Modular Scenario for CARLA Simulator

This file represents a modular scenario for the CARLA Simulator, a powerful open-source autonomous driving simulator. The scenario defines a set of entities and their behaviors within the simulated environment. It specifies the behavior of the entities by referencing scenarios from other files. This documentation provides a detailed explanation of its structure and its elements.

File Structure

The file follows a hierarchical structure. It is basically similar to the standard structure of an Openscenario file. The only difference can be found in the <ScenarioReference> tag.

The root element is <OpenSCENARIO>, which contains various sub-elements that define the scenario.

The following parts are the descriptions of the tags in the modular scenario file. For more details about the structure, the following link provides the description of the standard tags in any Openscenario file: Overview UML OpenSCENARIO (asam.net)

<FileHeader>
The <FileHeader> element contains information about the XML file, it stores the metadata information about the modular scenario.
The attributes in this tag can be described in this table.

Name	Anotation	Type	Use
revMajor	It represents the major revision number	UnsignedShort	required
revMinor	It represents the minor revision number	UnsignedShort	required
date	Represents the date and time when the file was created or last modified	DateTime	required
description	Provides a brief description of the file content	Strting	required
author	Represents the name of the author	String	required
license	Represents the licensing information	License	optional

Example : <FileHeader revMajor="1" revMinor="0" date="2020-03-20T12:00:00" description="CARLA:ModularScenario" author="Nikolai"/>

<ParameterDeclarations>
The <ParameterDeclarations> element provide general parameters of the scenarios like weather or some characteristics of the ego vehicle.

The details about attributes for this tag and for all the other tags can be found in this link: [Overview UML OpenSCENARIO (asam.net)]

```
Example: <ParameterDeclaration name="exampleName" parameterType="exampleType" value="exampleValue" />
```

<CatalogLocations>
The <CatalogLocations> element specifies the location of catalogs used within the scenario. In this case, it contains a <ManeuverCatalog> element, which refers to a directory path named "catalogs."
The attributes in this tag can be described in this table.

```
Example <CatalogLocations>
  <ManeuverCatalog>
    <Directory path="catalogs"/>
  </ManeuverCatalog>
 </CatalogLocations>
```

<RoadNetwork>
The <RoadNetwork> element defines the road network for the scenario. It consists of a <LogicFile> element specifying the logic file associated with the road network and a <SceneGraphFile> element that can be used to add some modifications to the road network.

```
 example <RoadNetwork>
  <LogicFile filepath="Town05"/>
  <SceneGraphFile filepath=""/>
 </RoadNetwork>
```

<Entities>
The <Entities> element contains the definition of various entities present in the scenario. Each entity is defined within a <ScenarioObject> element, which includes a <Vehicle> element specifying the vehicle properties, such as name, category, performance, bounding box, axles, and properties.

```
 Example<Entities>
  <ScenarioObject name="hero">
   <Vehicle name="vehicle.tesla.model3" vehicleCategory="car">
    <ParameterDeclarations/>
    <Performance maxAcceleration="200" maxDeceleration="10.0" maxSpeed="69.444"/>
    <BoundingBox>
     <Center x="1.5" y="0.0" z="0.9"/>
     <Dimensions height="1.8" length="4.5" width="2.1"/>
    </BoundingBox>
    <Axles>
          <FrontAxle maxSteering="0.5" positionX="3.1" positionZ="0.3" trackWidth="1.8" wheelDiameter="0.6"/>
```

```
            <RearAxle maxSteering="0.0" positionX="0.0" positionZ="0.3" trackWidth="1.8"
wheelDiameter="0.6"/>
      </Axles>
      <Properties>
        <Property name="type" value="ego_vehicle"/>
        <Property name="color" value="0,0,255"/>
      </Properties>
    </Vehicle>
  </ScenarioObject>
  </Entities>
```

<Storyboard>
The <Storyboard> element describes the sequence of actions and events that occur within the scenario.

```
Example<Storyboard>
  <Init>…
  </Init>
  <Story name="MyStory">…
  </Story>
  <StopTrigger/>
</Storyboard>
```

<Init>
The <Init> tag represents the initialization phase of the scenario. It is responsible for setting up the initial state of the entities and environment conditions before the main story begins. Within the <Init> tag, you can define a set of actions using <Actions>.

```
Example<Init>
  <Actions>…
  </Actions>
</Init>
```

<Actions>
<Actions> tag is a crucial sub-tag within the <Init> or <Act> sections of the <Storyboard>. It allows you to define a set of actions that are executed sequentially in the scenario. Let's explore the <Actions> tag and its usage:

```
Example<Actions>
  <GlobalAction>···
  </GlobalAction>
  <Private entityRef="hero">···
  </Private>
  <Private entityRef="vehicle_0">···
  </Private>
  <Private entityRef="vehicle_5">···
  </Private>
</Actions>
```

Within the <Actions> tag, you can include various types of actions, depending on the desired behavior and objectives of the scenario. Some commonly used action types include:

