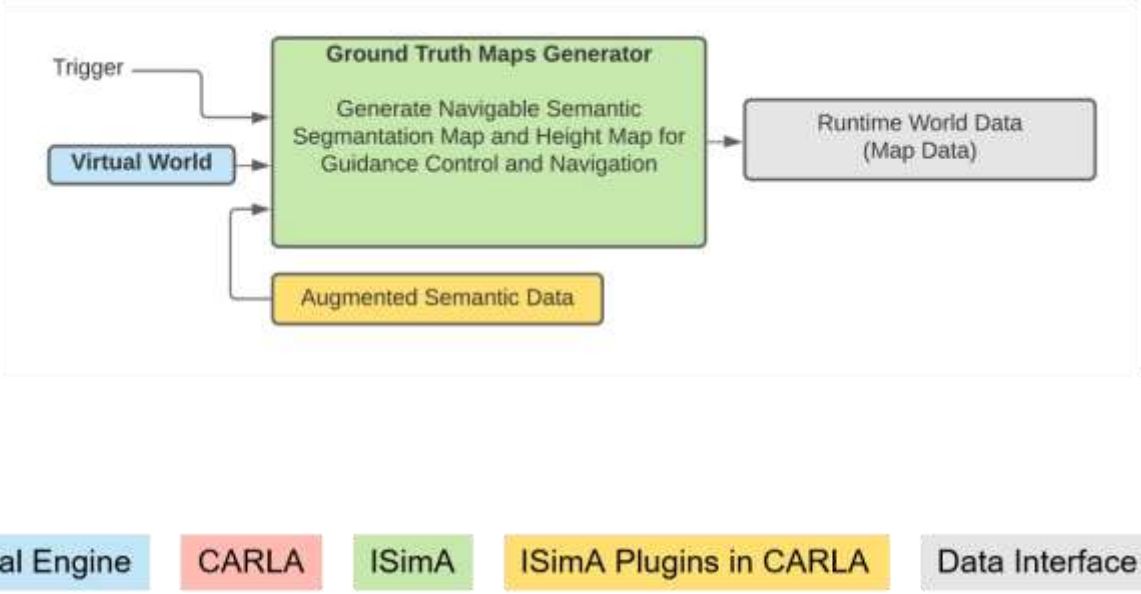

Figure 3: Ground Truth Maps Generator

3.1.4 Multi-agent Communication Simulator

Provides a simulation pathway for vehicle-to-vehicle and vehicle-to-infrastructure communications within the ISimA environment and is shown in Figure 4. This module is capable of limiting communication between transceivers based on geography, distance, jamming, and other considerations. It is designed to support line-of-sight, point-to-point, satellite, and other communication paradigms by allowing for custom communication emulators as plug-ins.

It has two components: one which simulates how a message would pass from point A to point B in the Virtual World considering the transmission media. The other emulates the network graph which can be modified to support any type of network structure.

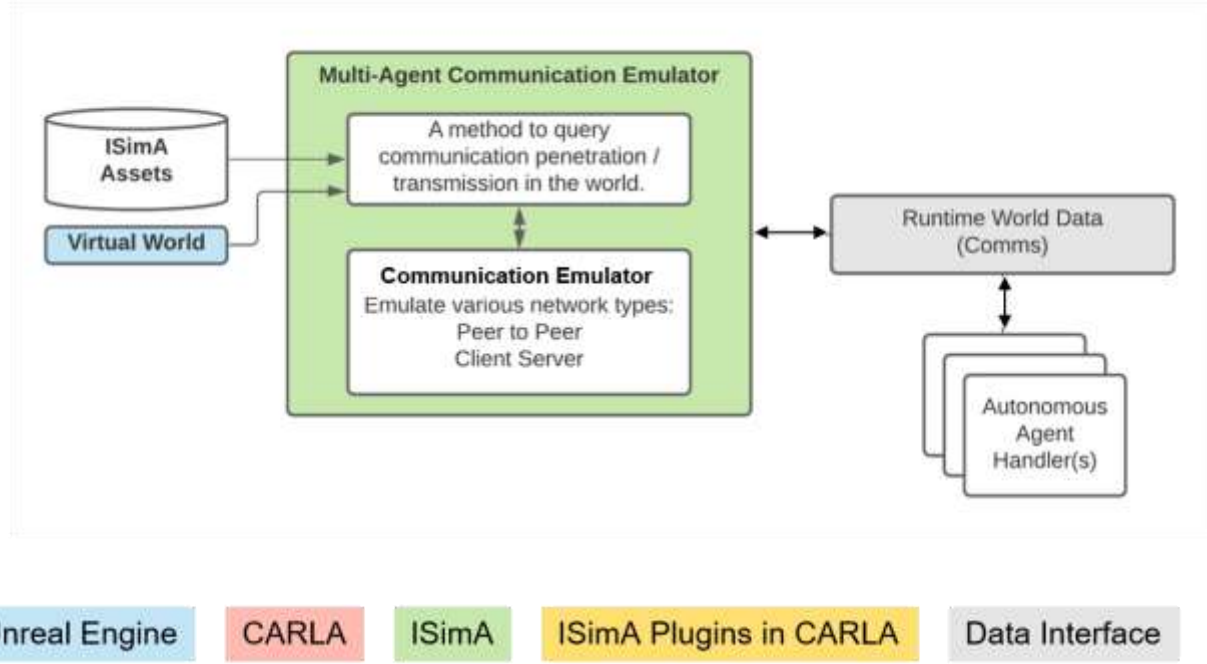

Figure 4: Multi-Agent Communication Emulator

3.1.5 Other Autonomous Agents

The CARLA/Unreal Engine framework natively supports Autonomous Agents for standard wheeled vehicles with Ackermann type steering systems. ISimA extends this functionality by allowing for plug-ins to be added to support a wider array of vehicle types such as tracked, legged, or aerial vehicles. Furthermore, this function allows additional controlled vehicles to be spawned in the simulator allowing for more than one controlled vehicle in a given environment.

