

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Interface OpenSCENARIO for CARLA simulator using ROS control

Miguel Jorge Maia de Magalhães Barros Lançós

Master in Electrical and Computers Engineering

Advisor: Luís Paulo Gonçalves dos Reis

Advisor: João Manuel Leite da Silva

July 28, 2020

Resumo

O potencial que os carros autónomos apresentam para o futuro da nossa sociedade inspirou muitos investigadores a mudar o foco da sua pesquisa para contribuir para o desenvolvimento desta tecnologia. Nos últimos anos, sua popularidade e financiamento aumentaram substancialmente com a participação de startups e algumas das principais empresas automobilísticas. As pesquisas mais promissoras nesta área baseiam-se em algoritmos de aprendizagem computacional que requerem treinar os sistemas na vida real ou com datasets de tamanho considerável. Mas este tipo de treino não é possível em estradas públicas devido aos riscos para a segurança dos utilizadores destas vias e questões éticas, impedindo o seu rápido desenvolvimento.

Deste modo, os simuladores poderão ser a solução ao permitirem o treino dos sistemas de condução autónoma em ambientes virtuais, com cenários realistas e complexos, de uma forma barata e sem pôr em perigo vidas humanas. Existem vários simuladores de condução urbana no mercado mas cada um apresenta uma interface única e uma API personalizada o que resulta numa elevada incompatibilidade dentro da indústria.

Esta tese propõe o desenvolvimento de uma camada de comunicação para o simulador CARLA, fornecendo métodos standard para a interface com os algoritmos de controlo. Esta implementação, ainda que desenhada especificamente para este simulador, permitirá uma interação completa com o mesmo utilizando apenas os métodos de comunicação standard: ROS e OpenSCENARIO.

Abstract

The potential impact self-driving cars could have in our society's future inspired many researchers to switch their focus to further develop this technology. In the recent years its popularity and financing has increased substantially with startups and some major car companies getting involved. Regardless of this trend, the technology still faces many challenges ahead. The leading advances are based on machine learning techniques, which require training on a variety of situations in real-life or with large datasets. However, security liabilities and ethical questions prevent these vehicles from performing freely on the public roads, halting the development progress.

Urban driving simulators present themselves as the solution, allowing to train ADSs in realistic and complex urban scenarios while reducing costs and without endangering human lives. Many urban driving simulators are available nowadays but all of them developed their own interface and APIs resulting in a largely non-standard industry.

This thesis proposes the development of a communication layer for CARLA simulator providing a standard method to interface with the control algorithms. This implementation, although tuned specifically for this simulator, will allow full interaction with the simulator using only ROS and OpenScenario standard.

Acknowledgements

My biggest thanks to my supervisors, Professor Luis Paulo Reis and Dr. João Manuel da Silva, for providing guidance throughout the whole thesis development and for this opportunity.

To Pedro Santos, from CISTER. His workshops were essential to the success of this work.

To all my colleagues in FEUP and Altran for the help and support, especially Jorge Godinho that contributed a lot with his CARLA experiences.

Finally, I want to thank my family that always supported me and help me navigating the world to reach where I am and where I will end up.

And thanks to the SARS-CoV-2 virus that surely provided the development of this project with endless challenges that I had to overcome. I will certainly come out much more wise after this experience.

Miguel Lançós

*“I think it’s very important to have a feedback loop,
where you’re constantly thinking about what you’ve done
and how you could be doing it better.”*

Elon Reeve Musk

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation	2
1.3	Goals	2
1.4	Document Structure	3
2	Literature Review	5
2.1	Autonomous Driving	5
2.1.1	System Architectures	5
2.1.2	Sensors	5
2.1.3	Localization & Mapping	6
2.1.4	Perception	6
2.1.5	Assessment	7
2.2	Robot Operating System (ROS)	8
2.2.1	Design Goals	8
2.2.2	Nomenclature	9
2.3	OpenSCENARIO	10
2.3.1	Entities	10
2.3.2	Storyboard	11
2.3.3	Events & Triggers & Actions	12
2.3.4	Re-Use Mechanisms	15
2.4	Urban Driving Simulators	16
2.4.1	Comparison	19
2.4.2	Overview	24
2.5	Car Learning to Act (CARLA) Simulator	26
2.5.1	Unreal Engine 4	27
2.5.2	ROS Bridge	29
2.6	Conclusion	31
3	Approach and Work Plan	33
3.1	Research Approach	33
3.2	Work Plan	33
4	WP1: Scenario Configuration	37
4.1	Scenario Runner	38
4.2	Parameters	38
4.2.1	Implementation	39
4.3	Catalogs	40

4.3.1	Implementation	40
4.4	Entities	41
4.4.1	Entities in CARLA	44
4.4.2	Implementation	44
4.4.3	Vehicle	46
4.5	Storyboard	49
4.5.1	Implementation	50
4.5.2	Init	51
4.5.3	Story	53
4.5.4	Act	54
4.5.5	ManeuverGroup	56
4.5.6	Maneuver	58
4.5.7	Event	59
4.5.8	Daytime Animation	61
4.5.9	Naming	64
4.6	Actions	65
4.6.1	Global Actions	66
4.6.2	Private Actions	68
4.6.3	Environment Action	70
4.6.4	Entity Action	75
4.6.5	Parameter Action	77
4.6.6	Longitudinal Action	81
4.7	Conclusion	88
5	WP2: Scenario Constructor	89
5.1	Objectives	90
5.1.1	Main Canvas	90
5.1.2	Side Bar	91
5.1.3	Attribute Window	91
5.2	Implementation	92
5.2.1	Visual Architecture	93
5.2.2	Functionality	95
5.3	Content	101
5.3.1	OpenSCENARIO Elements	101
5.4	Conclusion	105
6	Conclusions & Future Work	107
6.1	Summary	107
6.2	Future Research Directions	108
	References	109
A	Absolute Target Speed Action	113
B	Relative Target Speed Action	123

List of Figures

1.1	Proposed stack layers structure	2
2.1	OpenSCENARIO structure & Storyboard content (taken from "XSD Diagram") .	10
2.2	OpenSCENARIO <i>Storyboard's</i> structure.[1]	13
2.3	Trigger conditions (AND, OR) relationship	14
2.4	Example of a velocity condition and different edge condition responses.	14
2.5	Current vs Sulkowski et al. 2018 simulator review model	16
2.6	Carla client-server architecture [2]	26
2.7	Carla epic vs low quality level [2].	27
2.8	Waypoints and traffic trigger boxes in a intersection [2]	29
3.1	Proposed system structure	34
4.1	OpenSCENARIO's add-on content directory.	37
4.2	View of a CARLA's map with the platform spawned.	44
4.3	Entities Base Class - UML class diagram.	45
4.4	OpenSCENARIO vehicle coordinates reference[1].	48
4.5	VehicleConfiguration UML class diagram.	49
4.6	Storyboard UML class diagram.	51
4.7	Storyboard behavior tree.	52
4.8	Init UML class diagram.	53
4.9	Init behavior tree.	53
4.10	Story UML class diagram.	54
4.11	Story behavior tree.	54
4.12	Act UML class diagram.	55
4.13	Act behavior tree.	56
4.14	ManeuverGroup UML class diagram.	58
4.15	ManeuverGroup behavior tree.	58
4.16	Maneuver UML class diagram.	59
4.17	Maneuver behavior tree.	59
4.18	Event UML class diagram.	61
4.19	Event behavior tree.	61
4.20	Real vs intended CARLA's sun movement.	62
4.21	Behavior tree with the TimeOfDay action.	63
4.22	TimeOfDay Action - UML class diagram.	63
4.23	Parallel and Sequence UML class diagrams.	65
4.24	Actions Base classes - UML class diagram.	66
4.25	Example of a state machine implementation.	66
4.26	Global actions UML class diagram.	67

4.27	PrivateAction UML class diagram.	69
4.28	Private UML class diagram.	69
4.29	Private behavior tree.	70
4.30	Environment action - UML class diagram.	73
4.31	SetWeather action behavior - UML class diagram.	76
4.32	Entity Action - UML class diagram.	77
4.33	AddEntityAction behavior - UML class diagram.	77
4.34	DeleteEntityAction behavior - UML class diagram.	78
4.35	ParameterAction UML class diagram.	79
4.36	SetParameter behavior - UML class diagram.	80
4.37	AddParameter behavior - UML class diagram.	80
4.38	MultParameter behavior - UML class diagram.	80
4.39	LongitudinalAction UML class diagram.	81
4.40	DynamicBehavior UML class diagram.	85
4.41	AbsoluteTargetSpeed UML class diagram.	85
4.42	Speed evolution of a vehicle. Parameters: kp=1.0, Audi A2. Test script: listing A.1 from Appendix A	87
4.43	RelativeTargetSpeed UML class diagram.	88
4.44	Speed evolution of a vehicle following a leading vehicle. Parameters: kp=1.0, Audi A2. Test script: listing B.1 from Appendix B	88
5.1	Scenario Constructor libraries diagram.	89
5.2	Main window - Interface mock-up.	90
5.3	Main window showing the attribute window - Interface mock-up.	92
5.4	Interface UML class diagram.	93
5.5	Interface structure - static elements.	94
5.6	Example of the creation of a script.	96
5.7	OSCElement UML class diagram.	97
5.8	OSCLabel, ListLabel and CanvasLabel UML class diagram.	98
5.9	Example modal window with the <i>FileHeader</i> element attributes.	98
5.10	Attribute and AttrType UML class diagram.	99
5.11	AttrWindow UML class diagram.	100
5.12	OSCChildren UML class diagram.	100
5.13	Example of a data type class.	103
5.14	OSCEnumeration data type class.	103
5.15	OSCBoolean data type class.	104
5.16	OSCUnsignedShort data type class.	104
5.17	OSCDatetime data type class.	104
5.18	OSCString data type class.	105
5.19	OSCDouble data type class.	105

List of Tables

2.1	Urban driving simulators review	25
2.2	Currently available CARLA sensors	28
2.3	Vehicle configurable physics parameters	30
4.1	Vehicle custom properties	50
4.2	CARLA's weather parameters.[2]	73
4.3	Conversion weather parameters between CARLA and OpenSCENARIO.	74
4.4	Speed dynamics evolution: Shape vs Dimension	86

Acronyms

AD Autonomous Driving.

ADAS Advanced Driver-Assistance System.

ADS Automated Driving System.

AI Artificial Intelligence.

API Application Programming Interface.

CARLA Car Learning to Act.

DATMO Detection and Tracking of Multiple Objects.

DCNN Deep Convolutional Neural Network.

EU European Union.

FEUP Faculdade de Engenharia da Universidade do Porto.

FOV Field of View.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

GUI Graphical User Interface.

IDL Interface Definition Language.

IMU Inertial Measurement Unit.

LIDAR Light Detection And Ranging.

MoI Moment of Inertia.

MOT Multiple Object Tracking.

NHTSA National Highway Traffic Safety Administration.

NPC Non-Player Character.

RADAR Radio Detection And Ranging.

ROS Robot Operating System.

RPC Remote Procedure Call.

RPM Revolutions per Minute.

SLAM Simultaneous Localization and Mapping.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

UE4 Unreal Engine 4.

XML eXtensible Markup Language.

Chapter 1

Introduction

1.1 Context

In the last century, cars have revolutionized the mobility of individuals and groups of people. Since their invention, the number of vehicles on public roads has increased considerably and with it the number of road accidents. For example, in 2015 there were more than one million road accidents in the European Union (EU) resulting in close to 1.5 million injured and 35 thousand dead people [3]. And, according to the National Highway Traffic Safety Administration (NHTSA), about 94% of worldwide's road accidents are due to human error [4]. Autonomous driving is hoping to reduce these numbers largely.

Recent advances in Artificial Intelligence (AI) algorithms and computer processing power paved the way for Automated Driving System (ADS) of level 4 and above [5]. However, research shows that an autonomous vehicle would need millions or even billions of driven miles to prove its safety and readiness to be implemented in everyday's use [6]. This extensive testing cannot be accomplished in real-world situations at the risk of injuring or killing human beings. As such, simulators present themselves as the answer by allowing to perform the necessary driven miles needed to train and test the driving algorithms, in a variety of normal and/or challenging scenarios.

CARLA is an open-source free software, developed on Unreal Engine 4 (UE4), that allows simulating quite realistically urban driving environments. It has an integrated set of the most common sensors, allows creating different demanding environments and has ROS Bridge communication capabilities. This set of characteristics and many others make it one of the most suitable options to develop city ADS control algorithms [7].

In order to strengthen its position as a strategic Engineering provider, *Altran Portugal* created a joint effort with some major Portuguese scientific organizations, *CISTER*, *NOVALincs* and *HasLab*, to give origin to *Vortex-CoLab*, a collaborative laboratory whose goal is to accelerate the technology transfer of cyber-physical systems and cybersecurity [8]. The development of autonomous driving solutions is the first target research area of *Vortex-CoLab*.

This thesis was proposed by *Altran Portugal* in the scope of assisted and Autonomous Driving (AD) research projects, developed within the company in collaboration with the *Vortex-CoLab*

association. This simulation extension work is one of the pieces composing the company’s initiatives for creating a software infrastructure to drive forward the development of AD platforms. Modules already tackled include Perception, V2X communication, World Modeling, data labeling, decision & control, infotainment, data visualization. The work developed during this thesis enables the testing, development and validation of most of the previously mentioned modules, further extending the projects ecosystem towards an integrated and harmonized set of solutions.

1.2 Motivation

Although simulators present themselves as a solution to effectively and quickly develop, train and validate ADSs in a safe environment, it’s not uncommon for each simulator to implement its own personalized interface thus creating a challenge to design versatile ADS architectures.

Some standards were developed with the objective of tackling this emerging software incompatibility in mind. The most popular examples are ROS, which provides standard communication for robotic systems, and OpenSCENARIO, with a standard very useful to define testing scenarios.

As proposed in figure 1.1, developing an ADS with ROS to interface with the hardware and using OpenSCENARIO to construct and define training and testing scenarios in simulators ensures that no aspect of the bottom layer is taken in consideration resulting in extraordinarily versatile architectures. In theory the simulation layer replacement, with another simulator or real vehicle, would only affect the communication layer while leaving the software layer intact.

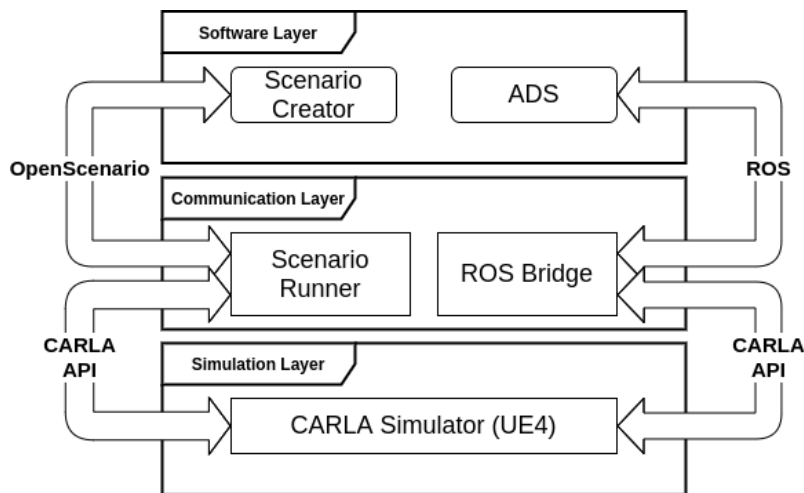


Figure 1.1: Proposed stack layers structure

1.3 Goals

The main purpose of this thesis is to extend CARLA’s functionalities while also developing a communication layer on top of the simulator. The final result should be a system that allows:

- complete interaction between ADSs and CARLA simulator using only the ROS framework;

- full configuration of simulator's settings and scenario definition by using Openscenario standard;
- exportation of labelled datasets;
- configuration of the simulator using a Graphical User Interface (GUI);

Ideally one should be able to completely interact with CARLA using only the ROS framework.

1.4 Document Structure

In this chapter (1) it is presented an overview of the current position of autonomous driving development and how this thesis could be useful to further advance this promising technology.

The chapter 2 performs an exposition of the state-of-the-art on ADS design with the objective to better understand what should consist an urban driving simulation, a description of the necessary communication method for this project, a short review of the market's availability on urban driving simulators and, to conclude, a more in-depth review of the chosen simulator, CARLA.

The chapter 3 provides a brief description of this thesis research method, objectives and milestones.

The first phase of the development, the interpretation and execution of the OpenSCENARIO scripts, is presented in the chapter 4, alongside a detailed description of the OpenSCENARIO's features and its implementation strategy.

In the chapter 5 it is described the implementation for a GUI, based on Qt framework, intended to facilitate the creation of OpenSCENARIO scripts without requiring a deep knowledge of the standard.

Finally, in the chapter 6 a final summary of the work performed in this thesis is presented followed by a description of the planned future work in further development of this project.

Chapter 2

Literature Review

2.1 Autonomous Driving

The following content presented in this section was obtained from a literature review on autonomous driving [9].

2.1.1 System Architectures

Common ADSs architectures can be categorized according to their connectivity and algorithm design.

As far as connectivity goes, an ADS can be defined as an ego-only or a connected system. An ego-only system consists in self-sufficient vehicles with all the necessary requirements to carry out all the intended automated driving operations. On the other hand, a connected system is able to use its communication capabilities to improve the automated driving operations. Probably due to the challenges presented by communications in driving scenarios, there isn't yet a connected system implementation.

In the algorithmic design category, there can be modular and end-to-end systems. A modular ADS is structured as a pipeline of separate software components, for individual tasks, linking sensor inputs to actuator outputs, which usually are motor commands. End-to-end systems are the opposite, instead of being composed by several modules, a single software unit is developed which is fed sensor data, directly, and outputs discrete or continuous actuator outputs.

2.1.2 Sensors

It is essential for an autonomous driving vehicle to be robust and reliable and so it is important to have high sensor redundancy. In an ADS there are two types of sensors, exteroceptive and proprioceptive.

Exteroceptive sensors are used for perceiving the environment around the vehicle, to identify and detect both dynamic and static objects. In the current state of art, the most common exteroceptive sensors are cameras, Light Detection And Rangings (LIDARs), Radio Detection And Rangings (RADARs) and ultrasonic sensors.

Since the knowledge of the surrounding environment is not enough for a vehicle to safely perform all of its automated driving tasks, proprioceptive sensors are used. This type of sensors are used to determine states of the vehicle, such as position, speed, acceleration, orientation... Currently, there is a wide variety of sensors used for different goals, wheel encoders for vehicle odometry, Inertial Measurement Units (IMUs) for velocity and position changes, tachometers for velocity and altimeters to measure altitude. To increase robustness a combination of several proprioceptive sensors is used.

2.1.3 Localization & Mapping

Another important component of an ADS is its localization. Localization, in this situation, is the task of finding ego-position relative to a reference frame in an environment. As such, it is essential for a vehicle to use the correct road lane and position itself accurately in it or to use local and global navigation. The most common solutions for this challenge are an GNSS-IMU fusion, SLAM and a priori map-based localization.

A GNSS-IMU fusion consists in using the IMU data to calculate the position changes from a dead-reckoning, but to prevent error propagation, regularly updated by GPS information. Although its performance is far superior to only IMU localization and more robust than only GPS, it does not meet the performance criteria for vehicle localization and, as such, can be only used for high-level route planning, or initial pose estimation for other localization technologies.

Simultaneous Localization and Mapping (SLAM), as described in the name, is the act to ego-localization while simultaneously generating an online map without any a priori environment information. Despite being a very versatile solution that works anywhere, using SLAM outdoors involves inaccurate and inefficient algorithms due to its high computational requirements and environmental challenges.

Lastly, a priori map based localization uses detailed a priori generated maps to match sensor readings in order to find the vehicle position. To improve performance it can be used with other solutions in order to provide an estimate of initial position, such as GNSS. It can be implemented either by using landmark search, which requires a sufficient number of landmarks, or point cloud matching, at the cost of higher computational requirement. Of course, this approach requires an extra step of generating the map and it faces challenges in fast changing environments, such as high vegetation or constructions.

2.1.4 Perception

As previously mentioned in the chapter 2.1.2, perception is a major task for ADSs and consists in extracting information from the surrounding environment, which can be critical for safe navi-

gation. The most common approach is by using 2D cameras but 3D vision solutions is becoming increasingly popular. This critical task can be categorized into five topics: object detection, semantic segmentation, 3D object detection, road and lane detection and object tracking.

Object detection is the task of identifying the localization and size of static and dynamic objects of interest. It is a basic task which is the support for other ADS tasks and essential for scene understanding. Frequent approaches are based in Deep Convolutional Neural Networks (DCNNs) either single stage or region proposal detection frameworks.

Sometimes an object is poorly defined by a bounding box and, as an alternative, semantic segmentation allows to classify each pixel of an image with a class label.

3D object detection can be extremely useful for ADSs. To bridge the gap between 2D and 3D images, depth estimation is introduced. There are some implementations of 3D object detection using algorithms on top of a single 2D camera data but stereo or multi-view systems are more robust although very hardware expensive. Recently a solution with 3D LIDAR was introduced, providing depth naturally without any algorithm processing.

Object tracking, also known as Multiple Object Tracking (MOT) and Detection and Tracking of Multiple Objects (DATMO), consists in applying motion models by estimating speed and heading of objects to allow their tracking. Sensor fusion for this task is the most used solution in order to correctly and accurately estimate all the necessary complex data from the frame of the object.

Road and lane detection is necessary for vehicle understanding of road semantics, which is clearly important to properly navigate the road. This task can be divided in several levels: understanding of the current driving lane, determination of neighboring lanes and calculation of direction and merging lanes.

2.1.5 Assessment

To assure maximum safety an ADS has to be able to evaluate the overall risk level of specific situations and predict the intentions of human drivers and pedestrians. This risk assessment has been studied in three different subjects: overall risk assessment, human driving behavior assessment and driving style recognition.

2.2 ROS

Robot programming is difficult nowadays with so many hardware implementations and increasingly more complex tasks. The result is a long and over-complicated code which is not easily reusable. And, since the development of a full stack is beyond the expertise of a single researcher, a system that allows large-scale software integrations was necessary.

ROS is not a traditional operating system. It functions on top of a host operating system and it provides a structured communications layers for heterogeneous computer clusters. Initially developed as a response to the challenges presented in the STAIR project, from Stanford University, and Personal Robot Program, from Willow Garage, it puts its emphasis on large-scale integrative robotics research and versatility to be applicable to much more than only service-robot and mobile-manipulations domains.

2.2.1 Design Goals

To solve all the previously mentioned problems and challenges, ROS was designed with the principles of peer-to-peer, tools-based, multi-lingual, thin and free and open-source.

Being peer-to-peer it allows multi-process and multi-host communication on a heterogeneous network, that is a cluster of devices without a central data server. However, it still needs a main device, called master, to enable processes to find each other at runtime.

ROS is multi-lingual to provide more comfort and agility to the programmer. Every language has trade-offs between programming time, ease of debugging, syntax and runtime efficiency and the appropriate use of a language depends on the project requirements and the programmer preferences. As such, ROS is currently implemented in C++, Python, Octave and LISP and to facilitate portability all peer-to-peer connection negotiation and configuration is in XML-RPC format which is supported by most major languages. Available messages are described, by their fields, in short text files using language-neutral Interface Definition Language (IDL) which is implementation independent and enables easy creation of new messages.

It is tools-based because, to manage its complexity, it is constituted by a microkernel full of small tools that build and run various components. Although this reduces the framework efficiency, it gains in stability and complexity management.

It is thin because it encourages the development of standalone libraries with no dependencies on ROS by performing builds inside the source code tree and making use of CMake. The main idea is to develop processes that only expose configuration options and route data in and out through ROS. Also, this independent structure allows for ROS to be able to update source code from external repositories or even apply patches.

Finally, it is free and open-source since it is distributed under the BSD license, allowing the development of both commercial and non-commercial projects. Its independency of processes supports the use of individual licenses for each process, from GPL to BSD.

2.2.2 Nomenclature

Nodes: also could be interpreted as a software module, are essentially processes.

Messages: are a predefined data structure which contains information to be transported between nodes.

Topics: are the subject of the message, usually a string, it identifies the content of the message. In ROS nodes can subscribe several topics they might be interested and can publish several topics.

Services: allowing synchronous communication, similar to web services, a service responds with the required information when requested by a node. A service name is unique and no two nodes can run the same service.

2.3 OpenSCENARIO

Scenarios have been used in various disciplines, from military to economic and other technological fields, as a means to stimulate thinking about possible occurrences and analyse relevant courses of action.[10] The term *scenario* has been defined numerous times throughout the years and Ulbrich et al. [11], based on the work of Geyer et al. [12], produced the definition that follows this paragraph, mainly focusing on simulation scenarios.

"A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions&events as well as goals&values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time."

- Ulbrich et al. [11]

According to this definition a scenario is constructed by a sequence of scenes which contain actions, events, goals and values in order to characterize a single temporal sequence of actions and events depending on the relevant goals and values of the scene.

OpenSCENARIO is a standard providing support for scenario description containing all the components present in the previous definition. All dynamic actors are named *entities* and can be organized into catalogs for vehicles, pedestrians and miscellaneous objects, and ego vehicles are defined by the keyword *ego*. The scenery is defined by road network characteristics, including OpenDRIVE files, and environment settings, specifically weather and time information. Actions&events and goals&values are represented in a series of maneuvers, which can also be organized into a maneuver catalog, and all these maneuvers are sequenced chronologically into a storyboard.[13]

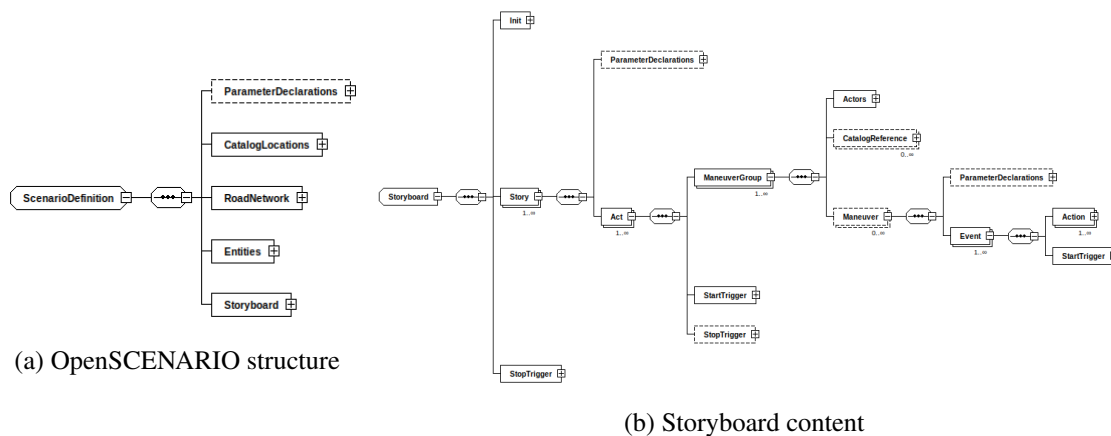


Figure 2.1: OpenSCENARIO structure & Storyboard content (taken from "XSD Diagram")

2.3.1 Entities

According to the OpenSCENARIO user guide, entities, often referred to as Actors, are objects that may change their position and orientation dynamically over time. In this case, entities can be

classified as *Vehicles*, *Pedestrians* or *MiscObjects*. The latter group, identical to the OpenDRIVE format, comprises the following object classes: obstacle, pole, tree, vegetation, barrier, building, parking space, patch, railing, traffic island, crosswalk, street lamp, gantry, sound barrier, wind and road mark. In a scenario these actors can be used either in triggers, to enable actions, or as the target to perform actions, resulting in a change of the entity's state.[1]

Besides entity type specific properties, e.g. vehicle's maximum acceleration, meant to allow for definition of test-instance or use-case specific properties, all entity types contain user-customized set of properties, presenting a powerful instrument for providing features related to the specific simulator, hardware or software setup responsible for executing the scenario. These properties can be introduced in the form of a list of name-value pairs, instances of properties, or as a reference to external files dependent on the software implementation. However, no scenario should be dependent on these properties, i.e. custom properties can influence the scenario but its execution should always be possible without the knowledge of their meaning.[1]

In order to keep behaviour consistency, and other software-hardware specific behaviours, entities can be assigned a default, or a user-assigned, controller responsible for managing longitudinal or lateral movements when these are not under the effect of other user-assigned actions. Longitudinal and lateral control are considered independent, and a controller can be assigned for one or both movement directions.[1]

Entities can also be grouped into *EntitySelections*, allowing for actions and triggers to treat multiple entities as one. These groups can be used as target for actions, where the action is only considered finished when all entities involved completed it, or their aggregated information can be used in triggers.[1]

2.3.2 Storyboard

The *Storyboard* element is OpenSCENARIO's main component, it provides a concept similar to that of classical storytelling, such as theater plays, in which answers for the question "who?", "what?" and "when?" are given. The *Storyboard* should contain at least one *Story* element which in turn should contain at least one *Act* element and so on, according to the following structure:

$$\textit{Storyboard} \supset \textit{Story} \supset \textit{Act} \supset \textit{ManeuverGroup} \supset \textit{Maneuver} \supset \textit{Event} \supset \textit{Action}$$

Init is the first element of the *Storyboard*. A *Storyboard* needs to contain exactly one *Init* element to define the initial state of the scenario. This element is intended to correctly position the entities for the following sequence of events. Although any available action (see chapter 2.3.3) can be executed in this environment, the execution of all actions in parallel due to the nonexistence of triggers, only makes it suitable for simple actions, such as positioning or setting initial speed.[1]

Following an *Init* element, the *Story* elements come through. At least one *Story* element is necessary but many more can exist in a *Storyboard*. This element permits authors to group different acts in order to provide better organizational structure in large scenarios.[1]

Contained in the *Story* element, the *Act* element can be found. With the function to answer the "when?" question, this element provides temporal localization in its *Story's* timeline. To fulfill this objective, it possesses both start, that initiate the execution of *ManeuverGroups*, and stop triggers, to "close" the act.[1]

The *ManeuverGroup* elements, contained in the *Act* elements, also play a role in the classical storytelling concept by providing an answer to the question "who?". These elements allow to assign entities, flexibly, i.e. during runtime, to sequences of actions contained in the *Maneuver* element.[1]

Finally, the *Maneuver* element defines "what?" happens in the scenario. It contains a series of *Event* elements that describe the actions to perform.[1]

The figure 2.2 contributes to a better understanding of a *Storyboard* hierarchical structure.

2.3.3 Events & Triggers & Actions

Event elements serve as containers to combine actions and produce meaningful behavior for the scenario. These elements contain start triggers for better control over the beginning of the its actions. Although an *Event* can be run multiple times, by taking advantage of the "maximumExecutionCount" parameter, an *Event* can only have one instance running at the same time, which means an *Event* can only be triggered when it is not running.

To create a broader combination of behaviours, these elements are characterized with a priority level that defines rules for their execution in relation to other *Event* elements inside the same *Maneuver*:

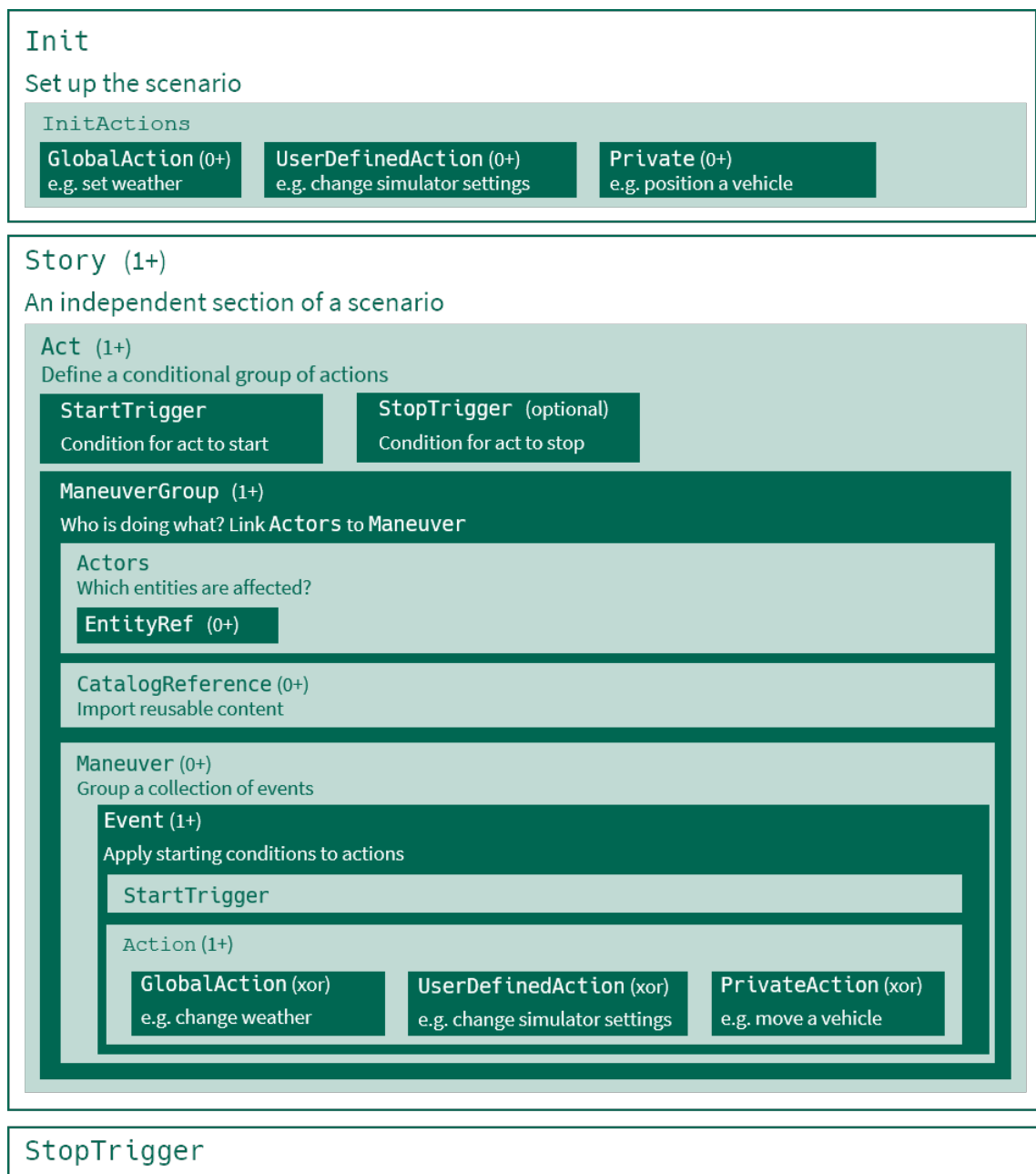
overwrite When this *Event* is triggered, all the other running *Events* are terminated.

skip The triggered *Event* will hold its execution until all the other running *Events* terminate.

parallel The *Event* is always executed independently of other running *Events*.[1]

Action elements are singular behaviours that allow to create or modify dynamic elements of a scenario. *Actions* can be present in the *Init* element, where they are immediately executed in parallel, due to the lack of triggers, with the sole purpose of setting up the initial state of the scenario or inside *Event* containers subjected to their own triggers and their parent element's triggers, creating a timeline of actions. OpenSCENARIO defines three types of actions, *PrivateAction*, *GlobalAction* and *UserDefinedAction*.[1]

PrivateActions are related to an entity, or group of entities, and describe their motion, position and visibility in the scenario. These are organized into several categories, *LongitudinalAction*, *LateralAction*, *VisibilityAction*, *SynchronizeAction*, *ActivateControllerAction*, *ControllerAction*, *TeleportAction*, *RoutingAction*. On the other hand, *GlobalActions* do not require an entity and serve to set or modify other scenario related quantities. These are categorized into the following: *EnvironmentAction*, *EntityAction*, *ParameterAction*, *InfrastructureAction* and *TrafficAction*. Finally, users are able to create their own customized actions by providing a script file or commands

Figure 2.2: OpenSCENARIO *Storyboard*'s structure.[1]

with *UserDefinedActions*. However, these are dependent on the implementation and its contents are not only dependent on the simulator, hardware but also the software responsible for running the scenario.[1]

Triggers play an important role in creating the temporal dimension of a scenario as they are used to start and stop ongoing scenario elements, referred as *startTriggers* and *stopTriggers*, respectively. A *startTrigger* allows the execution of its parent element, *Act* or *Event*, and their respective children. On the other hand, a *stopTrigger* halt the execution of its parent elements, *Story* or *Act*, and all the involved children.[1]

A *Trigger* is a container for *ConditionGroup* elements which, in turn, are containers for *Condition* elements. A *Condition* is represented by a logical expression assessed during runtime producing a boolean output. The value of each *Condition* is backpropagated to its parent *ConditionGroup* whose value is then backpropagated to its parent *Trigger* where a final value is reached and influence the scenario. A *Trigger* provides a flexible combination of conditions through a (AND, OR) relationship, as demonstrated by the figure 2.3.[1]

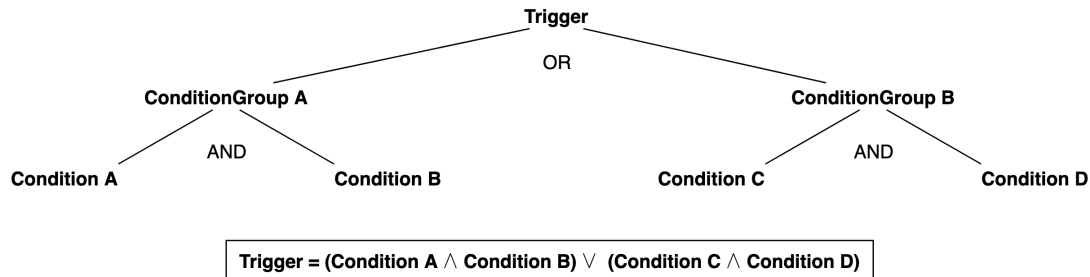


Figure 2.3: Trigger conditions (AND, OR) relationship

ConditionGroups only present a "true" value once all its child *Conditions* also present a "true" value, effectively representing an AND relationship. *Triggers*, however, are activated as soon as one of its child *ConditionGroups* present a "true" value, an OR relationship.[1]

A *Condition* is characterized by a name, delay and a edge condition. The delay allows to postpone the activation of the condition, starting once its parameters are met, for the required duration. The edge condition, as exemplified in figure 2.4, allows to define a rule for the condition output in relation to the evaluation of its parameters.[1]

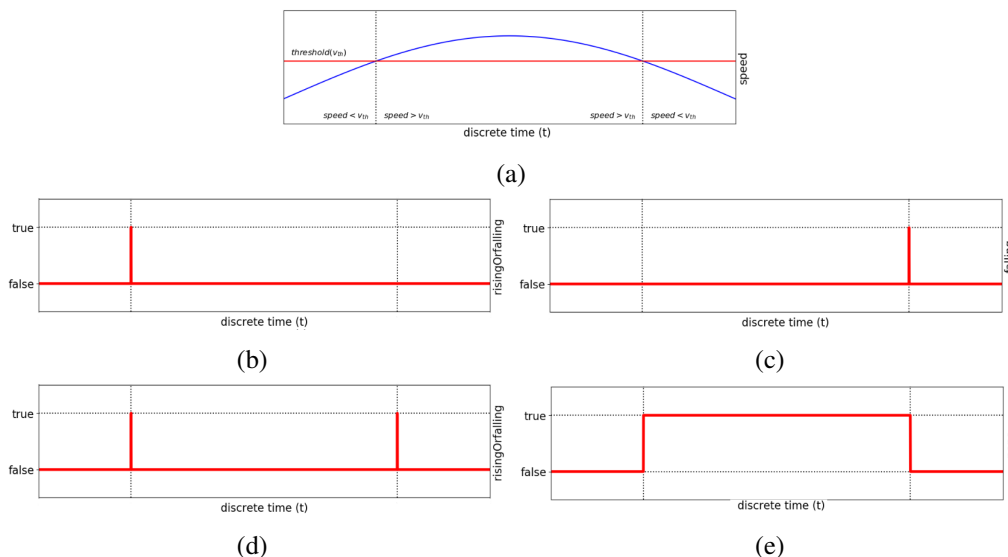


Figure 2.4: Example of a velocity condition and different edge condition responses. (a) Speed evolution of an actor compared to a simple condition with a specified threshold. (b) Output of a rising edge condition. (c) Output of a falling edge condition. (d) Output of a rising or falling edge condition. (e) Output of a none edge condition.

A rising edge only triggers its condition in the moment its parameters are met. Opposite to the rising edge, a falling edge only triggers the condition in the moment its parameters stop being met. A rising or falling edge, is a mix of both previous methods, and triggers the condition either the moment its parameters are met or the moment its parameters stop being met. Finally, the last method, none, allows to mirror the evaluation of a conditions parameters, i.e. a condition is triggered if its parameters are currently being met.[1]

In OpenSCENARIO two types of conditions are defined, *ByEntityConditions* and *ByValueConditions*. The first group contains conditions that are directly related to the state of entities present in the scenario, while the second contains conditions related to other non-entity parameters, such as traffic signals or simulation time. User defined conditions are also provided in the *ByValueCondition* element.[1]

2.3.4 Re-Use Mechanisms

This file format was designed with two methods that allow the users to use repeatedly the same value or even complex elements throughout the scenario without rewriting everything. These are parameters and catalogs.[1]

Parameters share the same concept as the "variables" in most common programming languages. They are values that are declared inside *ParameterDeclaration* elements, in the beginning of the script, catalogs, or throughout the scenario. These values can then be manipulated, by *ParameterAssignment* elements or by actions during the execution of the scenario, and can be used to replace any value from the script or to trigger conditions. Parameters allow for an extension of the scenarios providing easy manipulation of multiple values and integration with external tools to permit re-simulations of the same scenario with different settings.[1]

Catalogs offer the possibility to outsource the description of certain elements from the scenario to a separate file, which can then be referenced from a scenario. This feature allows reusability of complex, long or specific elements while also increasing the readability of the scenario file. There are eight catalog types, and each catalog contains a list of elements corresponding to its type. All vehicles, pedestrians, "MiscObjects", controllers, maneuvers, trajectories and routes can be stored in their own catalogs.[1]

2.4 Urban Driving Simulators

One of the most difficult problems ADS development faces is navigation in densely populated urban environments due to complex multi-agent dynamics at traffic intersections, the need to track every actor present in the Field of View (FOV), recognition of traffic rule semantics and occurrences of rare events such as road construction, unpredictable pedestrians crossing the road, rogue drivers, etc. Thus, to correctly develop a safe ADS it is essential to closely simulate the complexity of urban driving. To complete this task urban driving simulators should be able to emulate pedestrians, intersections, crosswalks, traffic and other common city objects. Ideally they should also have control and customization over the environment, scenario specification and scripting, a wide range of common autonomous driving sensors and feedback upon collision or violation of traffic rules.

To better analyze the previous mentioned criteria and relevant others, the comparison presented next was based on Sulkowski et al. 2018 [5] work but adapted to better correspond to the dissertation goals. This model is exposed and compared to the original in figure 2.5 and further explained below.

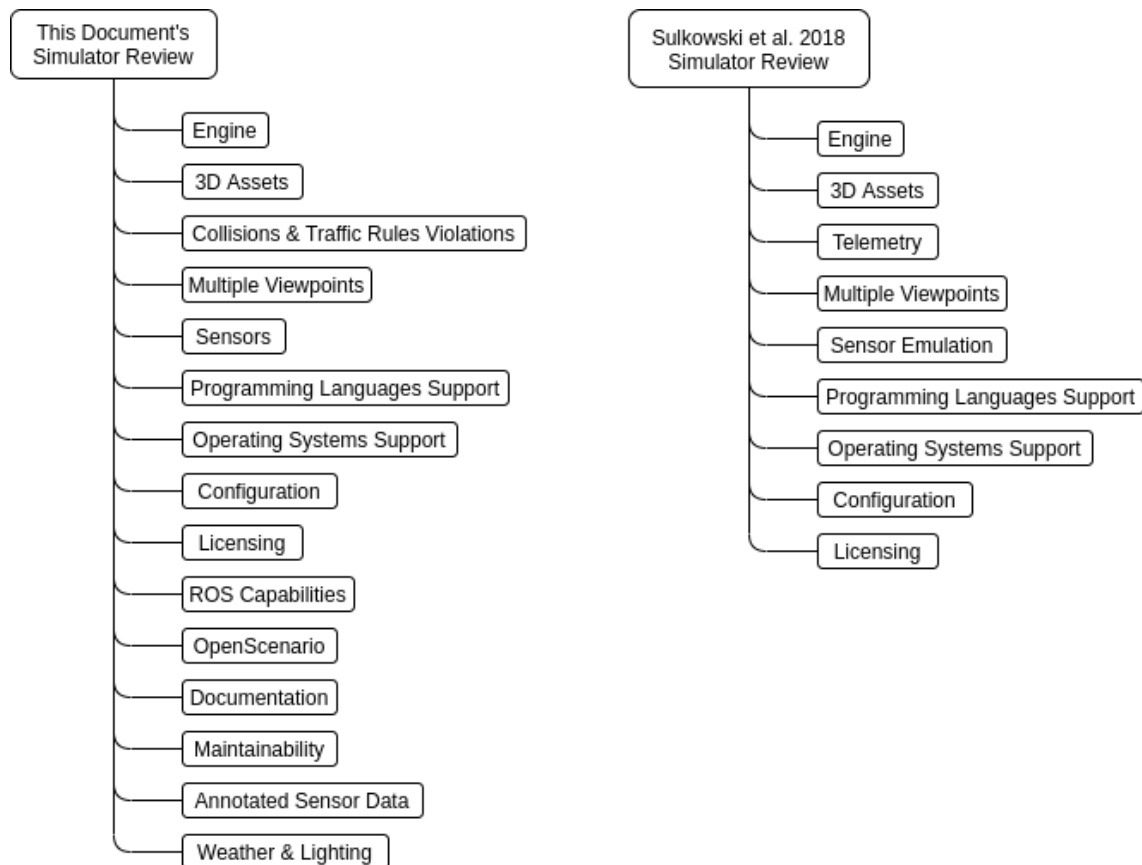


Figure 2.5: Current vs Sulkowski et al. 2018 simulator review model

Engine

Most simulators suitable for urban driving simulation are based on game engines which

have a major influence on many aspects of the resulting images and simulation, for instance image fidelity, physics realism, etc.

This parameter reveals, to the reader, what game engine framework was used as a base for the respective simulator, and that can be useful for project and plugins compatibility, further research and development or just to have a better idea of the expected results.