<GlobalAction>
The <GlobalAction> tag represents an action that affects the entire scenario or environment. It is typically used to modify global parameters or settings. For example, you can use a <GlobalAction> to set the time of day, weather conditions, or road conditions for the scenario.

```
Example <GlobalAction>
   <EnvironmentAction>
     <Environment name="Environment1">···
     </Environment>
   </EnvironmentAction>
 </GlobalAction>
```

<Private>
The <Private> tag is used to specify actions related to individual entities or actors within the scenario. You can reference specific entities using the entityRef attribute within the <Private> tag.

```
 Example<Private entityRef="hero">
   <PrivateAction>
     <TeleportAction>
       <Position>
         <WorldPosition h="3.14" x="-32" y="-4" z="0"/>
       </Position>
     </TeleportAction>
   </PrivateAction>
</Private>
```

<Story>

The <Story> tag represents the main story or behavior of the scenario. It consists of one or more <Act> tags, each representing a specific segment or phase of the story.

```
Example<Story name="MyStory">
  <Act name="Behavior">···
  </Act>
</Story>
```

<Act>
The <Act> tag represents an individual segment or phase within the story. It may contain multiple <ManeuverGroup> tags and is often used to organize and structure the sequence of actions.

```
Example<Act name="Behavior">

  <ManeuverGroup maximumExecutionCount="1" name="ManeuverSequence">···
  </ManeuverGroup>

  <ManeuverGroup maximumExecutionCount="1" name="ManeuverSequence">···
  </ManeuverGroup>

  <ManeuverGroup maximumExecutionCount="1" name="ManeuverSequence">···
  </ManeuverGroup>

  <StartTrigger>···
  </StartTrigger>
  <StopTrigger>···
  </StopTrigger>
</Act>
```

<ManeuverGroup>
The <ManeuverGroup> tag defines a group of maneuvers executed within an act. It allows you to group related actions together, such as controlling specific actors/entities in the scenario. It can be configured with a maximum execution count to control how many times the group is executed.

```
Example<ManeuverGroup maximumExecutionCount="1" name="SimpleScenario1Sequence">
  <Actors selectTriggeringEntities="false">···
  </Actors>
  <ScenarioReference scenarioFileName="simple_1.xosc">···
  </ScenarioReference>
</ManeuverGroup>
```

Within a <ManeuverGroup>, you can define the following sub-tags:

<Actors>: Specifies the actors or entities to which the group of maneuvers applies. You can use the <EntityRef> tag to reference the entity by its name or ID.

```
Example<Actors selectTriggeringEntities="false">
  <EntityRef entityRef="vehicle_0"/>
</Actors>
```

<ScenarioReference>: References a simple scenario file to be executed within the current maneuver group. It requires the attribute scenarioFileName to specify the name of the simple scenario file. The details of this attribute can be found in the table.

Name	Annotation	Type	Use
Scenario FileName	It represents the name of the scenario file that contains the details of the scenario being referenced	String	Required
ParameterReference	It represents the attributes that need to be changed in the simple scenario	ParameterReference	Optional

<ParameterReference> tag is used to provide parameters or settings that will be used when executing the referenced scenario . It has a key-value pair.

This pair of attributes can be specified in the following table.

Name	Annotation	Type	Use
key	It represents the name of the parameter being set.	String	Required
value	It represents the value of the parameter being set	String	Required

The use of strings makes the tag flexible because it can be used for different type of parameters and it can accept different data types as values for them.

```
Example<ScenarioReference scenarioFileName="simple_1.xosc">
  <ParameterReference key="value" value="20"/>
  <ParameterReference key="delay" value="10.0"/>
  <ParameterReference key="conditionEdge" value="10.555"/>
</ScenarioReference>
```

<StartTrigger>

The <StartTrigger> tag defines the conditions that trigger the start of the scenario. It allows you to specify one or more conditions that need to be met before the scenario execution begins. Commonly used conditions include time-based conditions or conditions based on entity behavior.

```
Example<StartTrigger>
  <ConditionGroup>···
  </ConditionGroup>
</StartTrigger>
```

<StopTrigger>
The <StopTrigger> tag defines the conditions that trigger the end of the scenario. It specifies the conditions that, when met, indicate that the scenario execution should stop. Similar to the <StartTrigger>, you can define one or more conditions within the <StopTrigger> tag.

```
Example<StopTrigger>
  <ConditionGroup>
    <Condition name="EndCondition" delay="0" conditionEdge="rising">
      <ByEntityCondition>
        <TriggeringEntities triggeringEntitiesRule="any">
          <EntityRef entityRef="hero"/>
        </TriggeringEntities>
        <EntityCondition>
          <TraveledDistanceCondition value="20000.0"/>
        </EntityCondition>
      </ByEntityCondition>
    </Condition>
  </ConditionGroup>
</StopTrigger>
```

These conditions can include events such as reaching a specific distance, time-based conditions, or conditions based on the behavior of entities.