3.1.6 Weather Physics Engine

The Weather Physics Engine allows custom weather effects to modify terrain characteristics and the underlying motion control physics simulation as depicted in Figure 5. It performs this by interfacing with the Custom Vehicle Physics as well as simulating the effect of weather on Custom Sensors. This module also relies on the CARLA Weather Bridge to pass information to the CARLA system.

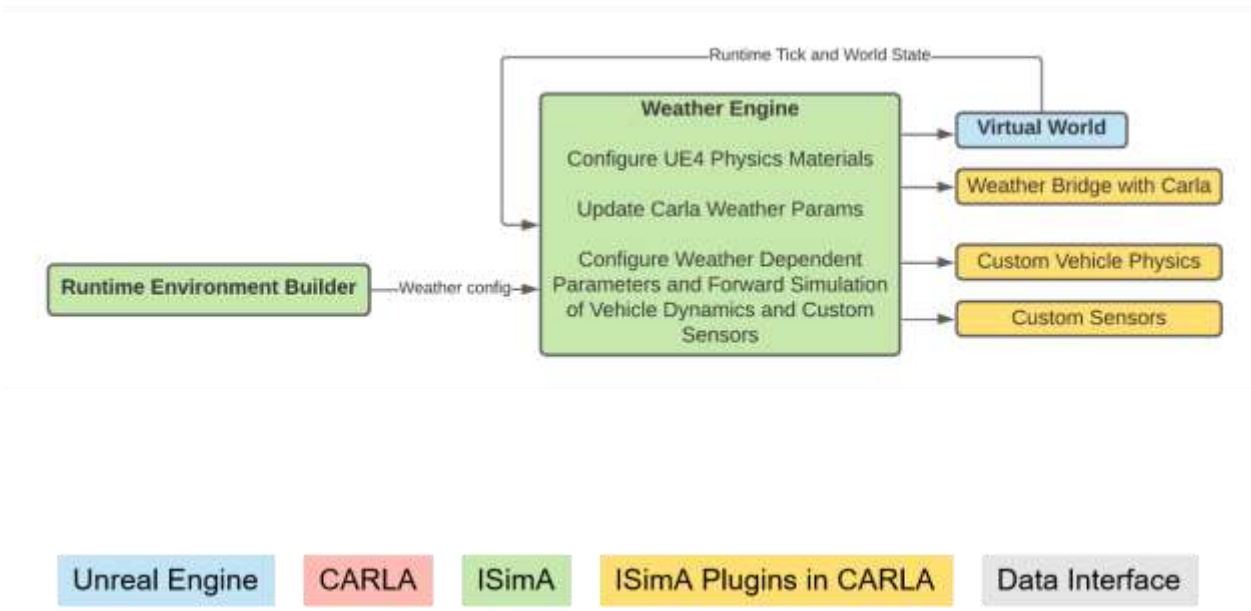

Figure 5: Weather Engine

3.2 CARLA-BASED COMPONENTS OVERVIEW

Within CARLA there are four planned plug-ins to provide additional capabilities:

3.2.1 Custom Vehicle Physics

This module allows for replacement of the standard NVidia PhysX engine for wheeled vehicles with a custom physics simulation for higher fidelity vehicle simulation and to allow for weather-based physics. This module groups all autonomous agents including wheeled vehicles or Other Autonomous Agents such as legged and flying robots into a single framework as shown in Figure 6. This module however does not include pedestrians, as CARLA natively supports pedestrian motion and controls. CARLA natively provides support for wheeled vehicles (CARLA Vehicle Class), which builds upon the wheeled vehicle model of NVidia PhysX (UE4 Vehicle Class). However, ISimA adds a layer between the two allowing for the integration with user-defined custom vehicle and terramechanics models as well as integration with weather-modified terrain functionality. ISimA does allow a standard wheeled vehicle to use the native CARLA interface for control if desired.

This setup does allow for a user to replace the ISimA custom vehicle physics with a module of their own (or from a third-party) without impacting the remainder of the simulation system.