3D Assets

As already mentioned the realism of the produced scenario is imperative to achieve a superior ADS behavior. In this case, little details are very important and a good simulation should be able to integrate all objects that are normally observed in a real-life environment.

To fulfill this parameter the software needs to be able to provide traffic signs, road objects and other dynamic Non-Player Characters (NPCs), such as pedestrians and traffic.

Collisions & Traffic Rules Violations

As referred in the chapter 2.1, many ADSs algorithms rely on reinforced learning methods. Therefore, the ability to provide feedback on the performed actions is essential for this type of simulators, most importantly information about collisions (position, objects involved and impact) and reports of traffic rules violations.

If more than the basic information is provided about collisions and infractions this category is considered accomplished.

Multiple Viewpoints

Being able to watch the ego vehicle behaviors, or any other object surrounding it or just present in the simulator is an advantage that facilitates debugging of agent decisions and supervision of algorithms.

This topic evaluates whether the simulator is capable of providing the user with several viewing angles, including free cam to watch all the simulator's world.

Sensors

Various types of sensors are used in ADSs. Usually a vehicle is composed of multiple sensors and according to the literature, reviewed in chapter 2.1, the vehicle constitution varies quite extensively from project to project. This means that simulators need to be equipped with a wide variety of, both proprioceptive and exteroceptive, sensors.

Simulators are expected to support a considerable amount of proprioceptive sensors to provide position, speed, acceleration, orientation, etc. And, as far as exteroceptive sensors go, several types of cameras, RADAR and LIDAR should be supported.

Programming Languages Support

Different programming languages bring different capabilities to projects. Each project is written in its most convenient language, whether it is for its features, programmers experience or even taste, and if simulators support multiple languages then there is a better compatibility with preexisting projects and it gives more freedom to the users.

Operating Systems Support

According to the literature review on autonomous driving, chapter 2.1, ADSs can be designed in modular structures which allows for modules to be developed at a different time. Therefore, some parts of the software might already be developed when researchers start inquiring about simulation solutions. To increase compatibility with projects it is important for a simulator to support several operating systems.

Configuration

In order to simulate diverse driving scenarios a great deal of control of the simulator's settings during runtime is essential. If a simulator is capable of providing this kind of management, the researcher is able to easily simulate sequences of different scenarios by changing the traffic, lighting or even the weather.

A complete software should allow for these types of configurations to be easily accessible either through an interface or through it's Application Programming Interface (API).

Licensing

The software license, which specifies the copyrights for the source code and object code, can vary from free and open-source to paid and closed-source. The type of licensing is very important both for companies who might want or not to commercialize the product and for researchers who might want to further develop the simulator capabilities. In this sense, the more open-source and free the better. [14]

ROS Capabilities

ROS standard implements a structured communication protocol to provide some kind of abstraction between different connected systems. This popular tool in robotic systems allows for the development of modular software unaware of other software specifications. By using this framework ADS control algorithms could be developed to be testing in different systems, simulators and real cars, without the need to constantly be adapted. [15]

OpenSCENARIO

OpenSCENARIO is a definition standard to provide description of complex maneuvers involving dynamic content in a (virtual) world.[1] It contains a common vehicle database, collection of Advanced Driver-Assistance System (ADAS) sample scenarios and an open source scenario validation tool that is helpful to define different testing situations. It incorporates OpenDrive at its core to handle the static content. [16]

Documentation

Any software development should be followed with guidelines for its features and use cases. This documentation facilitates the usage of the software and enables the user to harness the full capabilities of the software while reducing the learning difficulty and time.

Maintainability

Updating and continuous development of these types of simulators is a requirement. Autonomous driving is currently a research focus and new technologies, methods and hardware keep being developed. The consequence is the need to constantly improve the software capabilities in order to keep up with the state of art.

Annotated Sensor Data

As stated in chapter 2.1, popular algorithms in ADSs are based on supervised learning. The requirement to use these algorithms is to have huge annotated datasets of labelled data but, unfortunately, creating these datasets on the roads is dangerous and expensive so the solution relies on creating the datasets on simulators, which consists in exporting annotated sensor data.

Weather & Lighting

Many difficult environments for autonomous systems are related to the weather and lighting variances. Training an ADS on broad daylight has different challenges from the sunset and clear sky is completely unrelated with a rainy day. Thus, it is a requirement to have the ability to simulate different types of weather and lighting.

2.4.1 Comparison

2.4.1.1 CARLA

CARLA is an open-source simulator that has been developed to support training, prototyping and validation of autonomous driving models. It includes a multitude of urban 3D assets, sensor suites and a wide range of environmental conditions. It was implemented as an open-source layer over UE4 which permits future extensions by the community. A server-client system structure was used where the server runs the simulation and renders the scene. The client sends commands and meta-commands to the server in order to receive sensor readings and control the ego-vehicle. Pedestrians navigate the streets encouraged to walk along sidewalks and marked road crossings. The appearance of NPCs is randomized when they are added to the simulation to increase visual diversity. [7]

Engine: Developed on top of UE4[7];

3D Assets: Contains buildings, vegetation, traffic signs, traffic vehicles, pedestrians and other infrastructure objects [7];

Collisions & Traffic Rules Violations: Provides cumulative impact with cars, pedestrians and static objects and detects infractions, such as opposite lane intersection, speeding and traffic lights violation [7];

Multiple Viewpoints: Support for changes in angle view and camera position relative to the car. Also provides final scene, depth view, segmentation view [5];

Sensors: Sensor suite of rgb cameras, depth cameras, semantic segmentation, LIDAR, RADAR, collision, lane invasion, obstacle, Global Navigation Satellite System (GNSS) and IMU [2];

Programming Languages Support: It includes C++ and Python API [5];

Operating Systems Support: Supports Windows and Linux, since it was developed as a layer over UE4 [7];

Configuration: Meta-commands allows the user to control the number of vehicles and pedestrians automatically spawned in the city and how they are spawned, the current weather and lighting conditions and the camera view [7];

Licensing: The simulator is free and open-source supported by the MIT license [5];

ROS Capabilities: Accompanied by a ROS package to provide a ROS bridge it is possible to publish different types of sensor data [17];

OpenSCENARIO: This compatibility is provided by a traffic scenario definition and an execution engine for CARLA [18];

Documentation: CARLA contains extensive documentation of its features and use cases [2];

Maintainability: CARLA Github is quite active, at the time of the writing, and the most recent release was in April 2020 [19];

Annotated Sensor Data: Provides access to exact locations and bounding boxes of all dynamics and static objects [7];

Weather & Lighting: Lighting conditions such as position and color of the sun and intensity and color of diffuse sky radiation can be altered. As well as ambient occlusion, atmospheric fog, cloudiness and precipitation [7].

CARLA simulator provides the necessary conditions to simulate various challenging scenarios for ADSs and is convenient to train and validate different types of algorithms with its feedback information and annotated data. Researchers and developers are granted quite a lot of freedom with this simulator as it can run in the two most popular operating systems, supports ROS and OpenScenario standards and is free and open-source.

2.4.1.2 Microsoft AirSim

The Microsoft AirSim is an open-source platform that aims to aid in the development of autonomous vehicles. Although initially focused on autonomous flying drones, it contains many features needed in autonomous driving simulation [5]. Made publicly available on February 2017 [20], this platform is implemented as a plugin on top of UE4 and Unity [21] and follows a modular structure in order to facilitate user extensibility. AirSim provides a high-fidelity physical and visual simulation that generates large training and validation datasets adequate for machine learning models.[22]

Current API supports Remote Procedure Call (RPC) communication protocol which, by creating an abstraction on top of the simulator, grants the ability for users to develop control software on any programming language. [20]

Engine: Provides implementations on UE4 and Unity [21];

3D Assets: Contains, by default, several ego-vehicle models, NPC vehicles and pedestrians. Some static content is also provided as traffic signs, road objects and other infrastructure content [5];

Collisions & Traffic Rules Violations: Although it informs of impact position, impact normal and penetration depth for each collision, the simulator does not incorporate traffic rules violation analysis [22];

Multiple Viewpoints: The simulator supports sub-windows in over-lay configurations to choose between normal, depth and segmentation view. Camera position can also be altered [5];

Sensors: AirSim only provides proprioceptive sensor models, i.e. accelerometer, gyroscope, barometer, magnetometer, IMU and Global Positioning System (GPS) [22];

Programming Languages Support: With RPC protocol it is possible to use any programming languages [20];

Operating Systems Support: Windows, Linux & OSX are supported [21];

Configuration: It allows extensive customization through the use of its settings file, such as the vehicle, physics engine, flight controller, etc [20];

Licensing: Free and open-source software [22];

ROS Capabilities: No support for ROS communication is provided;

OpenSCENARIO: No support for OpenSCENARIO is provided;

Documentation: There are complete sets of documentation both on the simulator as on the API. [21];

Maintainability: The most recent release happened on May 2019 and commits have been pushed on January 2020, so the simulator is still maintained and in development [21];

Annotated Sensor Data: Sensor feeds may be labelled and can include 1st and 2nd order derivatives for moving objects [21];

Weather & Lighting: Advanced control of lighting, sun position according to time of the day, and weather, such as wind, rain, cloudiness, snow and leaves, is easily accessible [21].

With both UE4 and Unity development plugin, AirSim is compatible with a wider range of software which might be appealing for researchers. On the other hand the lack of exteroceptive sensors highlights a major flaw in this simulator that can be corrected only by designing new sensor models, but at a greater cost for the user. In the end, AirSim is a great solution as an urban driving simulator if the user intends to develop sensor models.

2.4.1.3 SynCity

SynCity is a paid and closed-source simulation platform used to generate data for neural network training and validation in autonomous applications, ADAS and smart sensors. [23] As a commercial software, it provides little documentation or insight on its architecture. [24]

Engine: Based on UE4 [25];

3D Assets: Contains several types of dynamic, vehicles and pedestrians, and static, infrastructure, objects [23];

Collisions & Traffic Rules Violations: No information is provided;

Multiple Viewpoints: Allows to change angle view and camera position relative to the vehicle [23];

Sensors: Syncity provides different types of sensors, LIDAR, RADAR, rgb cameras, GPS and infrared [23];

Programming Languages Support: Provides C++ and Python API [23];

Operating Systems Support: Windows and Linux are supported [26];

Configuration: It is possible to configure several environment settings from its GUI [23];

Licensing: This software is paid and closed-source [24];

ROS Capabilities: ROS communication is supported [23];

OpenSCENARIO: This feature is not supported;

Documentation: There is little documentation on this simulator [23];

Maintainability: Since CVEDIA keeps posting results from its simulator it is assumed that it is maintained and still in development [23];

Annotated Sensor Data: Syncity is able to export a variety of labelled data, such as bounding boxes, proprioceptive data, segmentation and many other sensor types [23];

Weather & Lighting: In its GUI lighting and weather can be extensively configured [23].

CVEDIA's software is considerably complete and provide a lot of features to simulate several different ADSs and scenarios. It is also a powerful tool to generate datasets for machine learning algorithms. Unfortunately it is a paid and closed-source simulator which provides little documentation on use cases and system architecture.

2.4.1.4 Voyage DeepDrive

Voyage DeepDrive is an open-source free urban driving simulator focused on Deep Reinforcement Learning. Successor of DeepDrive simulator, it is the first release of this new version collaborating with Voyage. To motivate autonomous drive advancements, Voyage's concentrated on developing challenges for the community. [27]

Engine: Based on UE4; [28]

3D Assets: Although it provides several types of 3D objects and vehicles, it does not contain pedestrians, an essential part for urban driving simulators; [28]

Collisions & Traffic Rules Violations: No information was found;

Multiple Viewpoints: It allows changing points of view between chase cam, lateral orbit camera, hood camera and free camera; [29]

Sensors: It provides several proprioceptive sensors, for example speed, acceleration, rotations, but as far as exteroceptive sensors go, only rgb and depth cameras are supported; [28]

Programming Languages Support: It only allows interfacing with the UE4 [28], so it is possible to use C++ and Python using UnrealEnginePython; [30]

Operating Systems Support: Only Linux is currently supported; [28]

Configuration: No information is provided;

Licensing: This is an open-source and free software using the MIT license; [29]

ROS Capabilities: No support;

OpenSCENARIO: No support;

Documentation: Little documentation is provided; [28]

Maintainability: This is a recent simulator, launched in 2019; [27]

Annotated Sensor Data: No annotated sensor data is provided;

Weather & Lighting: No weather and lighting editing is possible;

Since Voyage DeepDrive is focused on reinforcement learning algorithms, it does not provide labelled datasets. It is also quite incomplete on several other features, like weather and lighting control. But this lack of features are probably due to its young age and might be developed later in time.

2.4.1.5 rFPro

rFPro is a closed-source and paid simulator. It does not provide almost any 3D models focusing in developing the physics and realism and leaving the 3D model development for the client. This means that besides paying for the simulator, the client will also have to invest in developing all the necessary 3D models for their scenarios. [31]

Engine: No information was found;

3D Assets: No assets are provided; [31]

Collisions & Traffic Rules Violations: Collision detection and traffic infractions are provided; [31]

Multiple Viewpoints: Multiple cameras are available, such as driver, follower and free cam; [31]

Sensors: Several different sensors are provided, cameras, LIDAR, RADAR, GPS, infrastructure sensors, etc; [31]

Programming Languages Support: Simulink and C++ API are provided; [31]

Operating Systems Support: Windows and other operating systems like dSpace and Speedgoat are supported; [31]

Configuration: Runtime configuration allows to change weather and lighting conditions; [31]

Licensing: This is a paid and closed-source software; [31]

ROS Capabilities: No support;

OpenSCENARIO: No OpenSCENARIO is supported;

Documentation: No documentation is accessible; [31]

Maintainability: No information was found;

Annotated Sensor Data: Labelled sensor feeds are provided, and 1st and 2nd derivatives from moving objects can be exported; [31]

Weather & Lighting: This simulator allows to configure various parameters of weather and lighting conditions; [31]

Although no assets are provided, rFPro is a very complete simulator which appears to provide a great deal of features and realism. In addition to the provided API's, interface with the simulator is possible using User Datagram Protocol (UDP). However, there is not much information available since it is a closed-source simulator.

2.4.2 Overview

The table 2.1 presents a final comparison between all the previously analyzed urban driving simulators. In this table the (✓) mark is used when a simulator fulfills all the requirements in a category, the (✗) mark indicates it lacks some features in the corresponding category and the (X) mark that the simulator does not provide any features in this category. On the rare occasion information on a specific category is not found, it is signaled with the (- -) mark.

Of the five reviewed simulators, Voyage DeepDrive and rFPro were the ones that provided the least number of features expected from an urban driving simulator. Unlike SynCity and rFPro, CARLA, Microsoft AirSim and Voyage DeepDrive are free and open-source which can be a deciding factor since they have an impressionable amount of features. Microsoft AirSim has the main advantage of providing extra flexibility with programming languages but CARLA is the one that features the best score in this review, fully supporting all the required categories.

Table 2.1: Urban driving simulators review

Simulators	CARLA	Microsoft AirSim	SynCity	Voyage DeepDrive	rFpro
Engine	UE4	UE4 & Unity	UE4	UE4	--
3D Assets	✓	✓	✓	✗	✗
Collisions & Traffic Rules Violations	✓	✓	--	✓	✓
Multiple Viewpoints	✓	✓	✓	✓	✓
Sensors	✓	✗	✓	✗	✓
Programming Languages Support	C++ & Python	Any with RPC support	C++ & Python	C++ & Python	FMI, C++ Simulink...
Operating Systems Support	Linux & Windows	Linux, OSX & Windows	Linux & Windows	Linux	Windows & others...
Configuration	✓	✓	✓	✗	✓
Licensing	Free & Open-Source	Free & Open-Source	Paid & Closed-Source	Free & Open-Source	Paid & Closed-Source
ROS Capabilities	✓	✓	✓	✗	✗
OpenScenario	✓	✗	✗	✗	✗
Documentation	✓	✓	✗	✗	✗
Maintainability	✓	✓	✓	✓	✓
Annotated Sensor Data	✓	✗	✓	✗	✗
Weather & Lighting	✓	✓	✓	✗	✓

2.5 CARLA Simulator

The following information contained in this section was obtained from CARLA's Documentation [2].

CARLA is an open-source simulator that has been developed to support the development, training and validation of ADSs and also contains open digital assets [7]. This simulator was developed on a scalable client-server architecture to enable a series of clients to run and collaborate on a shared environment, figure 2.6. The server is responsible for running the simulation providing sensor rendering, physics computation and update the world and actors state. The client side consists of modules controlling the logic of actors and setting the world conditions by issuing commands to the server. Communication client-server is performed through Transmission Control Protocol (TCP) ports, defined as 2000 and 2001 as default, using its C++ and Python APIs.

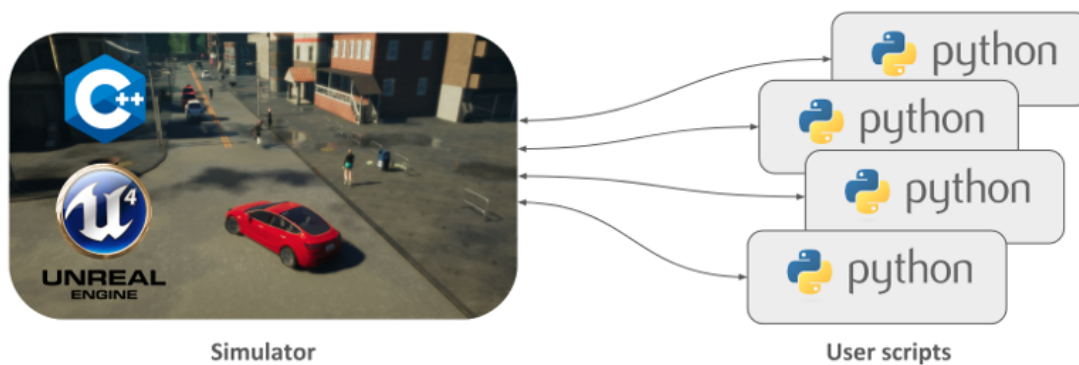


Figure 2.6: Carla client-server architecture [2]

When launching the software the user is able to choose several settings: graphics API, quality level, run off-screen, no rendering, notion of time and synchronous mode. Vulkan is the default graphics API, that runs faster at the cost of more memory consumption, but OpenGL can be selected as an alternative. Regarding the quality level, there are two levels, epic and low, seen in figure 2.7, where epic is the default. If the OpenGL graphics API is being used it is possible to run the server off-screen, meaning the simulator is running but without launching any window. To achieve higher frequencies, the user can opt for not rendering which results in faster simulation of traffic and road behavior because of the reduced rendering overhead at the cost of disabling the cameras and other render-based sensors. Usually games try to simulate real time by using variable time-steps, this feature is defaulted in CARLA but it does not render repeatable simulations. As an alternative, fixed time-step can be chosen to allow to simulate longer periods in less time and gain repeatability, by reducing floating-point arithmetic errors introduced in real-time. A synchronous mode can also be enabled where the simulation halts until a *tick message* is received from the client, a feature extremely useful to synchronize sensors data.

The 3D assets, models and maps are maintained in an open repository[32]. CARLA has a extensive asset library to provide the most realistic and complete scenes. More than 90 static objects are available ranging from small items, such as bags, newspapers and leaves, to small



Figure 2.7: Carla epic vs low quality level [2].

It is noticeable a smaller rendering radius in low quality level.

infrastructures, for instance ATMs, benches and fountains to bigger infrastructures, buildings, houses, tunnels, etc. In order to simulate real roads many traffic signs and traffic lights are supported. Fourteen types of pedestrian, adults and children, with several outfits and accessories, like smartphones, guitar cases and umbrellas, are randomized for increased visual fidelity. Finally, 27 different vehicle models were designed identical to real consumer vehicles from various brands: BMW, Audi, Mercedes, Tesla, and more. In case the user requires specific items not contained in the open repository, there are tutorials present in the documentation explaining how to design and import new 3D assets. Seven different cities can be used to train and validate machine learning algorithms.

All sensors models present in the simulator are configurable in multiple ways and can be easily placed in the desired portion of the vehicle. The currently supported sensors are described in table 2.2.

A new algorithm was implemented, named Traffic Manager, to control NPC vehicles with the objective to improve the way cars roam through the city. Built on the client side, it provides a more configurable traffic controller and reducing server overhead at the same time. With this technology a set of parameters are configurable: distance to leading vehicle, above speed limit driving allowance, velocity related to the speed limit, respectability of "keep right" rule, running traffic lights, ignoring pedestrians or other vehicles, forcing lane changes, allowing lane changes, forcing overtaking and avoiding tailgating.

CARLA carries out a complete environment control with an actor responsible for modifying all the lighting and environmental actors. The users are able to adjust weather and illumination according to their needs with the assistance of various settings, such as sun angle, direction, brightness and color, cloud percentage and color, horizon gradient colors, sky's color and light intensity, precipitation amount and accumulation and wind angle and intensity.

2.5.1 Unreal Engine 4

UE4 is a game engine with state-of-the-art rendering quality and realistic physics that also provides a varied set of interoperable plugins that allow to extend its capabilities. CARLA was developed

Table 2.2: Currently available CARLA sensors

Sensor type	Description	Configurable parameters
RGB camera	This sensor acts as a regular color camera capturing images.	Its resolution, camera settings and lens distortion can be fully configured. Furthermore, additional advanced camera attributes are provided by UE4.
Depth camera	Also known as depth buffer or z-buffer, this sensor codifies the distance of each pixel into an image.	Besides resolution and FOV its frequency and lens distortion are also configurable.
Semantic segmentation camera	By tagging more than 13 different objects beforehand, this sensor classifies the scene into a segmented image, each label with different colors. More labels can be defined by the user.	Resolution, FOV, frequency and lens distortion are configurable.
LIDAR	This sensor, by using ray-cast, provides a point-cloud similar to a LIDAR sensor. A laser for each selected channel is distributed in the vertical FOV and a rotation is performed while ray-casting each laser's path.	The channel number, range, points per second, rotation frequency, vertical FOV, and frequency are configurable parameters.
RADAR	An implementation of a low-fidelity RADAR, by using ray-casting, provides range, azimuth, altitude and velocity.	Both vertical and horizontal FOV are configurable, as well as points per second and range parameters.
GNSS	Calculates current GNSS position by adding the position to a reference location present in the OpenDrive map definition.	No parameters are configurable in this sensor.
IMU	This sensor provides accelerometer, gyroscope and compass information.	No parameters are configurable
Obstacle	This sensor creates "fake" actors, with all the relevant information, every time it detects an object ahead of the vehicle that is not an actor.	Range, radius, visibility and frequency are configurable parameters. An option to detect only dynamic objects is also available.
Collision	This sensor creates "fake" actors every time the ego collides against any object. These actors have all the information about the collisions.	No parameters are configurable in this sensor.
Lane invasion	By working on the client side, this sensor provides the percentage of the vehicle trespassing to the neighboring lane based on the OpenDrive description.	No parameters are configurable in this sensor.

as a plugin on top of UE4 which means it can use these features to develop a realistic environment, ideal for autonomous driving.

The simulator's maps can be created by generating map files, with the help of VectorZero's RoadRunner software or other tools, and importing it to UE4. CARLA is then able to generate waypoints, based on the OpenDrive files, which allows Traffic Manager to autonomously navigate the vehicles through the map. In figure 2.8 it is presented an intersection with all the possible vehicle paths defined as connections of waypoints, represented as red boxes. Waypoints are structured in a graph-style pathways to allow vehicles to travel the road by iterating through the nodes. Represented as hollow yellow boxes are the traffic triggers and every road segment entering an intersection is accompanied by two triggers. The first one, farthest from the intersection, signals traffic lights and other cars that a vehicle is approaching and the second one, just before the intersection, signals the respective car to wait if needed in order to drive safely.



Figure 2.8: Waypoints and traffic trigger boxes in a intersection [2]

Vehicles are models created accordingly to UE4's user guide and so aren't different from other simulators. Two-wheeled vehicles are designed as a four-wheeled vehicles while adding only a little extra complexity to appear a normal motorbike. Through CARLA's API is it easy to define specific vehicle body, wheel and gear physics parameters, as described in table 2.3.

Pedestrians follow an identical navigation system by using generated waypoints indicating all the possible pathways. A location-based cost algorithm is implemented in order to encourage pedestrians to walk along sidewalks and marked road crossings while also permitting crossing the road at any point for a more realistic environment. These actors roam freely in the city while trying to avoid other actors. In case a collision occurs, the pedestrian involved is eliminated and another one is spawned at a different location. All maps have defined specific locations where pedestrians are able to spawn and specific locations for vehicles to spawn, called spawn points.

2.5.2 ROS Bridge

ROS Bridge was first released in 2018 as an add-on for CARLA simulator to support ROS communications and control. Currently compatible with CARLA 0.9.4 or newer versions it features sensor publishing and control over pedestrians and vehicles. This ROS package provides several example scripts to facilitate learning and its configuration is conveniently organized in a single YAML file.

In asynchronous mode, by default, this package, publishes data every "world.on_tick()" and "sensor.listen()" callbacks. On synchronous mode, ROS controls the time-step by *ticking* the simulator only when all sensor data has finished receiving. This feature can slow down the simulation

Table 2.3: Vehicle configurable physics parameters

Component	Parameter	Description
Body	<i>torque_curve</i>	Defines a curve that relates torque (Nm) with the RPM of the vehicle's engine.
	<i>max_rpm</i>	Indicates the maximum RPM of the engine.
	<i>moi</i>	Defines the MoI of the vehicle's engine.
	<i>damping_rate_full_throttle</i>	Sets the damping rate when the throttle is at maximum value.
	<i>damping_rate_zero_throttle_clutch_engaged</i>	Sets the damping rate when the throttle is at zero and the clutch is engaged.
	<i>damping_rate_zero_throttle_clutch_disengaged</i>	Sets the damping rate when the throttle is at zero and the clutch is disengaged.
	<i>use_gear_autobox</i>	Allows to set the vehicle with automatic transmission.
	<i>gear_switch_time</i>	Defines the switch time between gears.
	<i>clutch_strength</i>	Sets the clutch strength of the vehicle.
	<i>final_ratio</i>	Defines the ratio from the transmission to the wheels.
	<i>forward_gears</i>	Allows to list the available gears in the vehicle.
	<i>mass</i>	Indicates the vehicle mass in Kg.
	<i>drag_coefficient</i>	Sets the drag coefficient of the vehicle's chassis.
	<i>center_of_mass</i>	Indicates the position of the center of mass of the vehicle.
	<i>steering_curve</i>	This curve defines the maximum steering angle for specific forward speeds.
	<i>wheels</i>	List all the wheel objects that define the wheel physics.
Wheel	<i>tire_friction</i>	Indicates the value of friction of the wheel
	<i>damping_rate</i>	Sets the damping rate of the select wheel.
	<i>max_steer_angle</i>	Defines the maximum angle a wheel can steer.
	<i>radius</i>	Indicates the wheel radius in centimeters.
	<i>max_brake_torque</i>	Indicates the maximum brake torque of the wheel.
	<i>max_handbrake_torque</i>	Indicates the maximum brake torque of the wheel when the handbrake is used.
	<i>position</i>	Allows to set the position of the wheel.
Gear	<i>ratio</i>	Allows to set the transmission ratio of the respective gear.
	<i>down_ratio</i>	Indicates the minimum RPM for the gear to down-shift.
	<i>up_ratio</i>	Indicates the maximum RPM for the gear to up-shift.

but it ensures reproducible results. A control topic is published that allows to pause, play and execute a single step of the simulation.

Except for the obstacle sensor, all the other sensors CARLA supports are able to publish with this package. A full list of actors, objects and their status are published in their respective topics. Full control of the vehicle is possible and it even allows to override any other client's control. An implementation of AckermannDrive messages is supported to provide smoother control of the vehicle with a PID controller. Pedestrians can also be controlled and their odometry information is supplied. To enable debugging it is possible to draw markers on the simulator, such as arrows, points, cubes and line strips, although this feature can interfere with sensor data. Finally, it allows spawning vehicles and configure their sensors, spawn infrastructure sensors and provides waypoint calculation.[17]

2.6 Conclusion

In this section we first reviewed the current state-of-the-art on autonomous driving to clarify what features would be important in a simulator to properly simulate urban environments to train ADSs. This was followed by a comparison of several urban driving simulators. In this comparison CARLA distinctively performed well in all categories emerging as the best solution for this thesis development. Thus, a further review of CARLA was carried out to provide a base knowledge of its features and capabilities.

Chapter 3

Approach and Work Plan

3.1 Research Approach

In order to develop ADSs that are compatible with different hardware implementations or even with hardware simulations, these systems need to be designed with versatility in mind and taking advantage of standard protocols of communication. Most available simulators only provide their specifically designed interface not fully compatible with any available standard.

Altran's interest in researching AD technology, requiring an urban driving simulator, is halted by this interface incompatibility. Instead of designing different control systems for each application, simulator and vehicle, the chosen approach was to develop one ADS with standard communication and create communication layers to bridge these two incompatible interfaces.

This thesis consists in evaluating the feasibility of implementing such communication layer on CARLA simulator, which demonstrated to be the most suitable software for the project needs. The proposed solution is to develop a system, presented in figure 3.1, taking advantage of CARLA's API, to allow: communication with control algorithms, exportation of labelled datasets using the ROS standard, definition of testing scenarios, control of the simulator and import other designed static meshes using the OpenSCENARIO standard. In figure 3.1 a color distinction is used to differentiate between the modules intended to be developed in this thesis, in white background, and the modules already developed that this thesis intends to take advantage, with grey background.

3.2 Work Plan

In order to complete the objectives presented in the chapter 1.3 the thesis work was divided into three major work packages, each responsible for: the scenario parsing, scenario GUI and ROS Bridge node. These work packages are, then, subdivided into smaller tasks presented below:

WP1: Scenario Configuration Support

This work package consists in further developing the scenario runner add-on for the CARLA

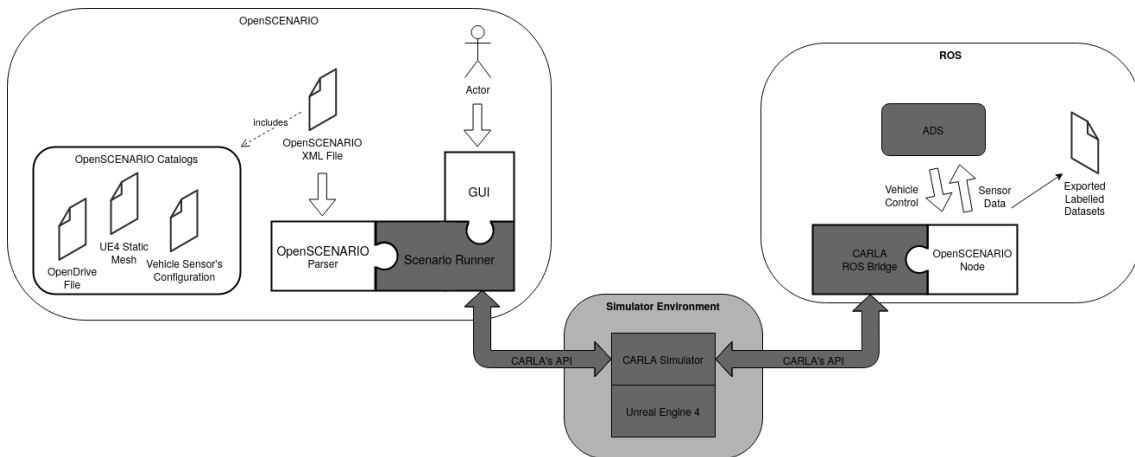


Figure 3.1: Proposed system structure

simulator, more specifically improving the OpenSCENARIO support provided by this software. The user should be able to configure all simulator's settings through an OpenSCENARIO eXtensible Markup Language (XML) file. These settings should include: environment conditions; NPC's, pedestrians and vehicles spawning; ego vehicle selection; ego vehicle's sensors setup and automatic map generation from an OpenDRIVE file.

T1.0 - API Structure & Plan: During this task, a more deep study of the scenario runner code and functionality will be studied in order to design our approach to the add-on development.

T1.1 - OpenSCENARIO Parser: This task consists in the development of the add-on, to parse the XML OpenSCENARIO script and execute its contents on the CARLA Simulator.

T1.2 - OpenSCENARIO v1.0.0: On mid March, a new version of OpenSCENARIO was released by ASAM e.V., OpenSCENARIO v1.0.0, with the goal to correct several bugs and errors present in the previous version and increase its coherence as a scenario description. This new update was provided with abundant documentation which proved useful to better understand some implementation details and correct a few misunderstandings on the standard. This documentation also defined how to act in certain situations, such as conflicting actions and unique naming. However, the two versions of the format are incompatible, and OpenSCENARIO v0.9.1 scripts had to be converted before executing as OpenSCENARIO v1.0.0 scripts.

Given all these advantages of this update and the incompatibility between the versions, it was considered essential to perform the portability of the parser to execute this new version.

WP2: Scenario Constructor Development

In order to create scenarios in an easier manner, an user graphical interface should be able

to help the user by providing a structured and fluid method of generating multiple scenarios without the need of advanced knowledge about the OpenSCENARIO standard.

T2.1 - Interface Structure: This task consists in the development of the interface graphical appearance and structure.

T2.2 - Interface Functionality: This tasks will consist in developing interface functionality related to the OpenSCENARIO standard but independent of any of its versions.

T2.3 - OpenSCENARIO v1.0.0: Introduction of OpenSCENARIO v1.0.0 structure and version specific data.

WP3: ROS Bridge Interface Development

The ROS Bridge provided by the CARLA simulator is capable of publishing sensor data and vehicle commands. The third work package will consist in further augmenting these features by providing the ability to export annotated sensor data originating labelled datasets to train and validate ADSs algorithms. Additionally, the topics to control all the objects and actors present in the simulator's world will be added.

T3.1 - Scenario Connection: Development of the capability to control and execute several OpenSCENARIO scripts.

T3.2 - Integration Node: Capability to publish all OpenSCENARIO entities sensors and odometry.

T3.3 - Correct Datasets: This task will consist in developing the feature to export corrected labelled datasets.

Chapter 4

WP1: Scenario Configuration

CARLA's scenario runner plugin allows to execute a great variety of scenarios that are required to fully test autonomous vehicles. This scenario construction capability brings a new potential for urban driving simulators, as it allows to extensively train ADSs even in situations with very low occurrence in real-life roads.

Adding OpenSCENARIO support to this tool increases convenience and standardizes the definition of scenarios to run in the CARLA simulator. As such, this work package was focused in developing an OpenSCENARIO parser that would work on scenario runner, making use of already existing and tested platforms. To improve compatibility with the open-source software and promote software modularity it was designed as an add-on with minimal editions to the source code.

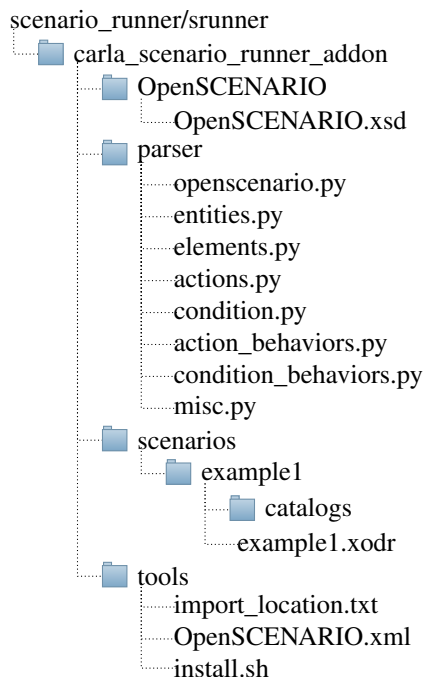


Figure 4.1: OpenSCENARIO's add-on content directory.

In figure 4.1 it is described the structure of this add-on folder that allows to execute OpenSCENARIO scripts on scenario runner. Although the GUI will be provided in the same directory, for simplification purposes, is not shown in this figure, and will be further explained in the next chapter. This folder is accompanied by an installation script that places it inside the "srunner" directory of the source code along with the necessary changes to properly execute it. All the user's XML scripts should be put in a folder with the script name, inside the "scenarios" folder, as exemplified by the "example1", and all the required catalogs inside a folder named "catalogs" in the same directory.

OpenSCENARIO standard definition files are present in the "OpenSCENARIO" folder and although they can be replaced with newer versions, it is important to verify the compatibility of the add-on with these versions.

Finally, all the add-on's source code is present inside the "parser" directory.

4.1 Scenario Runner

The scenario runner is a software module developed by CARLA for its urban driving simulator. It was designed to facilitate the creation and execution of complex traffic scenarios ensuring the repeatably required for ADS testing purposes. The CARLA Challenge[33] was also a motivation for the development of this addon, as it is used to evaluate the contestants' ADS agents performance.[18]

It allows to execute scenarios written in Python and OpenSCENARIO. The scenario execution is based in behavior trees, using the "py_trees" python library, where the event sequences are represented by atomic behaviors and conditions organized in a hierarchical tree. Scenarios can be programmed in Python by constructing its behavior tree and OpenSCENARIO scripts are converted into their version of a python behavior tree.

With scenario runner it is possible for users to add their own scripts and create groups of scripts. These groups of scripts can then be executed in sequence automatically.

4.2 Parameters

OpenSCENARIO provides a re-use mechanism, equivalent to the "variables" present in most programming languages, denominated "parameters", that allow to re-use and manipulate the same value across the script. This value can be declared in *Parameter Declarations* elements in the beginning of the script or inside specific elements, such as the *Story* element (listing 4.1).

```

1 <ParameterDeclarations>
2 <ParameterDeclaration name="var1" parameterType="boolean" value="true"/>
3 <.../>
4 </ParameterDeclarations>

```

Listing 4.1: Example of parameters in a OpenSCENARIO script.

The location of this declaration determines the parameters scope, parameters are global, i.e. can be used anywhere in the script, when declared in the preamble. All the other parameters are local and can only be used inside the element where they were declared and its children elements.

For example in the listing 4.2, the parameter "varLocal" can only be available to values inside the "FirstStory" and if tried to be used in values inside the "SecondStory" it should be indicated that the parameter does not exist.

Parameters can have several different types supported by the W3 2001 XMLSchema: boolean, dateTime, double, integer, string, unsignedInt, unsignedShort.

These can be accessed in any value throughout the script, according to their scope, using a reference to the parameter name (\$varLocal). All the values containing the parameter reference in the script will be replaced by the value of the parameter if its scope is valid and the parameter type matches the value type, otherwise an error should be thrown.

```

1 <.../>
2 <Storyboard>
3 <.../>
4 <Story name="FirstStory">
5   <ParameterDeclarations>
6     <ParameterDeclaration name="varLocal" parameterType="integer" value="10"/>
7   </ParameterDeclarations>
8 <.../>
9 </Story>
10 <Story name="SecondStory">
11   <ParameterDeclarations/>
12 <.../>
13 </Story>
14 <.../>
15 </Storyboard>

```

Listing 4.2: Example of local parameters inside the *Story* element.

4.2.1 Implementation

4.2.1.1 Parameter Declarations

Parameter declarations present in an OpenSCENARIO script, irrespective of their scope and location, are translated into a dictionary by the utility function `parameter_declaration(node: xml.etree.ElementTree.Element): dict`, found in the **misc** module, in the format of a tuple (name, value).

All the values in the dictionary are converted to the requested data type, as referenced in the **parameterType** attribute. This conversion is accomplished through a mapping of XSD data types into Python data types, present in listing 4.3.

```

1 parameter_type = {"boolean": (lambda x: (x == 'true'))},
2   "dateTime": (lambda x: time_parser(x)),
3   "unsignedInt": (lambda x: int(x) + 2**32),
4   "unsignedShort": (lambda x: int(x) + 2**32),
5   "integer": (lambda x: int(x)),
6   "double": (lambda x: float(x)),
7   "string": (lambda x: str(x))}

```

Listing 4.3: Dictionary with XSD to Python mapping data types.

4.2.1.2 Parameters Scope

In order to provide global and local scope for the parameters, every class, responsible for processing an OpenSCENARIO element, receives as an initialization argument, the parameters dictionary from its parent element. When an element contains local parameter declarations, a new copy of a dictionary is created with the parents parameters and the newly declared local parameters, listing 4.4. This new dictionary will, then, be passed to descendants of this class.

```

1 # Get local parameter declaration
2 decl_node = xml_node.find('ParameterDeclarations')
3 if (decl_node is not None):
4     decl = misc.parameter_declaration(decl_node)
5     self.parameters = dict(self.parameters, **decl)

```

Listing 4.4: Merging parent's with local parameters.

4.2.1.3 Parameter Reference

When processing the XML content of an element, its class needs to verify all attributes values for the presence of a parameter reference. Since all classes have their own parameter dictionary, with global and local declarations, in order to obtain the value of an attribute the function `get_attrib(node: xml.etree.ElementTree.Element, name: str, parameters: dict, default: ?, var_type: str): ?`, from the *misc* module, will always be used. This function verifies if a parameter reference is present in the attribute (passed in **name**) and if this parameter exists in the classes parameter dictionary (passed in the **parameters** argument).

4.2.1.4 Parameter Editing

During the scenario's execution, Actions can edit parameters and Conditions can read those parameters. Since this edition is only applied to global parameter declarations, a reference of the parameter dictionary directly from the OpenSCENARIO main class is passed to the respective actions and conditions.

4.3 Catalogs

Another re-use method present in the OpenSCENARIO standard, is the ability to provide catalogs for one or multiple scripts. A catalog is a collection of entries that represent a piece of code. They are useful to contain very lengthy pieces of code, making the script cleaner and easier to understand, or to contain pieces of code common to many scripts and spare the writer the work of rewriting the same code multiple times.

One OpenSCENARIO script may contain multiple catalogs, one for each available type. As shown in listing 4.5, there are 8 different types of catalogs, each containing unbounded entries of the element type referred in the name. E.g. A *VehicleCatalog* can contain multiple entries of the element *Vehicle*, and a *ManeuverCatalog* can contain multiple entries of the element *Maneuver*.

4.3.1 Implementation

One OpenSCENARIO script can have up to 8 catalogs and each catalog can have unlimited entries. Not all the entries are used in the script so processing all of them in the preamble would be a waste of time. As such, catalogs are parsed into dictionaries of entries.

Catalog files are expected to be present inside each scenario folder, and so, the path to be provided is the relative path of the file in relation to the scenario folder. In the following example


```

1 <xsd:complexType name="CatalogLocations">
2   <xsd:all>
3     <xsd:element name="VehicleCatalog" type="VehicleCatalogLocation"
4       minOccurs="0"/>
5     <xsd:element name="ControllerCatalog" type="ControllerCatalogLocation"
6       minOccurs="0"/>
7     <xsd:element name="PedestrianCatalog" type="PedestrianCatalogLocation"
8       minOccurs="0"/>
9     <xsd:element name="MiscObjectCatalog" type="MiscObjectCatalogLocation"
10      minOccurs="0"/>
11     <xsd:element name="EnvironmentCatalog" type="EnvironmentCatalogLocation"
12      minOccurs="0"/>
13     <xsd:element name="ManeuverCatalog" type="ManeuverCatalogLocation"
14      minOccurs="0"/>
15     <xsd:element name="TrajectoryCatalog" type="TrajectoryCatalogLocation"
16      minOccurs="0"/>
17     <xsd:element name="RouteCatalog" type="RouteCatalogLocation" minOccurs="0"/>
18   </xsd:all>
19 </xsd:complexType>

```

Listing 4.5: XSD definition to import catalogs.[34]

(listing 4.6) three catalogs present inside a "Catalogs" folder are imported into a script. Since all catalogs have the .xodr file format, only the name of the catalog file is necessary.

```

1 <CatalogLocations>
2   <VehicleCatalog>
3     <Directory path="Catalogs/VehicleCatalog"/>
4   </VehicleCatalog>
5   <EnvironmentCatalog>
6     <Directory path="Catalogs/EnvironmentCatalog"/>
7   </EnvironmentCatalog>
8   <PedestrianCatalog>
9     <Directory path="Catalogs/PedestrianCatalog"/>
10  </PedestrianCatalog>
11 </CatalogLocations>

```

Listing 4.6: Example to import three catalogs.

These catalogs would, then, be parsed by the Python's XML library but, instead of processing its entries, each catalog is represented by a dictionary containing all its entries and a pointer to its parsed dictionary. All these dictionaries are then contained into a dictionary member of the main OpenSCENARIO class, named "catalogs". A short example of the contents of this "catalogs" variable can be seen in listing 4.7, where two catalogs, *VehicleCatalog* and *EnvironmentCatalog*, were imported and the first catalog contains two entries, **audi.a2** and **chevrolet.impala**, and the second catalog contain only one entry, **sunny**.

4.4 Entities

OpenSCENARIO's dynamic actors, denominated entities, are the main focus of these types of scripts, with actions used to manipulate them and triggering conditions. In the script's preamble,

```

1 catalogs = {
2   'VehicleCatalog': {
3     'audi.a2': <Element 'Vehicle' at 0x7f6b8900d550>,
4     'chevrolet.impala': <Element 'Vehicle' at 0x7f6b8900da50>
5   }, 'EnvironmentCatalog': {
6     'sunny': <Element 'Environment' at 0x7f6b88ff9490>
7   }
8 }

```

Listing 4.7: Example of two imported catalogs.

an *Entities* element (listing 4.8) allows to declare entities intended to be used in the script and their specific properties.

```

1 <xsd:complexType name="Entities">
2   <xsd:sequence>
3     <xsd:element name="ScenarioObject" type="ScenarioObject" minOccurs="0"
4       maxOccurs="unbounded"/>
5     <xsd:element name="EntitySelection" type="EntitySelection" minOccurs="0"
6       maxOccurs="unbounded"/>
7   </xsd:sequence>
8 </xsd:complexType>

```

Listing 4.8: XSD definition of the *Entities* element.[34]

Scenario Object: An entity that is used in the script is considered an scenario object and is described in a *ScenarioObject* element, listing 4.9. Entities can later be classified into groups, with the *EntitySelection* element, allowing to use a group of entities in the scenario as it was only a single entity. The *Entities* element can contain unlimited *ScenarioObjects* and *EntitySelection* elements.

```

1 <xsd:complexType name="ScenarioObject">
2   <xsd:sequence>
3     <xsd:group ref="EntityObject"/>
4     <xsd:element name="ObjectController" type="ObjectController" minOccurs="0"/>
5   </xsd:sequence>
6   <xsd:attribute name="name" type="String" use="required"/>
7 </xsd:complexType>

```

Listing 4.9: XSD definition of the *ScenarioObject* element.[34]

A *ScenarioObject* element is identified by its "name" attribute. Later in the script, when it is intended to specify an entity to be used on an action or condition, this name is used as an entity reference. This element contains the entity type, present in the *EntityObject* group, and an optional default controller that manages the entity movements when not under the influence of an action.

