POLY-VERIFICATION USER GUIDE

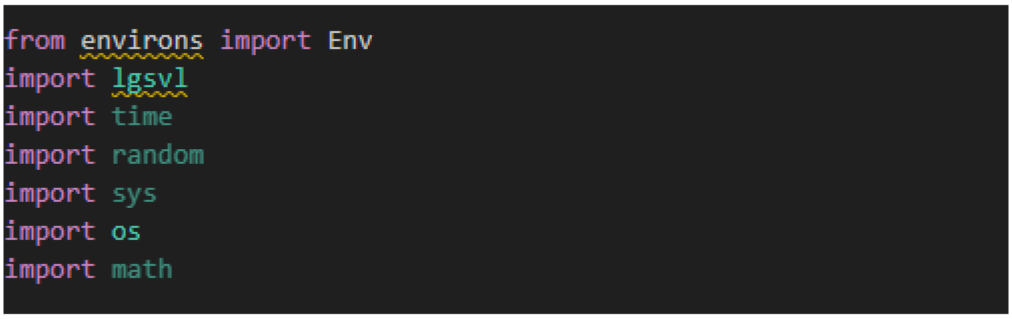
Welcome to PolyVerif

This user guide offers a comprehensive step-by-step tutorial for developing a customized Python script to execute scenarios within the OSSDC simulator.

To run scenarios in Python API mode, follow these steps:

1. Import the necessary module.
2. Set the default weather values and select the scene.
3. Load the simulator.
4. Set the weather and Time of day (if any)
5. Spawn the map location.
6. Add the ego vehicle.
7. Connect to the bridge.
8. Add NPC/Pedestrians.
9. Stop the simulation.

1. Import the necessary module



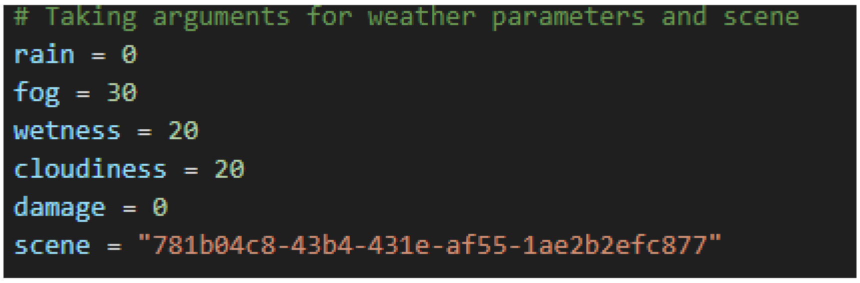
```python
from environs import Env
import lgsvl
import time
import random
import sys
import os
import math
```

2. Set the default weather values and select the scene

You can set rain, fog, wetness, cloudiness, and damage (referring to road damage) as an int (0-100) range. Additionally, set the scene by providing the map name as a string.

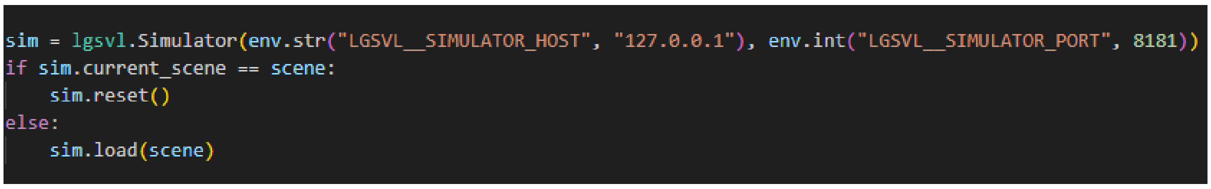
List of scenes are given below:

- **BorregasAve** - "aae03d2a-b7ca-4a88-9e41-9035287a12cc"
- **CubeTown** - "06773677-1ce3-492f-9fe2-b3147e126e27"
- **JTA_R2(Jacksonville Transportation Authority)** - "781b04c8-43b4-431e-af55-1ae2b2efc877"
- **SanFrancisco** - "5d272540-f689-4355-83c7-03bf11b6865f"



```python
# Taking arguments for weather parameters and scene
rain = 0
fog = 30
wetness = 20
cloudiness = 20
damage = 0
scene = "781b04c8-43b4-431e-af55-1ae2b2efc877"
```