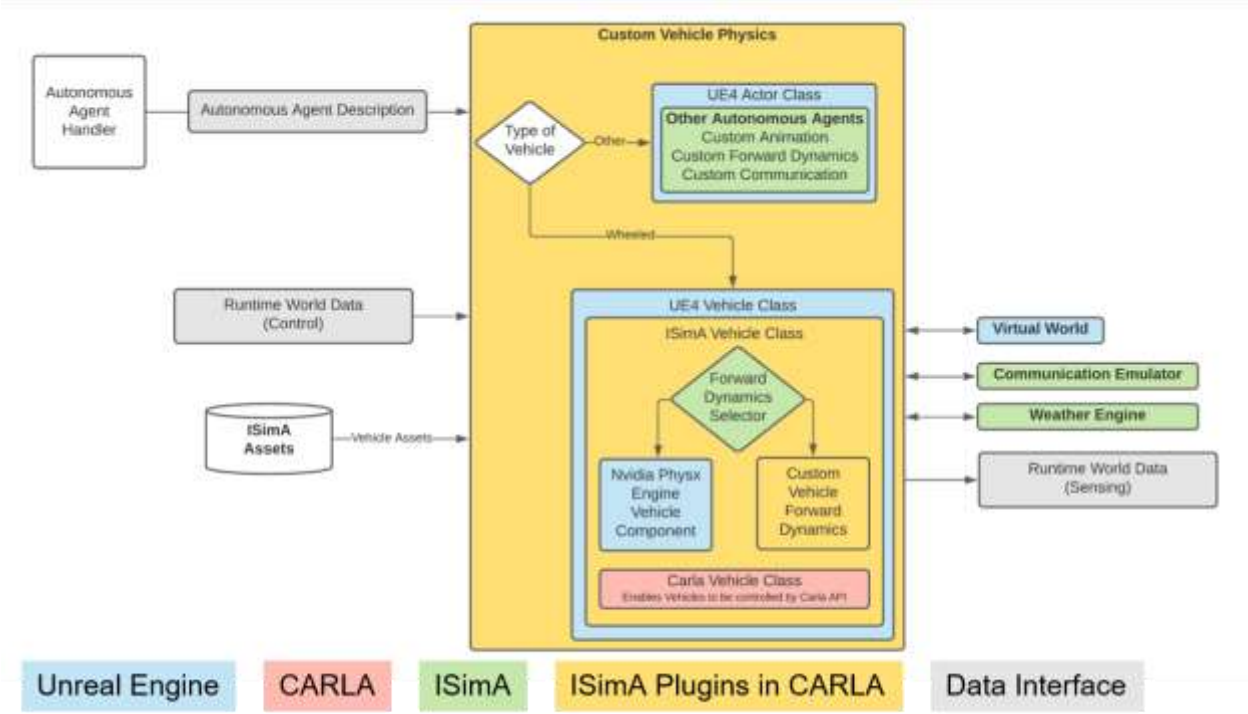

Figure 6: Custom Vehicle Physics

3.2.2 Weather Bridge

Ties CARLA weather phenomena to the Weather Physics Engine and serves as a bridge between the Unreal Engine- and CARLA-based components.

3.2.3 Custom Sensors

Allows for additional sensor modalities as well as weather impacts to sensors. CARLA and Unreal Engine do not natively support weather impact to sensor systems requiring this plug-in capability to provide that feature.

3.2.4 Augmented Semantic Data

Provides higher level labeling of components in the Virtual World to allow for training and testing of Autonomous Agents, as well as for use with the custom physics to apply different effects for different terrain types. An example of this is the generation of semantic segmentation maps or semantic camera features.

4 Framework for Intelligent Scenario Generator (ISG)

Outside of the base simulator system, the second principal component developed under this project was the Intelligent Scenario Generator (ISG) system. This group of sub-components allows for the user to specify the variable parameters to consider while testing an autonomous vehicle stack, then run a series of scenarios intelligently modifying these test parameters to evaluate the stacks performance. As part of ISG, the scenario is analyzed both during and post-run to determine the critical metrics of the autonomous system under test. Based on these, the ISG system determines which parameters to

modify, as well as by how much and in what direction in response to how well or poorly the autonomous system performed.

For example, if the user desires to test an autonomy stacks performance in woodland environments, they may decide to determine at what tree density the stack fails to plan successfully on average. They could specify that the variable parameter is the density of trees, the ISG would generate an initial scenario with a baseline density, run the scenario collecting data on the autonomous stack's performance, then either if the autonomous system performed well, would increase the density and re-run the scenario. The ISG system is not limited to modifying single parameters, nor to using parameters with simple linear effects (i.e., higher tree density monotonically results in lower performance, but cloud cover may initially improve performance by eliminating glare before reducing performance due to low light levels. Other variables may have even more complicated impacts on stack performance.)

4.1 ISG COMPONENTS OVERVIEW

The ISG is comprised of 4 primary components as laid out in Figure 7:

1. Intelligent Parameter Generator (IPG)
2. Scenario Configuration Generator (SCG)
3. Scenario Controller (SC)
4. Scenario Analysis (SA)

These components collectively perform the following tasks during a test run:

1. Generate a scenario from the operator input
2. Configure the simulation system to execute the scenario

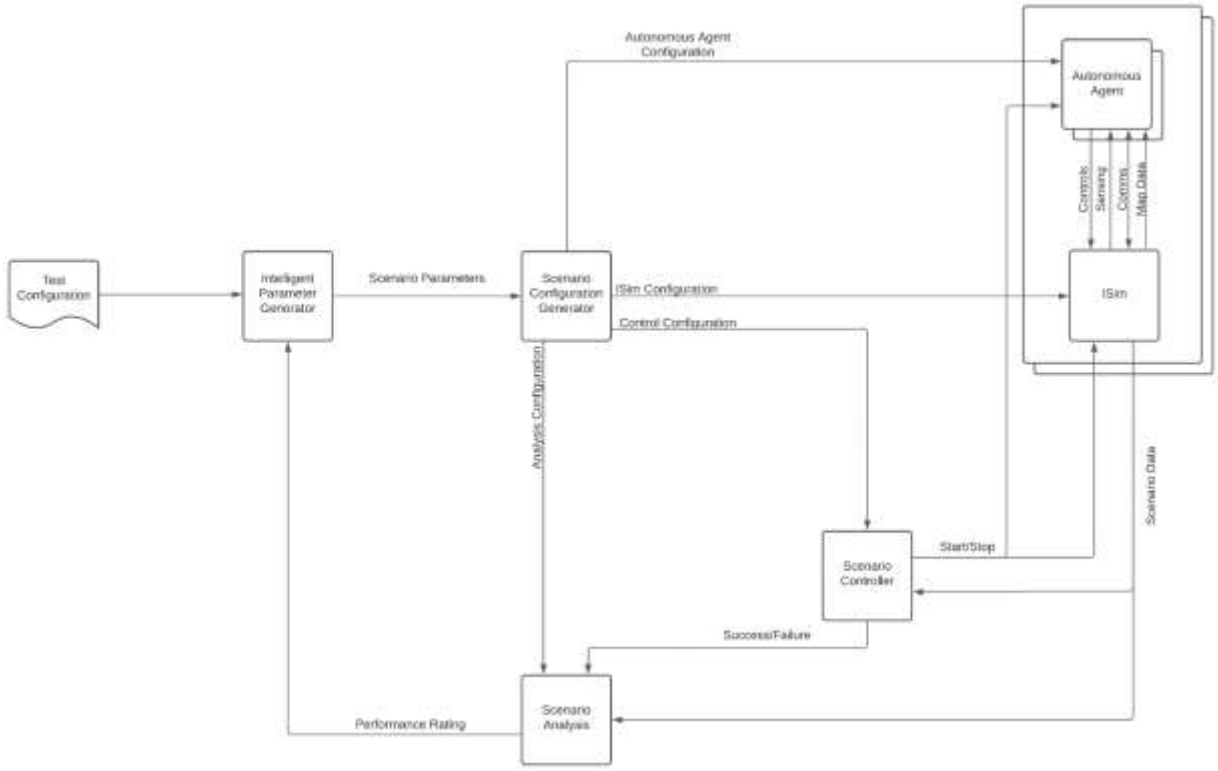

Figure 7: Intelligent Scenario Generator Overview

3. Monitor performance of the autonomy stack during execution
4. Terminate the simulation environment when required criteria are met
5. Perform post-simulation analysis of stack performance
6. Adjust scenario parameters to meet operator requirements based on the analysis

4.2 Assumptions and Requirements for ISG

The ISG sub-system is designed to be highly modular allowing for easy expansion of capabilities to meet operator needs. The initial configuration file that the Intelligent Parameter Generator uses to initialize the test setup is human readable YAML-like configuration file that the operator will use to specify what the parameters to adjust during the test run are, what the analysis termination conditions are, the number and type of vehicles (including whether they are “friendly”, “hostile”, “neutral”, etc.). An example of the configuration file parameters is shown in Appendix B.

However, due to this high level of configurability, the ISG is tightly coupled with the ISimA simulation system. In order to have the ability to modify scenario parameters such as surface type, obstacle placement, and non-player-controlled vehicles, the ISG must communicate with a simulation system that supports the required API layer for these items.

4.3 Block Components Description

4.3.1 Intelligent Parameter Generator (IPG)

The Intelligent Parameter Generator is the core functional component of the ISG. It is this module that configures the other modules based on the operator input, as well as determines how to change the test parameters to achieve the desired testing result. The operator input is provided in the form of a YAML-like test configuration file, while the other communication paths are transmitted via a middleware system, such as ROS, to the other components. The components of the IPG are detailed in Figure 8: Intelligent Parameter Generator.