In OpenSCENARIO it is possible to create three types of entities, as described in the *Entity-Object* group (listing 4.10) *Vehicle*, *Pedestrian* and *MiscObject*. Each of these types has their own catalog and have different properties.

```

1 <xsd:group name="EntityObject">
2   <xsd:choice>
3     <xsd:element name="CatalogReference" type="CatalogReference" minOccurs="0"/>
4     <xsd:element name="Vehicle" type="Vehicle" minOccurs="0"/>
5     <xsd:element name="Pedestrian" type="Pedestrian" minOccurs="0"/>
6     <xsd:element name="MiscObject" type="MiscObject" minOccurs="0"/>
7   </xsd:choice>
8 </xsd:group>

```

Listing 4.10: XSD definition of the *EntityObject* group.[34]

Entity Selection: After the entities are declared with *ScenarioObject* elements, they can be joined into groups. These groups allow to refer multiple entities at once in the script, either for actions or conditions. In order to create a group, one should utilize the *EntitySelection* element (listing 4.11) inside the *Entities* element.

```

1 <xsd:complexType name="EntitySelection">
2   <xsd:sequence>
3     <xsd:element name="Members" type="SelectedEntities"/>
4   </xsd:sequence>
5   <xsd:attribute name="name" type="String" use="required"/>
6 </xsd:complexType>

```

Listing 4.11: XSD definition of the *EntitySelection* element.[34]

Similar to the *ScenarioObject* element, the *EntitySelection* is also identified by its "name" attribute that can be used as an entity reference throughout the script. The *Members* element allows to define what entities belong to the *EntitySelection*, which is of *SelectedEntities* type, listing 4.12.

```

1 <xsd:complexType name="SelectedEntities">
2   <xsd:choice>
3     <xsd:element name="EntityRef" type="EntityRef" minOccurs="0"
4       maxOccurs="unbounded"/>
5     <xsd:element name="ByType" type="ByType" minOccurs="0" maxOccurs="unbounded"/>
6   </xsd:choice>
7 </xsd:complexType>

```

Listing 4.12: XSD definition of the *SelectedEntities* element type.[34]

The *SelectedEntities* element type allows to identify entities by two methods.

First, a list of the entity names can be provided. This list can contain unlimited *EntityRef* elements which contain the entity reference, content of the "name" attribute from the *ScenarioObject* elements.

Alternatively, the entities can be identified by its type, grouping all script's entities of the correspondent type ("vehicle", "pedestrian" or "miscellaneous"). Multiple types can be grouped into the same *EntitySelection*.

These two methods can be used in conjunction resulting in a group containing the union of the selected entities.

4.4.1 Entities in CARLA

In CARLA simulator, entities are referred as actors. There are also three types of actors, vehicle, pedestrian and props. Although the last type has a different name, prop actors are the equivalent of the OpenSCENARIO's *MiscObject* entities.

Available actors in CARLA are described by blueprints which allow to edit actor properties before spawning. The simulator contains a library where all the accessible blueprints are listed and after selecting the desired blueprint and edit its settings, the actor can be spawned into a position of the simulator's world. After spawning some settings, such as physics, can be edited.

Unlike the OpenSCENARIO scripts, in CARLA an entity only exists after being spawned, and it is necessary to have a defined position in the world to spawn, whereas in OpenSCENARIO an entity is required to be defined before a position is provided.

CARLA's actors have different blueprint attributes and after-spawn settings depending on the actor type. However, all actor have a bounding box, important to determine the object dimensions in case of collision, that is predefined and not editable during runtime.

4.4.2 Implementation

In the OpenSCENARIO main class, entities are created by its member function `_initialize_actors (config: dict): void`, overridden from the `BasicScenario` class. This function initializes all the entities by spawning them into the simulator's world, in a position that won't disturb the scenario, and creates a dictionary of all the entities spawned into the simulator.

In order to spawn entities before executing the script it was necessary to find a position that would be guaranteed it would not disturb any scenario event. However, with the ability to generate any map from an OpenDRIVE file, there was no position in the map that could meet these requirements. As an alternative, it is spawned a platform 200m under the map, shown in figure 4.2, that allows to spawn all the entities without interfering with the scenario or any entity sensors.



Figure 4.2: View of a CARLA's map with the platform spawned.

Each entity is, then, processed accordingly to its type. This is accomplished by looping through the list of *ScenarioObject* elements and mapping the entity type with the correct initialization class using the dictionary present in listing 4.13.

```

1 object_type = {"Vehicle": entities.VehicleConfiguration,
2   "Pedestrian": entities.PedestrianConfiguration,
3   "MiscObject": entities.PropConfiguration}

```

Listing 4.13: Linking dictionary for entity types.

Although each entity type has specific properties, some are common between between all types, in both OpenSCENARIO and CARLA. The *DynamicActorConfiguration* class (figure 4.3) is the base class for all types of entities, responsible for all the activities from selecting the blueprint to configuring actor settings after spawning. This class is abstract and each entity type is required to override the `_blueprint_editor(): void` member function with their specific blueprint attributes.

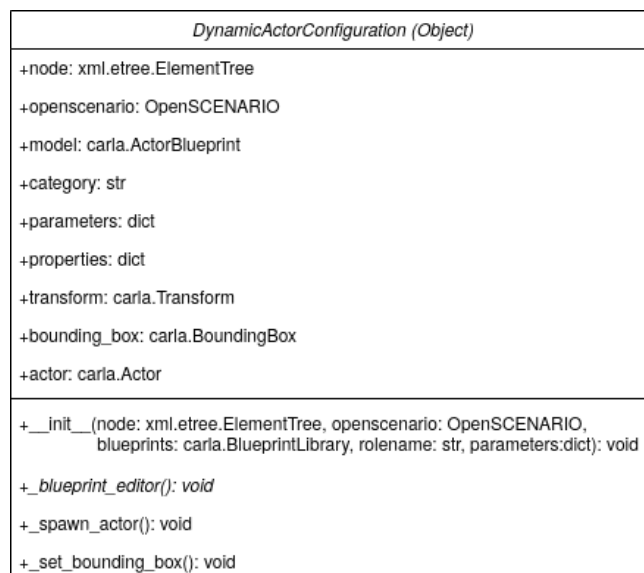


Figure 4.3: Entities Base Class - UML class diagram.

In the initialization, this class, starts by fusing the global with the local parameters dictionary. Since all entity types are required to provide information for the bounding box, this configuration is common to all classes. The next step is to form the dictionary with all the entity's properties, to be used later by the derived classes. After obtaining the location to spawn the entity, it sets the "rolename" of the blueprint and execute the abstract `_blueprint_editor(): void` member function. Finally it spawns the actor.

The derived classes are required to provide a filtered `carla.BlueprintLibrary` and a location to the base class when executing its initialization method. Any configurations required after the spawning of the actor, can be performed in the derived class.

4.4.3 Vehicle

The Vehicle entity type is the most complex entity, with both OpenSCENARIO and user defined properties. The *Vehicle* element (listing 4.14) contains local parameter declarations, as all elements that can be referenced in a catalog, and is identified by its "name" attribute.

```

1 <xsd:complexType name="Vehicle">
2 <xsd:all>
3 <xsd:element name="ParameterDeclarations" type="ParameterDeclarations"
   minOccurs="0"/>
4 <xsd:element name="BoundingBox" type="BoundingBox"/>
5 <xsd:element name="Performance" type="Performance"/>
6 <xsd:element name="Axles" type="Axles"/>
7 <xsd:element name="Properties" type="Properties"/>
8 </xsd:all>
9 <xsd:attribute name="name" type="String" use="required"/>
10 <xsd:attribute name="vehicleCategory" type="VehicleCategory" use="required"/>
11 </xsd:complexType>

```

Listing 4.14: XSD definition of the *Vehicle* element.[34]

This element allows to completely characterize a vehicle by using four types of properties. The OpenSCENARIO defines a vehicle using the *BoundingBox*, *Performance* and *Axles* elements. Implementation specific properties can be set in the *Properties* element.

Each vehicle, in a OpenSCENARIO script, is required to provide a *Performance* element, listing 4.15. In this element it is defined some characteristics of the vehicle in question. The "maxAcceleration" attribute defines the maximum acceleration a vehicle is capable of performing, by its own engine, and is provided in m/s^2 . The "maxDeceleration" attribute, as indicated by the name, defines the maximum deceleration a vehicle is capable of producing by utilizing its own brakes, also provided in m/s^2 . Finally, the "maxSpeed" attribute defines the maximum velocity the vehicle is capable of reaching on its own, provided in m/s . [34]

```

1 <xsd:complexType name="Performance">
2 <xsd:attribute name="maxAcceleration" type="Double" use="required"/>
3 <xsd:attribute name="maxDeceleration" type="Double" use="required"/>
4 <xsd:attribute name="maxSpeed" type="Double" use="required"/>
5 </xsd:complexType>

```

Listing 4.15: XSD definition of the *Performance* element.[34]

The "vehicleCategory" attribute (listing 4.16) allows to classify each vehicle into multiple common vehicle categories, such as *bicycle*, *car*, etc.

Another required element to define a vehicle entity in a script, is the *Axles* element (listing 4.17) where a vehicle's axles are described. Every vehicle is required to provide a front and rear axles, but it is possible to provide additional axles, such as bigger trucks with more than 4 wheels.

Each vehicle's axle is defined by a set of parameters provided as attributes in the *Axle* element, listing 4.18. Five values are required to correctly define an axle. It is assumed an axle is composed by two wheels.

```

1 <xsd:simpleType name="VehicleCategory">
2   <xsd:union>
3     <xsd:simpleType>
4       <xsd:restriction base="xsd:string">
5         <xsd:enumeration value="bicycle"/>
6         <xsd:enumeration value="bus"/>
7         <xsd:enumeration value="car"/>
8         <xsd:enumeration value="motorbike"/>
9         <xsd:enumeration value="semitrailer"/>
10        <xsd:enumeration value="trailer"/>
11        <xsd:enumeration value="train"/>
12        <xsd:enumeration value="tram"/>
13        <xsd:enumeration value="truck"/>
14        <xsd:enumeration value="van"/>
15      </xsd:restriction>
16    </xsd:simpleType>
17    <xsd:simpleType>
18      <xsd:restriction base="parameter"/>
19    </xsd:simpleType>
20  </xsd:union>
21 </xsd:simpleType>

```

Listing 4.16: XSD definition of the *VehicleCategory* attribute type.[34]

```

1 <xsd:complexType name="Axles">
2   <xsd:sequence>
3     <xsd:element name="FrontAxle" type="Axle"/>
4     <xsd:element name="RearAxle" type="Axle"/>
5     <xsd:element name="AdditionalAxle" type="Axle" minOccurs="0"
6       maxOccurs="unbounded"/>
7   </xsd:sequence>
8 </xsd:complexType>

```

Listing 4.17: XSD definition of the *Axles* element.[34]

```

1 <xsd:complexType name="Axle">
2   <xsd:attribute name="maxSteering" type="Double" use="required"/>
3   <xsd:attribute name="positionX" type="Double" use="required"/>
4   <xsd:attribute name="positionZ" type="Double" use="required"/>
5   <xsd:attribute name="trackWidth" type="Double" use="required"/>
6   <xsd:attribute name="wheelDiameter" type="Double" use="required"/>
7 </xsd:complexType>

```

Listing 4.18: XSD definition of the *Axle* element.[34]

The "maxSteering" attribute defines the maximum angle of steering of the wheels in the correspondent axle. This angle, provided in radians and as a positive double equal or inferior to π , indicate the maximum any wheel can turn to one side, and it is assumed the wheels can turn a simetrical angle to the opposite side.

The "positionX" and "positionZ" attributes allow to define the position of the axle respective to the vehicle point of origin. In OpenSCENARIO the vehicle's relative coordinates were conventionalized as displayed in figure 4.4, with the origin placed in the projection of the vehicle's rear axis on the floor. The "positionX" attribute, in meters, indicates the horizontal displacement of the axle, while the "positionZ" indicates the vertical displacement of the axle. The center of the axle

will always be aligned with the center of the Y-axis of the vehicle.

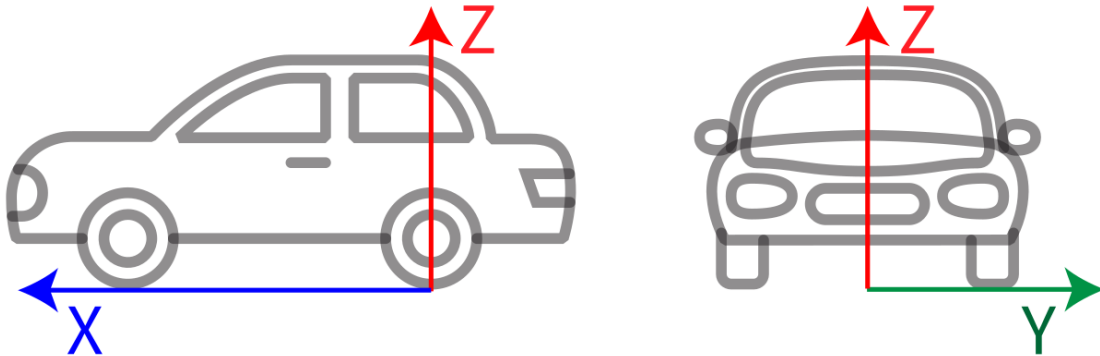


Figure 4.4: OpenSCENARIO vehicle coordinates reference[1].

The "trackWidth" attribute indicates the width of the axle or the distance between the axle wheels, also provided in meters. At last, the "wheelDiameter" attribute indicates the diameter of both wheels which is also provided in meters.

In order to better characterize the vehicle it is also possible to provide implementation specific properties. This is defined in the *Properties* element (listing 4.19) and can be provided in a tuple (name, value) or by a file. The meaning and function of each properties should be defined by the developer.

```

1 <xsd:complexType name="Properties">
2 <xsd:sequence>
3 <xsd:element name="Property" type="Property" minOccurs="0"
   maxOccurs="unbounded"/>
4 <xsd:element name="File" type="File" minOccurs="0" maxOccurs="unbounded"/>
5 </xsd:sequence>
6 </xsd:complexType>

```

Listing 4.19: XSD definition of the *Properties* element.[34]

4.4.3.1 Vehicles in CARLA

In CARLA simulator, vehicles are highly customizable, allowing to change some appearance settings, such as color, and to edit most physics and performance related parameters as shown in the table 2.3.

However most appearance construction are not editable at runtime, e.g. the number of axles, the axles length, wheels size, and many other characteristics, cannot be altered after designing. CARLA allows to create vehicles appearing to have different number of wheels but all vehicles are required to have four wheels. In case a truck, with more than four wheels, is designed the extra wheels only affect the appearance and are not functional. On the other hand, a two wheeled vehicle would have four wheels with the wheels sharing the same position two by two.[2]

4.4.3.2 Implementation

Derived from the `DynamicActorConfiguration` class, the `VehicleConfiguration` class (figure 4.5) deals with the spawning of vehicles. In CARLA the vehicle actor is the most complex type with blueprint properties and post-spawn properties, mainly physics settings.

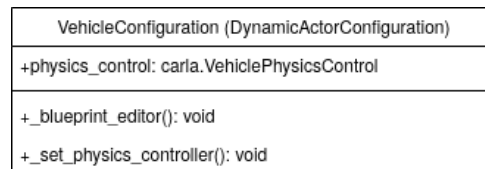


Figure 4.5: VehicleConfiguration UML class diagram.

In order to correctly identify the desired vehicle blueprint to spawn the "name" attribute is used to select the brand and model of the vehicle, using the "brand.model" format. However less specific identifications are possible: if only the brand is provided a vehicle of that brand will be randomly selected and if no information is provided a vehicle will be randomly selected from CARLA's library.

Since vehicle actors can be spawned in the simulator while having their physics disabled, in order to keep them in the simulation until requested to participate in the scenario, they are initially positioned under the city, at -150 meters of altitude distributed into a grid format forming a square 600 meters long. The actors are separated, in all directions, by at least 10 meters from each other, predicting the possibility of custom vehicles into the simulator, such as bigger trucks.

After selected the vehicle's blueprint and initial spawn location, it is possible to alter the actor's color, which is a blueprint property. After spawn it is possible to edit the vehicle's physics settings. All these settings can be defined as user defined properties and the table 4.1 contains a more detailed description of each available property and how to format its content.

As all vehicle's in CARLA require to have four wheels, in case only one wheel is characterized all four wheels will be characterized with the same values. Otherwise, in case only two wheels are characterized the front wheels and rear wheels will be paired and characterized with the same values. For more information see CARLA's documentation.[2]

4.5 Storyboard

The dynamic description of the scenario, in OpenSCENARIO scripts, is present in the *Storyboard* element. All the previous elements, *ParameterDeclarations*, *Catalogs*, *RoadNetwork* and *Entities*, can be considered the preamble of the script and allow to initialize the scenario in order to be ready to execute the storyboard.

The *Storyboard*, as described in the XSD schema (listing 4.20) is composed of an *Init* element, one or multiple *Story* elements, and, finally, a *StopTrigger* element. The *Init* element is used to establish the initial conditions of the scenario, position and velocities of entities, environment settings, among other OpenSCENARIO actions. The *Story* elements contain the timed actions

Table 4.1: Vehicle custom properties

Property	Default Value
<i>color</i>	random choice of recommended values (ex: "255,255,255")
<i>torque_curve</i>	"[[0.0, 500.0], [5000.0, 500.0]]"
<i>max_rpm</i>	"5000.0"
<i>moi</i>	"1.0"
<i>damping_rate_full_throttle</i>	"0.15"
<i>damping_rate_zero_throttle_clutch_engaged</i>	"2.0"
<i>damping_rate_zero_throttle_clutch_disengaged</i>	"0.35"
<i>use_gear_autobox</i>	"true"
<i>gear_switch_time</i>	"0.5"
<i>clutch_strength</i>	"10.0"
<i>final_ratio</i>	"4.0"
<i>forward_gears</i>	"[[1.0, 0.5, 0.65]]"
<i>mass</i>	"1000.0"
<i>drag_coefficient</i>	"0.3"
<i>center_of_mass</i>	"[0.0, 0.0, 0.0]"
<i>steering_curve</i>	"[[0.0, 1.0], [10.0, 0.5]]"
<i>wheels</i>	"[[2.0, 0.25, 70.0, 30.0, 1500.0, 3000.0]]"

of the scenario. The StopTrigger allows to terminate the scenario, whether because the objectives were accomplished or because an error has occurred.

```

1 <xsd:complexType name="Storyboard">
2 <xsd:sequence>
3 <xsd:element name="Init" type="Init"/>
4 <xsd:element name="Story" type="Story" maxOccurs="unbounded"/>
5 <xsd:element name="StopTrigger" type="Trigger"/>
6 </xsd:sequence>
7 </xsd:complexType>

```

Listing 4.20: XSD definition of the *Storyboard* element.[34]

4.5.1 Implementation

A class Storyboard (figure 4.6) is responsible for handling the creation of behavior tree nodes to correctly impersonate the required behavior. Since it is the first class responsible to process an element, it only receives the storyboard parsed node and a reference to the OpenSCENARIO main class.

In the initialization, four behavior tree nodes are created, as shown in figure 4.7, in order to ensure the correct execution of an OpenSCENARIO storyboard.

The Python's library "py_trees" provides an implementation of behavior trees that is used by the scenario runner and in this implementation of OpenSCENARIO. In this library there are two main nodes, that are used in this project, the parallel and the sequence nodes. The parallel node

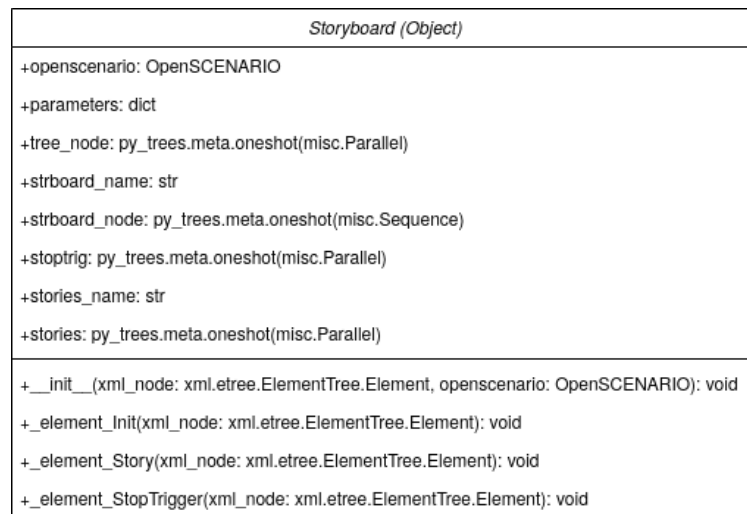


Figure 4.6: Storyboard UML class diagram.

allows to execute all its children in parallel while the sequence node execute its children one by one by the order they are presented, left to right. The parallel nodes present a particularity where it is possible to choose its success behavior, if they are "SuccessOnOne" it means that the node will succeed once one of its children succeeds, and if they are "SuccessOnAll" it means they will only succeed once all their children have succeeded.

The root node of the tree, **OpenSCENARIO**, is a parallel node with success on one. This allows to execute both its children, **Storyboard** and **StopTrigger**, at the same time and stop the scenario when one of them succeeds. E.g. If the *Storyboard's StopTrigger* element is triggered, the **StopTrigger** node would succeed ending the scenario.

The **Storyboard** node is a sequence node in order to allow to first execute the *Init* element and only after start the execution of the *Story* elements. Although the **Init** node is not initialized directly in this class, it is ensured to be positioned as the first child of the **Storyboard** node.

The **StopTrigger** node is also a parallel node with success on one. This allows to create the "OR" relationships between its children, the *ConditionGroup* elements.

Finally, the **Stories** node is a parallel node with success on all. This node allows to execute, after the completion of the **Init** node, all the **Story** nodes at the same time and only succeed once all these nodes have also succeeded.

4.5.2 Init

The *Init* is the first element of the *Storyboard* and it is mandatory that all scenarios have an *Init* element. This element has the main purpose to set the initial conditions for the execution of the scenario, such as entities initial positions and velocities, environment settings, and many more.

As shown in the listing 4.21, the *Init* contains one *Actions* element which is, in turn (listing 4.22) a container for global, private and user defined actions.

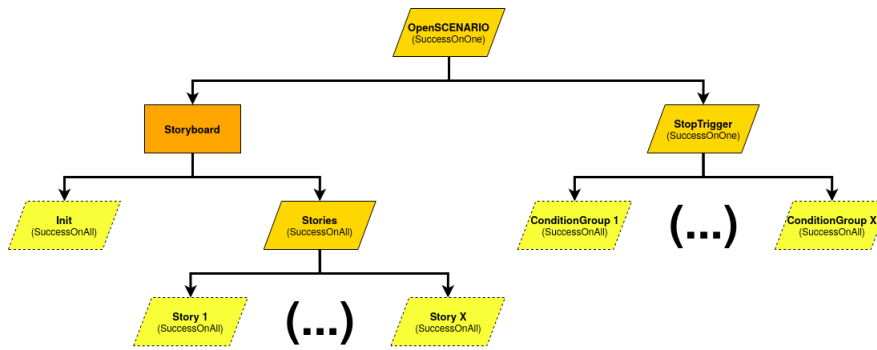


Figure 4.7: Storyboard behavior tree.

```

1 <xsd:complexType name="Init">
2   <xsd:sequence>
3     <xsd:element name="Actions" type="InitActions"/>
4   </xsd:sequence>
5 </xsd:complexType>

```

Listing 4.21: XSD definition of the *Init* element.[34]

```

1 <xsd:complexType name="InitActions">
2   <xsd:sequence>
3     <xsd:element name="GlobalAction" type="GlobalAction" minOccurs="0"
4       maxOccurs="unbounded"/>
5     <xsd:element name="UserDefinedAction" type="UserDefinedAction" minOccurs="0"
6       maxOccurs="unbounded"/>
7     <xsd:element name="Private" type="Private" minOccurs="0"
8       maxOccurs="unbounded"/>
9   </xsd:sequence>
10 </xsd:complexType>

```

Listing 4.22: XSD definition of the *InitActions*.[34]

Although *GlobalAction* and *UserDefinedAction* elements can be defined regularly and unlimited in this element, private actions require an entity to act upon provided by the *Private* element, listing 4.23.

```

1 <xsd:complexType name="Private">
2   <xsd:sequence>
3     <xsd:element name="PrivateAction" type="PrivateAction" maxOccurs="unbounded"/>
4   </xsd:sequence>
5   <xsd:attribute name="entityRef" type="String" use="required"/>
6 </xsd:complexType>

```

Listing 4.23: XSD definition of the *Private*.[34]

The *Private* element encapsulates *PrivateAction* elements to the same entity. In case the user wants multiple private actions for different entities, multiple *Private* elements would need to be used. The *Private* element is further explained in next sections.

In conclusion, the *Init* element is a container for OpenSCENARIO actions, and since it contains no triggers, all of its actions are executed in parallel in the beginning of the execution of the

script.

4.5.2.1 Implementation

The implementation of the `Init` class is quite standard, figure 4.8. Its initialization receives its xml node, a reference for the main `OpenSCENARIO` class and the parameters dictionary from its parent, in this case the `Storyboard` class.

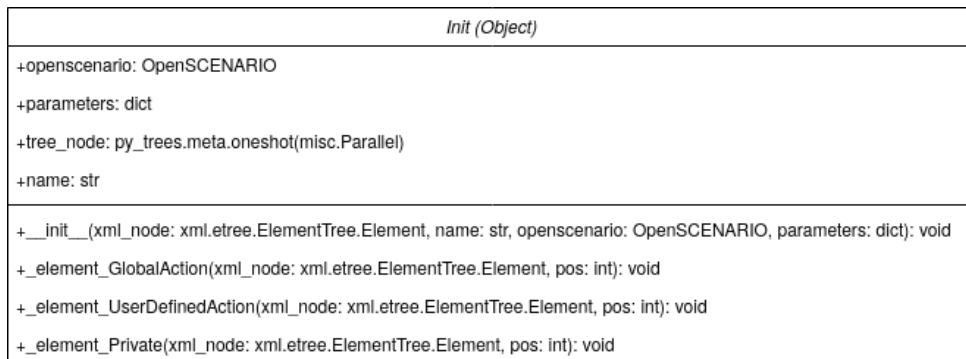


Figure 4.8: Init UML class diagram.

During the initialization this class constructs a simple behavior tree with only one node, figure 4.9. The member functions are responsible for creating this node children accordingly.

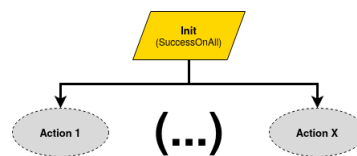


Figure 4.9: Init behavior tree.

The **Init** node is a parallel node with success on all. Being all the `Init` actions its children, the parallel node allows to execute all the actions simultaneously while only ending when all the actions are successfully complete.

4.5.3 Story

Following the `Init` element, the `Story` elements show up in the storyboard, listing 4.24. These elements contain the scenario's main story, providing actions with timelines through an intricate hierarchy of elements.

An `OpenSCENARIO` script can contain unlimited `Story` elements. These elements provide the user with an extra layer of organization. They are identified by their name, an attribute, and beside local parameter declarations, can contain an unlimited amount of `Act` elements, another division of the story for the user convenience.

```

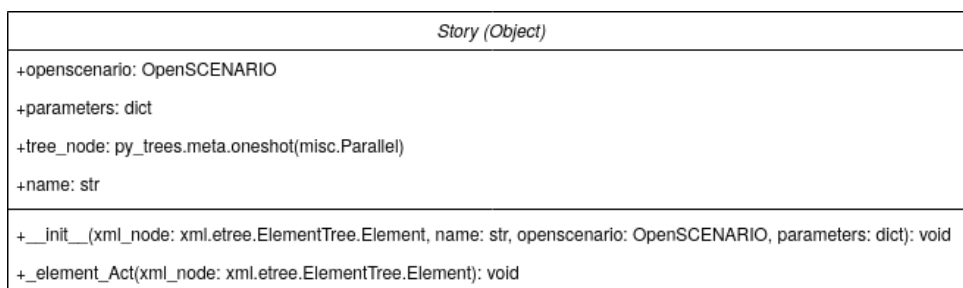
1 <xsd:complexType name="Story">
2 <xsd:sequence>
3 <xsd:element name="ParameterDeclarations" type="ParameterDeclarations"
  minOccurs="0"/>
4 <xsd:element name="Act" type="Act" maxOccurs="unbounded"/>
5 </xsd:sequence>
6 <xsd:attribute name="name" type="String" use="required"/>
7 </xsd:complexType>

```

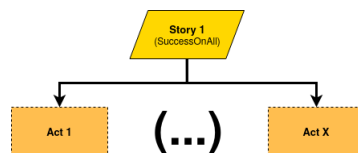
Listing 4.24: XSD definition of the *Story* element.[34]

4.5.3.1 Implementation

The *Story* class also follows the design pattern, with all the standard initialization arguments, figure 4.10. However it also receives an extra argument, "name". This argument is a string with the story name provided in the script.

Figure 4.10: *Story* UML class diagram.

During its initialization a simple behavior tree is created, figure 4.11, with one main node descended by all its **Act** nodes.

Figure 4.11: *Story* behavior tree.

The main node represents the *Story* element in question. It is parallel with success on all, allowing to execute all its children acts simultaneously and, also, only succeeding when all acts have succeeded.

4.5.4 Act

An *Act* (listing 4.25) contained in the *Story* elements, is the first element to provide temporal and spatial context in the scenario. This is accomplished by its triggers, both the *StartTrigger* and *StopTrigger*. This element is identified by its name, provided as an attribute, and it is a container for unlimited *ManeuverGroup* elements.

```

1 <xsd:complexType name="Act">
2   <xsd:sequence>
3     <xsd:element name="ManeuverGroup" type="ManeuverGroup"
4       maxOccurs="unbounded"/>
5     <xsd:element name="StartTrigger" type="Trigger"/>
6     <xsd:element name="StopTrigger" type="Trigger" minOccurs="0"/>
7   </xsd:sequence>
8   <xsd:attribute name="name" type="String" use="required"/>
9 </xsd:complexType>

```

Listing 4.25: XSD definition of the *Act* element.[34]

The *StartTrigger* element is required within an *Act* and indicates when its execution should initiate. As a rule, an *Act* is in standby until the *StartTrigger* is triggered.

On the other hand, the *StopTrigger* is not required. This trigger indicates when the execution of the *Act* element should terminate. When it is triggered, the execution of all descendant elements of the *Act* element should immediately stop its execution, however this trigger is only active when the *Act* element is being executed, i.e. this trigger can only be triggered if the *Act* was already triggered by the *StartTrigger*.

4.5.4.1 Implementation

Identical to the *Story* class, the *Act* class, (figure 4.12) receives the same standard parameters as the design pattern indicates. Its three member functions allow to create the necessary children nodes to fill the behavior tree as intended.

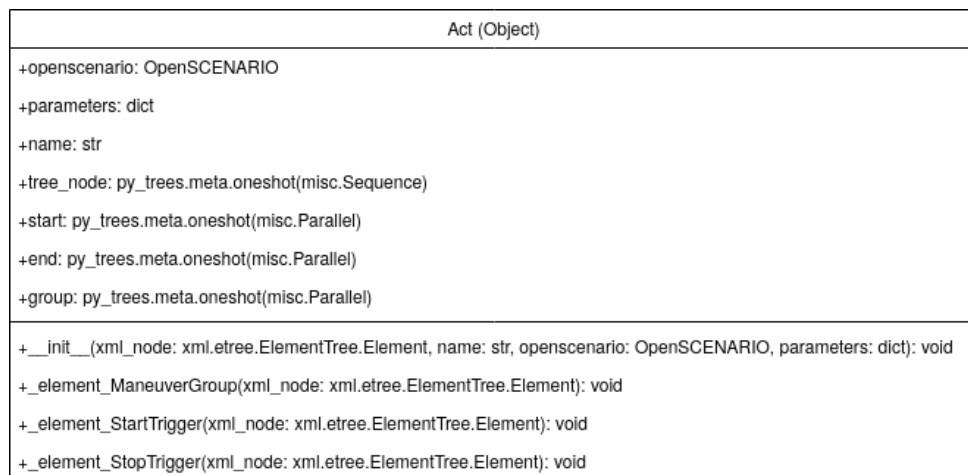


Figure 4.12: Act UML class diagram.

However, this class initialization is a bit more complex than the previous. It generates a bigger behavior tree with five nodes, and many children, for three of those five nodes, can be created with its member functions. The behavior tree generated in the initialization method is presented in the figure 4.13.

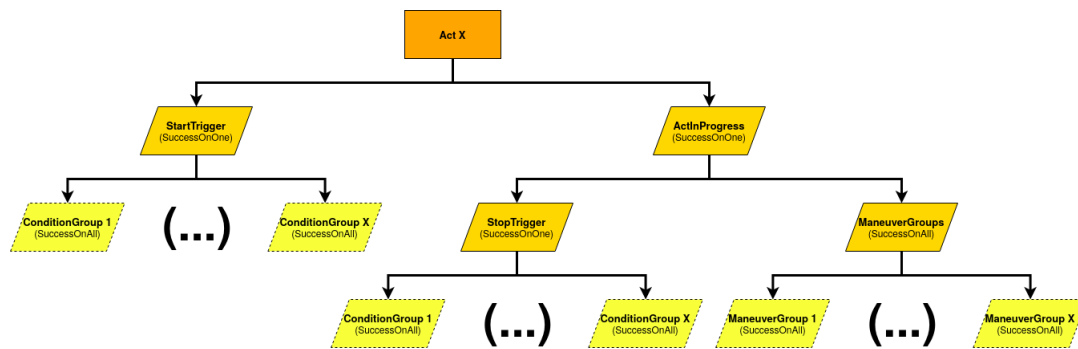


Figure 4.13: Act behavior tree.

In order to simulate the *Act* element in standby mode waiting for the *StartTrigger* to be triggered, the **Act** node is a sequence with the **StartTrigger** node as the first child. This ensures that, when the scenario is executed, only the **StartTrigger** node is executed and content of the act is only executed when this node succeeds. This node is a parallel with success on one as it is common for the trigger nodes.

The next child, after the **StartTrigger** node is the **ActInProgress** node, which, contains the **StopTrigger** and **ManeuverGroups** nodes as its descendants. The **ActInProgress** is a parallel node in order to allow the execution of the **StopTrigger**, only in parallel with the execution of the **ManeuverGroups** node. It is parallel with success on one allowing for either the success of the **StopTrigger** node or the finish of the execution of the **ManeuverGroups** node to end the execution of the **Act**.

Finally, the **ManeuverGroups** node contains all the *ManeuverGroup* elements in the *Act*. This node is parallel with success on all in order to execute all the **ManeuverGroup** node simultaneously and only finish once all these nodes are completed.

4.5.5 ManeuverGroup

Now that the *Act* already provided a temporal and spacial timeline, the *ManeuverGroup* element (listing 4.26) allows to identify the actors, or entities, that will be performing or affected by these maneuvers described inside these elements.

The *ManeuverGroup* element is constituted by two attributes, a string that identifies it by its name, and an integer denominated "maximumExecutionCount".

This integer indicates that the *ManeuverGroup* can be executed more than once and how many times it can be executed. These executions are sequential and never in parallel, i.e. no two instances of the same *ManeuverGroup* can be executed at the same time, and it is not necessary to execute it the indicated number of times. In case the parent *Act* ends prematurely, by either its own *StopTrigger* or *Storyboard's StopTrigger*, the executions of this element will also stop even if the


```

1 <xsd:complexType name="ManeuverGroup">
2   <xsd:sequence>
3     <xsd:element name="Actors" type="Actors"/>
4     <xsd:element name="CatalogReference" type="CatalogReference" minOccurs="0"
5       maxOccurs="unbounded"/>
6     <xsd:element name="Maneuver" type="Maneuver" minOccurs="0"
7       maxOccurs="unbounded"/>
8   </xsd:sequence>
9   <xsd:attribute name="maximumExecutionCount" type="UnsignedInt" use="required"/>
10  <xsd:attribute name="name" type="String" use="required"/>
11 </xsd:complexType>

```

Listing 4.26: XSD definition of the *ManeuverGroup* element.[34]

execution count has not reached its maximum. Otherwise, once the execution count has reached its maximum, the element is complete and stops its execution.

The *Actors* element (listing 4.27) is required in the *ManeuverGroup* element and its main function is to identify the entities which will be present in the containing *Maneuver* elements.

```

1 <xsd:complexType name="Actors">
2   <xsd:sequence>
3     <xsd:element name="EntityRef" type="EntityRef" minOccurs="0"
4       maxOccurs="unbounded"/>
5   </xsd:sequence>
6   <xsd:attribute name="selectTriggeringEntities" type="Boolean" use="required"/>
7 </xsd:complexType>

```

Listing 4.27: XSD definition of the *Actors* element.[34]

The *Actors* element can contain a list of entities by providing a series of *EntityRef* elements with the name of the desired entities. However, the list of entities can also be provided by activating the "selectTriggeringEntities" attribute. This attribute, indicates that the entities, previously identified in the *TriggeringEntities* element of all conditions in the parent *Act's StartTrigger*, who enabled the trigger, are automatically selected for the maneuvers in this element.

The *ManeuverGroup* element main objective is to contain the *Maneuver* elements and provide them with all the necessary information, in this case the entities. It can contain unlimited *Maneuver* elements and these elements can be described directly in the script or insert as references from the *ManeuverCatalog*.

4.5.5.1 Implementation

The *ManeuverGroup* class (figure 4.14) follows the design pattern quite closely, receiving the usual arguments and containing a simple initialization.

First the actors are obtained, this is done with a loop to search for the actors in the Open-SCENARIO main class actors dictionary, and adding them to a list of CARLA actors. Then the behavior tree is constructed, present in figure 4.15.

The behavior tree is constituted by only the **ManeuverGroup** node. This node is parallel with success on all allowing its children to execute simultaneously and only succeed once all children

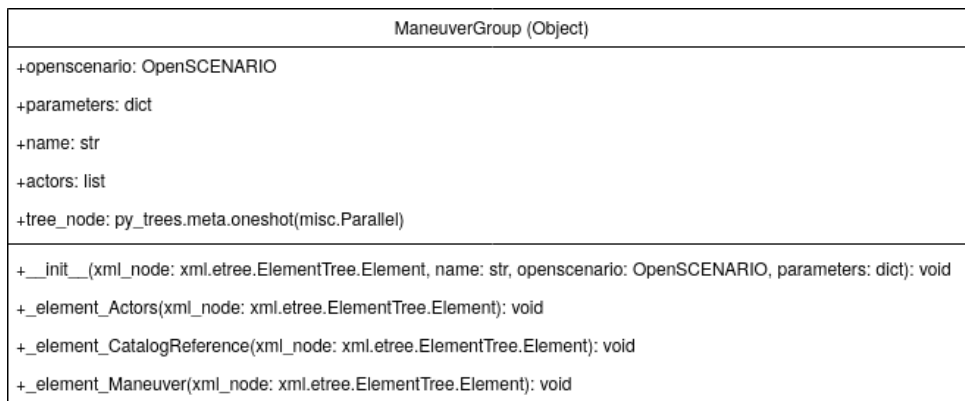


Figure 4.14: ManeuverGroup UML class diagram.

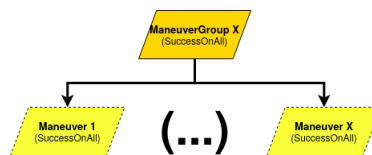


Figure 4.15: ManeuverGroup behavior tree.

have been completed. The **ManeuverGroup** node contains a list of **Maneuver** nodes as children. Every **Maneuver** node is created by the "_element_Maneuver" member function and, in case it is a catalog reference, the "_element_CatalogReference" inserts the catalog entry in the XML node as if it was a normal *Maneuver* element and executes the previous function to create the corresponding node.

4.5.6 Maneuver

The *Maneuver* element, described in the listing 4.28, is a container for *Event* elements. It is identified by its "name" attribute and contains local parameter declarations.

```

1 <xsd:complexType name="Maneuver">
2   <xsd:sequence>
3     <xsd:element name="ParameterDeclarations" type="ParameterDeclarations"
4       minOccurs="0"/>
5     <xsd:element name="Event" type="Event" maxOccurs="unbounded"/>
6   </xsd:sequence>
7   <xsd:attribute name="name" type="String" use="required"/>
8 </xsd:complexType>

```

Listing 4.28: XSD definition of the *Maneuver* element.[34]

4.5.6.1 Implementation

The *Maneuver* class, with the following UML class diagram in figure 4.16, follows the design pattern with the exception that it also receives a list of actors during its initialization. This list is

provided by its parent class, *ManeuverGroup*, to indicate which actors should the *Event* elements act upon, and it will be passed to its descendants, *Event* classes.

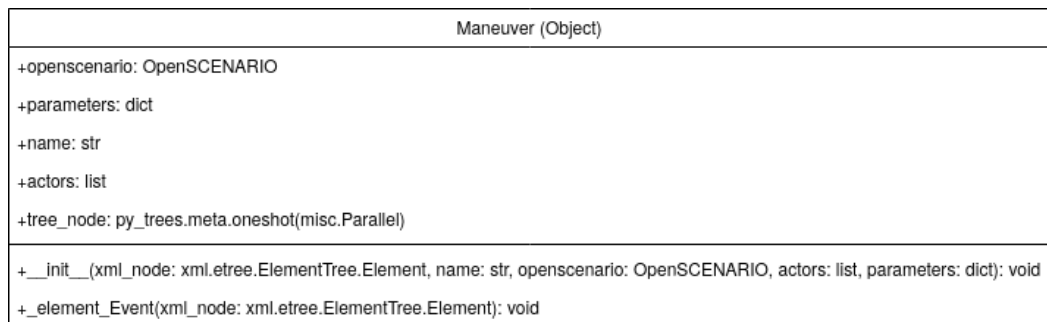


Figure 4.16: Maneuver UML class diagram.

In the initialization, this class, starts by fusing its parent's parameters dictionary with the new local parameter declarations, if any exist. Later a simple behavior tree is generated (figure 4.17) with only one main node.

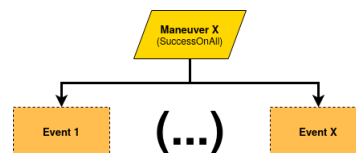


Figure 4.17: Maneuver behavior tree.

The **Maneuver** is a parallel node with success on all. This configuration allows to execute all its children, the **Event** nodes, simultaneously and only succeed when all the events have been executed.

4.5.7 Event

The *Event* element, in turn, is a container for the *Action* elements. However, it also plays a role on managing the scenarios timeline by providing start triggers. Similar to the *ManeuverGroup* element, exposed in the chapter 4.5.5, it contains a "maximumExecutionCount" attribute to allow multiple executions of the same event in sequence.

```

1 <xsd:complexType name="Event">
2   <xsd:sequence>
3     <xsd:element name="Action" type="Action" maxOccurs="unbounded"/>
4     <xsd:element name="StartTrigger" type="Trigger"/>
5   </xsd:sequence>
6   <xsd:attribute name="maximumExecutionCount" type="UnsignedInt"/>
7   <xsd:attribute name="name" type="String" use="required"/>
8   <xsd:attribute name="priority" type="Priority" use="required"/>
9 </xsd:complexType>
  
```

Listing 4.29: XSD definition of the *Event* element.[34]

This element is also identified by its "name" attribute, but, unlike the previously exposed elements, it also provides a "priority" attribute. This attribute, as explained in chapter 2.3.3, defines how the execution of the *Event* element affects and is affected by the execution of other *Event* elements present in the same *Maneuver*. The "priority" attribute content is defined by the *Priority* type shown in listing 4.30.

```

1 <xsd:simpleType name="Priority">
2   <xsd:union>
3     <xsd:simpleType>
4       <xsd:restriction base="xsd:string">
5         <xsd:enumeration value="overwrite"/>
6         <xsd:enumeration value="parallel"/>
7         <xsd:enumeration value="skip"/>
8       </xsd:restriction>
9     </xsd:simpleType>
10    <xsd:simpleType>
11      <xsd:restriction base="parameter"/>
12    </xsd:simpleType>
13  </xsd:union>
14 </xsd:simpleType>

```

Listing 4.30: XSD definition of the *Priority* type.[34]

The *Priority* type (listing 4.30) is an enumeration and describes three valid values for the "priority" attribute of the *Event* element. These values define how the execution of its *Event* element affects and is affected by the execution of other *Event* elements present in the same *Maneuver* element. For further explanation of each value see chapter 4.2.

The *Action* elements (listing 4.31) contained in this element, encapsulates the various types of action that exist in the OpenSCENARIO.

```

1 <xsd:complexType name="Action">
2   <xsd:choice>
3     <xsd:element name="GlobalAction" type="GlobalAction" minOccurs="0"/>
4     <xsd:element name="UserDefinedAction" type="UserDefinedAction" minOccurs="0"/>
5     <xsd:element name="PrivateAction" type="PrivateAction" minOccurs="0"/>
6   </xsd:choice>
7   <xsd:attribute name="name" type="String" use="required"/>
8 </xsd:complexType>

```

Listing 4.31: XSD definition of the *Action* element.[34]

With this element it is possible to either insert *GlobalAction*, *UserDefinedAction* and *PrivateAction* elements, which represent the global, user defined and private action types of the OpenSCENARIO, respectively. One *Action* element can contain only one of the three mentioned elements but an *Event* can contain unlimited *Action* elements.

4.5.7.1 Implementation

The *Event* class (figure 4.18) implement the behavior of both, *Event* and *Action*, elements. Apart from the usual design pattern initialization arguments it also receives a list of the actors and a string with the corresponding priority.