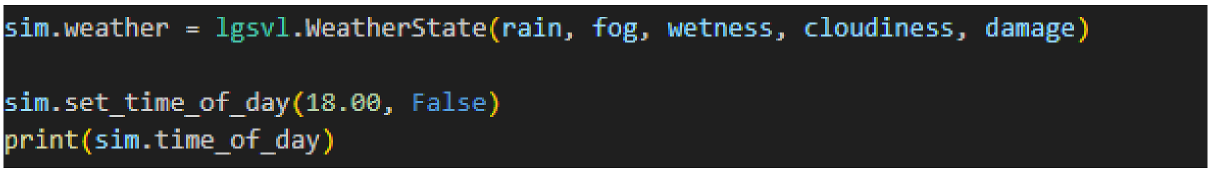
3. Load the simulator



```python
sim = lgsvl.Simulator(env.str("LGSVL__SIMULATOR_HOST", "127.0.0.1"), env.int("LGSVL__SIMULATOR_PORT", 8181))
if sim.current_scene == scene:
    sim.reset()
else:
    sim.load(scene)
```

4. Set the weather and Time of day (if any)

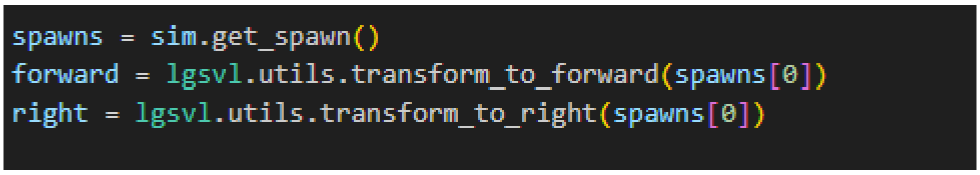
Setting weather and time of day refer image below



```python
sim.weather = lgsvl.WeatherState(rain, fog, wetness, cloudiness, damage)

sim.set_time_of_day(18.00, False)
print(sim.time_of_day)
```

5. Spawn the map location



```python
spawns = sim.get_spawn()
forward = lgsvl.utils.transform_to_forward(spawns[0])
right = lgsvl.utils.transform_to_right(spawns[0])
```

Agents

You can create vehicles and pedestrians by calling the add_agent method of the Simulator object.

The currently available Agent Types are:

1. AgentType.EGO - EGO vehicle
2. AgentType.NPC - NPC vehicle
3. AgentType.PEDESTRIAN – pedestrian

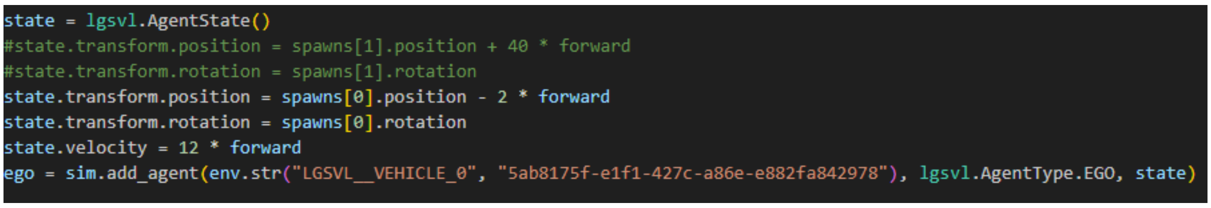
All agents have the following common functionality:

1. state - property to get or set agent state (position, velocity, ...)

2. transform - property to get transform member of the state (shortcut for state. transform)
3. on_collision - method to set a callback function to be called when the agent collides with something (other agent or static obstacle), see callbacks section for more information.

6. Add the ego vehicle

Load the Ego vehicle into the scene as shown in image



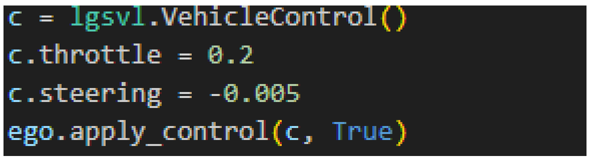
```python
state = lgsvl.AgentState()
#state.transform.position = spawns[1].position + 40 * forward
#state.transform.rotation = spawns[1].rotation
state.transform.position = spawns[0].position - 2 * forward
state.transform.rotation = spawns[0].rotation
state.velocity = 12 * forward
ego = sim.add_agent(env.str("LGSVL__VEHICLE_0", "5ab8175f-e1f1-427c-a86e-e882fa842978"), lgsvl.AgentType.EGO, state)
```

EGO vehicle has following additional functionality:

1. apply_control - method to apply specified throttle, break, steering or other actions to vehicle.
2. connect_bridge - method to connect to ROS or Cyber RT bridge
3. bridge_connected - bool property, True if bridge is connected
4. set_initial_pose - method to publish an initial pose of EGO vehicle to ROS