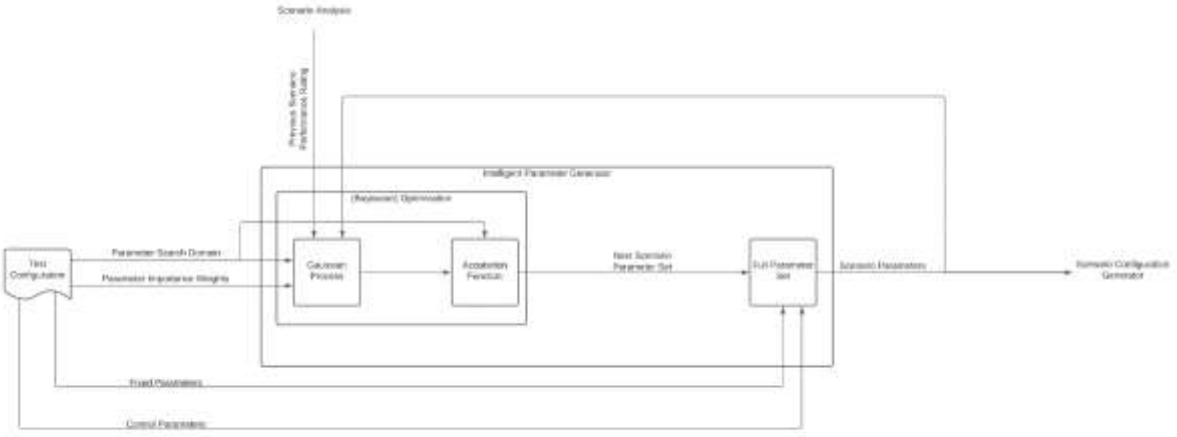

Figure 8: Intelligent Parameter Generator

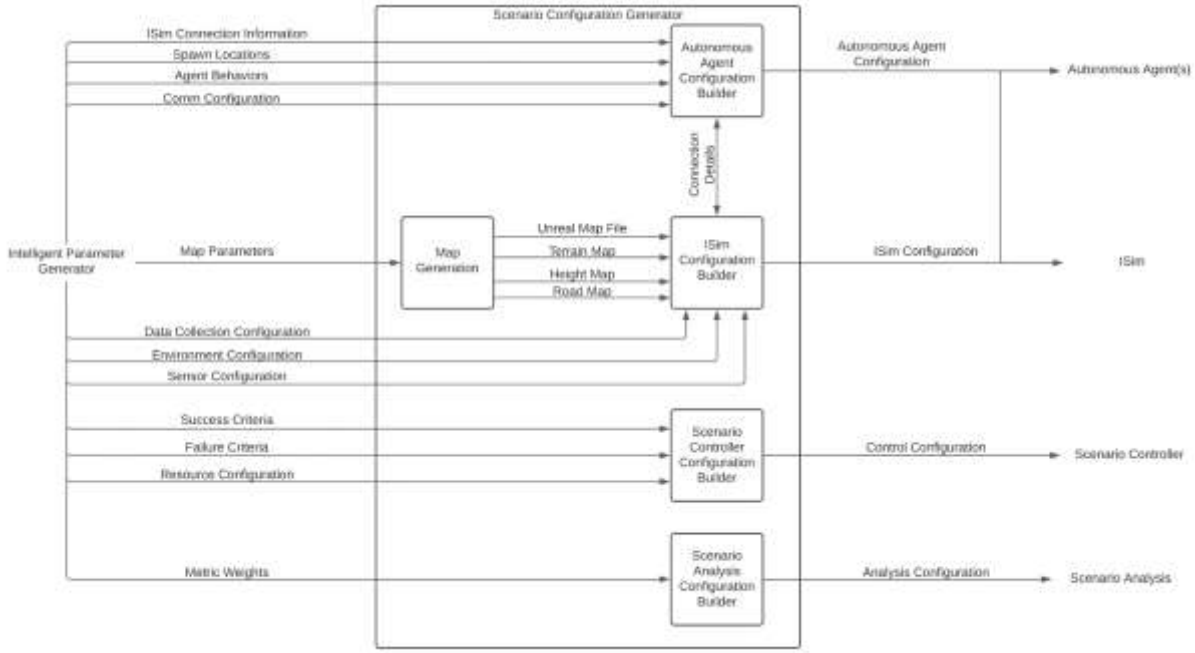

Figure 9: Scenario Configuration Generator

The test configuration file is a human readable text file with fields for configuring the scenario environment, setting the type and number of dynamic vehicles in the scenario, specifying the termination conditions and variable test parameters, as well as other configuration information. Further details are included in Scenario Test Configuration File.

Based on the test configuration file, the IPG will generate the initial configuration for the scenario and transmit this to the Scenario Configuration Generator. Once the scenario is complete and the Scenario Analysis module communicates its final results to the IPG, it uses an update scheme, such as Bayesian Optimization system (comprised of a Gaussian Process along with a user specified Acquisition Function), to determine how to modify the variable test parameters for the next run. These modified variable parameters are combined with the fixed parameters and once again passed to the SCG for the next simulation run. This process repeats until the desired test end-state is reached[2].

4.3.2 Scenario Configuration Generator (SCG)

The Scenario Configuration Generator serves the primary purpose of translating the user- and IPG-specified configuration requirements into valid simulation components for execution. As part of this process, the SCG is responsible for the following items as detailed in Figure 9.

1. Autonomous agent configuration – to include setting up required non-player-controlled (NPC) vehicles
2. Map generation – generates all required maps for the Autonomous Agents and the ISimA system
3. ISimA configuration – start and goal locations for vehicles, environmental parameters, etc.
4. Scenario Controller configuration – all required configuration parameters for the Scenario Controller including completion criterion

[2] The end-state can have total elapsed time, number of iterations, and other test related criteria as part of it.

5. Scenario Analysis configuration – all required configuration parameters for the Scenario Analysis module including metrics to be measured from test runs

4.3.3 Scenario Controller (SC)

The Scenario Controller is the portion of the ISG that directly controls the simulator system. It provides the run/stop/reset commands to the simulator based on the configuration provided by the SCG. It monitors scenario performance by receiving scenario data from the ISimA system to determine when the stopping criteria are met. The SC is detailed in Figure 10.