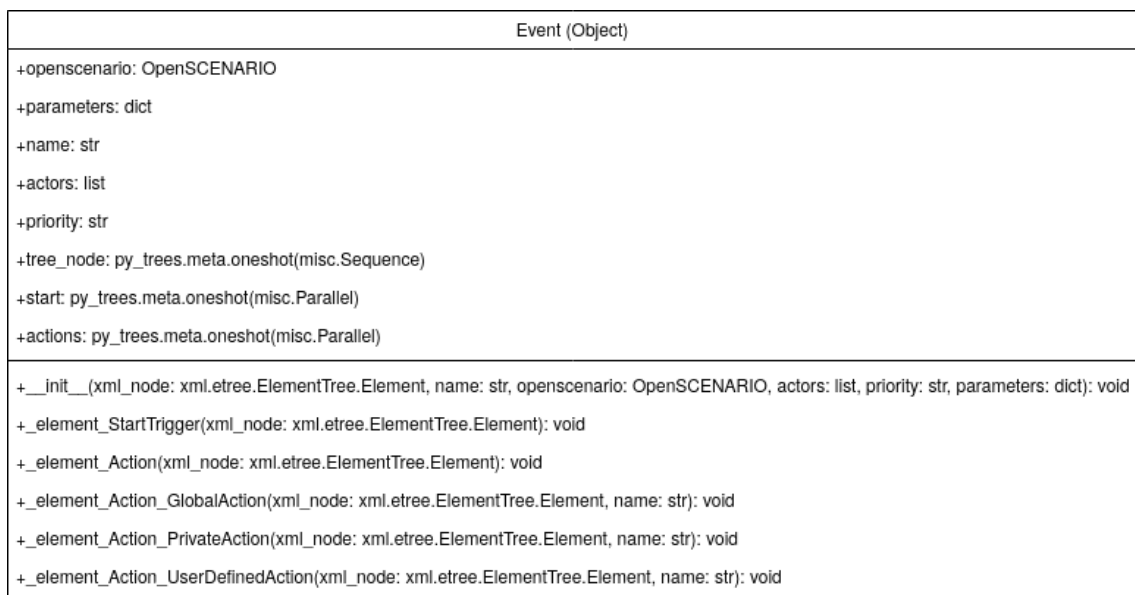


Figure 4.18: Event UML class diagram.

In the initialization, this class, constructs a dynamic behavior tree which size depends on the number of actors present in the list from its parent, as seen in figure 4.19.

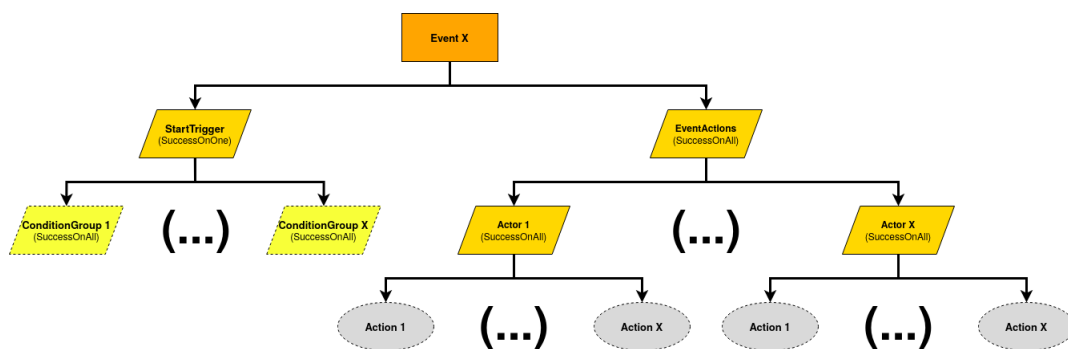


Figure 4.19: Event behavior tree.

The **Event** node is a sequence in order to provide a standby mode until the **StartTrigger** node succeeds. The followed node, **EventActions**, contains, as children, all the actions performed on all the actors. It has as many direct children as there are actors in the list, each node representing one actor. These nodes then have same actions as children, the actions described in the *Event* element, but acting upon the actor represented in the node. These actions are created by the corresponding member functions of the class.

4.5.8 Daytime Animation

OpenSCENARIO presupposes the existence of an animated day. While setting the weather parameters, an attribute, denominated "animation", allows to enable or disable the animated passage

of time. Although CARLA does provide extensive configurable weather parameters, animated passage of time is not natively implemented.

When a user defines the environment settings within a script, the CARLA weather parameters are updated to correspond to the desired settings and these parameters remain static until further changes by the script. However, when the environment settings enable the weather animation, the passage of time in the simulation should be accompanied by the correspondent visual cues, the sun's position, the movement of the clouds and the passage from day to night time and vice-versa.

In CARLA simulator the cloud's movement is not controllable, however it is possible to control the angle of the sun and transition between day and night. As shown in the figure 4.20, the influence of the sun's position not only affects the lighting angle of the map objects but also defines the night and day on the simulator. When the sun is above the map, on top of the middle dashed line, it is day time in the simulator, when the sun transverses the dashed line and is now under the map, it is night time as no light is capable of reaching the top of the map.

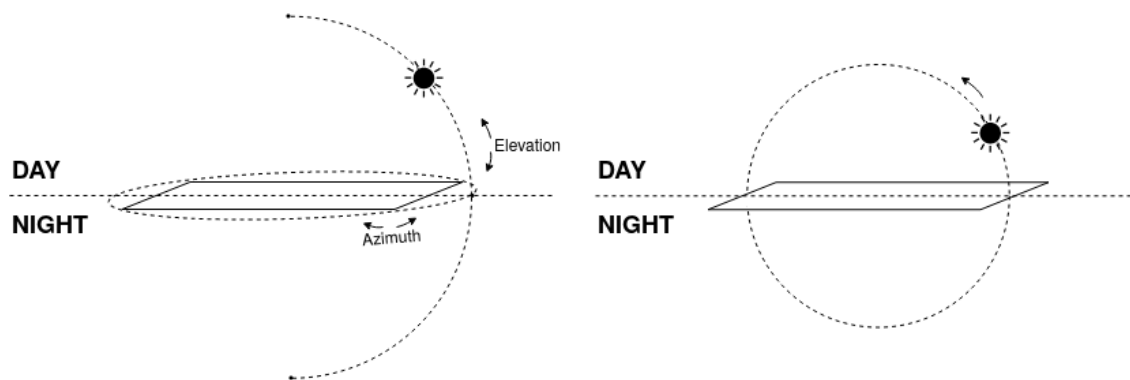


Figure 4.20: Real vs intended CARLA's sun movement.

On the left is a description of how the sun behaves in the CARLA simulator by manipulating the elevation and azimuth angle. On the right is a description of the intended movement of the sun to simulate passage of time.

The figure 4.20 shows two diagrams explaining the movement of the sun. Ideally the sun's movement would be as suggested by the diagram on the right, where only one angle, ranging from 0° to 360° , would be responsible for the sun revolving around a full day. However, in CARLA simulator, the sun is only capable of performing a full revolution by using both its angles, elevation and azimuth, since the elevation angles only ranges from -90° to 90° . Thus, to simulate a full revolution from the sun, and provide an adequate simulation of the passage of time, one would need to increment the sun's elevation angle until 90° were reached, add, at that moment, 180° to the azimuth angle and start decreasing the elevation angle until it reaches -90° , followed by a subtraction of 180° to the azimuth angle and repeat.

4.5.8.1 Implementation

To control the weather dynamically while the simulation is running, a constant update on the weather parameters is required. To implement this ability in a behavior tree it is necessary to

create an action independent of other environment setting actions, and only dependent on the execution of the scenario. The solution was to develop a permanent action that is always executed throughout the whole script in the root of the tree, as shown by the behavior tree in the figure 4.21.

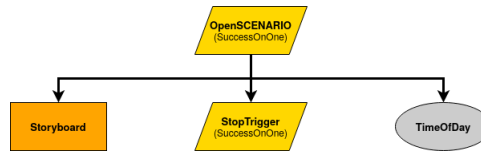


Figure 4.21: Behavior tree with the TimeOfDay action.

The **TimeOfDay** action, being a child of the **OpenSCENARIO** node is ensured to be execute the whole scenario and, by always return its status as "RUNNING", it allows a continuous control of the CARLA's weather parameters. This action's main objective is to regularly update the sun's angle, if the animation setting is enabled. In the figure 4.22, the diagram of this class shows how the behavior is implemented. In the initialization method it receives it's name, used to create the action node with the correct name, and an instance of the simulator's world, required to control the weather parameters.

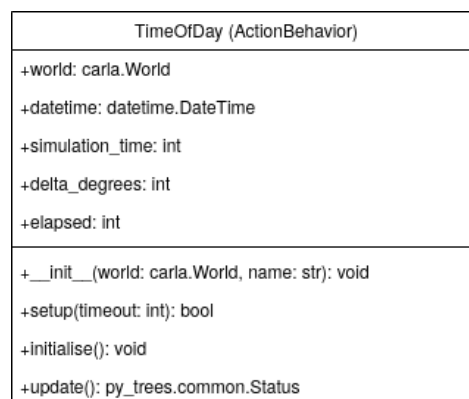


Figure 4.22: TimeOfDay Action - UML class diagram.

Two variables in tree's blackboard are updated by this action. The first, "TimeOfDay", contains the simulator's time and date. The second, "Animation", contains the current state of the animation setting, enabled or disabled.

In the setup method, this action, first updates the "TimeOfDay" blackboard entry with the real world time and date, this occurs the moment the script is executed. Although this value can be changed in the script, in case the user does not specify any date and time for the script, the default value is the current date and time. The "Animation" value is also set as false, remaining disabled until the script enables it.

During its execution, in case the "Animation" variable remains disabled, all this action does it to incrementally update the current time with the time passed, in the simulator. However, once the "Animation" is enabled, this action is responsible for incrementing the sun's angle according to the passage of time in the simulator.

Since the sun's angle can perform a 360° degree turn and the resolution of the angle of the sun in the simulator is limited to 1° it is important to increment the sun's angle every 240 seconds, as demonstrated by the equation 4.1.

$$\text{Sun's rate} = \frac{\text{FullDay}}{\text{FullTurn}} = \frac{24 * 60 * 60}{360} = 240 \text{ seconds/degree} \quad (4.1)$$

4.5.9 Naming

When writing an OpenSCENARIO script it is important to pay attention to the name of each element. Throughout the scenario description elements can be reference and, as such, it is essential to ensure that all elements can be provided with a unique name. The user needs to make sure all element names are unique in the same scope, i.e. the parent element. For example, in the same *Story* all elements should have unique names, however, between two *Story* elements their children can share the same name.

When referencing an element the name of the reference should make sure only one element corresponds to the search. In case the element name is not enough to correspond to only one element, prefixes should be added until it is unique. These prefixes are references to their parent elements and should be separated by an "::" symbol. As a result the "::" symbol cannot be used in names.[1]

As an example consider the listing 4.32 where two *Story* elements are defined with one *Act* element each, that share the name. In order to reference the first *Act* element using only its name is not enough because it is not globally unique. To make sure the reference is unique one should add the *Story* element name as a prefix. The final reference would be "Story01::Act1".

```

1 <.../>
2 <Storyboard>
3 <Story name="Story01">
4 <Act name="Act1">
5 <.../>
6 </Act>
7 </Story>
8 <Story name="Story02">
9 <Act name="Act1">
10 <.../>
11 </Act>
12 </Story>
13 <.../>
14 </Storyboard>

```

Listing 4.32: Example to correctly reference an element.

4.5.9.1 Implementation

In order for to allow conditions, which are leaf nodes in the behavior tree, to access the execution status of all intermediate and leaf nodes, one entry per node is added to the shared blackboard as soon as the node starts its execution and updated with their current status every cycle. This variable

entry is managed by the `ActionBehavior` base class, figure 4.24 from chapter 4.6, and `Parallel` and `Sequence` classes, figure 4.23. Further explanation about the `ActionBehavior` class is provided in the chapter 4.6.

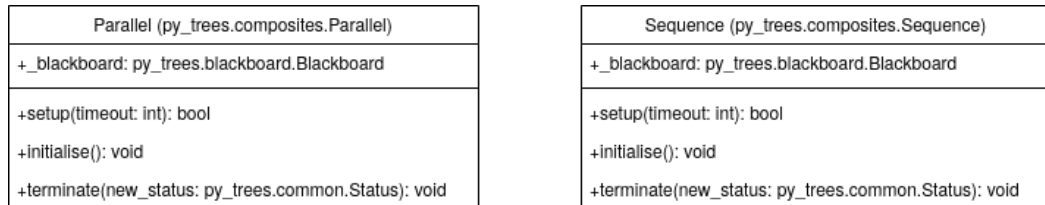


Figure 4.23: Parallel and Sequence UML class diagrams.

To make sure all node names are unique the full path of the node in the behavior tree, from the root element, is always added. Additionally, to provide more information and facilitate the reading, all elements are referenced by their type and name, in case it exists. In the example in listing 4.32, the first *Act* element would have the name "OpenSCENARIO::Storyboard::Stories::Story(Story01)::Act(Act1)".

There are some exceptions when this pattern is not exactly as described: when elements do not possess any identification and when actions or conditions act upon an entity. In the case of elements that do not contain any identification attributes, e.g. actions contained in the *Init* element, a sequential number, according to the order presented in the script, is provided as an identifier. For example, in an *Init* element where the first action was an *Environment* element, it would have the name "(...)::EnvironmentAction(0)". Another special case is when leaf nodes, actions or conditions, are related to an entity and, in this case, a suffix is added containing the name of the entity, "(...)::Entity(Actor01)".

4.6 Actions

The leaf nodes of the behavior tree are composed by actions and conditions. These nodes are derived from the `Behavior` class from the "py_trees" python module. The scenario runner plugin contains an implementation of the class `AtomicBehavior` that enables logging for all the important events from each action.

In order to take advantage of this feature and implement some basic OpenSCENARIO features, all actions are derived from two base classes that are, in turn, derived from the `AtomicBehavior` class. The `ActionBehavior` class (figure 4.24) is responsible for setting and updating a variable in the behavior tree's common blackboard, with the name of the corresponding action and its execution state. This feature is required in order to permit some conditions and actions to access the state of other actions.

For other actions, with more complex movements and decisions, another base class is provided. The `StateMachineBehavior` class (figure 4.24), derived from the `ActionBehavior` class, adds the capability to easily construct behaviors based on state machines.

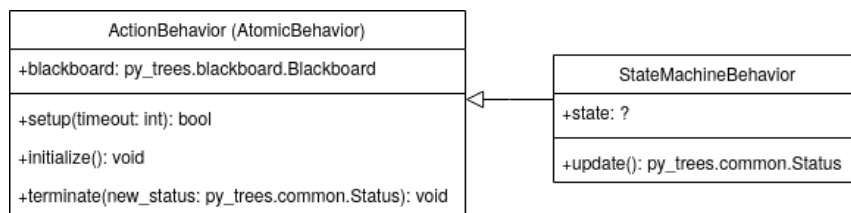


Figure 4.24: Actions Base classes - UML class diagram.

The usage of the StateMachineBehavior class is quite simple, but it defers from the usage of the library's Behavior class. An example of a state machine and its implementation, by deriving from the StateMachineBehavior class, is presented in the figure 4.25.

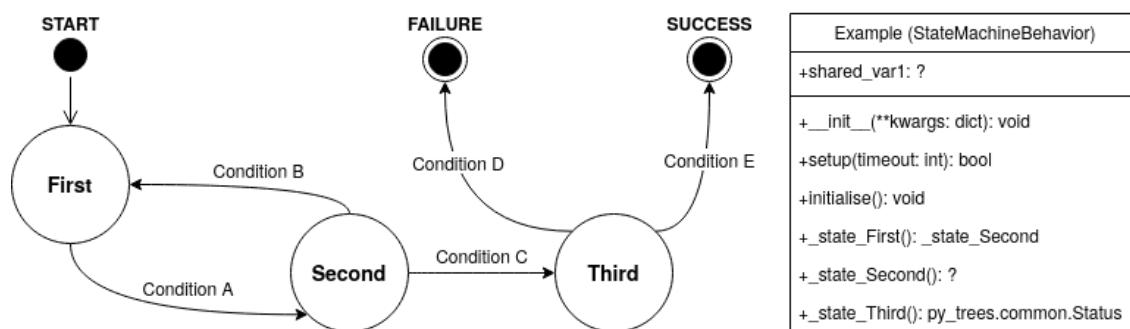


Figure 4.25: Example of a state machine implementation.

The derived class ("Example" class in figure 4.25) should contain one member function for each state from the state machine. As it is easy to see, these member functions are denominated with the state's name prefixed with the term "_state_".

The transitions to another state or to finish the behavior, by either success or failure, are evaluated within each member function and the result should be returned. As such, each state's member function should return either another state, which is a pointer to another state's member function, or the final result of the behavior, e.g. the "_state_First" returns the "_state_Second" to transition to the second state while the "_state_Third" return a `py_trees.common.Status` to terminate the execution of the state machine and the behavior, consequently. In case a state needs to execute for more than one cycle, it should return a pointer to itself.

Finally, the state's member functions should not accept any argument and, in case it is necessary to transmit information from one state to another, member variables should be used to contain all the global information.

4.6.1 Global Actions

One of the main groups of actions available in the OpenSCENARIO are the global actions (listing 4.33). This group contains all the actions that do not explicitly target an entity's state.[1] Meaning that, although most actions are not associated with any specific entity, even the actions that reference an entity do not alter its state directly.

```

1 <xsd:complexType name="GlobalAction">
2 <xsd:choice>
3 <xsd:element name="EnvironmentAction" type="EnvironmentAction"
4   minOccurs="0"/>
5 <xsd:element name="EntityAction" type="EntityAction" minOccurs="0"/>
6 <xsd:element name="ParameterAction" type="ParameterAction" minOccurs="0"/>
7 <xsd:element name="InfrastructureAction" type="InfrastructureAction"
8   minOccurs="0"/>
9 <xsd:element name="TrafficAction" type="TrafficAction" minOccurs="0"/>
10 </xsd:choice>
11 </xsd:complexType>

```

Listing 4.33: XSD definition of the *GlobalAction* element.[34]

The global actions are divided into five categories: Environment - allows to set the weather state, road condition parameters, current time and enable the animation of the passage of time; Entity - allows to insert or remove an instance of an entity from the scenario; Parameter - allows to set, add or multiply a global parameter value; Infrastructure - allows to modify and control traffic signal states and traffic signal controllers phase; Traffic - allows to create several types of background traffic.[1]

4.6.1.1 Implementation

The class present in figure 4.26 implements the behavior expected from the *GlobalAction* element.

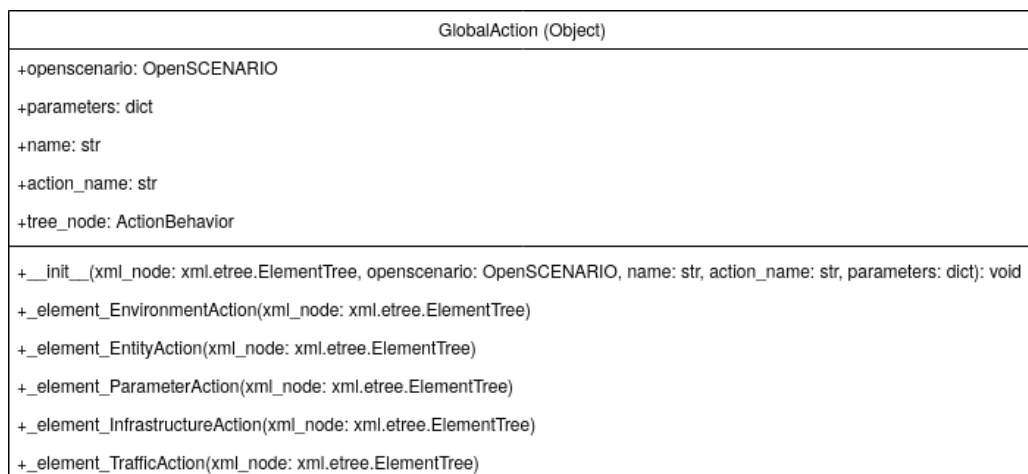


Figure 4.26: Global actions UML class diagram.

This class does not create any behavior tree node or performs any other complex operation. Its main objective is to correctly parse the *GlobalAction* element and determine what type of action it should create. This action will be created by the correspondent classes and its behavior tree node will be stored in this class' "tree_node" member variable.

4.6.2 Private Actions

Another main group of OpenSCENARIO's actions are the private actions (listing 4.34). These actions have to assigned to an entity and allow to change its motion, position and visibility. Both, the longitudinal and lateral movements of an entity can be controlled by these types of actions.[1]

```

1 <xsd:complexType name="PrivateAction">
2   <xsd:choice>
3     <xsd:element name="LongitudinalAction" type="LongitudinalAction"
4       minOccurs="0"/>
5     <xsd:element name="LateralAction" type="LateralAction" minOccurs="0"/>
6     <xsd:element name="VisibilityAction" type="VisibilityAction" minOccurs="0"/>
7     <xsd:element name="SynchronizeAction" type="SynchronizeAction"
8       minOccurs="0"/>
9     <xsd:element name="ActivateControllerAction" type="ActivateControllerAction"
10      minOccurs="0"/>
11    <xsd:element name="ControllerAction" type="ControllerAction" minOccurs="0"/>
12    <xsd:element name="TeleportAction" type="TeleportAction" minOccurs="0"/>
13    <xsd:element name="RoutingAction" type="RoutingAction" minOccurs="0"/>
14  </xsd:choice>
15 </xsd:complexType>

```

Listing 4.34: XSD definition of the *PrivateAction* element.[34]

The private actions are divided into more categories than the previously described global actions: Longitudinal - allows to control the speed and acceleration of an entity; Lateral - allows to control the heading of an entity; Visibility - intends to enable the visibility of an entity to certain types of sensors; Synchronize - allows to coordinate the longitudinal movement of an entity in order to encounter another at a desired location; ActivateController - enables or disables an entity controller; Controller - assigns a new controller to an entity while providing the ability to override certain aspects; Teleport - allows to move the entity instantaneously to a desired position; Routing - provides the entity with a path to follow, either describing only lateral movements or describing both lateral and longitudinal movements.[1]

Private All private actions have to assigned to an entity. When actions are contained in the *Storyboard*, this is not a problem because the entities are assigned in the *ManeuverGroup* elements. However, when actions are contained in the *Init* element no entity is assigned so it is necessary to encapsulate the private actions in a *Private* element (listing 4.23) that has the main objective to assign an entity to all the actions contained in it.

In the *Init* several *Private* elements can be contained in order to assign several entities to private actions. And, the *Private* element can contain several private actions and all will be assigned the same entity.

4.6.2.1 Implementation

Similarly to the *GlobalAction* class (figure 4.26), the *PrivateAction* class (figure 4.27) also does not create any behavior tree node or performs any complex operations. The objective of this class

is, only, to correctly parse the XML content and determine what category of private action is necessary to create. Its member functions, each responsible for one category, should instantiate the correct class that creates the desired behavior and store the behavior action in the "tree_node" member variable.

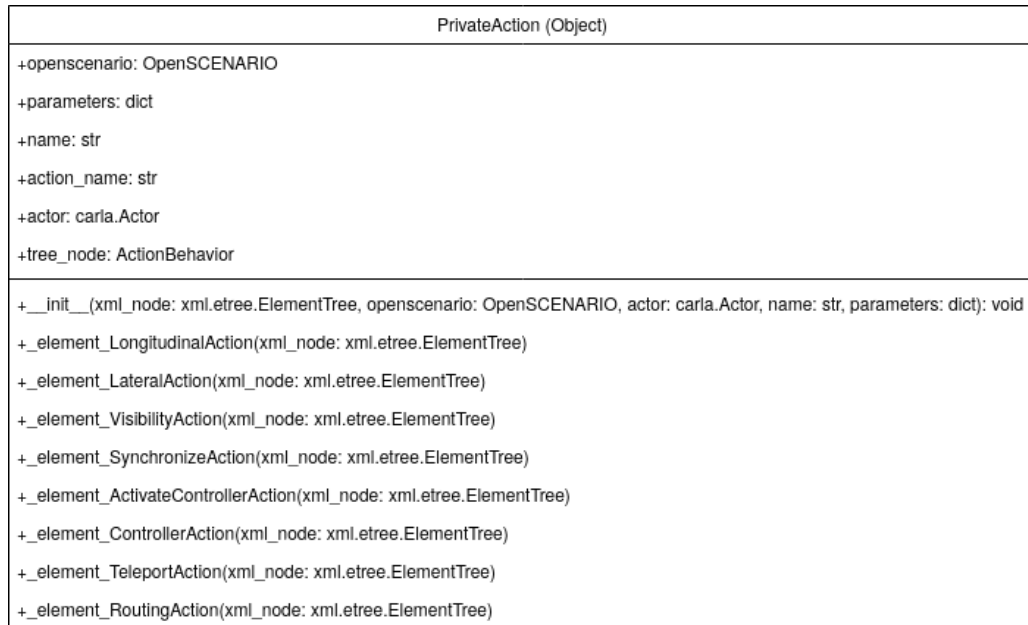


Figure 4.27: PrivateAction UML class diagram.

Private On the other hand, the Private class (figure 4.28), since it is responsible to assign an entity to a private action, retrieves the actor referenced in the script and assigns this actor to all the actions present inside the *Private* element.

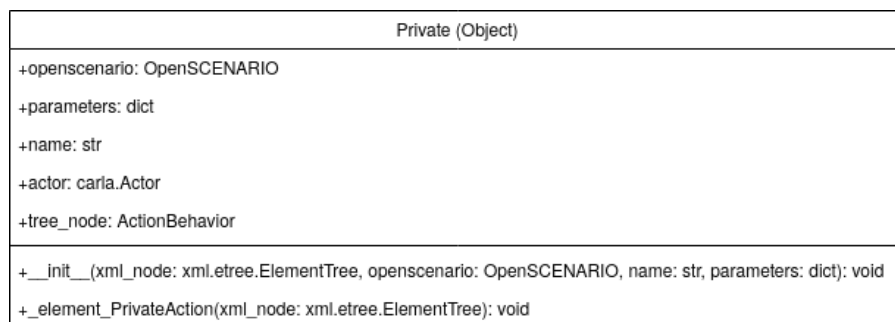


Figure 4.28: Private UML class diagram.

Since many private actions can be present in a *Private* element, the Private class generates the behavior tree node presented in figure 4.29 where all the private actions will be contained.

Since all these actions are assigned to the same entity and it is not a good idea to perform multiple actions on the same entity at the same time, this node is of sequence type to allow the execution of one action at a time.

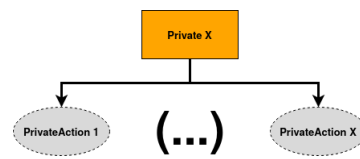


Figure 4.29: Private behavior tree.

4.6.3 Environment Action

As previously explained an environment action (listing 4.35) allows to change the weather characteristics, the road condition properties and time passage in the simulation.

```

1 <xsd:complexType name="EnvironmentAction">
2   <xsd:choice>
3     <xsd:element name="Environment" type="Environment" minOccurs="0"/>
4     <xsd:element name="CatalogReference" type="CatalogReference" minOccurs="0"/>
5   </xsd:choice>
6 </xsd:complexType>
  
```

Listing 4.35: XSD definition of the *EnvironmentAction* element.[34]

An *EnvironmentAction* element is a container for the *Environment* element, where all the previously mentioned settings are defined, or a catalog reference to an *Environment* element.

An *Environment* element (listing 4.36), besides the *ParameterDeclarations* element common to all catalog elements, contains three types of elements that define different characteristics of the simulation environment: *TimeOfDay* element, *Weather* element and *RoadCondition* element.

```

1 <xsd:complexType name="Environment">
2   <xsd:all>
3     <xsd:element name="ParameterDeclarations" type="ParameterDeclarations"
4       minOccurs="0"/>
5     <xsd:element name="TimeOfDay" type="TimeOfDay"/>
6     <xsd:element name="Weather" type="Weather"/>
7     <xsd:element name="RoadCondition" type="RoadCondition"/>
8   </xsd:all>
9   <xsd:attribute name="name" type="String" use="required"/>
10 </xsd:complexType>
  
```

Listing 4.36: XSD definition of the *Environment* element.[34]

TimeOfDay This element (listing 4.37) allows to set the simulated date and time of the simulation, and this value is then incremented according to the speed of time in the simulation. When the environment action is used inside the *Init* element, it defines the initial time and date of the simulation. Alternatively the environment action can be used in the *Story* elements to reset the date and time of the simulation to a specific moment.

The attribute "animation" enables the simulation of visual cues to infer the passage of time, see chapter 4.5.8.

```

1 <xsd:complexType name="TimeOfDay">
2   <xsd:attribute name="animation" type="Boolean" use="required"/>
3   <xsd:attribute name="dateTime" type="DateTime" use="required"/>
4 </xsd:complexType>

```

Listing 4.37: XSD definition of the *TimeOfDay* element.[34]

Weather In an OpenSCENARIO script, the weather (listing 4.38) is highly customizable with a variety of parameters that allow to alter different elements, such as clouds, sun, fog and precipitation.

```

1 <xsd:complexType name="Weather">
2   <xsd:all>
3     <xsd:element name="Sun" type="Sun"/>
4     <xsd:element name="Fog" type="Fog"/>
5     <xsd:element name="Precipitation" type="Precipitation"/>
6   </xsd:all>
7   <xsd:attribute name="cloudState" type="CloudState" use="required"/>
8 </xsd:complexType>

```

Listing 4.38: XSD definition of the *Weather* element.[34]

The "cloudState" attribute, from the *Weather* element, as shown in listing 4.39, allows to define the level of cloudiness in the sky using some common descriptions: free, cloudy, overcast and rainy. Optionally, this attribute can also instruct to disable the simulation of the sky through the keyword "skyOff".

```

1 <xsd:simpleType name="CloudState">
2   <xsd:union>
3     <xsd:simpleType>
4       <xsd:restriction base="xsd:string">
5         <xsd:enumeration value="cloudy"/>
6         <xsd:enumeration value="free"/>
7         <xsd:enumeration value="overcast"/>
8         <xsd:enumeration value="rainy"/>
9         <xsd:enumeration value="skyOff"/>
10      </xsd:restriction>
11    </xsd:simpleType>
12    <xsd:simpleType>
13      <xsd:restriction base="parameter"/>
14    </xsd:simpleType>
15  </xsd:union>
16 </xsd:simpleType>

```

Listing 4.39: XSD definition of the *cloudState* type.[34]

The *Weather* element also contains three elements to further define its meteorological characteristics. The *Sun* element (listing 4.40) allows to define the sun's azimuth, elevation and intensity, with the correspondent attributes.

The *Fog* element (listing 4.41) can characterize the presence of fog in the simulator by describing the affected area with a bounding box and the visual range, in meters, that an entity would be have when present inside the defined bounding box.

```

1 <xsd:complexType name="Sun">
2   <xsd:attribute name="azimuth" type="Double" use="required"/>
3   <xsd:attribute name="elevation" type="Double" use="required"/>
4   <xsd:attribute name="intensity" type="Double" use="required"/>
5 </xsd:complexType>

```

Listing 4.40: XSD definition of the *Sun* element.[34]

```

1 <xsd:complexType name="Fog">
2   <xsd:all>
3     <xsd:element name="BoundingBox" type="BoundingBox" minOccurs="0"/>
4   </xsd:all>
5   <xsd:attribute name="visualRange" type="Double" use="required"/>
6 </xsd:complexType>

```

Listing 4.41: XSD definition of the *Fog* element.[34]

At last, the *Precipitation* element (listing 4.42) describes the behavior of rain in the simulation by defining its intensity, in percentage value, and a precipitation type: dry, rain or snow.[34]

```

1 <xsd:complexType name="Precipitation">
2   <xsd:attribute name="intensity" type="Double" use="required"/>
3   <xsd:attribute name="precipitationType" type="PrecipitationType"
4     use="required"/>
5 </xsd:complexType>

```

Listing 4.42: XSD definition of the *Precipitation* element.[34]

RoadCondition This element (listing 4.43) not only allows to define the friction values of the road, using its "frictionScaleFactor" attribute, but also enables the user to define a series of personalized properties that would help better define the road in the simulation.

```

1 <xsd:complexType name="RoadCondition">
2   <xsd:sequence>
3     <xsd:element name="Properties" type="Properties" minOccurs="0"/>
4   </xsd:sequence>
5   <xsd:attribute name="frictionScaleFactor" type="Double" use="required"/>
6 </xsd:complexType>

```

Listing 4.43: XSD definition of the *RoadCondition* element.[34]

4.6.3.1 Weather Parameters in CARLA

The CARLA simulator also allows the configuration of multiple weather characteristics that are detailed in the table 4.2. The CARLA's Python API contains several predefined weather values that facilitate the configuration of specific meteorological conditions: ClearNoon, CloudyNoon, WetNoon, WetCloudyNoon, SoftRainNoon, MidRainyNoon, HardRainNoon, ClearSunset, CloudySunset, WetSunset, WetCloudySunset, SoftRainSunset, MidRainSunset and HardRainSunset. Although the simulator allows the manipulation of the friction of the road, this is only possible using

friction triggers, which means the entities would only be affected to this new road friction value if they happened to pass by the location of the friction triggers. CARLA allows to define the speed of the simulation but it has no definition of the current date and time or even visual passage of time, as discussed in chapter 4.5.8.

Table 4.2: CARLA’s weather parameters.[2]

Parameter	Values	Description
cloudiness	[0,100]	0 - clear sky & 100 - covered in clouds.
precipitation	[0,100]	0 - no rain & 100 - heavy rain.
precipitation_deposits	[0,100]	Creates puddles in the road using static noise, generating puddles always in the same places. 0 - no deposits & 100 - road completely capped with water.
wind_intensity	[0,100]	The wind in the simulator can only be detected by the RGB camera sensors through rain direction and movement of tree leaves. 0 - no wind & 100 - very strong wind
sun_azimuth_angle	[0,360]	0 is an origin point determined by UE4.
sun_altitude_angle	[-90,90]	-90 - midnight & 90 - midday.
fog_density	[0,100]	This property affects, mainly, the fog thickness and can only be detected by the RGB camera sensors. 0 - no fog & 100 - cannot see through fog.
fog_distance	[0, ∞]	Indicates the distance, in meters, from the center point of the sensor, until the fog starts.
wetness	[0,100]	Defines the wetness look of the road, also only affects the RGB camera sensors. 0 - dry road & 100 - completely wet road.

4.6.3.2 Implementation

In order to implement this action, the EnvironmentAction class (figure 4.30) is responsible for: creating the leaf node for the behavior tree, mapping OpenSCENARIO’s weather settings with CARLA’s weather parameters and update the simulation date, time and animation settings.

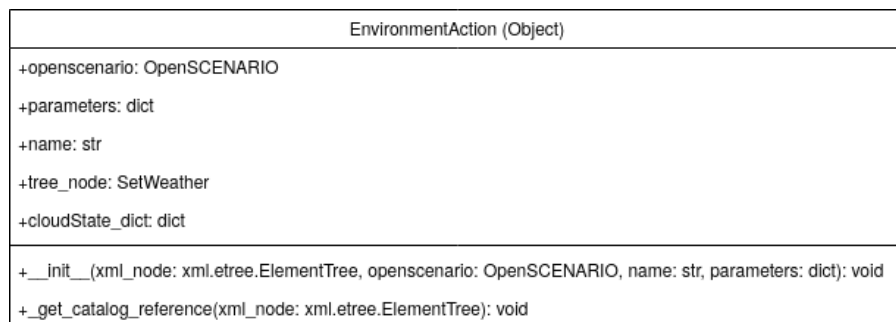


Figure 4.30: Environment action - UML class diagram.

As it can be inferred from the content in this section, although both, CARLA simulator and OpenSCENARIO, instances allow to characterize the desired weather conditions quite thoroughly, the specific parameters differ a little bit making the conversion not straight forward. The table 4.3 presents the connection made to link all the CARLA weather parameters to the properties OpenSCENARIO is able to provide. All CARLA's parameters were met but some adjustments in the OpenSCENARIO side was necessary.

Table 4.3: Conversion weather parameters between CARLA and OpenSCENARIO.

CARLA's weather parameters	OpenSCENARIO's weather parameters
cloudiness	Weather/cloudState
precipitation	Weather/Precipitation/intensity
precipitation_deposits	RoadCondition/Properties/precipitationDeposits
wind_intensity	RoadCondition/Properties/windIntensity
sun_azimuth_angle	Weather/Sun/azimuth
sun_altitude_angle	Weather/Sun/elevation
fog_density	RoadCondition/Properties/fogDensity
fog_distance	Weather/Fog/visualRange
wetness	RoadCondition/Properties/wetness

The attribute "cloudState" was used to determine the cloudiness level of the simulator. This attribute is not able to fully represent the simulator's cloudiness so a conversion of values was needed. The cloudiness presents float values from 0 to 100 while the "cloudState" attribute presents 5 string values. Another problem is that the simulator does not allow to disable the simulation of the sky making the "skyOff" keyword useless. The adopted solution was to associate a float value to the "cloudState" keywords, and the "skyOff" keyword associated with no clouds in the sky. In listing 4.44 is presented a Python dictionary that is used for the conversion between these two parameters.

```
1 cloudState_dict = {'skyOff': 0.0, 'free': 0.0, 'cloudy': 33.3,
2   'overcast': 66.6, 'rainy': 100.0}
```

Listing 4.44: Python dictionary to convert "cloudState" values into cloudiness float values.

The attribute "intensity" from the *Precipitation* element, inside the *Weather* element, is able to, proportionally, provide an equivalent value for the precipitation parameter, just by multiplying its value by 100.

The precipitation deposits, wind intensity, fog density and wetness cannot be provided by any of the OpenSCENARIO's weather parameters and they had to be implemented as a user-defined property in the *RoadCondition* element. The advantage is that its values share the same specification and do not need to be converted.

Although the fog density cannot be natively provided by an OpenSCENARIO attribute, the fog distance is equivalent to the "visualRange" attribute from the *Fog* element. Its values, also, do not need to be converted.

Finally, both the sun's azimuth and altitude had a corresponding OpenSCENARIO attributes, in the *Sun* element. Since the simulator's origin angle of the sun's azimuth is dependent of the UE4 implementation (table 4.2) and as a consequence it can vary from city to city, it was assumed that the 0° would always be pointing to north. This is identical to the OpenSCENARIO's "azimuth" attribute and thus it required no other conversion than from radians to degrees. However, the carla's sun's elevation ranges from -90 to 90 degrees while the "elevation" attribute from the *Sun* element ranges from $-pi$ to pi radians, i.e. CARLA's sun only performs half a circumference in the elevation direction while OpenSCENARIO's sun performs a full circumference. In order to convert the values from the OpenSCENARIO's sun's elevation to CARLA's sun's elevation it is needed to alter the CARLA's sun's azimuth, according to the equations 4.2 and 4.3. In the following equations: $C_azimuth$ and C_elev represent CARLA's azimuth and elevation, and $O_azimuth$ and O_elev represent OpenSCENARIO's azimuth and elevation, respectively.

$$C_azimuth = \begin{cases} O_azimuth * \frac{180}{\pi}, & \text{if } |O_elev| \leq 90 \\ \left(O_azimuth * \frac{180}{\pi}\right) + 180, & \text{otherwise} \end{cases} \quad (4.2)$$

$$C_elev = \begin{cases} -180 - \left(O_elev * \frac{180}{\pi}\right), & \text{if } O_elev < -90 \\ O_elev * \frac{180}{\pi}, & \text{if } -90 \leq O_elev \leq 90 \\ 180 - \left(O_elev * \frac{180}{\pi}\right), & \text{if } O_elev > 90 \end{cases} \quad (4.3)$$

Action node After parsing and converting all the values, the EnvironmentAction class creates a SetWeather action behavior (figure 4.31) that will be a leaf node in the behavior tree.

This action receives all the converted values from the EnvironmentAction class and is responsible for applying the values to the simulator. In case of the date, time and animation settings, it updates the variables "dateTime" and "animation" that should already be present in the behavior tree's shared blackboard.

This action succeeds once it verifies that the simulator's weather parameters are the same as the received settings.

4.6.4 Entity Action

The *EntityAction* element (listing 4.45) contains two global actions that are designed to add or remove an entity from the scenario.[34]

The *EntityAction* element is the only group of global actions that act upon entities. As such, its "entityRef" attribute allows to select the entity that the selected action will be performed on. In

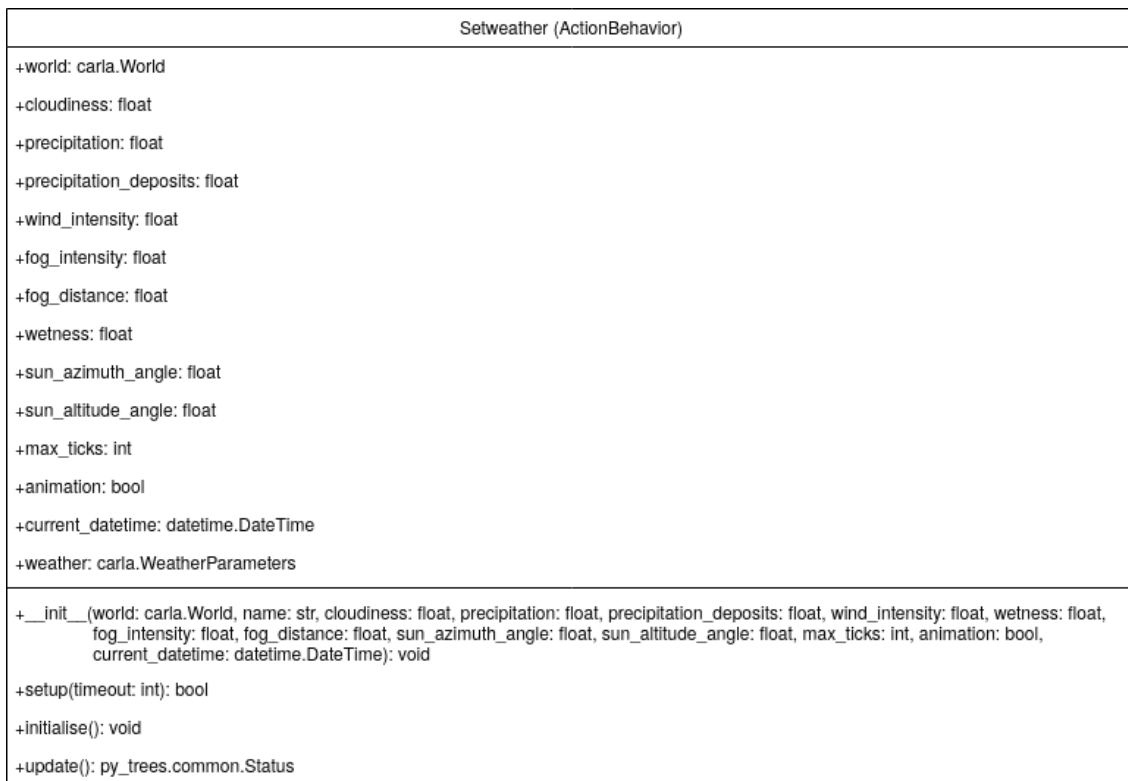


Figure 4.31: SetWeather action behavior - UML class diagram.

```

1 <xsd:complexType name="EntityAction">
2   <xsd:choice>
3     <xsd:element name="AddEntityAction" type="AddEntityAction" minOccurs="0"/>
4     <xsd:element name="DeleteEntityAction" type="DeleteEntityAction"
5       minOccurs="0"/>
6   </xsd:choice>
7   <xsd:attribute name="entityRef" type="String" use="required"/>
8 </xsd:complexType>

```

Listing 4.45: XSD definition of the *EntityAction* element.[34]

order to add an entity to a scenario, the *AddEntityAction* element (listing 4.46) allows to select a position for the entity to assume in the scenario.

```

1 <xsd:complexType name="AddEntityAction">
2   <xsd:all>
3     <xsd:element name="Position" type="Position"/>
4   </xsd:all>
5 </xsd:complexType>

```

Listing 4.46: XSD definition of the *AddEntityAction* element.[34]

When the entity is already on the scenario, the *DeleteEntityAction* element allows to remove it. This element does not have any attributes or descendant elements.

4.6.4.1 Implementation

The *EntityAction* element is implemented by the `EntityAction` class (figure 4.32) that is responsible for identifying the referenced entity and, in case of the *AddEntityAction* element, convert the position data into the correspondent CARLA's coordinates.

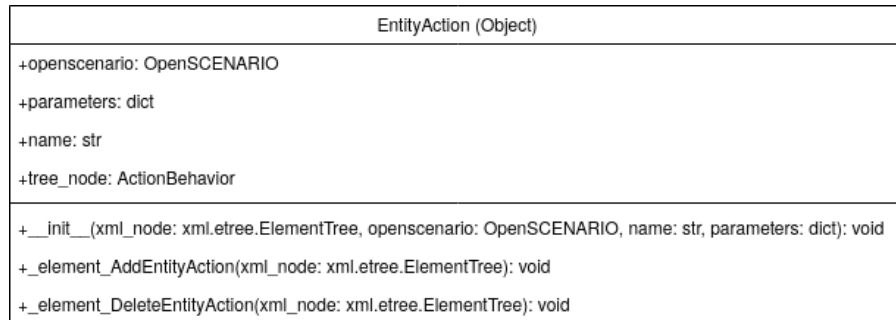


Figure 4.32: Entity Action - UML class diagram.

With this class two action behavior can be created, one to add an entity and another to remove the entity. The action behavior `AddEntityAction` (figure 4.33) performs the action of adding an entity to the scenario. As previously explained in the chapter 4.4, due to differences between the `OpenSCENARIO` and `CARLA`, all entities are first spawned in a specific location, under the loaded map, before the scenario begins its execution. Thus adding an entity to the scenario actually means moving the entity to the intended position in the scenario.

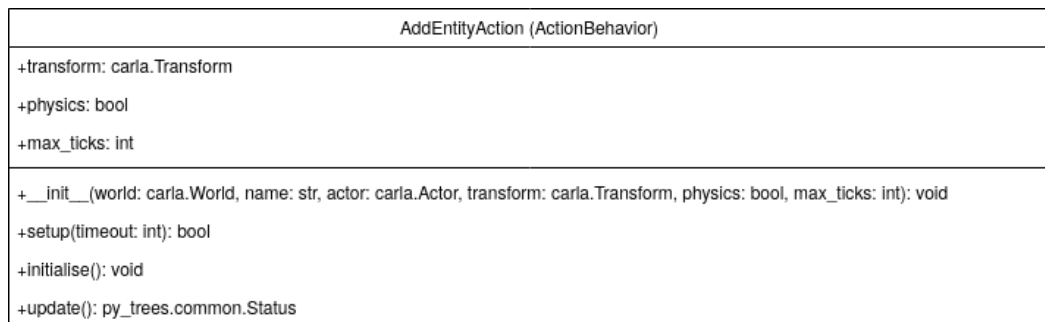


Figure 4.33: AddEntityAction behavior - UML class diagram.

Alternatively, the `DeleteEntityAction` (figure 4.34) completely removes an entity from the simulator, making it impossible from using it again in the scenario.

4.6.5 Parameter Action

The *ParameterAction* (listing 4.47) element provides the ability to change parameter values during runtime. In a normal implementation of the `OpenSCENARIO` this is only useful to trigger conditions that verify the parameters values, since all the other operations with parameters are performed previously to the scenario execution, see chapter 4.2.

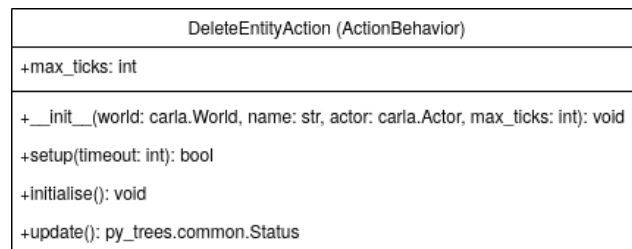


Figure 4.34: DeleteEntityAction behavior - UML class diagram.

```

1 <xsd:complexType name="ParameterAction">
2   <xsd:choice>
3     <xsd:element name="SetAction" type="ParameterSetAction" minOccurs="0"/>
4     <xsd:element name="ModifyAction" type="ParameterModifyAction" minOccurs="0"/>
5   </xsd:choice>
6   <xsd:attribute name="parameterRef" type="String" use="required"/>
7 </xsd:complexType>

```

Listing 4.47: XSD definition of the *ParameterAction* element.[34]

The *ParameterAction* element contains one attribute, "parameterRef", to identify the parameter that will be affected by the action. In this action two types of operations, on the parameter, are provided. The first, performed by the *SetAction* element, attributes the parameter with a new value independently of the previous value. The second operation, provided by the *ModifyAction* element (listing 4.48), alters the value of the parameter by taking advantage of the previous value.

```

1 <xsd:complexType name="ParameterModifyAction">
2   <xsd:all>
3     <xsd:element name="Rule" type="ModifyRule"/>
4   </xsd:all>
5 </xsd:complexType>

```

Listing 4.48: XSD definition of the *ModifyAction* element.[34]

When altering the parameter value with the *ModifyAction* element, two methods of editing the parameters are provided, contained in the *Rule* element (listing 4.49). The parameter's value can be added to another value, using the *AddValue* element, or it is possible to multiply the parameter's value by a ratio, with the *MultiplyByValue* element.

```

1 <xsd:complexType name="ModifyRule">
2   <xsd:choice>
3     <xsd:element name="AddValue" type="ParameterAddValueRule" minOccurs="0"/>
4     <xsd:element name="MultiplyByValue" type="ParameterMultiplyByValueRule"
5       minOccurs="0"/>
6   </xsd:choice>
7 </xsd:complexType>

```

Listing 4.49: XSD definition of the *ModifyRule* element.[34]

All these elements that allow to edit a parameter's value, the *SetAction*, *AddValue*, *MultiplyByValue* elements, are simple with only one attribute "value" and without any children elements.

This means that in these operations the parameter's type is never changed and the implementation should be careful about the values and perform the correct conversions.

4.6.5.1 Implementation

The correct parsing of all the previously described elements to edit a parameter's value is performed by the `ParameterAction` class (figure 4.35) that will then create the correct action behavior and store it in its `"tree_node"` member variable.

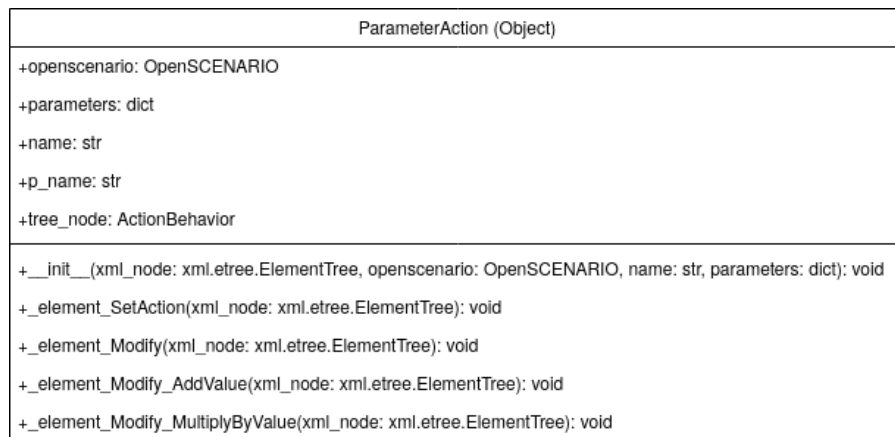


Figure 4.35: `ParameterAction` UML class diagram.

This class is responsible for finding, from the `"parameterRef"` attribute, the correct parameter to be used in the action behavior, or provide user feedback in case it does not exist.

Then it will parse the rest of the elements in order to determine the correct action to apply. In case the descendant element is the `SetAction`, the creation of the action behavior is simple and the responsibility is passed to the `"_element_SetAction"` member function, giving origin to the `SetParameter` action behavior (figure 4.36). However if the descendant element is the `ModifyAction` it will, first, be parsed by the `"_element_Modify"` member function until an action is created by the `"_element_Modify_AddValue"` function, generating the `AddParameter` action behavior (figure 4.37), or the `"_element_Modify_MultiplyByValue"` function, generating the `MultiParameter` action behavior (figure 4.38).

The `SetParameter` action (figure 4.36) allows to set a new value for the corresponding value and succeeds once it verifies that the new parameter's value is the desired value. Since the desired value is provided as a string, and without any reference to its type, this action is responsible for verifying if the value is able to be converted to the parameter's type and perform the conversion or, otherwise, fail.

On the other hand, the `AddParameter` action (figure 4.37) can only be performed with parameter types that allow mathematical calculations. However there is no constraint referred in the `OpenSCENARIO` standard indicating what types are able to be used in this action, so the following restrictions were decided during implementation. Beside the typical types that allow to

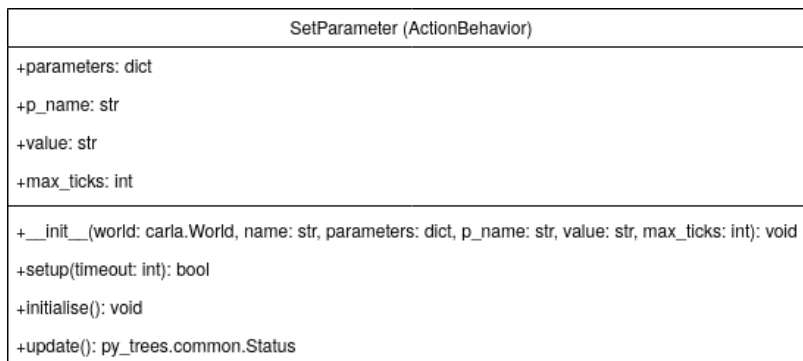


Figure 4.36: SetParameter behavior - UML class diagram.

perform the sum operation, in order to extend the compatibility of the action, the string type is also accepted resulting in a concatenation of both values.

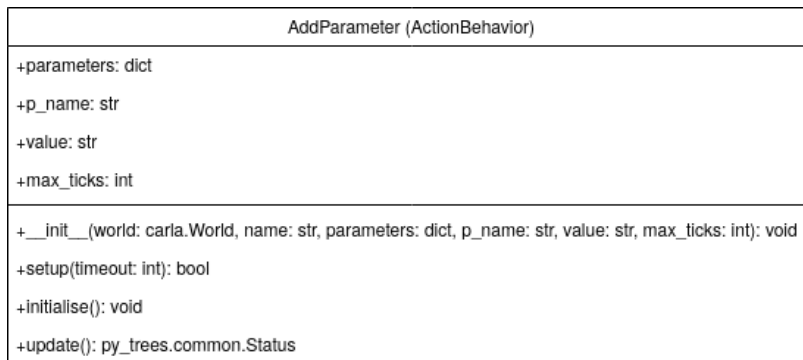


Figure 4.37: AddParameter behavior - UML class diagram.

Finally, the MultParameter action (figure 4.38) allows to multiply the previous parameter's value by a ratio. Similar to the AddParameter action, this action can, also, only be performed with types that allow for the multiplication operation. The compatibility of this action is reduced relative to the AddParameter action because the multiplication cannot be applied to the string type.

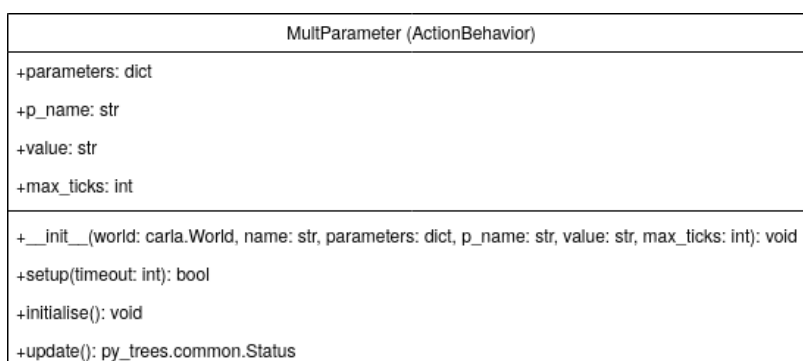


Figure 4.38: MultParameter behavior - UML class diagram.

4.6.6 Longitudinal Action

The OpenSCENARIO defines a group of actions that act upon an entity's longitudinal movements: distance, velocity and acceleration. These actions are contained in the *LongitudinalAction* element (listing 4.50) and are further distinguished into speed related actions, *SpeedAction* element (listing 4.51), and distance related actions, *LongitudinalDistanceAction* element.

```

1 <xsd:complexType name="LongitudinalAction">
2   <xsd:choice>
3     <xsd:element name="SpeedAction" type="SpeedAction" minOccurs="0"/>
4     <xsd:element name="LongitudinalDistanceAction"
5       type="LongitudinalDistanceAction" minOccurs="0"/>
6   </xsd:choice>
7 </xsd:complexType>

```

Listing 4.50: XSD definition of the *LongitudinalAction* element.[34]

4.6.6.1 Implementation

The *LongitudinalAction* class (figure 4.39) is responsible for correctly parsing all the elements that can exist contained inside the *LongitudinalAction* element. It receives the entity but it is required to find the name of the entity in order to correctly name the action node, see chapter 4.5.9.

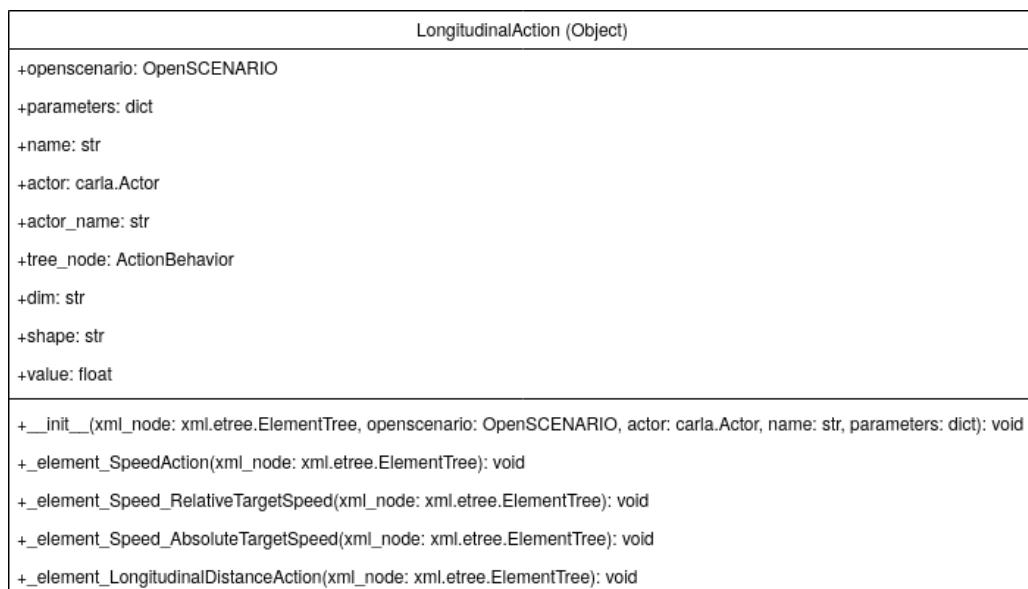


Figure 4.39: LongitudinalAction UML class diagram.

Since it parses, not only leaf element (elements that have no children), but also intermediate elements, such as *SpeedAction*, it contains more member functions that it is to be expected for a class that can only create three different actions: speed action with absolute target speed, speed action with relative target speed and longitudinal distance action.

4.6.6.2 Speed Action

The *SpeedAction* element (listing 4.51) generates actions with the ultimate goal of making an entity assume a specific final velocity, however it allows to characterize these actions by not simply controlling an entity's speed but also by defining two main aspects: the action dynamics, with the *SpeedActionDynamics* element (listing 4.52), and the action target, with the *SpeedActionTarget* element (listing 4.53).

```

1 <xsd:complexType name="SpeedAction">
2   <xsd:all>
3     <xsd:element name="SpeedActionDynamics" type="TransitionDynamics"/>
4     <xsd:element name="SpeedActionTarget" type="SpeedActionTarget"/>
5   </xsd:all>
6 </xsd:complexType>

```

Listing 4.51: XSD definition of the *SpeedAction* element.[34]

Action Dynamics The *SpeedActionDynamics* element (listing 4.52) is used to describe how the evolution of an entity's speed should evolve, either over time, distance or a constant rate, by describing the shape of the desired curve.[34]

```

1 <xsd:complexType name="TransitionDynamics">
2   <xsd:attribute name="dynamicsDimension" type="DynamicsDimension"
3     use="required"/>
4   <xsd:attribute name="dynamicsShape" type="DynamicsShape" use="required"/>
5   <xsd:attribute name="value" type="Double" use="required"/>
6 </xsd:complexType>

```

Listing 4.52: XSD definition of the *SpeedActionDynamics* element.[34]

The "dynamicsDimension" attribute defines the semantics of the "value" attribute and, consequently, the dimension that affects the evolution of the curve. It has three valid keywords: *rate*, *time* and *distance*. In case the *rate* dimension is selected, velocity of the entity should evolve according to the defined shape at a constant rate. On the other hand, if the *time* or *distance* dimensions are selected the velocity of the entity should evolve, according to the defined shape, by the passage of time or the distance travelled, respectively.[34]

The "dynamicsShape" attribute characterizes the shape of the curve that delineates how the evolution of an entity's speed should look like, i.e. defines a function $f(x)$ to shape the transition from the current speed to the desired speed according to the dimension variable selected. This attributes can assume four keywords: *linear*, *cubic*, *sinusoidal* and *step*. Each of these keywords indicate well known mathematical functions and little restrictions are provided by the standard on their implementation with the exception of ensuring the gradient of both *cubic* and *sinusoidal* functions is zero at the start and end.

The standard also provided mathematical equations to better characterize the functions:

linear: $f(x) = f_0 + rate * x$

cubic: $f(x) = A * x^3 + B * x^2 + C * x + D$

sinusoidal: $f(x) = A * \sin(x) + B$

step: $f(x) = f_{\infty}$

However no indication about the constant values are provided.[34]

At last, the "value" attribute has different meanings depending on the dimension. In case the *rate* dimension is selected it indicates the rate of evolution of the velocity, using the Δ/s unit. Alternatively, when the dimension is either *time* or *distance*, then this attribute indicates the amount time passed (*s*) or distance travelled (*m*), respectively, for the entity to reach the final velocity.[34]

Speed Target After the action dynamics is completely specified, it is necessary to determine what the goal velocity shall be. This is achieved by the *SpeedActionTarget* element (listing 4.53). Currently, in the OpenSCENARIO, it is possible to define this target through two elements: *RelativeTargetSpeed* (listing 4.54) and *AbsoluteTargetSpeed*. [34]

```

1 <xsd:complexType name="SpeedActionTarget">
2   <xsd:choice>
3     <xsd:element name="RelativeTargetSpeed" type="RelativeTargetSpeed"
4       minOccurs="0"/>
5     <xsd:element name="AbsoluteTargetSpeed" type="AbsoluteTargetSpeed"
6       minOccurs="0"/>
7   </xsd:choice>
8 </xsd:complexType>

```

Listing 4.53: XSD definition of the *SpeedActionTarget* element.[34]

The *AbsoluteTargetSpeed* is a simple element with a single attribute, "value". This element allows to define a specific terminal velocity by indicating it in the "value" attribute in *m/s*. This action is considered complete once the entity reaches the desired speed.[1]

However, the *RelativeTargetSpeed* element (listing 4.54) is a little more complex and allows to define a velocity relative to another entity's velocity. Depending on the settings of this action it can end when the speed is reached or continue, permanently, to match the entity's velocity until the action's execution is forced to stop. Instead of simply matching the relative entity's speed it is possible define different velocities in relation to its velocity, by either adding a difference or by multiplying it with a factor.[34]

```

1 <xsd:complexType name="RelativeTargetSpeed">
2   <xsd:attribute name="entityRef" type="String" use="required"/>
3   <xsd:attribute name="continuous" type="Boolean" use="required"/>
4   <xsd:attribute name="speedTargetValueType" type="SpeedTargetValueType"
5     use="required"/>
6   <xsd:attribute name="value" type="Double" use="required"/>
7 </xsd:complexType>

```

Listing 4.54: XSD definition of the *RelativeTargetSpeed* element.[34]

The "entityRef" attribute allows to select from what entity the speed will be relative to. Instead of simply matching the relative entity's speed it is possible to define different velocities in relation to its velocity. The "speedTargetValueType" can receive two keywords: *delta* and *factor*. When the *delta* is selected the desired velocity will be calculated by adding the entity's velocity to the content of the attribute "value" (equation 4.4), in m/s.[34]

$$v_f = v_R + \text{"value"} \text{ (m/s)} \quad (4.4)$$

Alternatively it is possible to define a final velocity by multiplying the entity's velocity by a factor, the content of "value" attribute. This is achieved when the "speedTargetValueType" attribute is set to *factor* (equation 4.5).[34]

$$v_f = v_R * \text{"value"} \text{ (m/s)} \quad (4.5)$$

Finally, in this element it is possible to alter the durability of the action. The "continuous" attribute allows to enable the permanent execution of the action. In case the "continuous" attribute is false, the action, similar to the *AbsoluteTargetSpeed* element, terminates once the main entity reaches the desired speed. But if the "continuous" attribute is true, the action will continuously recalculate the desired speed and follow the relative entity's velocity evolution.[34]

4.6.6.3 Speed Action Implementation

The speed actions are implemented into two different actions behaviors: *AbsoluteTargetSpeed* class (figure 4.41) and *RelativeTargetSpeed* class (figure 4.43). However in order to implement the action dynamics, due to being common to both behaviors and to its complexity, a base class is created, *DynamicBehavior* class (figure 4.40).

The *DynamicBehavior* class is derived from the *ActionBehavior* class, however it is not an action and cannot be executed on its own. Its main objective is to calculate the desired velocity values according to the action dynamics profiled.

Through the dimension, shape and value, this class implements a mathematical function, as presented in table 4.4, used to determine at any point of the action's execution what is the desired speed. The member function `set_constants` is called in its `setup` function and initializes the correct type of function and constants. When the *rate* dimension is selected, since its unit of measurement is increments per second, the dimension used in the class is time, in seconds, and the rate value is used to determine the duration of the action.

The member function `get_speed` receives the current value of the x axis. So when the *rate* and *time* dimensions are selected the member function receives the number seconds passed since the beginning of the action and when the *distance* dimension is selected that input of the member function is the number of meters travelled.



Figure 4.40: DynamicBehavior UML class diagram.

Absolute Target Speed The AbsoluteTargetSpeed class (figure 4.41) implements the behavior action that allows an entity to achieve a specific velocity indicated by the user. This implementation makes use of the DynamicBehavior class in order to determine the objective velocity every cycle. In case the action is controlling an pedestrian the control is simply indicating updating the desired velocity in the pedestrian controller. However, if the action is controlling a vehicle, the control is more complex since, similarly to a real life vehicle, the direct control is performed using the throttle and brake values. In order to implement a software controller for the vehicle achieve the desired velocities, a PID controller was implemented.

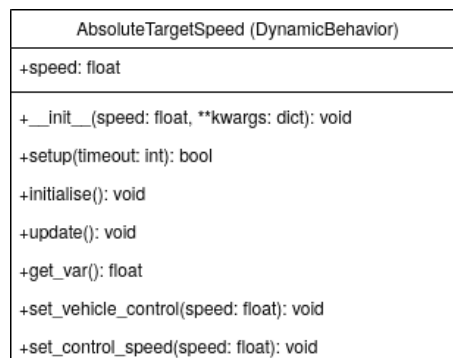
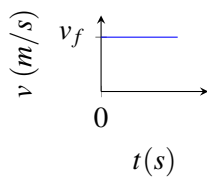
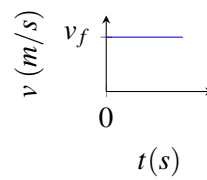
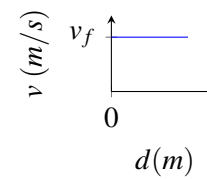
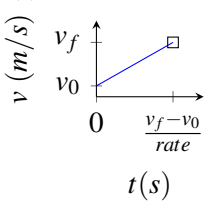
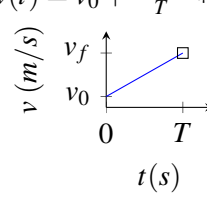
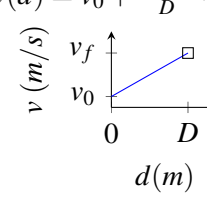
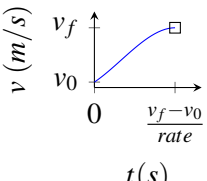
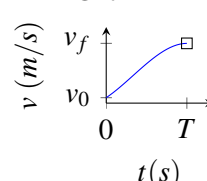
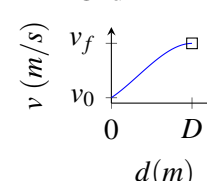
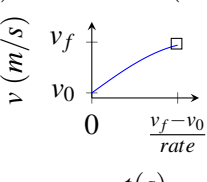
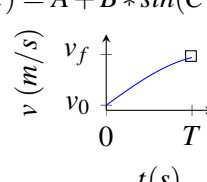
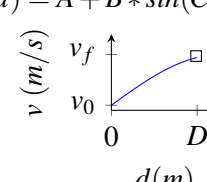


Figure 4.41: AbsoluteTargetSpeed UML class diagram.

Table 4.4: Speed dynamics evolution: Shape vs Dimension

	Rate	Time	Distance
Step	$v(t) = v_f$ 	$v(t) = v_f$ 	$v(d) = v_f$ 
Linear	$v(t) = v_0 + rate * t$ 	$v(t) = v_0 + \frac{v_f - v_0}{T} * t$ 	$v(d) = v_0 + \frac{v_f - v_0}{D} * d$ 
Cubic	$v(t) = A * t^3 + B * t^2 + C * t + D$  $A = -\frac{rate^3}{(v_f - v_0)^2}$ $B = \frac{rate^2}{v_f - v_0}$ $C = rate$ $D = v_0$	$v(t) = A * t^3 + B * t^2 + C * t + D$  $A = -\frac{v_f - v_0}{T^3}$ $B = \frac{v_f - v_0}{T^2}$ $C = \frac{v_f - v_0}{T}$ $D = v_0$	$v(d) = A * d^3 + B * d^2 + C * d + D$  $A = -\frac{v_f - v_0}{D^3}$ $B = \frac{v_f - v_0}{D^2}$ $C = \frac{v_f - v_0}{D}$ $D = v_0$
Sinusoidal	$v(t) = A + B * \sin(C * t)$  $A = v_0$ $B = v_f - v_0$ $C = \frac{\pi * rate}{2 * (v_f - v_0)}$	$v(t) = A + B * \sin(C * t)$  $A = v_0$ $B = v_f - v_0$ $C = \frac{\pi}{2 * T}$	$v(d) = A + B * \sin(C * d)$  $A = v_0$ $B = v_f - v_0$ $C = \frac{\pi}{2 * D}$

The figure figure 4.42 shows a vehicle's response to the absolute target speed action. This response was achieved with a proportional control equal to 1.0. The vehicle used was the Audi A2 vehicle, present in CARLA, modified according to the catalog described in listing A.2 from appendix A. The action dynamics' parameters chosen formed a sinusoidal shape with its peak in the 3 second value, using the time dimension.

Although the velocity function starts at 0 seconds, the vehicle only starts moving at 0.8 seconds. This is mostly due to the specific vehicle physics parameters, such as torque, friction, gear changing time and many others. The moment the vehicle starts moving a peak on the acceleration is noticed until it reaches a velocity close to the desired. Finally the vehicle acceleration stabilizes, after 1.8 seconds, and follows closely the evolution of the curve with minor difference, less than 10%. The action is finished exactly after 3 seconds when the curve reaches the final speed, 5m/s, and the vehicle's speed is within acceptable error, less than 5%.

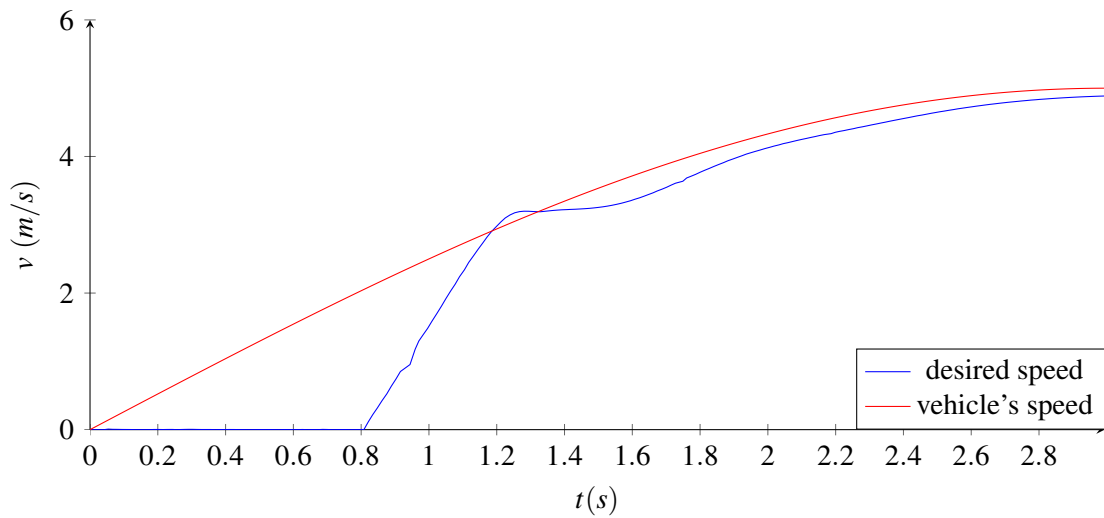


Figure 4.42: Speed evolution of a vehicle. Parameters: $k_p=1.0$, Audi A2. Test script: listing A.1 from Appendix A

Relative Target Speed Identically to the AbsoluteTargetSpeed, the RelativeTargetSpeed class (figure 4.43) is also derived from the DynamicBehavior class. Its implementation is very close to the previous action with the major difference that the goal speed is calculated according to the current velocity of the reference entity. In case the "continuous" attribute is enabled, this goal speed is continuously updated, according to the reference entity's speed, and the dynamic function recalculated.

The figure figure 4.44 shows a vehicle's response to the relative target speed action. This response was achieved with a proportional control equal to 1.0. The vehicle used was the Audi A2 vehicle, present in CARLA, modified according to the catalog described in listing A.2 from appendix A. The leading vehicle is tasked to achieve 10m/s of velocity according a step evolution. The terminal velocity is achieved around 2.5 seconds. The following vehicle is tasked to copy the leading vehicle's velocity and increase it by 1m/s. This vehicle follows a linear velocity curve, as can be observed until the 5 seconds into the script's execution, and when approaching its final velocity it regulates to maintain. The following vehicle's velocity in relation to its objective velocity, leading vehicle's velocity plus 1m/s, is maintained with less than 3% error.

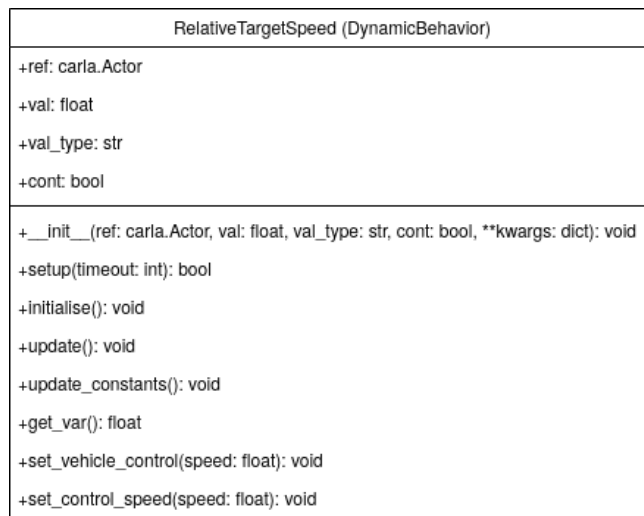
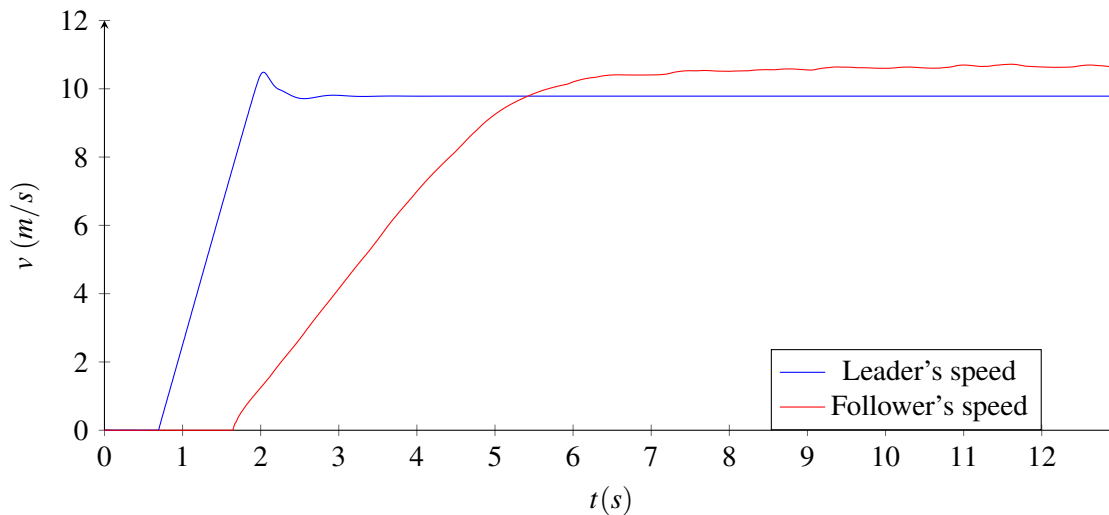


Figure 4.43: RelativeTargetSpeed UML class diagram.

Figure 4.44: Speed evolution of a vehicle following a leading vehicle. Parameters: $k_p=1.0$, Audi A2. Test script: listing B.1 from Appendix B

4.7 Conclusion

In this chapter the implementation of an OpenSCENARIO parser as an add-on for the CARLA's scenario runner plugin was described.

During the development of the parser, a new version of the Openscenario standard was released with major changes and incompatibility with the previous version introducing a major delay in the progress due to all the necessary adjustments needed to support this version.

At the end of this work package, the parser is capable of parsing all OpenSCENARIO scripts and implement the majority of behaviors and sequences in order to construct a scenario as a behavior tree to be executed by the scenario runner plugin. However there are still some OpenSCENARIO features to be implemented and incompatibilities with the CARLA simulator to be resolved.

Chapter 5

WP2: Scenario Constructor

As it is easily deduced from the chapter 4, the OpenSCENARIO standard requires a lot of knowledge about its contents and organization in order to be able to write even a short script that describes a simple scenario. Only a full understanding of all the available elements, attributes, constraints, actions and conditions allows to create a fluid scenario that behaves the way it is intended. Also even the simplest scenarios, with the help of pre-existing catalogs, will required a few hundreds lines of code making it an exhausting task.

The development of the interface has the main goal of minimize the previously mentioned issues. The proposed GUI will allow the construction of scenarios by requiring less knowledge and less work. The reduction in the necessary knowledge is achieved by providing: a list of all the available children one element can contain, simple edition and description of the attributes for elements and real-time validation of every element content. In addition, the amount of work required to build OpenSCENARIO scripts is also reduced with a more fluid chaining of all the elements and simple viewing with less information and quick access to specifics.

This GUI is designed to supplement the OpenSCENARIO parser addon (chapter 4) by providing a tool to enable the creation of scripts. As such, it is contained in the "tools" folder of the addon, figure 4.1. The application developed was organized in four C++ libraries: *Interface*, *OSCBASE*, *OSCTypes* and *OSCElements*. The libraries are linked as shown in figure 5.1.

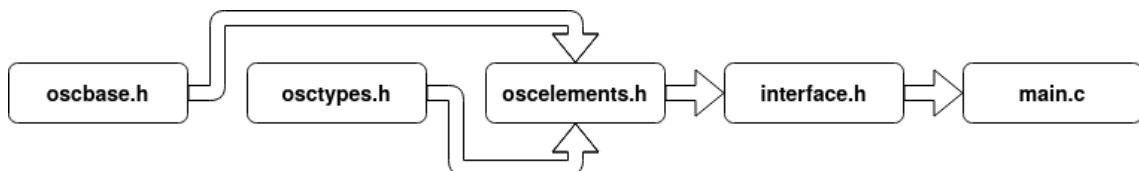


Figure 5.1: Scenario Constructor libraries diagram.

The *Interface* library allows the construction of the interface's static elements and some general functionality such as button actions. The *OSCBASE* contains most of the interface's functionality. The *OSCTypes* contains the OpenSCENARIO's data types. And, finally, the *OSCElements*

contain the OpenSCENARIO's elements. Both the *Interface* and *OSCBase* libraries are independent of any specific version of OpenSCENARIO, only implementing the functionality. The OpenSCENARIO content is only contained in *OSCTypes* and *OSCElements* libraries.

5.1 Objectives

The figure 5.2 presents a mock-up for the design of the interface that implements the basic functionally expected from this tool. The main window is composed by: a title bar, a menu bar, a side bar accompanied by a title and a main canvas. The attribute window is, then, shown in figure 5.3.

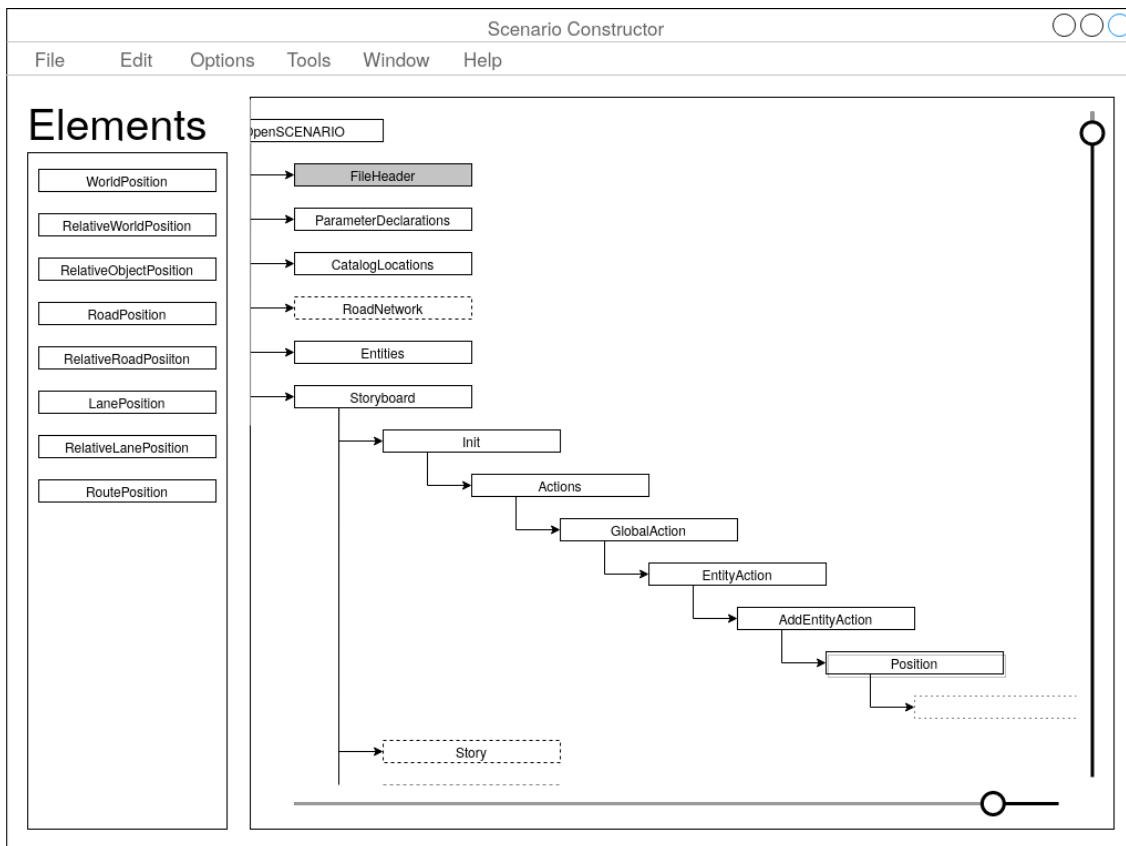


Figure 5.2: Main window - Interface mock-up.

5.1.1 Main Canvas

The OpenSCENARIO script content is intended to be represented in the main canvas. Here we see a series of rectangles organized in a hierarchical vertical tree, contained inside the main canvas, with slide bars to provide full view of the content without reformatting or reducing readability.

Each rectangle present in the tree represents an OpenSCENARIO element and the tree's hierarchy is a direct representation of the OpenSCENARIO's elements hierarchy. To provide the user with visual feedback of the elements completeness and validity, i.e. if it has all the necessary conditions to execute in the scenario, two methods are provided: a dark background color

for an invalid element, and a dashed border line for an incomplete element. These visual cues are not mutual exclusive and any of them indicates that further user action is required, otherwise the element will not be accepted by the parser.

An element is represented by a rectangle with dark background color when its attributes contents are not valid, e.g. "FileHeader" element in figure 5.2. These could mean that there are mandatory attributes that were not provided with a value or that the values inputted are not valid and should be altered. On the other hand an element represented by a rectangle with dashed border line, e.g. "RoadNetwork" element in figure 5.2, indicates that the element still requires more children elements in order to be valid. Since all the mandatory elements are automatically generated as children, the dashed line would mean that in order for the element to be valid it requires one or more optional children elements.

Alternatively, if an element is complete and valid it is represented by a rectangle with white background color and continuous border line, e.g. "ParameterDeclarations" element in figure 5.2.

5.1.2 Side Bar

The side bar is situated left to the main canvas. It contains a title with fixed text, "Elements", in order to indicate to the user that the side bar contains the elements available for choosing. Below the title is located the sidebar list where all the option elements will be vertically listed according to the selected element. For example, in the figure 5.2 the "Position" element is currently selected and the list of elements present in the side bar list are the available option elements, according to the OpenSCENARIO standard.

After selecting an element in the main canvas the user should be able to click on one of the elements in the side bar to add it as a child element to the selected elements.

Since the elements presented in the side bar do not yet exist in the scenario they do not have status, thus no dark background color or dashed border line is shown in the elements. All elements are shown in white background color and with continuous border line.

Once an element is clicked and added to the selected element, in the main canvas, as its child it should only remain in the side bar list if it is possible to add more of the same element, otherwise it should disappear.

Although elements can be added in the user's desired order, in the scenario as elements are added as children of another element, they should be placed in the correct order, required by the OpenSCENARIO standard.

5.1.3 Attribute Window

In order to keep the interface simple and easy to read, some details had to be hidden. A view with only the element names and their correspondent hierarchy provides the user with the optimal interface to quickly understand what is happening in the scenario. However with a simple double click in a specific element, present in the main canvas, a dialog window appears containing all the element's attributes information and current values, figure 5.3.

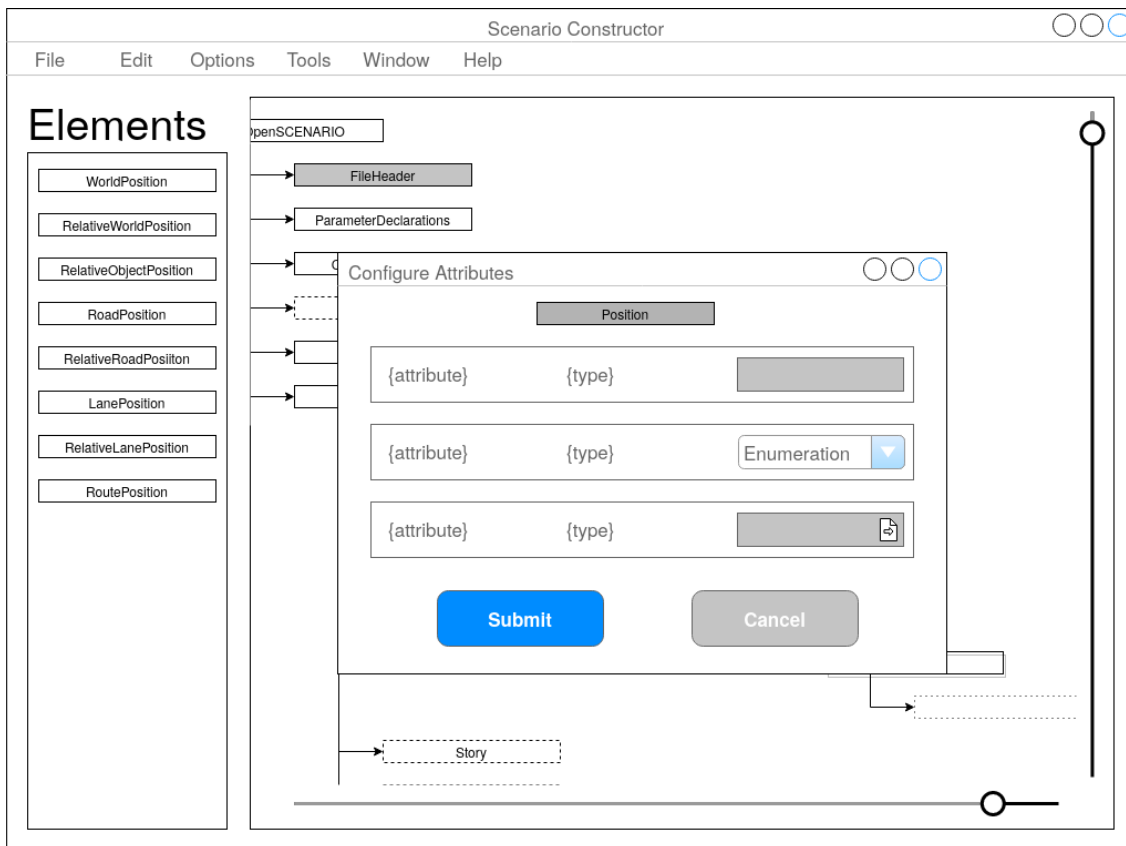


Figure 5.3: Main window showing the attribute window - Interface mock-up.

This dialog window indicates to what element the shown attributes belong to by presenting a rectangle with dark background color with the element's name in the top of with window. This is followed by a series of rectangles showing each attribute information, one rectangle per attribute. In the rectangle it should be presented with the attribute name, its OpenSCENARIO data type and the current values. In case the attribute value has not been edited since it was added to the scenario (main canvas) the value shown is the default value for the corresponding type.

The input method will vary according to the attribute's type. For example, a drop-down menu will be shown if it is an enumeration, allowing only certain keywords to be accepted.

5.2 Implementation

In this section all the graphical interface's static elements will be described and what techniques were used in its implementation. The C++ Qt library [35] was used to implement the graphical portions of this software due to the abundant and good quality documentation, its popularity and familiarity to the developers.

5.2.1 Visual Architecture

The Interface class (figure 5.4) is responsible for constructing the main window with all its static elements, interactive buttons and general actions. All the interface's structure and other global behaviors are all handled by this class. This class also initializes the root element, *OpenSCENARIO*, of the script allowing for the rest of the GUI functionality to be implemented in the elements' classes.

In order to implement the main window, the Interface class was derived from the **Qt::QMainWindow** object, which provides a simple framework to build the user interface containing a predefined layout that readily allows for a menu and status bar, toolbars and dock widgets.[35] Thus the resulting application (figure 5.5) can count with a menu bar, formed by three menus that contain several actions useful for the user, and a toolbar, used to allow quicker access to common actions, such as removing an element from the script.

A central widget, which is mandatory for any **Qt::QMainWindow** object[35], is used as a placeholder for the remaining elements of the interface, the main canvas, the side bar and its title.

5.2.1.1 Main Canvas

The main canvas objective is to display the OpenSCENARIO's script content in a simple manner for the user to view. The best discussed solution to achieve this was to organize the OpenSCENARIO elements in an interactive tree. Since OpenSCENARIO scripts can be quite long and, if organized in a tree, quite large, it is, also, essential that the main canvas has the ability to scroll both in horizontal and in vertical directions. Thus, an **Qt::QTreeWidget** object was used to build this part. This Qt object type provides a convenient method to construct classic item-based hierarchical lists while providing horizontal and vertical scroll areas when the items overflow in either directions.[35]

5.2.1.2 Side Bar

The side bar has the main objective to display vertical lists of the available elements to add to the currently selected element in the main canvas. When no element is selected, or the selected element cannot contain any more descendant elements, the side bar remains in place but it is shown empty, as in figure 5.5.

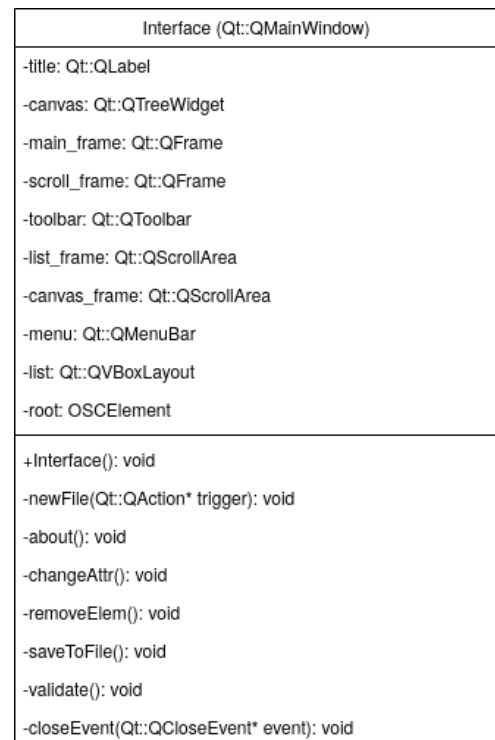


Figure 5.4: Interface UML class diagram.