4.3.4 Scenario Analysis (SA)

The final primary sub-component of the ISG is the Scenario Analysis module. This module, using the configuration provided by the SCG module, evaluates the autonomy stack under test both during run-time as well as post-run. Once the metrics are calculated for a given run, this data is provided back to the IPG to determine how the parameters should be modified for the next run.

As this component is likely to have very specific requirements for different users it will support a plug-in capability in order to allow for user designed metrics and analysis tools to be specified.

4.4 Interface with ISimA

There are three primary input points for the ISG with the baseline ISimA system:

1. Configuration and setup of the autonomous agents (AA)
    a. From the SCG to the AA modules
2. Scenario and simulator configuration

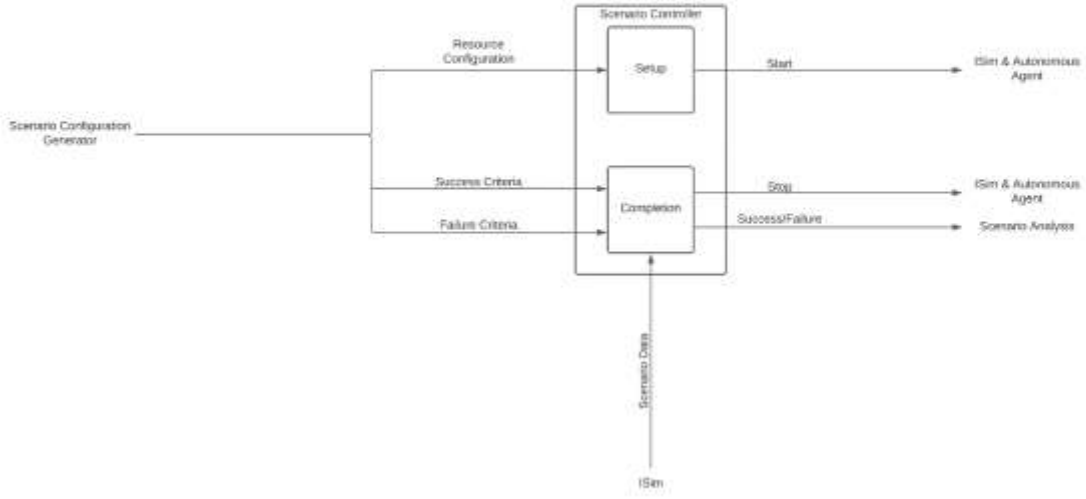

Figure 10: Scenario Controller

    a. From the SCG to ISimA
3. Start and stop commands
    a. From the SC to both the AA modules and the ISimA system

As the simulation is progressing there is a single output interface:

1. Simulation run data
    a. From ISimA to both the SC and the SA

5 VEHICLE TERRAIN MODELING

5.1 OVERVIEW

The native Unreal Engine/CARLA simulator system uses the NVidia PhysX engine for wheeled vehicles by default. This introduces a few known limitations in its accuracy:

1. Each wheel is only collision checked along a single ray
2. Terrain (and thus physics) not modified by weather
3. Terrain is not deformable (no high-centering on snow or mud, for example)

ISimA implements a modified terrain model to improve fidelity of off-road simulation and provide a proof-of-concept for more advanced terrain interaction models. This module plugs into CARLA and is implemented via an interface with the Custom Vehicle Physics module as detailed in Figure 6 and Section 3.2.1.

The prototype implementation provides improved fidelity by determining the terrain type under each point of contact of the vehicle. In this way, torsional effects can be accurately modeled (for example if the left wheels are on mud while the right wheels are on solid ground) as well as effects from having one (or more) wheels lose contact with the ground (for example, while traversing uneven terrain, it may be possible to lift one wheel off the ground and the wheel-terrain model would update accordingly).

6 DEMONSTRATION SCENARIO

6.1 OVERVIEW

To demonstrate the overall feasibility of our approach we have implemented certain key elements of the system using the target Unreal Engine 4 / CARLA simulator platform and custom modules. The proof-of-concept demonstration system generates a programmatically modified terrain and then executes a series of simulation runs while modifying two test parameters. The Scenario Analysis module can make the two test parameters easier or harder independently to test the capabilities of the autonomy stack under test using the simulation results to determine both direction and magnitude of the changes to make.

6.2 Example Scenario Setup

The demonstration test scenario consists of a hilly environment with a narrow traversable path crossing through it. The central section of this path widens considerably allowing for multiple valid trajectories through this portion. In the middle of this wider area, there is a band of icy ground with very low traction for our test autonomous vehicle. This ice patch width can be varied programmatically to allow for testing of the autonomous system capabilities for reasoning about the limited traction as well as testing the controller sub-system on its ability to maintain and/or recover from unpredicted loss of traction.

Additionally, this wider area may have randomly placed trees at a programmatically determined density. These trees impact the autonomous vehicle's ability to take its desired path across the ice patch as well as increasing the difficultly of the base planning problem of navigating through the environment without collision.

6.3 Components Implemented

The baseline ISimA simulator is a modified Unreal Engine 4 and CARLA codebase with custom export functionality implemented to generate the required map components for the Autonomous Agents. Additionally, prototype interfaces have been implemented to allow for data passing between the core ISimA system and the Intelligent Scenario Generator system.