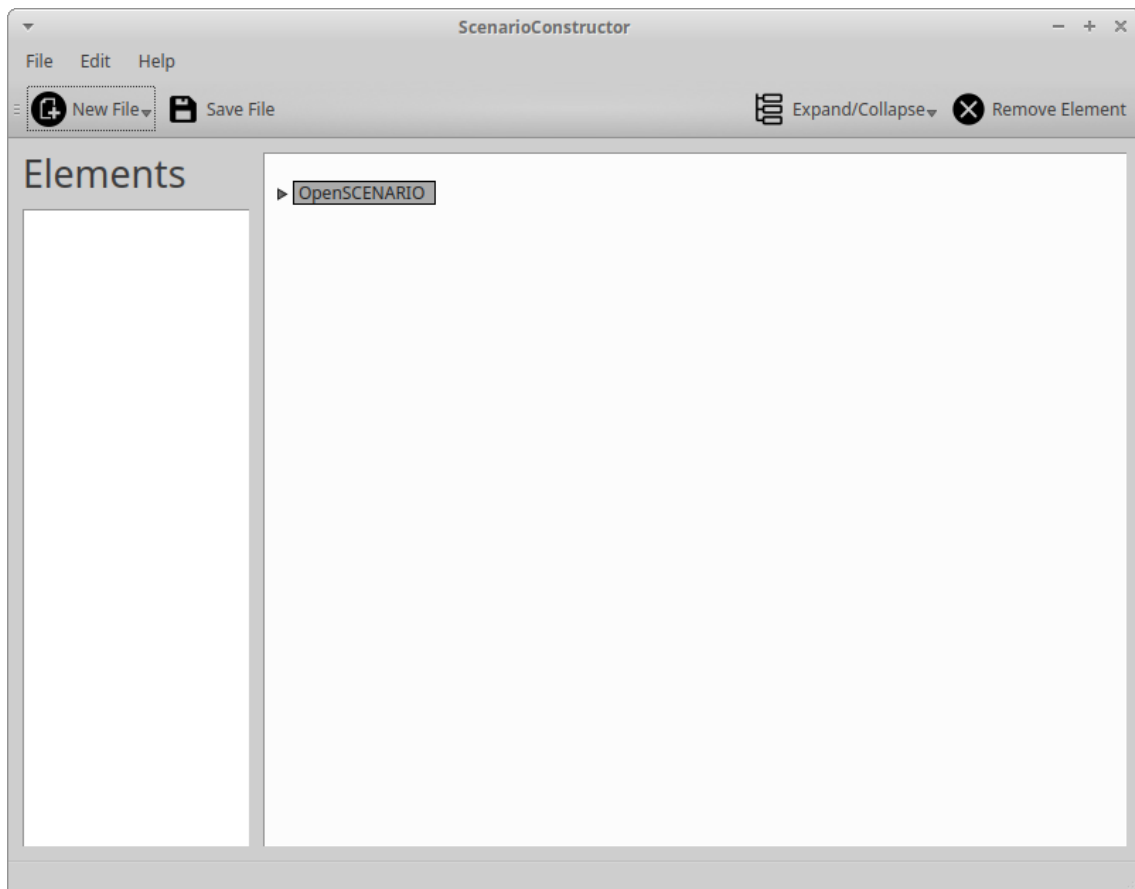


Figure 5.5: Interface structure - static elements.

In order to make it more intuitive for the user to understand the usage to this bar, a title "Elements" is placed above the list, indicating that the side bar is used to contain a list of elements. This is composed by a **Qt::QLabel** widget, which is commonly used to display text or images in a Qt interface. It is a simple object, without complex functionality, that can display a wide variety of content to the user.[35]

In some occasions the list of elements can be quite long and not be able to show all the elements in this side bar, so it should be constructed with the possibility to scroll vertically when this situation verifies itself.

A **Qt::QScrollArea** object is a frame that can contain a widget and provide a scrolling view if its size exceeds the frame size. A **Qt::QFrame** implements a frame that can be used as a placeholder for another widget. Finally, a **Qt::QVBoxLayout** allows line up widgets vertically.[35]

The side bar is constructed by a **Qt::QScrollArea** to allow scrolling when too many elements are displayed. The **Qt::QScrollArea** contains a **Qt::QFrame** that contains, in turn, a **Qt::QVBoxLayout** allowing to display the elements in a vertical fashion.

5.2.1.3 Toolbar

The toolbar, embedded in the **Qt::QMainWindow** object, is intended to provide easier access to some of the most common actions. It contains four buttons, grouped two by two on each extreme of the bar.

On the right side a drop-down menu, "Expand/Collapse", shows a list of two buttons, "Expand All" and "Collapse All" respectively. These buttons evoke the member functions, `void expandAll()` and `void collapseAll()`, of the root element, in the main canvas, that will perform, recursively, the correspondent action on all its children elements. The collapse or expansion of an element consists in hiding or showing its children elements, and this behavior is natively implemented by the **Qt::QTreeWidget** object.

The second button present on the right side of the toolbar, "Remove Element", allows to delete the currently selected element from the main canvas and, consequently, from the script's content. When this button is pressed, the element's object and all its children will be deleted, recursively. The next, remaining, element will become selected after the deletion is successful. In case the last element of the script is deleted, the selection will switch to the last, remaining, element. This automatic selection facilitates the deletion of several sequenced elements.

On the left side, a drop-down menu, "New File", presents two actions "OpenSCENARIO" and "Catalog". Both actions allow to create a new OpenSCENARIO script. However the first creates a scenario description script while the second creates a new catalog. When the user clicks on any of these actions, a warning window is presented informing that all data is lost if not saved, and after saving, or not, the current content of the scenario is deleted from the main canvas and a new root element is presented.

Next to the "New File" menu, the "Save File" button allowing to save the current contents of the main canvas into a file. When this button is clicked, a **Qt::QFileDialog**, which provides a dialog for the user to select files and directories [35], is executed allowing to select the location and filename to save the XML content. The XML content is generated by evoking the `string generateXML()` member function of the root element, that will generate, recursively, the xml of all elements and children.

5.2.2 Functionality

Most of the interface's functionality is independent of the current version of the OpenSCENARIO and is implemented in the "oscbase.h" library. A script content is displayed in the main canvas in a tree view, implemented by a **Qt::QTreeWidget** object. The figure 5.6 shows how a script content and the process of its creation look using the provided GUI. A portion of an OpenSCENARIO script is displaying in the main canvas. All the elements are organized in a tree view, in a vertical fashion. The "CatalogLocations" element is selected and all its children options are displayed in the side bar.

The **Qt::QTreeWidget** present in the main canvas contains only one column and each line is composed by a **Qt::QTreeWidgetItem** with a widget attached, in this case a **Qt::QLabel**.

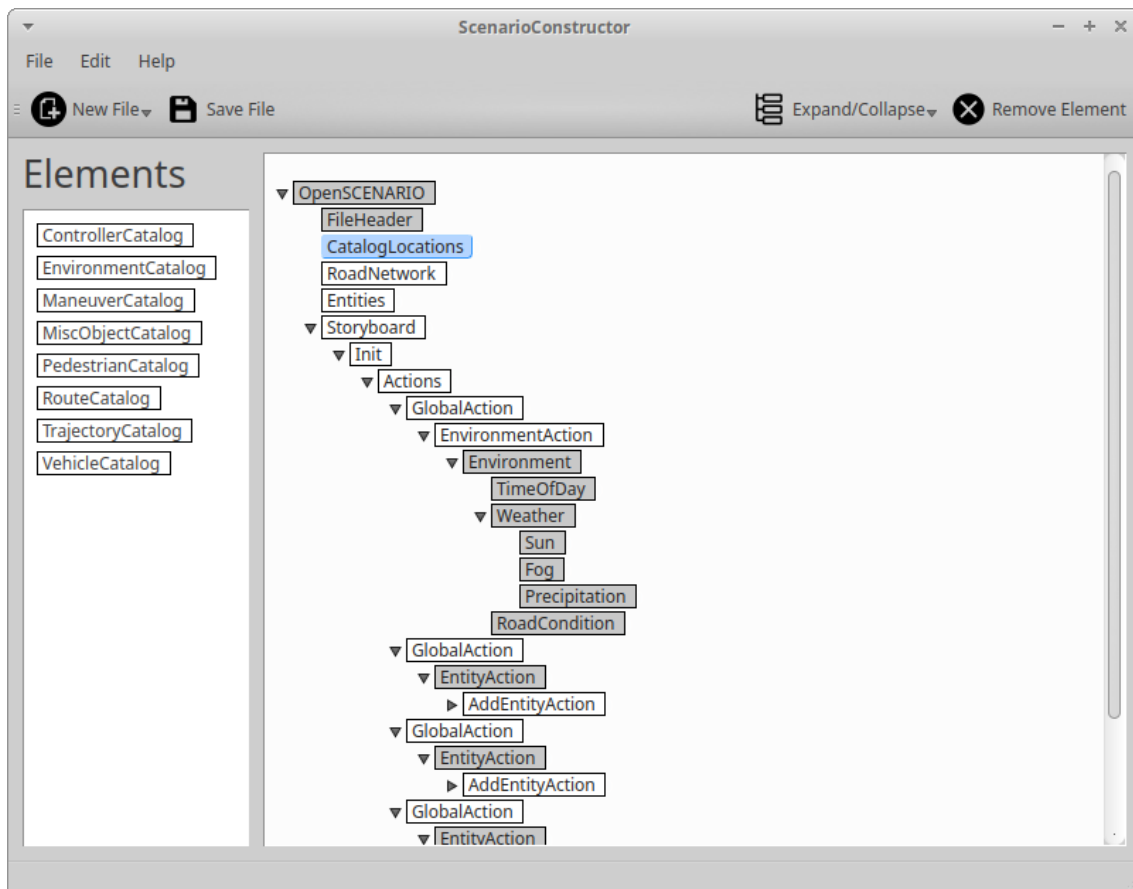


Figure 5.6: Example of the creation of a script.

Each element, both in the main canvas and in the side bar list, is mainly implemented by the `OSCElement` class, figure 5.7, which is derived from the `Qt::QTreeWidgetItem`.

The `OSCElement` class implements most of the interface's functionality since it is responsible for: containing all its child elements, adding the list of elements in the side bar, adding its children when selected from the side bar, deleting its child elements, generating XML, displaying the attribute window, between others. This is a virtual class designed to be the base of every OpenSCENARIO element and to simplify the creation of each element. It contains three constructors: one for the root element, one for elements in the sidebar and one for children elements in the main canvas.

5.2.2.1 Label

As previously stated each element is a `Qt::QTreeWidgetItem`, in this case a `OSCElement`, with a widget associated. This widget is a `Qt::QLabel`. The label is the only visible part of the element, it contains its name and shows its state by the background color and border style and color. This behaviors are implemented by the `OSCLabel` class, figure 5.8.

The `OSCLabel` class implements the basic OpenSCENARIO related behaviors, the completeness and validity of an element. Beside presenting the element's name, a dark background color

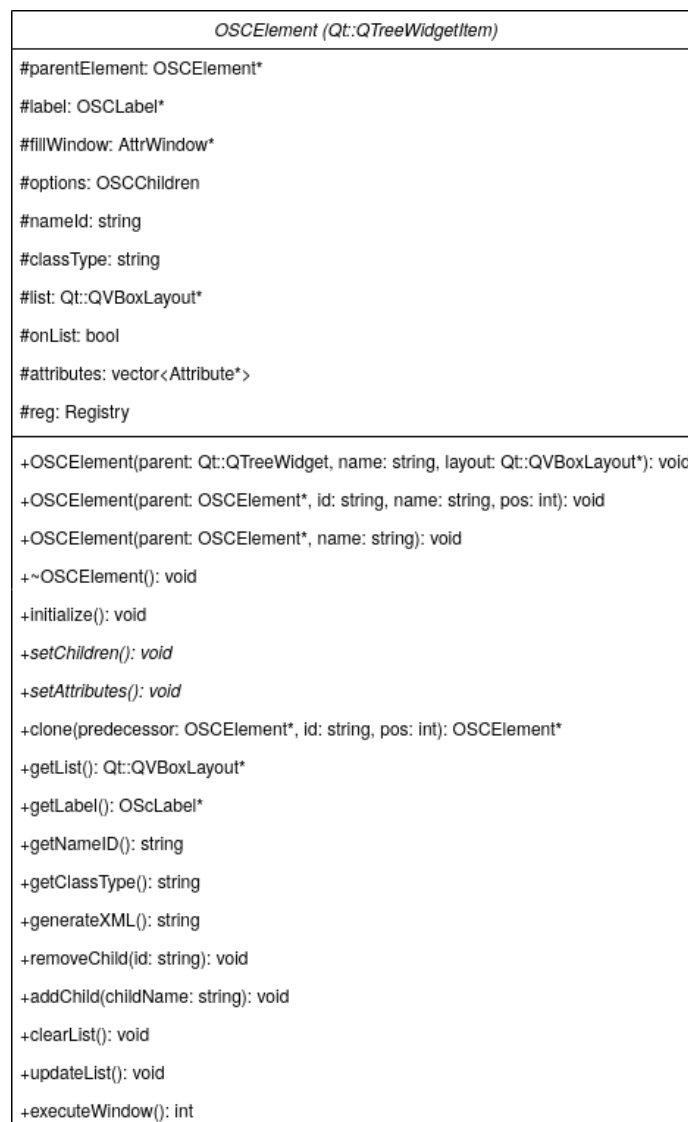


Figure 5.7: OSCElement UML class diagram.

indicates that the element's attributes require completion, and a dashed red contour indicate that the element's require more children to be complete.

However the labels present in the interface contain more behaviors, e.g. selecting a label and double-clicking to edit the elements attributes. Since the desired mouse events are different for an element present in the main canvas than for an element present in the side bar list, two classes are developed. The ListLabel class implements a label for an element that will be present in the side bar list, this class simply implements the behavior of adding a child to the selected element when the label is clicked. The CanvasLabel class implements the behavior for when an element is selected, highlighting the object, and opening the attribute window when the element is double-clicked.

The OSCElement class stores a reference to the label as a OSCLabel class, ignoring the specific type of the label, only worrying with updating the status of the element, background colour

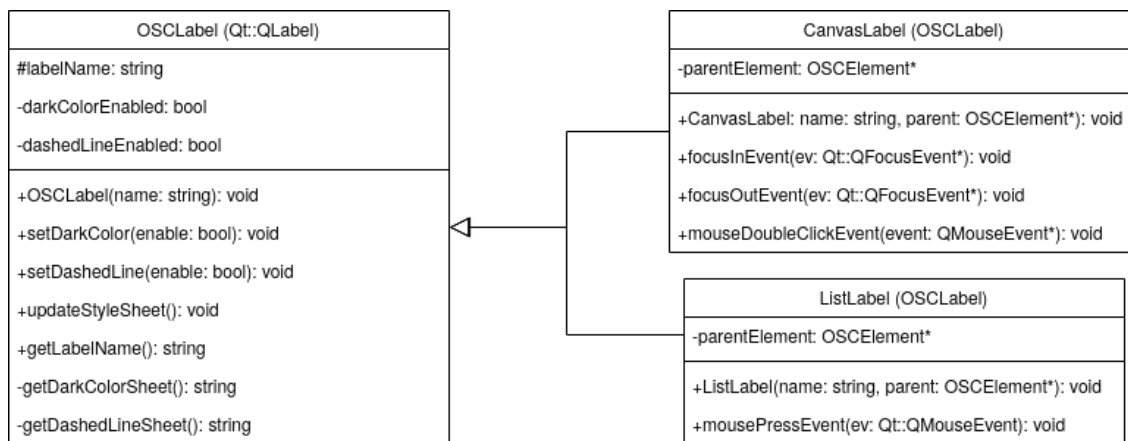


Figure 5.8: OSCLabel, ListLabel and CanvasLabel UML class diagram.

and border style.

5.2.2.2 Attributes

OpenSCENARIO elements can have attributes that allow to select values for a variety of reasons. In the GUI an element that contains attributes is presented with a dark background, if the attributes values are yet to be defined. These attributes can be edited by double-clicking in an element and editing their value in the modal window that will appear in the screen, figure 5.9.

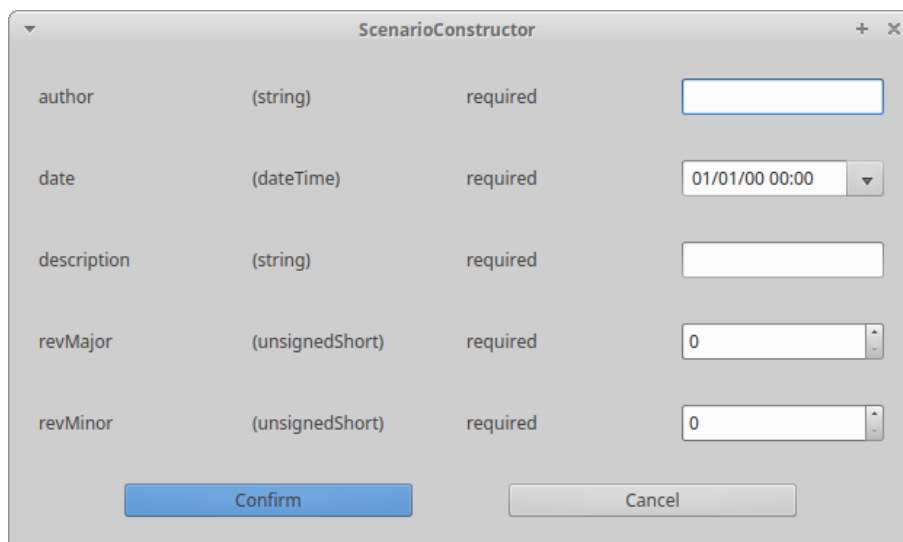


Figure 5.9: Example modal window with the *FileHeader* element attributes.

All the element's attributes are shown in a modal window, always with the same structure. The window is composed by a table with four columns and as many line as attributes. In each line all the relevant information about each attribute and their values are presented.

An attribute contains four types of information that is relevant to the user: name, data type, requirement rule and value. The OpenSCENARIO standard can define, freely, its own data types

for the attributes, and these types are contained in the "osctypes.h" library. The name of an attribute usually indicates its function. The data type defines what kind of data is accepted in the attribute's value. And the requirement rule indicates if the attribute is mandatory or optional. In this software, the attributes functionality and content is implemented by the Attribute and AttrType classes, figure 5.10.

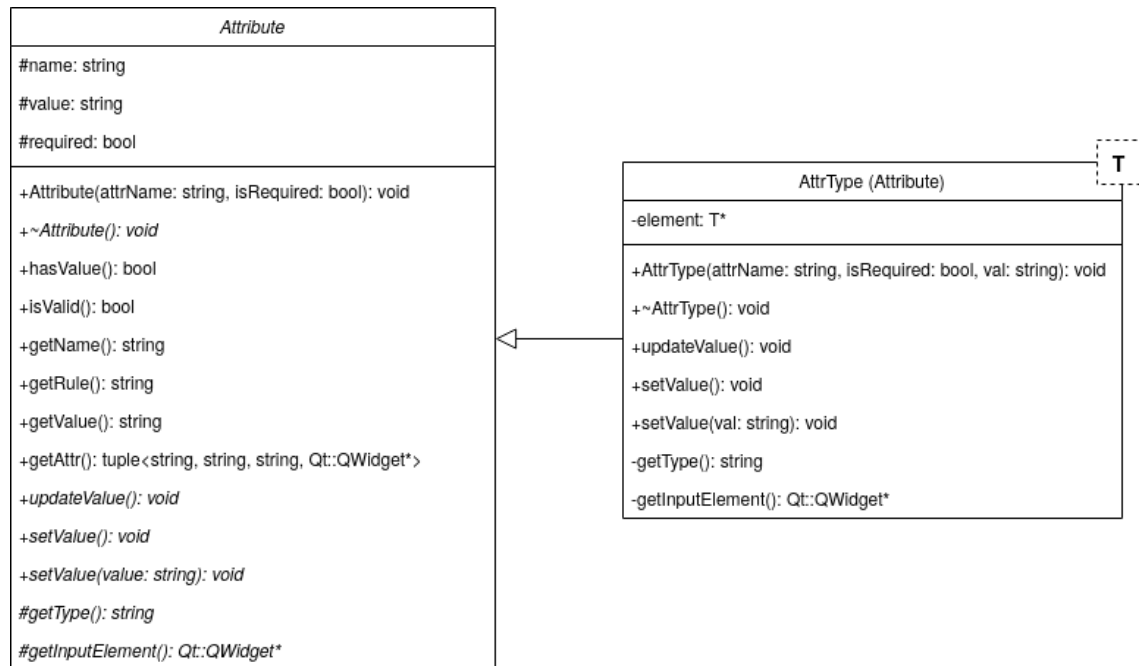


Figure 5.10: Attribute and AttrType UML class diagram.

The Attribute class, is a virtual class, that implements the relevant information of the attribute in order to generate its XML content. It's name, requirement rule, data type and current value, converted to string, are stored. It also evaluates the state of the attribute indicating if it has a value or if it is valid. An attribute is always valid if it is optional or if it is required but contains a value. The OSCElement class contains a vector of Attribute classes in order to store the element's attribute values.

The AttrType class is derived from the Attribute class with the objective of specializing the input of the attribute according to its data type. Since this class is a template it can store the OpenSCENARIO data type corresponding to the attribute's type and personalize the input values and validation in the attributes window.

The attribute window is a modal created using an **Qt::QDialog** class, which generates top-level windows for short-term tasks while blocking input to other visible windows in the application [35]. This window is implemented by the AttrWindow class (figure 5.11) and every OSCElement instance contains its own instantiation of this class.

The Attrwindow generates a window, as shown in figure 5.9, that displays vertically all the attributes of the element. This vertical display is placed by a **Qt::QVBoxLayout** and each line is an instance of the AttrLine class.

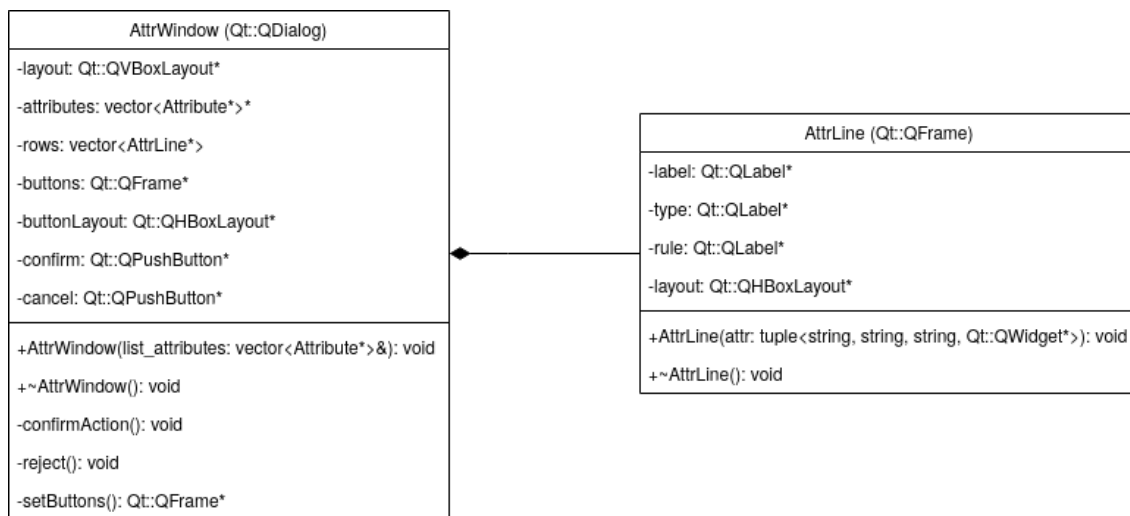


Figure 5.11: AttrWindow UML class diagram.

Each AttrLine class is responsible for displaying the information about its attribute. Using a **Qt::QHBoxLayout** it organizes the attribute's information into four horizontal columns. The first column displays the attribute name, the second the data type, the third the requirement rule and the last column displays the adequate widget for the user to input the attributes values.

Closing the window or clicking in cancel erases all the changes made by the user. The only way to accept the changes and update the attributes value is by clicking in the confirm button.

5.2.2.3 Children

In case an element can have child elements, these are managed by the OSCChildren class (figure 5.12) instance that is stored in the "options" member variable of the OSCElement class.

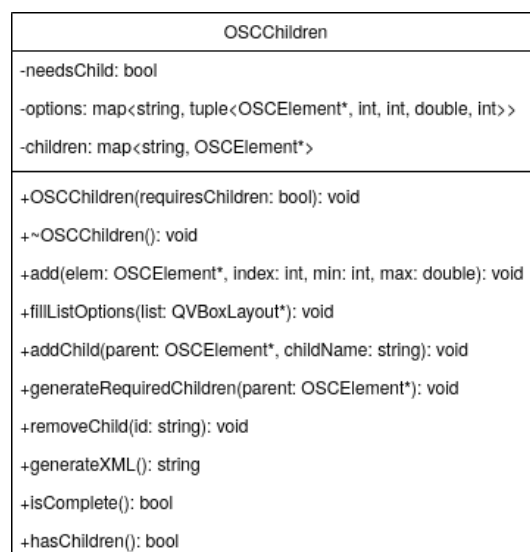


Figure 5.12: OSCChildren UML class diagram.

The OSCChildren class is responsible for all the aspects of managing an element's children. It stores a list of all the possible child elements, updates the side bar list with the available children, stores the current children of the element and evaluates the state of the child, i.e. if the minimum or the maximum number has been reached.

In each element's class all possible children elements are listed along with its settings: position and multiplicity. These elements are organized in the "options" member variable and identified by its reference name. When a element child is created, the OSCChildren instance will make a copy and update the information of the correspondent element in the "options" member variable and store it in the "children" member variable. When a child element is created all its mandatory children, with a minimum value ≥ 1 , are automatically added to the scenario, simplifying and accelerating the creation of scripts.

When an element is selected in the main canvas, it's OSCChildren instance will scan through all it's options children, verify if their maximum has been reached and, otherwise, add them to the side bar list for the user to select.

5.3 Content

The OpenSCENARIO content in this software is divided into elements and data types. The elements are created as classes by deriving from the OSCElement class and are instantiated in the main canvas in order to construct a script, while the data types are created as classes derived from an adequate widget for the user to input the corresponding value and are part of the attributes. The elements are placed in the "oscelements.h" library and the data types are placed in the "osctypes.h" library.

5.3.1 OpenSCENARIO Elements

Each OpenSCENARIO needs to be declared into a class in order to be used in the interface. However the classes follow a standard template that is quite simple and should enable future development of an automated method to update to a newer version of the standard. The listing 5.1 shows an example of the template all OpenSCENARIO's elements should produce.

```
1 class CatalogReference : public OSCElement
2 {
3 public:
4     CatalogReference(OSCElement *parent);
5     CatalogReference(OSCElement *parent, std::string id, int pos);
6
7     void setChildren() override;
8     void setAttributes() override;
9 };
```

Listing 5.1: Example of an element's class: *CatalogReference* element.

The class's name is the element's name. Two constructors should be defined, one to create elements as children in the main canvas and another one to add the element to the side bar list. Furthermore each element can have two functions `void setChildren()` and `void setAttributes()`, if the element has children and/or attributes, respectively. These functions are overridden from the base class, `OSCElement`, and are not mandatory for any element.

Attributes In case an element contains attribute, the attribute's information should be defined in the `void setAttributes()` function body. This can be done easily with the help of a series of defines with the following structure:

```
#define ATTR(type, name, req, default)
```

The "type" references the class that implements the corresponding data type of the attribute. The "name" is a string containing the attribute's name. The "req" is a boolean that if true indicates that the element is required. The "req" argument is optional, being the "true" value set as default. And the "default" is a string containing the default value for the attribute. This argument is also optional and a default value is only defined if one is provided. The listing 5.2 shows an example of definition of an element's, *OpenSCENARIO*, attributes. In this case two attributes are defined with the input class `OSCString` and with default values.

```
1 void OpenSCENARIO::setAttributes()
2 {
3     std::string name, value;
4     value = "http://www.w3.org/2001/XMLSchema-instance";
5     ATTR(OSCString, "xmlns:xsi", true, value);
6     name = "noNamespaceSchemaLocation";
7     value = "OpenSCENARIO.xsd";
8     ATTR(OSCString, name, true, value);
9 }
```

Listing 5.2: Example of an element's attributes: *OpenSCENARIO* element.

Children If an element can have children elements, i.e. if it can contain other elements, these elements should be described in the `void setChildren()` function body.

In order to provide an element with possible children, these should be added to the "options" member variable, which is an instance of `OSCChildren` class. This class provides a method to allows easy addition of child elements:

```
void add(OSCElement* elem, int index=0, int min=0, double max=INF);
```

In the first argument, a reference to an instance of the element's class should be passed. The second argument receives the index of the child, which is the position it takes in relation to other child elements. In case an element does not require a specific order for their children, the same index can be used for all declarations. The last two arguments define the multiplicity of the child element. The "min" indicates the number of elements that should be present as children of the main element in order for it to be considered complete. On the other hand, the "max" indicates the

maximum number of instances of these element that the main element can have. An example of an element with multiple child elements is presented in listing 5.3.

```

1 void OpenSCENARIO::setChildren()
2 {
3 options.add(new FileHeader(this), 0, 1, 1);
4 options.add(new ParameterDeclarations(this), 1, 0, 1);
5 options.add(new CatalogLocations(this), 2, 1, 1);
6 options.add(new RoadNetwork(this), 3, 1, 1);
7 options.add(new Entities(this), 4, 1, 1);
8 options.add(new Storyboard(this), 5, 1, 1);
9 }

```

Listing 5.3: Example of an element's children: *OpenSCENARIO* element.

5.3.1.1 OpenSCENARIO Data Types

Contained in the "osctypes.h" library, all the existent data types for element's attributes should be declared as classes, with the objective to define an input widget that allows the user to insert or select the correct values for the attribute data type in question.

The data types classes should follow the structure present in the example figure 5.13. As a convention, all data type classes should be prefixed by "OSC", as in the OSCExample class. The class should be derived from the Qt widget that is most appropriate for the type of expected input and correctly setup the widget in the constructor. Furthermore all classes should have: a `string` type member variable, where the name of the data type will be stored as a string; a `void editValue(string val)` member function, that allows to set the current value that appears on the widget and a `string getValue()` member function, that will return the current value of the widget converted to string.

OSCExample (Qt::SomeWidget)
+type: string
+OSCExample(): void
+editValue(val: string): void
+getValue(): string

Figure 5.13: Example of a data type class.

OSCEnumeration The OSCEnumeration class (figure 5.14) allows the user to select from a list of predefined words. Since OpenSCENARIO defines multiple data types that are enumerations, this class is to be used as a base to define all these enumerations. The derived class should list the available keywords in their constructor. This class uses the **Qt::QComboBox** as the input widget, which is a combined button and popup list that allow to present a list of options to the user while taking minimum amount of screen space.[35]

OSCEnumeration (Qt::QComboBox)
+editValue(val: string): void
+getValue(): string

Figure 5.14: OSCEnumeration data type class.

OSCBoolean The OSCBoolean class (figure 5.15) is an example of an enumeration and it makes use of the OSCEnumeration base class. In the constructor a list of two words is defined ("true" and "false").

OSCBoolean (OSCEnumeration)
+type: string
+OSCBoolean(): void

Figure 5.15: OSCBoolean data type class.

OSCUnsignedShort The class present in figure 5.16 allows to implement an unsigned short data type. It uses a **Qt::QSpinBox** widget for user input, providing a spin box that handle integers allowing the users to increase or decrease with the mouse and keyboard and input directly a specific value.[35]

OSCUnsignedShort (Qt::QSpinBox)
+type: string
+OSCUnsignedShort(): void
+editValue(val: string): void
+getValue(): string

Figure 5.16: OSCUnsignedShort data type class.

OSCDateTime The OSCDateTime data type (figure 5.17) allows to facilitate the input of both a date and a time by the user. It is derived from the **Qt::QDateTimeEdit**, a widget that allows the user to edit dates by using the keyboard, the arrow keys to navigate and increase or decrease the date values or to use a popup calendar to select a specific date, with the help of the **Qt::QCalendarWidget**.[35]

OSCDateTime (Qt::QDateTimeEdit)
+type: string
+OSCDateTime(): void
+editValue(val: string): void
+getValue(): string

Figure 5.17: OSCDateTime data type class.

OSCString In order to allow the user to input strings as normal text, the OSCString class (figure 5.18) is derived from the **Qt::QLineEdit**, a one-line text editor widget that allows the user to introduce a single line of plain text and enables several useful function such as copy, paste, undo, redo, etc.[35]

OSCString (Qt::QLineEdit)
+type: string
+OSCString(): void
+editValue(val: string): void
+getValue(): string

Figure 5.18: OSCString data type class.

OSCDouble Identically to the OSCUnsignedShort class, the OSCDouble class (figure 5.19) also allows the user to input a number, but this time with more precision. By deriving from the **Qt::QDoubleSpinBox** it provides the user with a spin box that takes doubles. It also allows the user to select a value by clicking the buttons up and down, using the arrow keys or by entering a specific value directly.[35]

OSCDouble (Qt::QDoubleSpinBox)
+type: string
+OSCDouble(): void
+editValue(val: string): void
+getValue(): string

Figure 5.19: OSCDouble data type class.

5.4 Conclusion

The GUI developed to support the development of OpenSCENARIO scripts was presented in this chapter. After a description of the goals hoped to achieve with this graphical interface, a description of the implementation of its features and functionality was described.

The GUI implementation architecture was designed with portability between OpenSCENARIO versions in mind and so the main functionality is independent of any specific version.

The achieved result was a simple interface that allows to easily create OpenSCENARIO scripts, validate elements in real time and manage the elements content quite easily.

Chapter 6

Conclusions & Future Work

6.1 Summary

This thesis focused on developing a product to allow to execute several complex OpenSCENARIO scripts in seamless integration with the CARLA simulator. Furthermore, a GUI was developed as an auxiliary tool to help simplify the process of building OpenSCENARIO scripts.

Initially it was performed an analysis of the current state of the art on autonomous driving technologies to help better understand what would be required to perform the best urban driving simulation. An introduction to ROS and OpenSCENARIO allowed to understand the capabilities of these technologies and the major interest to have compatibility with the simulator. Having an objective established for what an urban driving simulator should be capable of, it was performed a review on the most feature-rich urban driving simulators on the market. After comparison and highlighting the CARLA simulator as the most appropriate to our project, a deeper analysis on the simulator's capabilities was performed.

Using the selected simulator it was developed a software to, as add-on to the CARLA's scenario runner, improve the integration of the simulator with OpenSCENARIO. This product allowed to convert a scenario into a timeline of actions and conditions in CARLA, while ensuring the repeatability of the scenarios. The implemented result consisted in developing a parser that would build a behavior tree representing the scenario's timeline and actions and final conclusion is that it allowed to parse all OpenSCENARIO scripts and provided compatibility with a lot of actions and conditions in order to build complex timelines.

In order to facilitate the creation of OpenSCENARIO scripts while reducing the user expertise requirements, on the standard, and the work to write long and exhaustive XML scripts, a GUI was developed using the Qt framework. The process consisted in designing an interface that was easy to read and intuitive while also providing a fluid construction of scripts. The architecture design accounted for future updates on the OpenSCENARIO standard consisting in a code mostly dependent on XML structure and independent of specific OpenSCENARIO content, aside for the libraries where OpenSCENARIO is implemented.

Some challenges were presented during the development that incurred into significant delays. The introduction of the new OpenSCENARIO v1.0.0 version two months after starting the development of the parser had a major impact in the development where a lot more time had to be allocated into the first work package impacting all the development timeline.

Due to the time drifts suffered in the project, it was not possible to develop the third work package - to develop communication and control through ROS framework. This is an indispensable feature that will be added.

6.2 Future Research Directions

This project promises a lot of potential for the company and so the development should continue on the next months to complete all the goals established. Several projects in *Altran Portugal* and *Vortex-CoLab*, that are now starting their development, are planning to use CARLA simulator as their base for testing and this thesis output will be the gateway for this possibilities.

More time investment in the OpenSCENARIO parser will provide the possibility to increase the compatibility with the standard by developing more complex, control wise, actions, e.g Synchronize actions, and other OpenSCENARIO specific features that will be a challenge to implement, e.g. addition of triggering entities to the actions.

The recent release of CARLA simulator v0.9.9 introduces a better interface and compatibility with the traffic signs and lights paving the path to introduce the traffic related actions into the OpenSCENARIO path, e.g. Infrastructure actions.

Further work on the GUI will also allow for the development of more intuitive, interactive and user friendly capabilities in the creation of scenarios, such as dynamically design a vehicle route on the map and the correspondent translation to OpenSCENARIO.

As said in the summary, the integration with the ROS framework is essential. The capability to execute scenarios, control the simulation and read all the simulation's data by using the ROS as the communication interface simplifies all further development on projects and compatibility with other systems.

References

- [1] ASAM. ASAM OpenSCENARIO - Dynamic content in driving simulation - User Guide. <https://www.asam.net/standards/detail/openscenario>, March 2020. Accessed: 15-04-2020.
- [2] CARLA. CARLA Documentation. <https://carla.readthedocs.io/>. Accessed: 28-06-2020.
- [3] Directorate General for Transport. European Commission, Annual Accident Report. Technical report, European Commission, June 2017.
- [4] S Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. dot hs 812 115. *National Highway Traffic Safety Administration, US Department of Transportation*, 2015.
- [5] Tomasz Sulkowski, Paulina Bugiel, and Jacek Izydorczyk. In Search of the Ultimate Autonomous Driving Simulator. In *2018 International Conference on Signals and Electronic Systems (ICSES)*, pages 252–256. IEEE, sep 2018.
- [6] Nidhi Kalra and Susan M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, dec 2016.
- [7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- [8] Altran Portugal. Rumo ao Futuro: Altran Portugal apresenta Vortex Co-Lab. https://www.altran.com/pt/pt-pt/news_press_release/rumo-ao-futuro-altran-portugal-apresenta-vortex-colab/. Accessed: 2019-11-20.
- [9] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *CoRR*, abs/1906.0, 2019.
- [10] Matthias Jarke, X Tung Bui, and John M Carroll. Scenario Management: An Interdisciplinary Approach. *Requirements Engineering*, 3(3):155–173, 1998.
- [11] Simon Ulbrich, Till Menzel, Andreas Reschka, Fabian Schuldt, and Markus Maurer. Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, volume 2015-Octob, pages 982–988, 2015.

- [12] Sebastian Geyer, Marcel Baltzer, Benjamin Franz, Stephan Hakuli, Michaela Kauer, Martin Kienle, Sonja Meier, Thomas Weißgerber, Klaus Bengler, Ralph Bruder, and Others. Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. *IET Intelligent Transport Systems*, 8(3):183–189, 2013.
- [13] Aaron Heinz and Johann Schweiger Wolfram Remlinger. Track- / Scenario-based Trajectory Generation for Testing Automated Driving Functions. In *8. Tagung Fahrerassistenz*, München, 2017. Lehrstuhl für Fahrzeugtechnik mit TÜV SÜD Akademie.
- [14] T A Alspaugh, H U Asuncion, and W Scacchi. Analyzing software licenses in open architecture software systems. In *2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, pages 54–57, 2009.
- [15] M Quigley, B Gerkey, K Conley, J Faust, T Foote, J Leibs, E Berger, R Wheeler, and A Ng. ROS: An open-source Robot Operating System. *ICRA Workshop on Open Source Software*, 3:1–6, 2009.
- [16] Arpad Barsi, Vivien Poto, and Viktor Tihanyi. Creating OpenCRG Road Surface Model from Terrestrial Laser Scanning Data for Autonomous Vehicles. In Károly Jármay and Betti Bolló, editors, *Vehicle and Automotive Engineering 2*, pages 361–369, Cham, 2018. Springer International Publishing.
- [17] CARLA. GitHub - carla-simulator/ros-bridge: ROS bridge for CARLA Simulator. <https://github.com/carla-simulator/ros-bridge>, 2018. Accessed: 03-02-2020.
- [18] CARLA. GitHub - carla-simulator/scenario_runner: Traffic scenario definition and execution engine. https://github.com/carla-simulator/scenario_runner, 2018. Accessed: 03-02-2020.
- [19] CARLA. GitHub - carla-simulator/carla: Open-source simulator for autonomous driving research. <https://github.com/carla-simulator/carla>, 2017. Accessed: 03-02-2020.
- [20] Daniel Aros Banda and Joel Wachsler. Exploration of AirSim using C and Rust in the Context of SafetyCritical Systems, 2018.
- [21] Microsoft. AirSim Documentation. <https://microsoft.github.io/AirSim/>. Accessed: 03-02-2020.
- [22] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 621–635, Cham, 2018. Springer International Publishing.
- [23] CVEDIA. Simulation Services - CVEDIA’s SynCity, A Computer Vision Simulator. <https://www.cvedia.com/solution/syncity/>. Accessed: 03-02-2020.
- [24] Justin Carrillo, Burhman Gates, Gabe Monroe, Brent Newell, and Phillip Durst. Using Physics-Based M&S for Training and Testing Machine Learning Algorithms. In Jan Mazal, editor, *Modelling and Simulation for Autonomous Systems*, pages 445–455, Cham, 2019. Springer International Publishing.

- [25] C Pilz, G Steinbauer, M Schratler, and D Watzenig. Development of a Scenario Simulation Platform to Support Autonomous Driving Verification. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–7, nov 2019.
- [26] CVEDIA. GitHub - Cvedia/syncity-redis: Syncity World Simulator Redistributables. <https://github.com/Cvedia/syncity-redis>, 2018. Accessed: 03-02-2020.
- [27] Drew Gray. Introducing Voyage Deepdrive - Voyage. <https://news.voyage.auto/introducing-voyage-deepdrive-69b3cf0f0be6>. Accessed: 03-02-2020.
- [28] Voyage DeepDrive. Deepdrive from Voyage - Push the state-of-the-art in self-driving. <https://deepdrive.voyage.auto/>. Accessed: 03-02-2020.
- [29] Deepdrive. GitHub - deepdrive/deepdrive: Deepdrive is a simulator that allows anyone with a PC to push the state-of-the-art in self-driving. <https://github.com/deepdrive/deepdrive>, 2018. Accessed: 03-02-2020.
- [30] 20tab srl. GitHub - 20tab/UnrealEnginePython: Embed Python in Unreal Engine 4. <https://github.com/20tab/UnrealEnginePython>, 2016. Accessed: 03-02-2020.
- [31] rFPro. Driving Simulation for autonomous driving, ADAS, vehicle dynamics and motor-sport. <http://www.rfpro.com/>. Accessed: 03-02-2020.
- [32] CARLA. carla-simulator / carla-content — Bitbucket. <https://bitbucket.org/carla-simulator/carla-content/src/master/>, 2018. Accessed: 04-02-2020.
- [33] CARLA. CARLA AD Challenge. <https://carlachallenge.org/>. Accessed: 15-06-2020.
- [34] ASAM e.V. ASAM OpenSCENARIO. <https://www.asam.net/standards/detail/openscenario/>. Published: 13-03-2020. Accessed: 15-04-2020.
- [35] The Qt Company. Qt 4.8. <https://doc.qt.io/archives/qt-4.8/index.html>. Accessed: 03-07-2020.