For the Intelligent Scenario Generator system, we have implemented partial versions of all sub-components. For our demonstration system, the ISG is capable of modifying only 2 test parameters of the scenario allowing for changes in ice coverage and obstacle (tree) density. The demonstration system also uses a single map for all scenario tests and does not modify the elevation profiles or terrain types.

The SCG is currently limited to only instantiating a single Autonomous Agent and performs only the minimum configuration of the ISimA system.

The SA sub-component uses a simple linear regression due to the known monotonic relationship between the two available test parameters and overall autonomy system performance. Additionally, the analysis portion of the SA module only performs a rudimentary analysis of system performance.

6.4 Video Description

In the video located at the below link is a demonstration of the ISimA Framework and Intelligent Scenario Generator prototype implementation. In this video, the system initially runs three scenarios:

1. No trees and no ice along route
2. Minimal trees and ice along route
3. Medium trees and ice along route

After these first three runs are complete, the ISG system analyzes the performance to the autonomy stacks operation and generates a new parameter set for run #4. The simulator is run with the new parameters and the results are again analyzed by the ISG system. Since the vehicle was able to successfully navigate through the trees and ice as presented in run #4, the ISG system generates a harder set of parameters (medium high trees with a large ice patch). With this harder set of parameters, the Autonomous Agent attempts to navigate the course and, in this case, collides with a tree due to the

ice. Depending on the test setup, the system would either halt and report its results at this point, or re-run the simulation with a set of parameters in between scenarios #4 and #5 in difficulty.

https://drive.google.com/file/d/15UObrRg1p1Hn_Yrswh_5v9h8KgCbjGkq/view?usp=sharing

7 TECHNICAL FEASIBILITY OF FOLLOW-ON PHASE

The prototype implementation of the ISimA framework demonstrates the technical feasibility of expanding implementation of the full system. All critical elements of the ISimA framework have been tested as part of Phase I. For a Phase II project, the following tasks would be completed:

1. Cloud Capable Infrastructure
2. Multiple Vehicles
    a. Realistic communication simulation
    b. Dynamic intelligence of NPC vehicles
3. Rich real-world scenarios
    a. Deformable terrain modeling
    b. Weather effects on sensing
    c. Validate simulation with limited physical twin
4. Soft real-time capability
5. Integration with ADPT Drive Autonomy stack
6. Intelligent generation of test conditions

Appendix A. COMPARISON OF EXISTING SIMULATION SYSTEMS

Yellow – partial support for requirements; Red – no or limited support for requirements; Blank – unknown level of support

Name **Base Simulation System**	**Physics Engine**	**State of tire / road interaction physics** **Physics for driving simulation** **Any off-road limitations**	**Ease and quality of Custom environment** **Off-road**	**Open source / Free License**	**ROS**	**Easy to scale up to Photo realistic rendering**	**GPU vs CPU for Physics Engine** **Real time factor**	**Sensors** **Existing sensors + ability to add new**	**Pedestrian, weather and vehicles support**	**Plugin Support + Documentation**	**Operating System + Cloud Support**	**Ability to simulate Communication delays**	**How would we be able to modify road-tire interactions:** **1) Change params of an in-built model** **2) Change the model** **3) None**
Unreal Base for a lot of high-fidelity simulators	Yes. Very good physics engine made for Games and Simulators Uses Nvidia PhysX	Supports basic interaction. Some attempts to add a high-fidelity road-tire interaction model in progress. No	Has Unreal editor to edit environments. Tools to add in common elements like foliage, landscape. Extensive tools to create landscape and terrains	Free to use for our use-case	[1] Provides a wrapper for ROS. Not sure about 2.0	State of the art	GPU RTF Y	Camera exists. [1] provides GPS, Lidar, IMU. Can add new sensors	Yes, has a pedestrian modelling system. Lots of 3rd party pedestrian AI plugins exits	Support visual scripting language (Blueprint) for prototyping. Which can be converted to efficient C++. Support python scripting.	Windows recommended for manually modeling environments Supports Linux	have to emulate communication delay b/w agents	Change params of the PhysX model Can also change model
Carla Built on Unreal with driving features	Uses Unreal Engine	Same as Unreal. Except, addition such as [1] might break other features	One can create custom maps, but need to use specific description for road, traffic lights, foot-paths etc.	Yes MIT license But requires UE4 license	ROS bridge exists. No stable support for 2.0 yet.	Uses UE4	GPU RTF Y	Yes, has all autonomous driving related sensors. Can add new ones.	Has support to control weather. Controller pedestrian and other traffic might be somewhat restrictive.	Based on C++ and open source. Large community	Supports Both	-same-	Same as above for vehicle. Can add cuboidal patches in the env with diff friction
AirSim Built on Unreal	Uses Unreal	Ability to write new physics model:	Can load custom UE4 environments easily	But requires UE4 license	Partial support	Uses UE4	GPU RTF Y	Has the common sensors	No pedestrians. Can add other vehicles and weather	Opensource. Not good documentation support	Support both	-same-	Can change both model and params.
Nvidia Drive Sim		Very high-fidelity road-tire interaction.		Not open source									
Bullet	Has better n-body simulation than Nvidia PhysX	Has a very simple vehicle model. But Bullet is highly configurable	Not easy. Not developed to support off-road terrains	zlib license	Good ROS support	Does not support photoreal rendering. In future might get integrated with Unity	GPU RTF Y	Camera, Lidar. Not high-fidelity	None	Large community. Python and C++ plugins	Both	-same-	It is a barebones physics engine. Need to model everything ourselves.
Gazebo Link	ODE,Bullet,Simbody, DART	Vehicle plugin exists. But not does not have realistic models	Difficult	Yes	Yes	Uses Ogre3D. Not photoreal	CPU RTF N	Yes	Weather: not off the shelf.	Very configurable with plugins	Linux preferred	-same-	Custom model

Coppeliasim Link	Bullet Physics, ODE, Newton and Vortex	Vehicle plugin exists. But not does not have realistic models	Difficult	Yes. Source code only for EDU	Yes	No	CPU RTF N	Yes	Weather: not off the shelf.	Plugins can be written in Python, Lua, C, Java, Matlab	Linux preferred	-same-	Custom model
Unity	Uses Nvidia PhysX	High-fidelity pay-ware plug-ins available for vehicle physics	Has user friendly tools to generate off-road terrains	Not free. $1800 / year	None	Has photorealistic rendering	GPU	Camera yes. Rest: Not off the shelf. Need to develop or buy 3rd party assets	Weather yes. Off the shelf pedestrian no.	Mostly close source and 3rd party asset on the store	Windows is preferred.	-same-	Probably only params
rfpro	High quality vehicle dynamics model		Can use PC for terrain	Not free	Not sure	Yes	GPU	Yes	High quality weather	Closed source	-	-same-	Not enough information
SUMO			Not for off-road			Someone used Unity for viz						C2X communication technologies by coupling to a communication network simulator (OMNeT++ or ns-3)	
LGSimulator	Nvidia PhysX		Unity	Has free + pro versions	Yes	Yes	GPU	Yes	Not sure	Possible alternative in the future, but not enough support at current time			
ADAMS				Not free						Possible alternative in the future, but not enough support at current time			
Anvel				Not available									

Appendix B. SCENARIO TEST CONFIGURATION FILE

ISimA will be able to run a complete scenario, including setup, simulation, control of dynamic entities, logging, analysis, and cleanup, from a set of configuration files. This approach allows the user to build a proper level of confidence in the ability of their autonomous vehicle system by running a scenario repeatedly. The user can also modify parameters to gain confidence in the system through a variety of scenarios.

Each run will be generated from configuration files that describe the following scenario components:

Map:

- Manual Generation: select map from a list of pre-generated maps
    - Example parameters: map selection
- Semi-Manual Generation: select map from list of pre-generated maps and apply a transform
    - Example: The pre-generated map may be a path through the woods, and the transform may add an ice patch in different positions or sizes
    - Example parameters: map specific pass-through parameters (ice area, hill grade, tree density, etc.)
- Automatic Map Generation: The map is completely regenerated each run according to some constraints
    - Example parameters: map specific pass-through parameters (tree coverage percent, elevation changes)

NPCs:

- Parameters: Spawn location
- Sub-components: NPC blueprint, NPC Controller

NPC blueprint:

- Unreal blueprint for the NPC vehicle. While this will mainly be selected from a pre populated list, some blueprints may have the ability to take parameters that modify the physical behavior of the vehicle.
- Example parameters: physical attributes - mass, torque, hp, max speed

NPC Controller:

- Each NPC can be controlled in several ways, e.g.: AI blueprints built into Unreal (spline follower, chase player, move towards goal), registering with a group controller like Carla's traffic manager, or a custom script that provides input based on sensor information.
- Example parameters: goal, spline to follow, traffic manager instance

Success and failure criteria:

- ISimA allows for flexible success and failure criteria. A basic example uses reaching a spatial goal as the success criterion and any collision as the failure criterion. A slightly more complicated example would be to maneuver for a set time without collisions or being sighted by an NPC.
- Example parameters: spatial goal, maximum acceptable collision force, maximum time allowed

Additionally, ISimA will have built-in intelligence for the automated generation of test scenarios to dynamically provide test conditions that build up confidence in the ability of autonomous vehicles to accomplish their designated tasks. Each parameter is fixed by default, but can also be specified as a discrete (i.e., a list of options) or continuous range. The span of these discrete and continuous ranges defines the search domain across which the ISimA operates.

Scenario Set Specification:

- **Inputs:** *None (manually generated)*
- Represents requirement set
- YAML file
- Subsection for each component defines a generator for that component
    - Input parameters can be created in several ways:
        - Fixed (e.g., testing on a single map)
        - Select-from-list (e.g., select NPC vehicles from a list of pre-fabricated blueprints)
        - Generate-from-distribution (e.g., NPC start position may be selected from a gaussian to provide variance in the interactions)

Configuration Generator:

- **Inputs:** *scenario set specification, previous scenario results*
- Generates specific instance for each parameter according to distribution configuration
- Optionally, takes metrics from previous scenario runs to inform parameter generation
    - E.g., a single generate-from-distribution variable may remain at the previous value, while other parameters are regenerated after a failed test in order to explore causality.

Scenario Controller:

- **Inputs:** *scenario configuration, scenario measurements*
- From specific parameters, generates instances of each component, launches scenario, and takes in measurements to determine when to stop and clean up scenario