Appendix A

Absolute Target Speed Action

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <OpenSCENARIO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="OpenSCENARIO.xsd">
4   <FileHeader revMajor="1" revMinor="0" date="2020-04-21T10:00:00"
5     description="FollowVehicle" author="Miguel"/>
6   <ParameterDeclarations/>
7   <CatalogLocations>
8     <VehicleCatalog>
9       <Directory path="Catalogs/VehicleCatalog"/>
10    </VehicleCatalog>
11    <EnvironmentCatalog>
12      <Directory path="Catalogs/EnvironmentCatalog"/>
13    </EnvironmentCatalog>
14    </CatalogLocations>
15    <RoadNetwork>
16      <LogicFile filepath="Town03"/>
17    </RoadNetwork>
18    <Entities>
19      <ScenarioObject name="Lead">
20        <CatalogReference catalogName="VehicleCatalog" entryName="audi.a2">
21          <ParameterAssignments>
22            <ParameterAssignment parameterRef="color" value="255,0,0"/>
23          </ParameterAssignments>
24        </CatalogReference>
25      </ScenarioObject>
26    </Entities>
27    <Storyboard>
28      <Init>
29        <Actions>
30          <GlobalAction>
31            <EnvironmentAction>
32              <CatalogReference catalogName="EnvironmentCatalog" entryName="SunnyDay"/>
33            </EnvironmentAction>
34          </GlobalAction>
35          <GlobalAction>
```

```

34 <EntityAction entityRef="Lead">
35 <AddEntityAction>
36 <Position>
37 <WorldPosition x="40" y="7" z="2"/>
38 </Position>
39 </AddEntityAction>
40 </EntityAction>
41 </GlobalAction>
42 </Actions>
43 </Init>
44 <Story name="AccelerateStory">
45 <Act name="AccelerateAct">
46 <ManeuverGroup maximumExecutionCount="1" name="LeadAccelerate">
47 <Actors selectTriggeringEntities="false">
48 <EntityRef entityRef="Lead"/>
49 </Actors>
50 <Maneuver name="AccelerateManeuver">
51 <Event name="GainSpeed" maximumExecutionCount="1" priority="parallel">
52 <Action name="LongitudinalSpeed">
53 <PrivateAction>
54 <LongitudinalAction>
55 <SpeedAction>
56 <SpeedActionDynamics dynamicsDimension="time"
    dynamicsShape="sinusoidal" value="3"/>
57 <SpeedActionTarget>
58 <AbsoluteTargetSpeed value="5"/>
59 </SpeedActionTarget>
60 </SpeedAction>
61 </LongitudinalAction>
62 </PrivateAction>
63 </Action>
64 <StartTrigger>
65 <ConditionGroup>
66 <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
67 <ByValueCondition>
68 <SimulationTimeCondition value="2" rule="greaterThan"/>
69 </ByValueCondition>
70 </Condition>
71 </ConditionGroup>
72 </StartTrigger>
73 </Event>
74 </Maneuver>
75 </ManeuverGroup>
76 <StartTrigger>
77 <ConditionGroup>
78 <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
79 <ByValueCondition>
80 <SimulationTimeCondition value="1" rule="greaterThan"/>
81 </ByValueCondition>

```

```

82     </Condition>
83   </ConditionGroup>
84 </StartTrigger>
85 </Act>
86 </Story>
87 <StopTrigger>
88   <ConditionGroup>
89     <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
90       <ByValueCondition>
91         <SimulationTimeCondition value="15" rule="greaterThan"/>
92       </ByValueCondition>
93     </Condition>
94   </ConditionGroup>
95 </StopTrigger>
96 </Storyboard>
97 </OpenSCENARIO>

```

Listing A.1: OpenSCENARIO script controlling the speed of an entity.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <OpenSCENARIO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="OpenSCENARIO.xsd">
4   <FileHeader revMajor="1" revMinor="0" date="2020-02-21T10:00:00"
5     description="Vehicle Catalog" author="miguel"/>
6   <Catalog name="VehicleCatalog">
7     <Vehicle name="audi.a2" vehicleCategory="car">
8       <ParameterDeclarations>
9         <ParameterDeclaration name="color" parameterType="string" value="0,0,0"/>
10      </ParameterDeclarations>
11      <BoundingBox>
12        <Center x="1.0" y="1.0" z="1.0"/>
13        <Dimensions width="2.0" length="2.0" height="2.0"/>
14      </BoundingBox>
15      <Performance maxSpeed="100.0" maxDeceleration="10.0"
16        maxAcceleration="200.0"/>
17      <Axles>
18        <FrontAxle maxSteering="35.0" wheelDiameter="65.0" trackWidth="1.8"
19          positionX="2.0" positionZ="2.0"/>
20        <RearAxle maxSteering="35.0" wheelDiameter="65.0" trackWidth="1.8"
21          positionX="-2.0" positionZ="-2.0"/>
22      </Axles>
23      <Properties>
24        <Property name="torque_curve" value="[[0,1000],[1000,1000]]"/>
25        <Property name="color" value="$color"/>
26      </Properties>
27    </Vehicle>
28  </Catalog>
29 </OpenSCENARIO>

```

Listing A.2: OpenSCENARIO Vehicle catalog used in the script present in listing A.1.

```

1 <?xml version="1.0" encoding="UTF-8"?>

```

```

2 <OpenSCENARIO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="OpenSCENARIO.xsd">
3 <FileHeader revMajor="1" revMinor="0" date="2020-06-21T10:00:00"
  description="Environment Catalog" author="Miguel"/>
4 <Catalog name="EnvironmentCatalog">
5 <Environment name="SunnyDay">
6 <TimeOfDay animation="false" dateTime="2002-05-30T09:30:10+06:00"/>
7 <Weather cloudState="free">
8 <Sun intensity="1.0" azimuth="0.0" elevation="1.571"/>
9 <Fog visualRange="100000.0"/>
10 <Precipitation precipitationType="dry" intensity="0.0"/>
11 </Weather>
12 <RoadCondition frictionScaleFactor="0">
13 <Properties>
14 <Property name="wetness" value="0"/>
15 <Property name="precipitationDeposits" value="0"/>
16 <Property name="fogDensity" value="0.0"/>
17 <Property name="windIntensity" value="0"/>
18 </Properties>
19 </RoadCondition>
20 </Environment>
21 </Catalog>
22 </OpenSCENARIO>

```

Listing A.3: OpenSCENARIO Environment catalog used in the script present in listing A.1.

```

1 Time (s), Desired velocity (m/s), Vehicle's velocity (m/s)
2 0.0,0.0,0.0
3 0.0133389467373,0.0349209969889,0.0
4 0.0272630723193,0.0713721324163,0.0
5 0.0411896528676,0.107825899763,0.0
6 0.0539110759273,0.14112012407,0.00823077942435
7 0.0671811895445,0.175843674341,0.00490404422022
8 0.0803208621219,0.210217543922,0.00426451337279
9 0.0938969636336,0.245722657355,0.00287009890091
10 0.107899989933,0.282331264794,0.00238099604246
11 0.120805705898,0.316057740467,0.00194950460349
12 0.134303274564,0.35131543823,0.000454604762473
13 0.148279340006,0.387804529124,0.000321369850308
14 0.162361294031,0.424549068003,0.000302639406614
15 0.175256241113,0.45817611265,0.000610731779335
16 0.188272711821,0.492098875217,0.00255809502798
17 0.202097726054,0.528103749107,0.0025534689035
18 0.215275393799,0.562396988638,0.00165589952983
19 0.228244372644,0.596121008214,0.0027277824559
20 0.243112638593,0.634749944495,0.00304512174114
21 0.255933191627,0.66802795893,0.00056692238411
22 0.279195360839,0.728331135917,0.00361146939302
23 0.290701967664,0.758120651831,0.00457460911342
24 0.303168282844,0.790363670658,0.00442345462217

```

25 0.316234798171,0.824122885033,0.00303045026346
26 0.329268353991,0.857758515855,0.00216394165027
27 0.342336017638,0.891442063047,0.00217833075986
28 0.356884950772,0.928894601111,0.00209110292176
29 0.370289776474,0.963354262579,0.000201670372465
30 0.384201813489,0.999067599963,0.00188425654781
31 0.397306900471,1.03266097551,0.00118969824478
32 0.410427830182,1.06624625118,0.00229255082583
33 0.42419298552,1.10142641342,0.00214437787967
34 0.438253894448,1.13730335894,0.001205221867
35 0.451800595969,1.17181000229,0.00082077617932
36 0.465295124799,1.20612513696,0.0
37 0.478905294091,1.24067333691,0.000478253141768
38 0.491806724109,1.27336432388,0.000458895976814
39 0.505320490338,1.30754457816,0.00149599534906
40 0.518288217485,1.34028221204,0.000680188531022
41 0.532025345601,1.37489480398,0.00145845095991
42 0.546188207343,1.41050559981,0.000424166154335
43 0.559318798594,1.44345162733,9.02201705808e-05
44 0.573265828192,1.47837142482,0.00149059758584
45 0.586923347786,1.51248997431,0.000361665755875
46 0.600752011873,1.54695725644,0.000838725600669
47 0.613032856956,1.57749873634,0.000628778281761
48 0.627005995251,1.6121694162,0.00262027931508
49 0.640328641981,1.64514573443,0.00023289586725
50 0.653899269179,1.67865354303,0.00116570049973
51 0.674974026158,1.73052148882,0.000777989015265
52 0.686380226165,1.75850627168,0.00386688528784
53 0.699807613157,1.79136952444,0.00252789710701
54 0.712978891097,1.82351993788,0.000184510057257
55 0.725333250128,1.85359749626,0.000247255201619
56 0.739594004117,1.88821978844,0.00126740081533
57 0.752888347954,1.92040104987,0.00130788185571
58 0.766307923011,1.95279108024,0.000245638056153
59 0.780226326548,1.98628322859,0.00010894448413
60 0.794334349222,2.02012399673,0.000527257859261
61 0.808147112839,2.05314974732,0.00102601068585
62 0.821514971554,2.08500950585,0.118108965926
63 0.83433504682,2.11546779179,0.220716510682
64 0.847878825851,2.14754190195,0.318235207976
65 0.861235762015,2.17906776806,0.421594818399
66 0.875804369338,2.21333192002,0.524844079565
67 0.889185459353,2.2446897079,0.637951552529
68 0.902916752733,2.27675361105,0.743125543248
69 0.915802539326,2.30673616298,0.851079283866
70 0.944122658111,2.37226030291,0.953100837766
71 0.959341748618,2.40725808984,1.17727575654
72 0.970181431621,2.43209192148,1.29795982468
73 0.982298287563,2.45975899838,1.38564762491

74 0.995991856791,2.49090704171,1.48214791388
75 1.00930260308,2.52106157985,1.59118495763
76 1.02307051793,2.55212294479,1.69788099466
77 1.03688948229,2.58316611298,1.80809532034
78 1.05004879646,2.61260174651,1.91889835799
79 1.06327710673,2.64206670307,2.02438746825
80 1.07735094894,2.67327581703,2.12953679094
81 1.09026884474,2.70179385942,2.23993336002
82 1.10464385338,2.73338331919,2.34026607458
83 1.11732525751,2.76112253737,2.44876570353
84 1.13102698699,2.79095675052,2.54266675674
85 1.14485808369,2.8209269549,2.63997477634
86 1.1589416014,2.85129209358,2.73400543622
87 1.17207643017,2.87947208867,2.82421480227
88 1.18487843312,2.9068069756,2.90267906439
89 1.19871120993,2.93619593317,2.97294434982
90 1.21219811495,2.96470176885,3.04129825889
91 1.22517223936,2.99198429332,3.09998574703
92 1.23932812084,3.02159431064,3.14414271753
93 1.25326693151,3.05058808638,3.17594994721
94 1.26710239705,3.07920621147,3.19255630814
95 1.28027810622,3.10630946029,3.19854555019
96 1.29424080532,3.13487022001,3.19784921171
97 1.30682488717,3.16046750023,3.19409664061
98 1.32777451165,3.20277623566,3.19063298692
99 1.33998024464,3.22724893536,3.19636374615
100 1.35257264599,3.2523587586,3.20296553215
101 1.36589015741,3.27876062742,3.20963601084
102 1.37974866014,3.30606572024,3.21500641985
103 1.3921540929,3.33036024963,3.21911746753
104 1.40591239464,3.35713983303,3.22194869024
105 1.41727716662,3.37912924489,3.22482328461
106 1.43122499529,3.40595293169,3.22738819836
107 1.44447224122,3.43126112125,3.23106605612
108 1.45828713384,3.45747790798,3.23508465894
109 1.47287651896,3.48496803449,3.24013618887
110 1.4862858681,3.51005535627,3.24672645288
111 1.49940268788,3.53442798812,3.25409643975
112 1.51290605403,3.55934470222,3.26270989806
113 1.52659555525,3.58442324505,3.27317772823
114 1.54026430473,3.60928003313,3.2855191851
115 1.55384910107,3.63380098067,3.29961354236
116 1.56780759431,3.65880494474,3.31538971485
117 1.58129663114,3.68278229168,3.33341912594
118 1.59508329444,3.7070988384,3.35254632368
119 1.6076269215,3.72905515069,3.37374620647
120 1.62128324714,3.75277620051,3.3944199522
121 1.63455691468,3.77564869911,3.41829066448
122 1.64764336962,3.79802007138,3.44277034918

123 1.66128133889, 3.82114449546, 3.46797980897
124 1.67370260041, 3.84203636514, 3.49526005283
125 1.68743124884, 3.86493807978, 3.52089681216
126 1.70155810192, 3.88829557552, 3.54990442263
127 1.71444762591, 3.90942168542, 3.5803813178
128 1.72745517921, 3.93056072439, 3.60860412638
129 1.74915770348, 3.96542391938, 3.63734066062
130 1.76046609785, 3.98338722834, 3.68546346774
131 1.77324972954, 4.00352574008, 3.71045781541
132 1.78706560191, 4.02508867161, 3.73870810189
133 1.80011920165, 4.04526839367, 3.76910058924
134 1.81335580163, 4.06553804651, 3.79739342194
135 1.827273909, 4.08664067229, 3.82558907063
136 1.84093622584, 4.10714439038, 3.85463439341
137 1.85484932177, 4.12780845987, 3.88247485186
138 1.86923535075, 4.14894457248, 3.91007901255
139 1.88224283792, 4.16785270017, 3.93778842566
140 1.89625107218, 4.18799931199, 3.962093485
141 1.91026033554, 4.20792207513, 3.98737069353
142 1.9240065515, 4.22725068941, 4.01167267349
143 1.93617795315, 4.24418218928, 4.03462517223
144 1.94924444798, 4.26216697007, 4.05419303442
145 1.96287891269, 4.28072079737, 4.07440056079
146 1.9772654064, 4.30006142117, 4.0946442032
147 1.99092895258, 4.31820424955, 4.11516556623
148 2.00519182347, 4.33690709204, 4.13393223694
149 2.01862485893, 4.35430061864, 4.15282796054
150 2.0329628624, 4.37262824112, 4.17003407417
151 2.04521742091, 4.38809740074, 4.18780587187
152 2.0592460297, 4.40558415392, 4.20256530943
153 2.07291005179, 4.42238794181, 4.21900479706
154 2.08594284859, 4.43820455072, 4.23460735282
155 2.09909840766, 4.4539605421, 4.24914287169
156 2.11294816341, 4.47031958533, 4.26353567896
157 2.1263399655, 4.48591416205, 4.27842107945
158 2.14020980243, 4.50183288552, 4.29259219837
159 2.15291399229, 4.51620545661, 4.30708233231
160 2.16641067434, 4.53125566149, 4.32023127315
161 2.18624229077, 4.55295922703, 4.33409085393
162 2.19909316301, 4.56676114245, 4.35434318834
163 2.21123496816, 4.57961156251, 4.36743131182
164 2.22512091324, 4.59408099775, 4.3798382166
165 2.23853934649, 4.60783255746, 4.39398424593
166 2.25299523119, 4.62239279143, 4.40761777152
167 2.26533882972, 4.6346159192, 4.42226754959
168 2.27931995224, 4.6482267402, 4.43477182789
169 2.29318342078, 4.66147706487, 4.44890911212
170 2.30637667701, 4.67385873404, 4.46291088589
171 2.32028337009, 4.68666848794, 4.47619645536

172 2.33380252589,4.69888308027,4.49015258729
173 2.34762029257,4.71112416067,4.50365464738
174 2.36071947776,4.72250096551,4.51738424669
175 2.37389147282,4.73371699494,4.5303303821
176 2.38723642658,4.74485067803,4.54325599243
177 2.40087755304,4.75599202391,4.5562416869
178 2.41485114489,4.76715333724,4.56937760755
179 2.42814403027,4.77753410003,4.58267330211
180 2.44067497551,4.78710793154,4.59516019893
181 2.45486230031,4.79769849928,4.60677563225
182 2.46855927352,4.80767185664,4.61971504443
183 2.48132713977,4.81674601801,4.63198013393
184 2.49425996933,4.82571795632,4.6431992632
185 2.50687888917,4.83425883813,4.65434670165
186 2.52075769845,4.84340873818,4.66499691265
187 2.53426664323,4.85206916393,4.67642762567
188 2.54827529937,4.8607935542,4.68726797934
189 2.56092490256,4.86844686986,4.69818904985
190 2.57447613869,4.87640874013,4.70776590051
191 2.58793068584,4.88407092304,4.71773883437
192 2.60190256592,4.89177116534,4.72734710017
193 2.61526123527,4.89888864001,4.7370021569
194 2.62905739341,4.90598762857,4.74593469662
195 2.64227780234,4.91255016616,4.75485108874
196 2.65527269058,4.91877135483,4.76310323647
197 2.66922364477,4.9251967791,4.77092291676
198 2.68269111495,4.93115023042,4.77898433047
199 2.69603032991,4.93680528013,4.78646633589
200 2.70918914117,4.94214788858,4.793592197
201 2.72290393803,4.94746654166,4.80035137918
202 2.73711915873,4.95270999695,4.80710815534
203 2.75048889499,4.95739120807,4.81380780202
204 2.76426606998,4.96196092,4.8198339868
205 2.77665534802,4.9658497904,4.82577394328
206 2.79067050852,4.9699970972,4.83088683953
207 2.80329207983,4.97350298269,4.83640603213
208 2.81756617501,4.97720614769,4.84114207095
209 2.83091297094,4.98041722894,4.84623086257
210 2.84493535012,4.98352883679,4.8507425753
211 2.85822512116,4.98622992265,4.85522821584
212 2.87184522301,4.98874763296,4.85924793221
213 2.88534677587,4.99099301655,4.86312412245
214 2.89872720093,4.99297217213,4.86671899987
215 2.91225006711,4.99472339678,4.87005877395
216 2.9252958959,4.996175534,4.87321066095
217 2.93872907385,4.99742718301,4.87604073418
218 2.95282591973,4.99847481665,4.87873768176
219 2.96628783923,4.9992210687,4.881331745
220 2.97990790755,4.99972331629,4.88358568836


```
221 2.99388121068,4.99997433934,4.88564514626
```

Listing A.4: Results obtained with the script present in listing [A.1](#).

Appendix B

Relative Target Speed Action

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <OpenSCENARIO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="OpenSCENARIO.xsd">
4   <FileHeader revMajor="1" revMinor="0" date="2020-04-21T10:00:00"
5     description="FollowVehicle" author="Miguel"/>
6   <ParameterDeclarations/>
7   <CatalogLocations>
8     <VehicleCatalog>
9       <Directory path="Catalogs/VehicleCatalog"/>
10    </VehicleCatalog>
11    <EnvironmentCatalog>
12      <Directory path="Catalogs/EnvironmentCatalog"/>
13    </EnvironmentCatalog>
14    </CatalogLocations>
15    <RoadNetwork>
16      <LogicFile filepath="Town03"/>
17    </RoadNetwork>
18    <Entities>
19      <ScenarioObject name="Lead">
20        <CatalogReference catalogName="VehicleCatalog" entryName="audi.a2">
21          <ParameterAssignments>
22            <ParameterAssignment parameterRef="color" value="255,0,0"/>
23          </ParameterAssignments>
24        </CatalogReference>
25      </ScenarioObject>
26      <ScenarioObject name="Follow">
27        <CatalogReference catalogName="VehicleCatalog" entryName="audi.a2"/>
28      </ScenarioObject>
29    </Entities>
30    <Storyboard>
31      <Init>
32        <Actions>
33          <GlobalAction>
34            <EnvironmentAction>
35              <CatalogReference catalogName="EnvironmentCatalog" entryName="SunnyDay"/>
36            </EnvironmentAction>
37          </GlobalAction>
38        </Actions>
39      </Init>
40    </Storyboard>
41  </OpenSCENARIO>
```

```

34 </EnvironmentAction>
35 </GlobalAction>
36 <GlobalAction>
37   <EntityAction entityRef="Follow">
38     <AddEntityAction>
39       <Position>
40         <WorldPosition x="30" y="7" z="2"/>
41       </Position>
42     </AddEntityAction>
43   </EntityAction>
44 </GlobalAction>
45 <GlobalAction>
46   <EntityAction entityRef="Lead">
47     <AddEntityAction>
48       <Position>
49         <WorldPosition x="40" y="7" z="2"/>
50       </Position>
51     </AddEntityAction>
52   </EntityAction>
53 </GlobalAction>
54 </Actions>
55 </Init>
56 <Story name="Following">
57   <Act name="AccelerateAndFollow">
58     <ManeuverGroup maximumExecutionCount="1" name="LeadAccelerate">
59       <Actors selectTriggeringEntities="false">
60         <EntityRef entityRef="Lead"/>
61       </Actors>
62       <Maneuver name="Accelerate">
63         <Event name="GainSpeed" maximumExecutionCount="1" priority="parallel">
64           <Action name="LongitudinalSpeed">
65             <PrivateAction>
66               <LongitudinalAction>
67                 <SpeedAction>
68                   <SpeedActionDynamics dynamicsDimension="rate" dynamicsShape="step"
69                     value="0"/>
70                 <SpeedActionTarget>
71                   <AbsoluteTargetSpeed value="10"/>
72                 </SpeedActionTarget>
73               </SpeedAction>
74             </LongitudinalAction>
75           </PrivateAction>
76         </Action>
77       <StartTrigger>
78         <ConditionGroup>
79           <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
80             <ByValueCondition>
81               <SimulationTimeCondition value="2" rule="greaterThan"/>

```

```
82     </Condition>
83   </ConditionGroup>
84 </StartTrigger>
85 </Event>
86 </Maneuver>
87 </ManeuverGroup>
88 <ManeuverGroup maximumExecutionCount="1" name="Follow">
89   <Actors selectTriggeringEntities="false">
90     <EntityRef entityRef="Follow"/>
91   </Actors>
92   <Maneuver name="Accelerate">
93     <Event name="GainSpeed" maximumExecutionCount="1" priority="parallel">
94       <Action name="LongitudinalSpeed">
95         <PrivateAction>
96           <LongitudinalAction>
97             <SpeedAction>
98               <SpeedActionDynamics dynamicsDimension="rate" dynamicsShape="linear"
99                 value="2"/>
100              <SpeedActionTarget>
101                <RelativeTargetSpeed entityRef="Lead" continuous="true"
102                  speedTargetValueType="delta" value="1"/>
103              </SpeedActionTarget>
104            </SpeedAction>
105          </LongitudinalAction>
106        </PrivateAction>
107      </Action>
108    <StartTrigger>
109      <ConditionGroup>
110        <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
111          <ByValueCondition>
112            <SimulationTimeCondition value="2" rule="greaterThan"/>
113          </ByValueCondition>
114        </Condition>
115      </ConditionGroup>
116    </StartTrigger>
117  </Event>
118 </Maneuver>
119 </ManeuverGroup>
120 <StartTrigger>
121   <ConditionGroup>
122     <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
123       <ByValueCondition>
124         <SimulationTimeCondition value="1" rule="greaterThan"/>
125       </ByValueCondition>
126     </Condition>
127   </ConditionGroup>
128 </StartTrigger>
</Act>
</Story>
```

```

129 <StopTrigger>
130 <ConditionGroup>
131 <Condition name="WhenSimulationStarts" delay="0" conditionEdge="none">
132 <ByValueCondition>
133 <SimulationTimeCondition value="13" rule="greaterThan"/>
134 </ByValueCondition>
135 </Condition>
136 </ConditionGroup>
137 </StopTrigger>
138 </Storyboard>
139 </OpenSCENARIO>

```

Listing B.1: OpenSCENARIO script two entities: leader and follower.

```

1 Time (s), Leader velocity (m/s), Follower velocity (m/s)
2 0.0,0.0,0.0
3 0.0101852463558,0.0,0.0
4 0.0214531207457,0.00444875066807,0.0
5 0.0314681893215,0.00468035530011,0.00548037975358
6 0.0416690856218,0.00256189915597,0.00326754063605
7 0.0517456987873,0.00215150658489,0.00277457101855
8 0.0627213818952,0.00150381599659,0.00204744558495
9 0.072867157869,0.00189958407279,0.00237696518623
10 0.0830021090806,0.000644143685651,0.001007258831
11 0.0936365677044,0.000447506294269,0.00065612985352
12 0.104310875759,0.00054264029865,0.000769942726349
13 0.114423533902,0.000458590509859,0.000534864666558
14 0.12438405212,0.000950304416634,0.000740572489641
15 0.135189046152,0.00124244853733,0.00104665914474
16 0.145467417315,0.000664372780179,0.000552018174221
17 0.155487939715,0.00129310629043,0.00115568427605
18 0.165656557307,0.00163486506555,0.00151560632123
19 0.177131655626,0.0015648331087,0.00146456714346
20 0.187179675326,0.000518439792598,0.000493302613352
21 0.197241927497,0.00195580412815,0.00189943956746
22 0.20734719187,0.00194960051108,0.00189801418978
23 0.218065385707,0.00190359438322,0.00186992062239
24 0.22829997912,0.00131361841774,0.00129359242145
25 0.23836301174,0.00182914662262,0.00181780219385
26 0.2489733072,0.00197391304614,0.00197018459774
27 0.259786355309,0.00141200386843,0.00141873660299
28 0.270161435008,0.00123184917865,0.00124490956021
29 0.280372489244,0.00168803241575,0.00170833802258
30 0.291188934818,0.00181534062078,0.00183892475158
31 0.301369450055,0.0011769216614,0.00120660554439
32 0.311606050469,0.00181072611358,0.00184207458265
33 0.321976625361,0.00169227179511,0.00172747404448
34 0.33312301524,0.00150504342007,0.0015423722903
35 0.343225888908,0.000707635261393,0.000745827053444
36 0.353359749541,0.00177186413532,0.00181330687211

```

37 0.363551989198,0.00165871127805,0.00169942947341
38 0.374089187942,0.00152565693087,0.00156721660345
39 0.384301761165,0.00112318291855,0.0011662705031
40 0.394455496222,0.00140952727063,0.00145314342315
41 0.405685180798,0.0014012021143,0.00144291341916
42 0.415593987331,0.000323598430318,0.000358490774883
43 0.425800034776,0.0,0.001673393996
44 0.435802396387,5.57650201389e-05,0.0
45 0.44687080197,0.000290704033643,0.000421815403618
46 0.456834558398,0.000709423573051,0.000592809522242
47 0.466938488185,0.000605970508126,0.000699352745871
48 0.476923340932,0.000510910490396,0.000586543746778
49 0.488054843619,0.000665094512216,0.000725931735617
50 0.498040935025,0.000486772484013,0.000436214277324
51 0.508144436404,0.000851609029948,0.000888034394931
52 0.51868495252,0.000739281614122,0.000764664265426
53 0.528808468021,0.000333794242007,0.000349161805182
54 0.538869474083,0.000779305903464,0.000788746029219
55 0.548994198442,0.000832402139633,0.00083733436166
56 0.560303625651,0.00075909139777,0.000755838949173
57 0.570344315842,0.000489005444059,0.000498049441981
58 0.580453502014,0.000973015649992,0.000962580314827
59 0.590466281399,0.000875315866524,0.000855877584994
60 0.601533863693,0.00093038117683,0.000918234148114
61 0.611529598944,0.000217103422987,0.000239961086974
62 0.621508173645,0.00100035356196,0.000982921594969
63 0.631826637313,0.000968746207346,0.000937996397596
64 0.643184506334,0.000586469460556,0.000567925388474
65 0.653378718533,0.000517113084314,0.000550684826155
66 0.663487684913,0.000822498096809,0.000799058662077
67 0.674271411262,0.000868526872045,0.000832722345344
68 0.684521605261,0.000173961841748,0.000158883050669
69 0.694632834755,0.00107554872081,0.000682556075408
70 0.704627230763,0.10819041921,0.000786948572658
71 0.715763695538,0.189302753369,0.000847219362879
72 0.725896741264,0.279696065926,0.000383400837886
73 0.735853050835,0.36227845962,0.000743018448811
74 0.746014310047,0.443195173566,0.00085932650293
75 0.756893800572,0.525616329741,0.000599065202022
76 0.767114754766,0.613656578342,0.000189600393986
77 0.777013535611,0.696576552328,0.000554428797299
78 0.787705444731,0.776774208889,0.000826843641455
79 0.798303553835,0.8631125104,8.48322095788e-05
80 0.808342920616,0.948816825208,0.000125751794415
81 0.818409201689,1.03008897185,0.000666137145628
82 0.829272043891,1.11146096795,0.000586447633998
83 0.839597974904,1.19908004472,0.000268617424391
84 0.849914004095,1.2826405275,0.0
85 0.859908293001,1.36603185666,6.93050595438e-05

86 0.871367366984,1.4469051933,0.00039644813615
87 0.881545700133,1.53921088605,0.00112022921899
88 0.891747926362,1.6218189973,0.000374310622908
89 0.901713685133,1.70436752836,0.000333216235759
90 0.912466392852,1.78508685047,0.00054474565306
91 0.922544750385,1.87196361548,0.000277221477454
92 0.932818472385,1.95373956162,0.00045455788071
93 0.943328220397,2.03693474783,0.000233527918012
94 0.954030952416,2.12204936043,4.11427207802e-05
95 0.964175852016,2.20875730397,0.00018817239692
96 0.97439484857,2.29115425496,0.000415296314993
97 0.985275756568,2.37403098249,0.000313907655389
98 0.995400841348,2.46213630795,0.000374763735413
99 1.00562181603,2.54448831172,0.000451022664698
100 1.01582237985,2.62743179719,0.000320537169026
101 1.02729046904,2.71025030891,0.000318538519576
102 1.03767795302,2.80300591596,0.00101416355099
103 1.04785332084,2.88763637457,0.000264103628811
104 1.05810997635,2.97032560233,0.000456141785852
105 1.06886180304,3.05360786972,0.000337287152719
106 1.0789665496,3.14080059373,0.000189738885558
107 1.08917099703,3.22304595599,0.000496148776033
108 1.10030929465,3.30590860678,0.00035281700417
109 1.11060808133,3.39611059168,0.000627889732366
110 1.12060373649,3.47997969268,0.000324535718748
111 1.1308258744,3.56123818563,0.000590846247411
112 1.1419045236,3.64419336066,0.000315675858975
113 1.15194360912,3.73390316226,0.000584857121425
114 1.16194715723,3.81569163972,0.00054820054873
115 1.17218803987,3.89691108767,0.000528202829764
116 1.18328864966,3.97997211694,0.000245191475517
117 1.19353748392,4.06981396884,0.000652171297776
118 1.20357047673,4.15321732415,0.000306117983642
119 1.2137344759,4.23467732027,0.000486491451834
120 1.2244896125,4.31709102635,0.000311208287589
121 1.23459036835,4.40414444685,0.000311558335734
122 1.24454221688,4.48622090587,0.000385118970509
123 1.25525159296,4.56693419315,0.000486687875674
124 1.265866125,4.65350791185,0.0
125 1.2759653423,4.73952551424,6.4917544416e-05
126 1.28606228344,4.82146279561,0.000566129490958
127 1.29704027809,4.90322084573,0.000499237873223
128 1.30728529114,4.99182449131,0.000449614153933
129 1.31744238734,5.07493473883,0.000346616853739
130 1.32750142459,5.15963925222,0.000386582771138
131 1.33895243332,5.24056408123,0.000431797558029
132 1.34927295893,5.33344578301,0.00105220791206
133 1.35939040873,5.41645321444,0.000256876134724
134 1.36996113602,5.49849424871,0.000420784870456

135 1.38060795143,5.58375973465,8.1000450046e-05
136 1.39081444591,5.6700328273,0.000151890780985
137 1.40090753417,5.7523296171,0.000301382017949
138 1.41227298602,5.8340342106,0.000371549846735
139 1.42248087563,5.92581583132,0.000979129800826
140 1.43270243425,6.00829675407,0.000336985143925
141 1.44277561456,6.0907919976,0.000281423430871
142 1.4536026055,6.17215094768,0.000386834503471
143 1.4637430096,6.2595385608,0.000414194240081
144 1.47390666977,6.3413395517,0.000321083280933
145 1.48449407797,6.42331316247,0.000256300770852
146 1.49505705852,6.50868948926,0.000198735220708
147 1.50516722444,6.59381738665,0.000154605168996
148 1.51524663996,6.6752602865,0.000307900523567
149 1.52589271218,6.75643711345,0.000297942497455
150 1.53616375476,6.84215534348,0.000304332010233
151 1.54633726366,6.92479892693,0.000104226359607
152 1.55651679635,7.00663427322,0.000180662003755
153 1.56794184446,7.08848583436,0.000149802851529
154 1.57822059188,7.180331438,0.00115446290712
155 1.58832792472,7.26288919988,0.000178064870555
156 1.59843370877,7.3440584142,0.00032163587109
157 1.60927196778,7.4251809007,0.000284065402072
158 1.61932323314,7.51215371179,0.000486647561992
159 1.62947107013,7.59275931149,0.000356352906291
160 1.64020023216,7.67411974854,0.000712971018342
161 1.65084921289,7.76010408442,0.0486715859468
162 1.66104972828,7.84540177692,0.143746663887
163 1.67142790277,7.92707169083,0.197472682634
164 1.68238385115,8.01013206764,0.231270475239
165 1.69257657882,8.09777962756,0.275708084097
166 1.70290046651,8.17927789017,0.327339570457
167 1.71312850527,8.26179468287,0.38238939806
168 1.72410249151,8.34350943743,0.426704306254
169 1.73446521908,8.43114843037,0.46145659919
170 1.74452820886,8.51386359824,0.504141951115
171 1.75474177487,8.59415261933,0.535642627168
172 1.76558323111,8.67561085097,0.568999492371
173 1.77572599985,8.76203869234,0.603188776281
174 1.78576716222,8.84285701527,0.640012940496
175 1.79675117042,8.9228332469,0.671946269535
176 1.8068241924,9.0102805816,0.704532540328
177 1.81675207801,9.09044088078,0.737265289648
178 1.8266807925,9.16941195062,0.767929513646
179 1.83786588814,9.2483477387,0.797386554549
180 1.84814009909,9.33714835956,0.828712010931
181 1.85830400884,9.41855322472,0.858368837376
182 1.8682911247,9.49883752575,0.887648342998
183 1.87975124829,9.57734894755,0.916004826372

184 1.88985588495,9.66684482233,0.947913215663
185 1.89989438932,9.74505203226,0.976050079549
186 1.9099415401,9.82198704526,1.00388945895
187 1.92085818667,9.89809331985,1.03149641464
188 1.93078346271,9.979609811,1.06129270827
189 1.94093044661,10.0523248254,1.08833973174
190 1.95151073672,10.1251199585,1.11581386505
191 1.96225494891,10.1991300866,1.14444480174
192 1.97244994622,10.2719519443,1.173398288
193 1.98271530401,10.3319001351,1.20094194356
194 1.99344688561,10.3835504025,1.22858479211
195 2.00379521679,10.4273195076,1.25755907459
196 2.01403263211,10.4582740719,1.28543952418
197 2.02417278662,10.4777578292,1.31313684979
198 2.03562143166,10.4853263653,1.34053603947
199 2.04590068292,10.4807546372,1.37158704852
200 2.05592766032,10.4655646688,1.39943783185
201 2.06606035959,10.44190606,1.42675170034
202 2.07690636255,10.412232622,1.45433801542
203 2.0869796155,10.3754943654,1.48400095087
204 2.09706833679,10.338310726,1.51152273154
205 2.10768628027,10.3030811535,1.53967471487
206 2.11825539265,10.267601708,1.57051560476
207 2.12850626092,10.2290733411,1.6022910692
208 2.13878340926,10.193154203,1.63354523102
209 2.14953037631,10.1590002927,1.66528863994
210 2.1596740121,10.125744722,1.69840789803
211 2.16973007005,10.0974596817,1.72970089788
212 2.17989946902,10.0720385053,1.76029943018
213 2.19113270193,10.0492217999,1.79104647344
214 2.20138645545,10.0272929492,1.82463777022
215 2.21142328717,10.0098501754,1.85523854022
216 2.22175786272,9.99451501558,1.88486636758
217 2.23265825212,9.98382234052,1.91526304344
218 2.24278429803,9.97395077568,1.94679911935
219 2.25296525564,9.96506435941,1.97568811286
220 2.26381858252,9.95504020948,2.00435758379
221 2.27429435402,9.94362369409,2.03480441663
222 2.2844407009,9.93194683005,2.06431985957
223 2.29467289057,9.92014217832,2.09258201574
224 2.30571839772,9.90791028149,2.12098362254
225 2.31578926928,9.89447484539,2.15174021087
226 2.32601545844,9.88211230159,2.17969518676
227 2.336225315,9.86942169312,2.20817545896
228 2.3475740971,9.85677877047,2.23652152478
229 2.35767484549,9.84283884555,2.26812413537
230 2.36770037282,9.83078911383,2.29617055192
231 2.37828356586,9.81902071484,2.32410773481
232 2.38845539372,9.80708351433,2.35351258779

233 2.39852015022,9.79616484288,2.38192732581
234 2.40854259674,9.78579644259,2.4099369919
235 2.41983386222,9.7759888045,2.43793472962
236 2.42984795105,9.7655765323,2.46939472233
237 2.4399229316,9.75708497012,2.4978529209
238 2.44998789486,9.74916942592,2.52700601841
239 2.46093246434,9.74202636023,2.5559488437
240 2.47117296606,9.73507593508,2.58723204815
241 2.48135049455,9.72945970641,2.6169584894
242 2.49136404879,9.7245997427,2.64698374208
243 2.50275827385,9.72057806169,2.67650355801
244 2.51276602875,9.71682531389,2.7101527387
245 2.52284703497,9.71441057975,2.73948948381
246 2.53340743482,9.71258426291,2.76929513289
247 2.5436855983,9.71138641853,2.80136290285
248 2.5537404418,9.71091337046,2.83349696236
249 2.56378321536,9.71104018616,2.86524456439
250 2.57500216179,9.71171345912,2.89730781445
251 2.58516534045,9.71303999885,2.93269404816
252 2.59532028157,9.71480809484,2.9643194214
253 2.60556498449,9.71698627198,2.99571104822
254 2.61647521891,9.71959074366,3.02701540898
255 2.62678577937,9.72275883829,3.06018354983
256 2.63708701544,9.72609287524,3.09121678016
257 2.64779235795,9.72968249828,3.12212680321
258 2.65855162591,9.73366694392,3.15396429205
259 2.66888688877,9.7378640611,3.18593098864
260 2.67929237336,9.74203638439,3.2163932444
261 2.69010711554,9.74635938943,3.24703409122
262 2.70042319689,9.75093798076,3.27865939397
263 2.71052883379,9.75531439427,3.30887607679
264 2.72073483001,9.75961260746,3.33827811703
265 2.73186645471,9.76392798789,3.36798077747
266 2.74194578826,9.76856666527,3.40019836653
267 2.75197168346,9.77266365596,3.4294925676
268 2.76270509139,9.77664143785,3.45845351964
269 2.7728591254,9.78074987413,3.48947368543
270 2.7828703532,9.78445204537,3.51868783304
271 2.79295971431,9.78793105718,3.54754031441
272 2.80423180014,9.79123936136,3.57648480449
273 2.8141997708,9.79466115414,3.60885645608
274 2.82431183103,9.79741918868,3.63726560867
275 2.83422092628,9.79998553487,3.6659297098
276 2.8453074405,9.80222953902,3.69412726159
277 2.85563784279,9.80445351693,3.72558355914
278 2.8657056503,9.80620161048,3.75508540325
279 2.87575558852,9.80765311094,3.78373308805
280 2.88715326693,9.80884235045,3.81241842724
281 2.8972517401,9.80990189121,3.84485852886

282 2.90744626336,9.81053990647,3.87406729977
283 2.91828694195,9.81097574249,3.90406011839
284 2.92851553857,9.81118746499,3.93584301156
285 2.93855269067,9.81114455548,3.96553684817
286 2.94876833446,9.81090709605,3.99448136378
287 2.95994542446,9.81047794757,4.02398136681
288 2.9702532338,9.80981133407,4.0561302968
289 2.98053335771,9.80900834437,4.08618670538
290 2.99073606636,9.80806802513,4.11665851935
291 3.00165755954,9.80700468132,4.14695915123
292 3.01196049806,9.80574774156,4.17952049169
293 3.02206555288,9.80445456149,4.21006857177
294 3.03315111343,9.80310892886,4.24001349154
295 3.04327649437,9.8015658856,4.2726568644
296 3.05349194445,9.80010390399,4.30255548197
297 3.06356319413,9.7985951921,4.33241216732
298 3.0742028309,9.7970788505,4.36164468986
299 3.08435346838,9.79547095559,4.39252753181
300 3.09439643472,9.79394889136,4.42186066036
301 3.10473972652,9.79244590037,4.45091007512
302 3.11611732002,9.79093146513,4.48068998751
303 3.12629912794,9.78929972788,4.51378453593
304 3.13638886437,9.78790259429,4.54399492837
305 3.14712187275,9.7865664959,4.57381681911
306 3.15720795002,9.78520464817,4.60518031951
307 3.16744588595,9.78400587871,4.63434035877
308 3.17755735759,9.78285193211,4.66372163807
309 3.18874587771,9.78178763095,4.69277916035
310 3.19872591458,9.78069471926,4.72478408001
311 3.20889806189,9.77981447734,4.75346674974
312 3.21899488475,9.77899908519,4.78254666115
313 3.22981523629,9.77827810686,4.81145588528
314 3.23974239826,9.77759622931,4.84216756805
315 3.24985185452,9.77707266177,4.87010308727
316 3.26046720706,9.77662348095,4.89838286765
317 3.271012187,9.77624201117,4.92814324802
318 3.28103911504,9.77595686281,4.95750975255
319 3.29128832743,9.7757718505,4.98521115485
320 3.30203115847,9.77566027111,5.013292577
321 3.31223360635,9.77562784694,5.04280073389
322 3.32242924161,9.77567362418,5.07019904927
323 3.33255218342,9.77578520519,5.09815786617
324 3.34391895216,9.77595877541,5.12553316449
325 3.35405673739,9.77621817659,5.15680480416
326 3.36420220416,9.77651095641,5.18423223476
327 3.37435474712,9.77684855901,5.21221431686
328 3.38528192416,9.77722716972,5.24083397618
329 3.39547103271,9.77767539912,5.27305617198
330 3.4055687543,9.77812648954,5.30306300647

331 3.41626455076,9.77860046818,5.33365732947
332 3.42690968793,9.77912499187,5.36560636052
333 3.43688232545,9.77966668192,5.39799886172
334 3.44700339437,9.78018262276,5.42759662782
335 3.45769632235,9.78071477625,5.45792634544
336 3.46791431494,9.78127935498,5.48928985088
337 3.47780144773,9.78181246214,5.51958095155
338 3.4879333647,9.78231982021,5.54828227834
339 3.49921686109,9.78283194673,5.57795349378
340 3.50930803642,9.78338412814,5.61146086801
341 3.51926119439,9.78385429308,5.64271756841
342 3.5295158131,9.78430347729,5.6734726367
343 3.54037737101,9.78474312486,5.70583893875
344 3.55057168659,9.78518277282,5.73974685716
345 3.56064074021,9.78556806093,5.77196152459
346 3.57135723252,9.78592473882,5.80315003866
347 3.58197938185,9.78626997289,5.83639181533
348 3.59217975009,9.7865751524,5.8697618609
349 3.60226650257,9.78683741618,5.90142482817
350 3.6130138617,9.78706439431,5.93281361399
351 3.62311327271,9.78727229905,5.96567058403
352 3.63331125863,9.7874334735,5.99596788915
353 3.64326918311,9.78756603781,6.02608351947
354 3.65449858736,9.78766522328,6.05502612973
355 3.66457656119,9.78774533573,6.0876890276
356 3.67467405554,9.78778634714,6.11647714264
357 3.68468160648,9.78780256276,6.14541832339
358 3.69523604028,9.78779493653,6.17366904311
359 3.70537018403,9.78776060736,6.20308324733
360 3.71547459252,9.78770529745,6.23142599708
361 3.72653633263,9.78763091378,6.25930240026
362 3.73692815378,9.78752982736,6.28993540218
363 3.74736095872,9.7874144357,6.31826861577
364 3.75783217233,9.78728473897,6.3468826874
365 3.76859077718,9.78714073697,6.37519396758
366 3.77892213967,9.78698147611,6.40446273806
367 3.78926613368,9.78681935402,6.43215412994
368 3.79946050793,9.78664960255,6.46005723798
369 3.81077836826,9.78647699,6.4871826243
370 3.82098937407,9.78628053578,6.51747182853
371 3.83121112827,9.78610220102,6.5443883604
372 3.84196741413,9.78592195912,6.57151374756
373 3.85228231736,9.78573408784,6.59969342347
374 3.86230898928,9.78555670679,6.62692705347
375 3.87240697443,9.7853879086,6.65304679077
376 3.88364976458,9.78522292519,6.67950746754
377 3.89375602454,9.78504554442,6.70860362692
378 3.90371096879,9.78489391243,6.73500326896
379 3.91387063265,9.78475086354,6.7606619058

380 3.92468905076,9.78461353662,6.7869957449
381 3.93474021647,9.78447716371,6.81562984915
382 3.94470572192,9.78436081766,6.8433131383
383 3.95542627852,9.78425305465,6.87091584248
384 3.9662567256,9.78414910652,6.90122269353
385 3.97660208587,9.78405564875,6.93161871365
386 3.98679066077,9.78397554245,6.96013170231
387 3.99781054631,9.78390783398,6.98773583947
388 4.00814614259,9.78384489388,7.01711667496
389 4.01836456079,9.78379721256,7.04473225705
390 4.02853538003,9.78376002146,7.07160588053
391 4.04008355364,9.78373236726,7.09839510382
392 4.0501589356,9.78371043555,7.12835625267
393 4.06034213863,9.78370185477,7.15463144552
394 4.07047506049,9.78370090346,7.18078552245
395 4.08143091854,9.78370758175,7.20686950441
396 4.09172281064,9.78372284323,7.23466151982
397 4.1018797094,9.78374382644,7.26088521593
398 4.11288768891,9.7837714855,7.28637887378
399 4.12304049823,9.7838067742,7.31409078157
400 4.1332308352,9.783846831,7.33924540849
401 4.14322323166,9.78388974888,7.36458600222
402 4.15402954817,9.78393648162,7.38909069984
403 4.16434029676,9.7839898902,7.41567393381
404 4.17460726947,9.78404425235,7.4406597615
405 4.18469670229,9.78410052175,7.46519831606
406 4.19596765377,9.78415583738,7.48898918903
407 4.20605428144,9.78421973654,7.5156997405
408 4.21622411255,9.78427695968,7.53925696345
409 4.22681982629,9.78433609029,7.56314511199
410 4.23704912048,9.78439617459,7.58772229195
411 4.24716778938,9.78445435128,7.61161234839
412 4.25731396116,9.78450966715,7.63493115362
413 4.26837601978,9.78456498293,7.65844069395
414 4.27856179606,9.78462220633,7.68374028415
415 4.28875961807,9.78467370746,7.70723027465
416 4.29896733724,9.78472139399,7.73042319581
417 4.30982507579,9.78476621931,7.75376870481
418 4.32009422313,9.7848119985,7.77826911625
419 4.33018757124,9.78485110201,7.80161605681
420 4.34090684727,9.78488734425,7.8242429723
421 4.35163078364,9.78492263328,7.84839767675
422 4.36172026489,9.78495315378,7.87221525761
423 4.37191069964,9.78497890562,7.89478638362
424 4.38286706712,9.7850008428,7.91727263324
425 4.39311925974,9.78502087293,7.94157754294
426 4.40328734461,9.78503613448,7.96397700895
427 4.41358515341,9.78504758138,7.98632163859
428 4.42493349127,9.78505616716,8.00864481115

429 4.43506837077,9.78506189209,8.03338221447
430 4.44513143227,9.78506284844,8.05511935379
431 4.45542830508,9.78506189741,8.07682406788
432 4.46634281147,9.78505713182,8.09873573061
433 4.47653281596,9.78504950534,8.12209316643
434 4.48652860057,9.78504092478,8.14453602585
435 4.49757007137,9.78502948326,8.16746907422
436 4.50782437343,9.78501518098,8.19307050843
437 4.51789041329,9.78499992491,8.2175637703
438 4.52788174804,9.78498371509,8.24150962298
439 4.53876092564,9.7849646442,8.26554893596
440 4.54875617009,9.78494461984,8.29134492146
441 4.55875355471,9.78492459525,8.31507619894
442 4.56874847319,9.78490361706,8.33833159301
443 4.58010316174,9.78488263867,8.36145919496
444 4.59032262303,9.78485784593,8.38804959553
445 4.60052505787,9.78483591404,8.41285948188
446 4.61111950036,9.7848149359,8.43767032235
447 4.62144223694,9.78479395782,8.46385159585
448 4.63161436561,9.78477297975,8.48907728646
449 4.64168366324,9.78475295509,8.51401973587
450 4.65283371694,9.78473388419,8.53818112563
451 4.66301788576,9.7847129062,8.56473242928
452 4.67313945852,9.78469574266,8.588298727
453 4.68321996648,9.7846785791,8.61153600742
454 4.69412006624,9.78466332279,8.63412192783
455 4.70417482127,9.78464806671,8.65835008002
456 4.71437389124,9.78463471778,8.68010439629
457 4.72508859634,9.78462232257,8.702036097
458 4.73583461437,9.78461088136,8.72453237504
459 4.74606689252,9.78460039361,8.74698383054
460 4.75632837601,9.78459181319,8.76874863843
461 4.76725046337,9.78458323277,8.7913336072
462 4.77744843531,9.78457751354,8.81547306882
463 4.78763554897,9.78457274779,8.8385005429
464 4.7977945311,9.78456988939,8.86128482999
465 4.80910196435,9.78456798465,8.8841349211
466 4.81922569033,9.78456703389,8.90903064996
467 4.82932750974,9.78456703642,8.93121603014
468 4.83945768606,9.78456799254,8.95274623591
469 4.85022531729,9.7845689488,8.97408856828
470 4.8603713233,9.78457181249,8.99613566887
471 4.87038546521,9.7845756299,9.01670361756
472 4.88139912952,9.78458040085,9.03644472947
473 4.89154798724,9.78458612592,9.05795395897
474 4.90159612522,9.78459185063,9.07716101325
475 4.91158353351,9.78459852898,9.0960419104
476 4.92266277969,9.78460425363,9.11430959416
477 4.93282783218,9.78461188592,9.13443027283

478 4.94296982419,9.7846195178,9.15232411645
479 4.953072289,9.78462714971,9.17009112119
480 4.96428884286,9.78463478182,9.18812324755
481 4.97445211932,9.78464241402,9.20780237456
482 4.98450832535,9.78465004616,9.22575057675
483 4.9951432161,9.78465863175,9.24299782796
484 5.00548430998,9.78466626403,9.26070093772
485 5.01553074736,9.78467389611,9.27783470203
486 5.02564946096,9.78468057444,9.29399380937
487 5.03695082292,9.7846872528,9.31019773948
488 5.04708901886,9.78469393146,9.32778831717
489 5.05720592756,9.78470060973,9.34356881539
490 5.06746877171,9.78470538077,9.35886389262
491 5.07841373049,9.7847111056,9.37434016841
492 5.08862505946,9.78471587691,9.39037148535
493 5.09880510997,9.78472064791,9.40534517382
494 5.11038322467,9.78472446515,9.41983820955
495 5.12016586121,9.78472828288,9.43631298663
496 5.13046021201,9.78473114655,9.44975697779
497 5.14064994641,9.78473496382,9.46390287549
498 5.15124185197,9.78473687374,9.47750994629
499 5.16148516349,9.78473878398,9.49167301054
500 5.17169347592,9.78473974041,9.50495010927
501 5.18174762372,9.78474069672,9.51821576372
502 5.19277319591,9.784741653,9.53090062897
503 5.20296228118,9.78474165585,9.544840534
504 5.21299731173,9.78474165859,9.55730128547
505 5.22306407895,9.78474070749,9.56962375352
506 5.23394611944,9.7847397565,9.58162960105
507 5.24388892297,9.78473785204,9.5946568372
508 5.25396615826,9.78473690081,9.60616677301
509 5.26464715041,9.78473499606,9.61788174887
510 5.27533946652,9.78473309148,9.62993432764
511 5.28557155188,9.78473118699,9.64209943954
512 5.29563042894,9.78472832869,9.65334902164
513 5.30653023068,9.78472547032,9.66448606977
514 5.31682502665,9.78472261207,9.67618483451
515 5.32714027259,9.78471975377,9.68735812231
516 5.33731827978,9.78471689538,9.69817855002
517 5.34920761362,9.78471403705,9.70894461766
518 5.3589325808,9.78471117906,9.72111068525
519 5.36904668715,9.78470736694,9.73126163082
520 5.37924081646,9.78470450859,9.74231189268
521 5.38996973634,9.78470260392,9.75332591518
522 5.39998310711,9.78469974562,9.76449348079
523 5.410219457,9.78469784092,9.77498202758
524 5.42100672238,9.78469593627,9.78532275545
525 5.43172162864,9.78469307808,9.79627097563
526 5.44182451256,9.78469117344,9.80671470157

527 5.45205992274, 9.78468926877, 9.8166558387
528 5.46298723947, 9.78468736403, 9.82635569612
529 5.47314642742, 9.78468545948, 9.83678225593
530 5.48328024708, 9.78468546219, 9.84607107901
531 5.49337940384, 9.78468355752, 9.85541426134
532 5.50491514523, 9.78468260652, 9.86437406602
533 5.51511053462, 9.78468260945, 9.87468904665
534 5.52523404825, 9.78468070481, 9.88338182204
535 5.53548575006, 9.78468070748, 9.8921136978
536 5.54630575422, 9.78468071009, 9.90061383038
537 5.55637595523, 9.78468071286, 9.90968616953
538 5.56657394674, 9.78468071555, 9.91775524041
539 5.57761247363, 9.78468071816, 9.92602076823
540 5.58790980093, 9.7846807211, 9.93460101076
541 5.59798344504, 9.78468167741, 9.94276544874
542 5.60820126627, 9.78468168009, 9.95039964249
543 5.61905114073, 9.78468263638, 9.95780209319
544 5.62924380135, 9.78468359304, 9.96531135707
545 5.63919344451, 9.7846845493, 9.97255072899
546 5.64931324869, 9.78468550553, 9.979305633
547 5.66049038898, 9.7846864619, 9.98629800235
548 5.67073530424, 9.78468741844, 9.99368614899
549 5.6807326125, 9.78468837495, 10.0006727959
550 5.69146838039, 9.78468933126, 10.0071644847
551 5.70170597639, 9.78469028766, 10.014269388
552 5.71186066139, 9.78469029038, 10.0207095783
553 5.72193909716, 9.78469124665, 10.0272451356
554 5.73310528323, 9.78469220292, 10.0334287863
555 5.74321913626, 9.78469220601, 10.0404249705
556 5.75352466106, 9.78469220868, 10.0464140713
557 5.76362525485, 9.78469316498, 10.0526663865
558 5.7745725736, 9.78469316766, 10.0584914549
559 5.78479293361, 9.78469412414, 10.0649564406
560 5.7950290069, 9.78469412678, 10.0706537159
561 5.80567720812, 9.78469508329, 10.0765188381
562 5.81625574175, 9.78469603978, 10.0823028985
563 5.82646586839, 9.78469699625, 10.0877121641
564 5.83658138663, 9.78469795261, 10.0931309659
565 5.84736491367, 9.78469795519, 10.0982026292
566 5.85768593661, 9.78469795796, 10.1033114868
567 5.86799997091, 9.78469796063, 10.1084336947
568 5.87815947272, 9.78469891698, 10.1132564486
569 5.8895832561, 9.78469987339, 10.118198411
570 5.89965081029, 9.78469987634, 10.123441737
571 5.90984388534, 9.78470083281, 10.1283407837
572 5.91995318141, 9.78470083561, 10.1339331523
573 5.93061427679, 9.78470083825, 10.1405478603
574 5.94061637577, 9.78470084105, 10.1489993484
575 5.95075558219, 9.78470084368, 10.1575519249

576 5.96139837522, 9.78470084634, 10.1671020469
577 5.97211905941, 9.78470084914, 10.1773245114
578 5.98218471184, 9.7846998982, 10.1869433004
579 5.99241241626, 9.78469990087, 10.195308006
580 6.00314406957, 9.78469990349, 10.2036469632
581 6.01317608263, 9.78470086003, 10.2117808819
582 6.02314423025, 9.78470086261, 10.2192634393
583 6.03320989385, 9.78470086537, 10.2261737916
584 6.04437196068, 9.78470086794, 10.2330345531
585 6.05459515005, 9.78469991734, 10.2400717471
586 6.06488385797, 9.78469896626, 10.2459216135
587 6.07520987652, 9.784698969, 10.2517962752
588 6.08619515225, 9.78469897164, 10.2581334695
589 6.09643935226, 9.78469897447, 10.2645173951
590 6.10663145781, 9.78469897723, 10.2706314289
591 6.11740126647, 9.78469897986, 10.2761961454
592 6.1281785313, 9.78469802908, 10.2820145407
593 6.13832278922, 9.78469803186, 10.2872683598
594 6.1484766202, 9.78469803468, 10.292220816
595 6.15942528285, 9.7846970836, 10.2967116928
596 6.16961912252, 9.78469613285, 10.301065242
597 6.17962338869, 9.78469613547, 10.3051832317
598 6.18976625241, 9.78469613806, 10.308802448
599 6.20099072624, 9.7846961407, 10.3125036805
600 6.21116205771, 9.78469614373, 10.3161515085
601 6.22119080182, 9.78469614646, 10.319579989
602 6.23129927367, 9.78469614905, 10.322561195
603 6.24194544647, 9.78469615168, 10.3256272777
604 6.25200541317, 9.78469615455, 10.3284635251
605 6.26190757006, 9.78469615716, 10.3312559025
606 6.27300805785, 9.78469615982, 10.3336467815
607 6.28341399785, 9.7846961626, 10.3364134099
608 6.2937239306, 9.78469521176, 10.3396931145
609 6.30380960274, 9.78469521436, 10.3439531968
610 6.31488394458, 9.78469426353, 10.3482991093
611 6.32505924627, 9.78469426631, 10.3536978805
612 6.33524744026, 9.78469426918, 10.3583899775
613 6.34528660774, 9.78469427173, 10.3632832998
614 6.35657079984, 9.78469522822, 10.3675862971
615 6.36675991677, 9.78469523104, 10.3723327554
616 6.37699923012, 9.78469618749, 10.37602254
617 6.38705794606, 9.78469619007, 10.3797132782
618 6.39782190043, 9.78469523924, 10.382860421
619 6.40801013447, 9.78469524194, 10.3862078367
620 6.41831693798, 9.78469619858, 10.388875281
621 6.42937036604, 9.78469620132, 10.3915970851
622 6.43961275369, 9.78469620416, 10.3940337408
623 6.44987535384, 9.784696207, 10.3963807497
624 6.460607371, 9.78469620963, 10.3982986041

625 6.47115048114,9.78469621235,10.4002622344
626 6.48134926986,9.78469621521,10.4019597903
627 6.49156171083,9.78469621795,10.4036563912
628 6.50174507964,9.7846962206,10.4049553085
629 6.512859351,9.78469622334,10.4063448247
630 6.52284892369,9.78469622623,10.4074558684
631 6.53300816845,9.78469622901,10.4080595554
632 6.54361925181,9.78469718521,10.4083714174
633 6.55388979055,9.78469814176,10.4088644777
634 6.56412349455,9.78469814434,10.4090084919
635 6.57439414877,9.78469814723,10.4088625882
636 6.58565675467,9.78469814981,10.4089255391
637 6.59585931897,9.7846971992,10.4086823606
638 6.60594837274,9.78469720178,10.4081635684
639 6.61612375453,9.78469625096,10.4078927313
640 6.62698312756,9.78469720719,10.4073748919
641 6.63707502466,9.78469721026,10.4070773513
642 6.64713860489,9.78469721279,10.4065375766
643 6.65780408122,9.78469721562,10.4062381276
644 6.66847980116,9.78469721839,10.4056830935
645 6.67860366404,9.78469722137,10.4054351426
646 6.68858506158,9.78469627031,10.4049411428
647 6.69934511557,9.78469627308,10.4047017741
648 6.70940196514,9.78469627584,10.4042125427
649 6.71946005337,9.7846972323,10.4040609119
650 6.72930655908,9.78469818856,10.4036765841
651 6.74068714958,9.78469723752,10.4035363969
652 6.75092446152,9.78469819414,10.4031377639
653 6.76116440073,9.78469724332,10.4031539795
654 6.77129118238,9.78469724595,10.4029117489
655 6.78217987809,9.78469629501,10.4029355935
656 6.79251242802,9.78469629783,10.4027200659
657 6.80260307528,9.78469630071,10.4028440465
658 6.81332664564,9.78469630332,10.4027219788
659 6.8241416458,9.7846963062,10.4028564497
660 6.83430218045,9.78469535534,10.4027391505
661 6.84443830606,9.78469535818,10.4029499155
662 6.85529366788,9.78469440712,10.4029279835
663 6.86534498725,9.78469441004,10.4031749887
664 6.8755740663,9.78469441265,10.4031635473
665 6.88558937609,9.78469441542,10.4034086445
666 6.89680151176,9.7846953717,10.4034372575
667 6.90690587554,9.78469537484,10.4037395756
668 6.91715535335,9.78469537745,10.4037681886
669 6.92715922836,9.78469442654,10.4040657382
670 6.93796917703,9.78469538281,10.4041468036
671 6.948075925,9.78469443207,10.4044977596
672 6.95811302215,9.78469443471,10.4045978987
673 6.96903128643,9.78469443743,10.4049583909

674 6.97927171737, 9.78469444029, 10.4051291023
675 6.98954002652, 9.78469444307, 10.4056221559
676 6.99979352392, 9.78469444576, 10.4058825126
677 7.01060996018, 9.78469444855, 10.4064118058
678 7.02073531598, 9.78469445138, 10.4067370129
679 7.03095651604, 9.78469445414, 10.4073569054
680 7.04121250473, 9.78469350314, 10.4077498234
681 7.05232220516, 9.78469350594, 10.4084040483
682 7.0625191126, 9.78469446252, 10.4088694461
683 7.07261266746, 9.7846944653, 10.4090582776
684 7.08333880827, 9.78469446795, 10.4090668645
685 7.09349295311, 9.78469447087, 10.4093729986
686 7.10357028246, 9.78469447353, 10.410479266
687 7.11376253329, 9.78469447628, 10.4126860749
688 7.12527319323, 9.78469543263, 10.4152362073
689 7.13547826279, 9.78469638944, 10.4177377031
690 7.14562923741, 9.78469639212, 10.4195554136
691 7.1563713951, 9.78469734847, 10.421541925
692 7.16697332729, 9.78469830509, 10.4233186286
693 7.17702477053, 9.78469830787, 10.4252803451
694 7.18718138151, 9.78469831062, 10.4278180817
695 7.19789046049, 9.7846983133, 10.4313972323
696 7.20855630655, 9.78469831622, 10.4354322406
697 7.21868549008, 9.78469831903, 10.4402044407
698 7.22863451857, 9.78469832179, 10.445875003
699 7.23958311509, 9.78469927808, 10.4517134131
700 7.24974560924, 9.78469928118, 10.4588287962
701 7.2598389294, 9.78469928382, 10.4652937729
702 7.26990872156, 9.78469928638, 10.4710854557
703 7.2811984783, 9.78470024273, 10.4766568405
704 7.29125798773, 9.78470024574, 10.4822215519
705 7.3012797283, 9.78469929475, 10.4870395344
706 7.3112311773, 9.78469929752, 10.4912776828
707 7.3223414002, 9.78469930025, 10.4953394014
708 7.33254605811, 9.78469930317, 10.4992704685
709 7.34289559349, 9.78469930596, 10.5027952686
710 7.35346687865, 9.78469930868, 10.505820343
711 7.36420614738, 9.7846983579, 10.5088292053
712 7.37443975918, 9.78469836072, 10.512345423
713 7.38452400733, 9.78469836352, 10.5152636853
714 7.39530151058, 9.78469836626, 10.5176040199
715 7.40550909098, 9.78469741541, 10.5200330472
716 7.41555045638, 9.78469646453, 10.5217763807
717 7.42574062292, 9.78469646726, 10.523477752
718 7.43690369092, 9.78469646992, 10.524732803
719 7.44706385676, 9.78469647297, 10.5261003888
720 7.45717411768, 9.78469647563, 10.5268194732
721 7.46737855952, 9.7846964783, 10.52758338
722 7.47840792406, 9.78469648101, 10.527928623

723 7.48864580691,9.78469648394,10.5283587433
724 7.49894031323,9.78469648667,10.5282939048
725 7.51010645833,9.78469648941,10.5283292017
726 7.52020817436,9.7846964924,10.527941067
727 7.53042728454,9.78469554143,10.527762739
728 7.54067687225,9.78469649783,10.527207709
729 7.55144710932,9.784695547,10.5267718879
730 7.56164592132,9.78469554995,10.5259431524
731 7.57177508809,9.78469555283,10.5253461589
732 7.58187435474,9.78469555566,10.5253681005
733 7.59297242109,9.78469555836,10.5264076134
734 7.60308055487,9.78469556133,10.5277761446
735 7.61315927003,9.78469556414,10.5297845913
736 7.62383430358,9.78469556682,10.5316337746
737 7.6339240456,9.78469556978,10.5330852755
738 7.6440783944,9.78469461878,10.5339550341
739 7.65410685632,9.78469462159,10.5348610325
740 7.6656370135,9.78469557783,10.5353350159
741 7.67573624197,9.78469558103,10.5359253487
742 7.6858424563,9.78469558363,10.5359539655
743 7.69610391371,9.78469463278,10.5356325834
744 7.70706471615,9.78469463552,10.5349507119
745 7.71718705539,9.78469463846,10.5343718374
746 7.7273417525,9.78469464118,10.5334648977
747 7.73798226472,9.78469464391,10.5322623183
748 7.7488477435,9.78469464678,10.5311932532
749 7.75898712501,9.78469560349,10.5297722818
750 7.76911756117,9.7846956062,10.528155806
751 7.78006207384,9.78469465524,10.5267691656
752 7.79039935116,9.78469465818,10.525003916
753 7.80061712582,9.78469561464,10.5236468389
754 7.81080096494,9.78469657116,10.5220389446
755 7.82208009064,9.78469657388,10.5206885424
756 7.83217744902,9.7846965769,10.5189347353
757 7.84233453404,9.78469657971,10.5177226155
758 7.85298143886,9.78469658233,10.5172133541
759 7.86317363102,9.78469658521,10.5165972813
760 7.87326225918,9.78469754163,10.516439926
761 7.88329301123,9.78469659069,10.5159926536
762 7.89449975546,9.78469754708,10.5157399307
763 7.90464333445,9.78469659644,10.5151686805
764 7.91485186014,9.78469659918,10.5149493367
765 7.9249765845,9.78469660195,10.514447705
766 7.93583523761,9.78469660459,10.5141701869
767 7.94593281019,9.78469660756,10.5136037054
768 7.95604224596,9.7846966103,10.5133595663
769 7.96630772483,9.78469661296,10.5128607957
770 7.97741748486,9.78469661577,10.5121207454
771 7.98753206991,9.78469661891,10.5116000406

772 7.99772233982,9.78469662154,10.5119662537
773 8.00844879542,9.78469662441,10.5122180258
774 8.01854430139,9.7846966273,10.5128846472
775 8.02872392442,9.78469663011,10.5144095765
776 8.03879213333,9.78469663284,10.5159497648
777 8.04995129444,9.78469663564,10.5171266033
778 8.05997877847,9.78469663862,10.5185542591
779 8.07017547358,9.78469759489,10.5194545324
780 8.08037641365,9.78469759756,10.5205140698
781 8.0913055921,9.78469855403,10.5212503114
782 8.10155254975,9.78469760334,10.5217061731
783 8.11162252817,9.78469760604,10.5218091748
784 8.12227641977,9.78469856248,10.5216584988
785 8.13276270218,9.7846985654,10.5217176317
786 8.14273713157,9.78469856826,10.5225320751
787 8.15304357279,9.78469857098,10.5231338489
788 8.16398386285,9.7846985738,10.5240474747
789 8.17427789234,9.78469857678,10.5246854888
790 8.18441899959,9.78469857959,10.5255123306
791 8.19456245564,9.78469762868,10.5260139688
792 8.20590911619,9.78469763144,10.5262314118
793 8.2160106441,9.78469668085,10.5266853671
794 8.2260843832,9.78469668361,10.5267702494
795 8.23616198264,9.78469668635,10.5270697084
796 8.24706293456,9.78469573532,10.5279985935
797 8.25689978153,9.78469573832,10.5301128977
798 8.26701922249,9.78469574092,10.532210989
799 8.27751822304,9.78469478994,10.5349079888
800 8.28818523511,9.78469479285,10.5375162972
801 8.29812277202,9.78469479569,10.5404469487
802 8.30809719861,9.78469479844,10.5427176568
803 8.31866075844,9.7846948011,10.5450179293
804 8.3291575443,9.78469480403,10.5470244708
805 8.33929356374,9.78469480693,10.5490777424
806 8.34944355674,9.78469480963,10.5506265196
807 8.36060622521,9.78469481244,10.5522372859
808 8.37082635332,9.78469481552,10.5535714872
809 8.38081740681,9.78469481824,10.5549238078
810 8.3909349665,9.78469482101,10.5558574646
811 8.40157144237,9.78469482381,10.5568740913
812 8.41178856883,9.78469482664,10.5575559783
813 8.42180465441,9.78469482947,10.5583370468
814 8.43245552387,9.78469483214,10.5587423672
815 8.44299560599,9.78469483511,10.5592792951
816 8.45318313688,9.78469483792,10.5594242618
817 8.46315383352,9.78469579443,10.5597294457
818 8.47384954523,9.7846957972,10.5596970281
819 8.48402777314,9.78469675372,10.5597971717
820 8.49415822886,9.78469675654,10.5595463621

821 8.50430623628,9.78469580558,10.5594653064
822 8.51577426866,9.7846958084,10.5590790742
823 8.52598661464,9.78469581147,10.5588339868
824 8.5364194205,9.78469581431,10.5582617875
825 8.54647984728,9.78469677088,10.5578936746
826 8.55744932778,9.78469677357,10.5572509026
827 8.56769564468,9.78469677662,10.5567588117
828 8.57791680563,9.78469677926,10.5570697148
829 8.58856218681,9.78469678213,10.5584515956
830 8.59918984026,9.784696785,10.5601539114
831 8.60933956318,9.78469678798,10.5626191679
832 8.61937040649,9.78469679071,10.5648164416
833 8.63033372909,9.78469774709,10.5672874205
834 8.64053856209,9.78469775016,10.5695228425
835 8.65079373401,9.78469775291,10.5716524066
836 8.66087874305,9.7846968021,10.5733337435
837 8.6720282454,9.78469680482,10.5750074513
838 8.68206731696,9.78469680783,10.5763864742
839 8.69230105262,9.78469776432,10.5777149518
840 8.70231415424,9.78469776707,10.5786495617
841 8.71313509997,9.78469776987,10.5796118281
842 8.72316644713,9.78469872635,10.5802288644
843 8.73333294224,9.78469872919,10.5803709697
844 8.74363951385,9.78469873194,10.5806160719
845 8.75469418522,9.78469873483,10.5814190741
846 8.76463436894,9.78469873771,10.581997963
847 8.77469079103,9.78469778685,10.5820885691
848 8.7852941351,9.78469874325,10.5818530182
849 8.79530883022,9.78469874613,10.5812531632
850 8.80539277196,9.78469874895,10.5803576682
851 8.81561980676,9.78469875167,10.5796252517
852 8.82688077539,9.78469875455,10.578592427
853 8.83702523354,9.78469875748,10.5776654608
854 8.84700113628,9.78469876035,10.5765077036
855 8.85722996574,9.78469876305,10.5751363228
856 8.86830194667,9.78469876583,10.5735017273
857 8.87841326371,9.7846968615,10.571504735
858 8.8884581048,9.78469686426,10.5694676875
859 8.89916715398,9.78469782077,10.5677367695
860 8.90993797965,9.78469687,10.5657044897
861 8.92010222282,9.78469687303,10.5640326988
862 8.93049064372,9.78469782945,10.5622512349
863 8.94162370171,9.78469687869,10.5607444291
864 8.95186444186,9.7846968818,10.5589400763
865 8.96209455654,9.78469688455,10.5576936234
866 8.97216213308,9.78469688741,10.5562535739
867 8.98356596287,9.78469689012,10.555147311
868 8.99377658311,9.7846968933,10.5537110763
869 9.00388015434,9.78469689605,10.5522529069

870 9.01398659311,9.78469689879,10.5506869719
871 9.02477995772,9.78469690163,10.5490123181
872 9.03488570731,9.78469595087,10.5476046932
873 9.04490931984,9.78469595361,10.5471936591
874 9.05589276087,9.78469595634,10.5479413408
875 9.06609948073,9.78469595944,10.5504208974
876 9.07623626105,9.78469691588,10.5534678915
877 9.08628587052,9.78469596496,10.5574876353
878 9.09710207582,9.7846959677,10.5617000223
879 9.10709781107,9.78469597065,10.566878484
880 9.117191392,9.78469501981,10.5714761581
881 9.12719871383,9.78469502255,10.5763437237
882 9.13855548948,9.78469502528,10.5806982131
883 9.14877776615,9.78469502839,10.5850116963
884 9.15903064143,9.78469503128,10.5888245
885 9.16921535414,9.78469503407,10.593075995
886 9.17998402379,9.78469503684,10.5969641407
887 9.19014831167,9.78469503989,10.6011793981
888 9.2003088994,9.78469504265,10.6045287183
889 9.21139050182,9.78469504551,10.6077721809
890 9.22145365831,9.78469504862,10.6106933023
891 9.23142057844,9.78469505143,10.6126979407
892 9.241721984,9.78469505411,10.6146281923
893 9.25276602246,9.78469505699,10.61610068
894 9.26280289888,9.78469506007,10.6176132229
895 9.27299246565,9.78469506285,10.6194938831
896 9.28315158281,9.78469506568,10.6222328515
897 9.29437616654,9.7846950685,10.6250462066
898 9.30452782474,9.78469602517,10.6286902141
899 9.3147229366,9.78469698158,10.631585585
900 9.32567854598,9.78469698445,10.6338810944
901 9.33587282524,9.7846969874,10.6356682957
902 9.34584590048,9.78469699037,10.6366811114
903 9.35596638825,9.78469699313,10.6376138181
904 9.36726323143,9.78469699584,10.638040123
905 9.37741741259,9.78469795266,10.638442587
906 9.38746075425,9.78469795549,10.6382041792
907 9.39768138621,9.78469891186,10.6379791226
908 9.40858870186,9.78469796095,10.637279135
909 9.41867968999,9.78469796391,10.6360412745
910 9.42870108411,9.78469796676,10.6349865181
911 9.43930242397,9.78469796957,10.6344572365
912 9.44990060851,9.78469701876,10.6336389912
913 9.45982217882,9.78469702174,10.6330810987
914 9.46974953916,9.78469702452,10.6332775625
915 9.48093358055,9.78469702729,10.63337389
916 9.49111046176,9.78469703041,10.6338488275
917 9.50142701622,9.78469703314,10.6337296241
918 9.51149612106,9.78469703602,10.6336256794

919 9.52295454778,9.7846979925,10.633070646
920 9.53304091748,9.78469799569,10.6324746051
921 9.54332043976,9.78469799849,10.6314570385
922 9.55324893445,9.78469800124,10.6305138582
923 9.5641808575,9.78469800401,10.6292273544
924 9.57436504588,9.78469800705,10.6279103331
925 9.58454331383,9.78469800989,10.6262967181
926 9.59517801087,9.78469801271,10.6243683899
927 9.60596953705,9.78469801566,10.6220538227
928 9.61605847441,9.78469706487,10.6199624151
929 9.6261687912,9.78469706778,10.6187646009
930 9.63710639905,9.78469707058,10.617420874
931 9.647153317,9.78469611982,10.6163355933
932 9.65730885137,9.78469516904,10.6162421345
933 9.66738987714,9.78469517175,10.6161114823
934 9.67855476961,9.78469517465,10.6164233355
935 9.68855598755,9.78469517773,10.616411893
936 9.69875879213,9.7846951805,10.6166035836
937 9.70952751394,9.78469518331,10.6164319238
938 9.71968714893,9.7846951864,10.6163689837
939 9.72982644476,9.78469518918,10.6159341099
940 9.73993131332,9.78469519206,10.6156451486
941 9.75099011697,9.78469519473,10.6150367059
942 9.76110115182,9.78469519789,10.6145350753
943 9.77126653213,9.78469520067,10.6137330366
944 9.78135822713,9.78469615724,10.6126563391
945 9.7922221804,9.78469616,10.6117846819
946 9.80247871205,9.78469616299,10.610563026
947 9.81266145501,9.78469616592,10.6096818317
948 9.82334925421,9.78469616861,10.6085364693
949 9.83397814259,9.78469712535,10.6075703979
950 9.84406304359,9.78469712838,10.6063353898
951 9.8541962672,9.78469713104,10.6049392102
952 9.86519535724,9.78469713394,10.6037976619
953 9.87529616803,9.78469713696,10.6033179647
954 9.88541476801,9.78469713975,10.6026809108
955 9.89557658136,9.78469618897,10.6023328206
956 9.90698817372,9.78469714544,10.6017415431
957 9.91695455555,9.78469619491,10.6013295568
958 9.92700572126,9.78469619766,10.6006896418
959 9.93718909938,9.78469620054,10.6002795629
960 9.94816941023,9.78469620335,10.6005275201
961 9.95841330476,9.78469620638,10.6006495924
962 9.96859289333,9.78469620921,10.6004645813
963 9.97997666337,9.7846971658,10.6005103598
964 9.99015486799,9.78469716893,10.6002833874
965 10.0003043441,9.78469717175,10.6004092751
966 10.0104979631,9.78469717465,10.6002681336
967 10.0215107501,9.78469717736,10.6003434766

968 10.0317392666, 9.78469718043, 10.6001575129
969 10.0418845266, 9.78469718328, 10.6002853081
970 10.0521001332, 9.78469718621, 10.6011102402
971 10.0633418076, 9.78469718904, 10.6030471586
972 10.0733523462, 9.78469719216, 10.6054962012
973 10.0835327292, 9.78469719494, 10.6072395241
974 10.0942299012, 9.78469719777, 10.6086738567
975 10.1043867432, 9.78469720075, 10.6097991991
976 10.1145114433, 9.78469720358, 10.6110208628
977 10.1243442921, 9.78469720641, 10.6128252226
978 10.135466883, 9.78469625548, 10.6155384353
979 10.1456415653, 9.78469721226, 10.6188496038
980 10.1557345763, 9.7846972152, 10.6213491944
981 10.1658059889, 9.7846972179, 10.6238430632
982 10.1767151132, 9.78469722071, 10.625891566
983 10.1868876955, 9.78469722376, 10.6281136389
984 10.1969034262, 9.78469722661, 10.6296881657
985 10.2070444329, 9.78469722941, 10.6312789052
986 10.2180322511, 9.78469723225, 10.6324662404
987 10.2282058364, 9.78469723533, 10.6337823224
988 10.2382965768, 9.78469723817, 10.6345233373
989 10.2490179213, 9.784697241, 10.6353253873
990 10.2590004569, 9.7846972439, 10.6357526434
991 10.269020929, 9.78469629314, 10.6362428411
992 10.2790121986, 9.78469629585, 10.6363544295
993 10.2902508453, 9.7846953449, 10.6360902695
994 10.3004050003, 9.78469534809, 10.6358919139
995 10.3107792214, 9.78469535087, 10.6352987353
996 10.3210722795, 9.78469440004, 10.6348409786
997 10.3323363736, 9.78469440288, 10.6340389445
998 10.3420093348, 9.78469440601, 10.6328106177
999 10.3521306571, 9.78469440869, 10.6314449602
1000 10.3627571231, 9.7846944116, 10.630198512
1001 10.3734444836, 9.78469346086, 10.6286297213
1002 10.3835925115, 9.7846944177, 10.6273250981
1003 10.3938243939, 9.78469442047, 10.6258173414
1004 10.4047230473, 9.78469442338, 10.6240749802
1005 10.4147492684, 9.78469442646, 10.6225033267
1006 10.424971614, 9.78469538287, 10.6208267682
1007 10.4351283442, 9.78469538587, 10.6193857672
1008 10.446583868, 9.78469538862, 10.6177597528
1009 10.4566360181, 9.78469539186, 10.6162443647
1010 10.4668071922, 9.7846953947, 10.6146994122
1011 10.4769615848, 9.78469635124, 10.6134310252
1012 10.4879293656, 9.7846963541, 10.6119966986
1013 10.4981588898, 9.78469731086, 10.6107721806
1014 10.5081620542, 9.78469731364, 10.609445619
1015 10.5193626666, 9.78469731647, 10.6084642879
1016 10.5296753002, 9.78469731964, 10.6071920857

1017 10.5397608252,9.78469732256,10.6064424975
1018 10.5498588188,9.7846973254,10.6055222013
1019 10.5607479885,9.78469828192,10.6048994521
1020 10.5708572231,9.78469828498,10.6040592647
1021 10.5808847416,9.78469828782,10.6036568146
1022 10.5909925988,9.78469924431,10.6030817492
1023 10.6022444377,9.78469924716,10.6027889719
1024 10.612355643,9.78469925032,10.6032696258
1025 10.6223407155,9.78469925317,10.6035252122
1026 10.6324106362,9.78469925597,10.6040726234
1027 10.6432059081,9.7846992588,10.6044245316
1028 10.6532011759,9.78469926184,10.6050539598
1029 10.6630741283,9.78469926465,10.6054068221
1030 10.6738224439,9.78469831375,10.6059904741
1031 10.6840416789,9.78469831678,10.6064129557
1032 10.6943133017,9.78469831956,10.6071072348
1033 10.7046833746,9.78469832247,10.6085253542
1034 10.7156239841,9.78469737184,10.6111088651
1035 10.7259789612,9.7846973748,10.6141510946
1036 10.7362749726,9.78469737774,10.6166153975
1037 10.7465335261,9.78469738065,10.6191817441
1038 10.7578477934,9.78469738346,10.6213675746
1039 10.7681459403,9.78469738668,10.6238948224
1040 10.778252745,9.7846973896,10.6257592655
1041 10.7882831721,9.78469643879,10.627717169
1042 10.7988552107,9.78469644163,10.6293021855
1043 10.8088853909,9.78469549095,10.6310626792
1044 10.8188290652,9.78469549378,10.6333152691
1045 10.8297473434,9.78469549659,10.636462407
1046 10.8396770377,9.78469549967,10.6401245309
1047 10.8496260419,9.78469550247,10.6441252092
1048 10.8596237786,9.78469550528,10.649066212
1049 10.870388722,9.784695508,10.6542847354
1050 10.8803963829,9.78469551107,10.6605456278
1051 10.8904104494,9.78469551381,10.6662085656
1052 10.9004880721,9.78469551667,10.6711915337
1053 10.9117528964,9.78469551954,10.6759141492
1054 10.9218132636,9.78469552263,10.6804050241
1055 10.9319272358,9.78469552551,10.6841625211
1056 10.9419533107,9.7846955283,10.6872591216
1057 10.9525892558,9.78469553117,10.6900743879
1058 10.9625863545,9.78469648789,10.6923384306
1059 10.9726930317,9.78469649074,10.693706017
1060 10.9833586989,9.78469649351,10.6944575291
1061 10.9939568825,9.78469649656,10.6950669436
1062 11.0039641578,9.78469649959,10.694966823
1063 11.0141405547,9.78469554876,10.6947656112
1064 11.0249257497,9.78469650532,10.6939864716
1065 11.0348561499,9.78469746206,10.693050929

1066 11.0448650606, 9.78469746487, 10.6916013536
1067 11.0550366463, 9.78469746771, 10.6900993254
1068 11.0664064931, 9.78469747059, 10.6880823113
1069 11.0765037006, 9.78469747381, 10.6858192495
1070 11.0865632901, 9.78469747667, 10.6832824804
1071 11.0966939768, 9.78469652584, 10.6808286801
1072 11.1073851539, 9.78469557504, 10.6788574389
1073 11.1172170602, 9.78469653174, 10.6764656262
1074 11.1271612952, 9.78469558085, 10.6738668647
1075 11.1378376568, 9.78469558367, 10.6713377209
1076 11.1485227477, 9.78469558669, 10.6682983603
1077 11.1586470064, 9.78469558972, 10.6654640391
1078 11.168834297, 9.78469559249, 10.6634861179
1079 11.1798497774, 9.7846955954, 10.6613174608
1080 11.1898151422, 9.78469655221, 10.6586509849
1081 11.1999222403, 9.78469560138, 10.6565014008
1082 11.2099073157, 9.78469560425, 10.6549564464
1083 11.2211683998, 9.78469560709, 10.6544920064
1084 11.2313568341, 9.78469561029, 10.6542421432
1085 11.2416040162, 9.78469656686, 10.6548706154
1086 11.2514375094, 9.78469656976, 10.6553751099
1087 11.2620121427, 9.78469561879, 10.6562162523
1088 11.2722499808, 9.78469562181, 10.6567445894
1089 11.2821118664, 9.78469562473, 10.6568237458
1090 11.2927410277, 9.78469467378, 10.6569954088
1091 11.3034962118, 9.78469467683, 10.6577135286
1092 11.3136733538, 9.78469563358, 10.6595455418
1093 11.3236250617, 9.78469563648, 10.6613832773
1094 11.3343706802, 9.78469563932, 10.6636978512
1095 11.3443873096, 9.78469564228, 10.667153022
1096 11.3544792011, 9.78469564516, 10.6705032887
1097 11.3644571183, 9.78469564805, 10.6743637729
1098 11.3757676743, 9.78469469723, 10.6779581825
1099 11.385804845, 9.78469470045, 10.6822182592
1100 11.3958346304, 9.78469470331, 10.6852738436
1101 11.4058724055, 9.78469565984, 10.6881882848
1102 11.4167251671, 9.78469566269, 10.6904666249
1103 11.4267726177, 9.78469566577, 10.6927134945
1104 11.4368361784, 9.78469566863, 10.6940782142
1105 11.447350462, 9.78469662517, 10.6952941608
1106 11.4580907514, 9.78469662816, 10.6968581991
1107 11.4681211933, 9.78469663122, 10.6992776852
1108 11.47822541, 9.78469758775, 10.7014396777
1109 11.4888301734, 9.78469759063, 10.7039469012
1110 11.4987602374, 9.78469759355, 10.7061632544
1111 11.508754516, 9.7846975965, 10.7082947298
1112 11.5189653747, 9.78469664557, 10.7098215752
1113 11.5301299691, 9.7846966484, 10.71117962
1114 11.5401691031, 9.78469760519, 10.7129687275

1115 11.5501180217,9.7846976081,10.7139615144
1116 11.5601509549,9.78469856465,10.714893266
1117 11.5710247625,9.78469856745,10.7161883678
1118 11.5810803073,9.78469857059,10.7172040435
1119 11.591150362,9.78469857339,10.7182588189
1120 11.6014240142,9.78469952998,10.7187003809
1121 11.6124757091,9.78469953294,10.7189884008
1122 11.6224270919,9.78469953601,10.7184734263
1123 11.6327257575,9.78469858522,10.7177801131
1124 11.6434172289,9.78469858809,10.7163896629
1125 11.6536411336,9.78469859109,10.7147340904
1126 11.6637539426,9.78469859406,10.7124605354
1127 11.6738518765,9.78469764322,10.709644338
1128 11.6851043012,9.78469764606,10.7063093402
1129 11.695287982,9.78469764923,10.7025518639
1130 11.7053427212,9.78469765211,10.6986170023
1131 11.7154868832,9.78469765505,10.6947927661
1132 11.7263700012,9.784697658,10.6905632169
1133 11.7365314765,9.78469670747,10.6861343484
1134 11.7464229576,9.7846967104,10.6816644709
1135 11.7569700684,9.78469671324,10.6774892783
1136 11.7676031366,9.78469671627,10.6728029145
1137 11.7776435586,9.78469671932,10.6683568765
1138 11.7876650905,9.7846967222,10.6649150578
1139 11.7983728936,9.78469672507,10.6625566143
1140 11.8085845551,9.78469672803,10.6603431288
1141 11.8184694247,9.78469673098,10.6590508943
1142 11.828445985,9.78469673372,10.6576919023
1143 11.8396064732,9.78469673671,10.6566666969
1144 11.8498754147,9.7846967398,10.6552256886
1145 11.8599188775,9.7846957891,10.6541804567
1146 11.8698337693,9.78469674566,10.6528739177
1147 11.880724417,9.78469674842,10.6517848169
1148 11.8907966884,9.78469675157,10.6503380877
1149 11.9006201513,9.78469580081,10.6492871346
1150 11.9110223083,9.78469580354,10.6480273265
1151 11.921778026,9.78469580656,10.6469105699
1152 11.931896735,9.78469580956,10.6455229698
1153 11.941848929,9.7846958125,10.644526377
1154 11.9522457318,9.78469486171,10.6433352341
1155 11.9623916475,9.7846948646,10.6423415025
1156 11.9724819409,9.78469582122,10.6411722944
1157 11.9824874382,9.78469582404,10.6402825139
1158 11.9935739571,9.78469487337,10.6392182107
1159 12.0037221657,9.78469487645,10.6383293839
1160 12.0136982994,9.78469487939,10.6373099037
1161 12.0236985767,9.78469488218,10.6366003682
1162 12.034309349,9.78469488509,10.635722986
1163 12.0442930488,9.78469584173,10.6350821155

1164 12.0541869169, 9.78469584464, 10.6343020085
1165 12.064686737, 9.78469584751, 10.6338060971
1166 12.0751652075, 9.78469585044, 10.6331165896
1167 12.0851807417, 9.7846958535, 10.6327646836
1168 12.0951327607, 9.78469680998, 10.6322496991
1169 12.1057129372, 9.7846958592, 10.6315916637
1170 12.1158609781, 9.78469681585, 10.6307457539
1171 12.1258434681, 9.7846968188, 10.6302984809
1172 12.1358401207, 9.78469586803, 10.6297072027
1173 12.1470290087, 9.78469587083, 10.6294106107
1174 12.1571742585, 9.78469587409, 10.6289223301
1175 12.1674451092, 9.78469683071, 10.6288927676
1176 12.1776540698, 9.78469683358, 10.6286839142
1177 12.1886466173, 9.78469683655, 10.6287897738
1178 12.1987823388, 9.78469683975, 10.6287220648
1179 12.208889354, 9.78469684258, 10.629035826
1180 12.2195749283, 9.78469684552, 10.6291617135
1181 12.2303555151, 9.78469684873, 10.6295880094
1182 12.2403916745, 9.78469589815, 10.6298121258
1183 12.2504393132, 9.78469590103, 10.6303576313
1184 12.26125554, 9.78469590391, 10.6307124018
1185 12.2711291099, 9.78469590702, 10.6313637659
1186 12.2812328814, 9.78469590986, 10.6317500079
1187 12.291273321, 9.78469591277, 10.6319684036
1188 12.3024088787, 9.78469591565, 10.6320036935
1189 12.3123337347, 9.78469687253, 10.6318577858
1190 12.3224466164, 9.78469687529, 10.6321095605
1191 12.3323409501, 9.78469592455, 10.6330823138
1192 12.343305761, 9.78469688099, 10.6340111983
1193 12.3534187572, 9.78469688419, 10.6354445778
1194 12.3633984039, 9.78469593345, 10.6364983943
1195 12.3738444513, 9.78469593635, 10.6377477147
1196 12.3848342011, 9.78469593917, 10.6388043928
1197 12.3947774554, 9.78469594241, 10.6401557575
1198 12.4048252227, 9.78469594532, 10.6421365477
1199 12.4154791227, 9.78469594824, 10.6451444472
1200 12.4257095912, 9.78469595113, 10.6486463513
1201 12.435612658, 9.78469690782, 10.6527652841
1202 12.4456913527, 9.78469786439, 10.656679177
1203 12.4568030592, 9.78469786724, 10.6609745412
1204 12.4669287521, 9.7846978705, 10.6652956563
1205 12.4768360471, 9.78469787335, 10.6693192243
1206 12.4869002383, 9.78469787617, 10.6727333942
1207 12.4978871755, 9.78469883281, 10.676094159
1208 12.5079703201, 9.7846997897, 10.6791716836
1209 12.517921228, 9.78469979245, 10.6819230504
1210 12.5276678381, 9.78469979539, 10.6841098414
1211 12.5388777843, 9.78469979826, 10.6861469052
1212 12.5489937691, 9.78469980143, 10.6888420074

1213 12.5589893721,9.78469885063,10.6907293445
1214 12.5696154619,9.78469789981,10.6920501983
1215 12.5798949013,9.78469790285,10.6933395817
1216 12.5900643012,9.78469790569,10.6939871404
1217 12.5999913774,9.78469790874,10.6945736636
1218 12.6112778476,9.78469791159,10.6946270815
1219 12.6214415003,9.78469696115,10.6946185115
1220 12.6314815488,9.78469791774,10.6940177069
1221 12.6414842233,9.78469792063,10.693436929
1222 12.6523827724,9.78469792351,10.6924002937
1223 12.6623706762,9.78469792675,10.6912749666
1224 12.6722995741,9.7846979297,10.6897624456
1225 12.6824809583,9.78469793252,10.6878922957
1226 12.6937029604,9.78469793552,10.6860526629
1227 12.7038135417,9.78469793882,10.6836312879
1228 12.7140285531,9.78469794167,10.6816600463
1229 12.7247829204,9.78469794467,10.680282944
1230 12.7349909423,9.78469794772,10.6786349974
1231 12.7451203717,9.78469699705,10.6774095282
1232 12.7551433304,9.78469699992,10.6758445502
1233 12.7663156018,9.78469700288,10.6744378819
1234 12.7762094587,9.78469700605,10.6725438852
1235 12.7863326203,9.78469700887,10.6705659645
1236 12.7963394625,9.78469701186,10.6683171996
1237 12.8071003603,9.78469606103,10.665892004
1238 12.8169879308,9.7846960641,10.663085338
1239 12.8270686027,9.78469606694,10.6608432475
1240 12.8371102512,9.78469606993,10.6583818109
1241 12.8480406869,9.7846960728,10.6558002109
1242 12.857961121,9.78469607593,10.6528657504
1243 12.8680291092,9.78469703255,10.6501143954
1244 12.8780148914,9.78469703542,10.6472514599
1245 12.8888102276,9.78469703838,10.6447947898
1246 12.8989335168,9.78469704145,10.6420768124
1247 12.9090175638,9.78469704435,10.6400054268
1248 12.9200899675,9.78469704744,10.6378653761
1249 12.9301531706,9.78469609689,10.6354897671
1250 12.9400056619,9.78469609976,10.6333068009
1251 12.9500472657,9.78469610268,10.6315911358
1252 12.9608270731,9.78469610553,10.6298220646
1253 12.97085483,9.78469610872,10.627927108
1254 12.9810164068,9.78469611157,10.6261685275
1255 12.9907563627,9.78469611457,10.6248381471

Listing B.2: Results obtained with the script present in listing B.1.