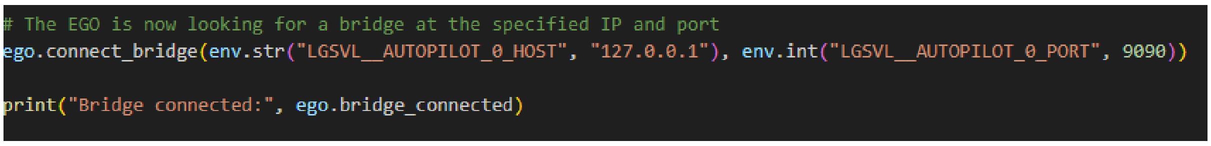
Apply control to ego vehicle

You can apply control to ego vehicle like steering, throttle, breaking etc**.**



```python
c = lgsvl.VehicleControl()
c.throttle = 0.2
c.steering = -0.005
ego.apply_control(c, True)
```

7. Connect to ego bridge



```python
# The EGO is now looking for a bridge at the specified IP and port
ego.connect_bridge(env.str("LGSVL__AUTOPILOT_0_HOST", "127.0.0.1"), env.int("LGSVL__AUTOPILOT_0_PORT", 9090))

print("Bridge connected:", ego.bridge_connected)
```

8. Add NPC/Pedestrians

1. NPC Vehicle

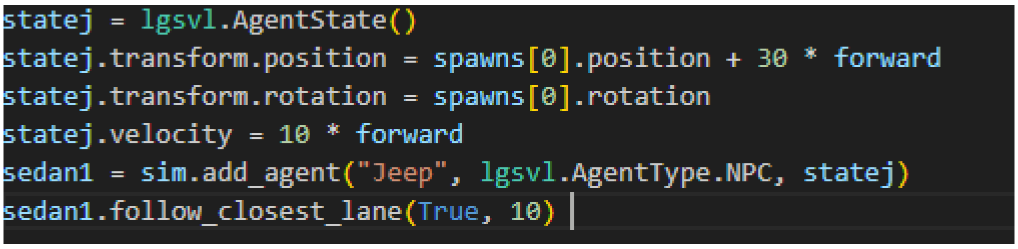
You can create multiple NPC vehicles on the map to drive along the lanes

NPC agents are called by their name directly. Available NPC vehicles:

1. Sedan
2. SUV
3. Jeep
4. Hatchback
5. SchoolBus
6. BoxTruck

If an incorrect name is entered, a Python exception will be thrown.

Load NPC vehicle to scene as shown in image

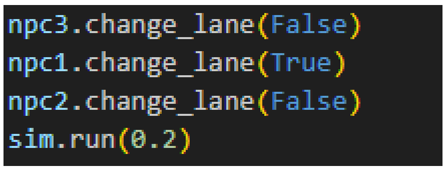


```python
statej = lgsvl.AgentState()
statej.transform.position = spawns[0].position + 30 * forward
statej.transform.rotation = spawns[0].rotation
statej.velocity = 10 * forward
sedan1 = sim.add_agent("Jeep", lgsvl.AgentType.NPC, statej)
sedan1.follow_closest_lane(True, 10)
```

NPC vehicle has the following additional functionality:

1. change_lane - method to make the vehicle change lanes
2. follow - method to make vehicle follow specific waypoints
3. follow_closest_lane - method to make vehicle follow lanes

You can use the Change_lane method like this.



```python
npc3.change_lane(False)
npc1.change_lane(True)
npc2.change_lane(False)
sim.run(0.2)
```

2. Pedestrians

You can create Pedestrian agents that will allow you to create pedestrians on sidewalks and make them walk.

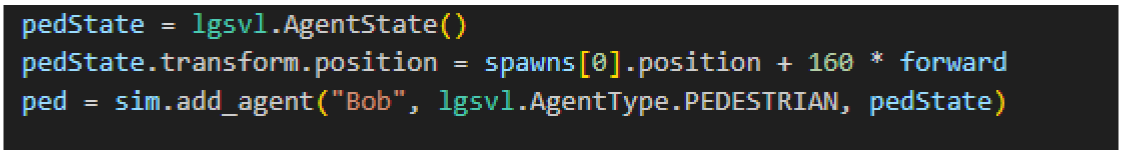
pedestrian agents are also called by their names directly. Available pedestrian types:

3. Bob
4. EntrepreneurFemale
5. Howard
6. Johny
7. Pamela
8. Presley
9. Red

10. Robin
11. Stephen
12. Zoe

If an incorrect name is entered, a Python exception will be thrown.

Add the pedestrian to scene as per shown in image



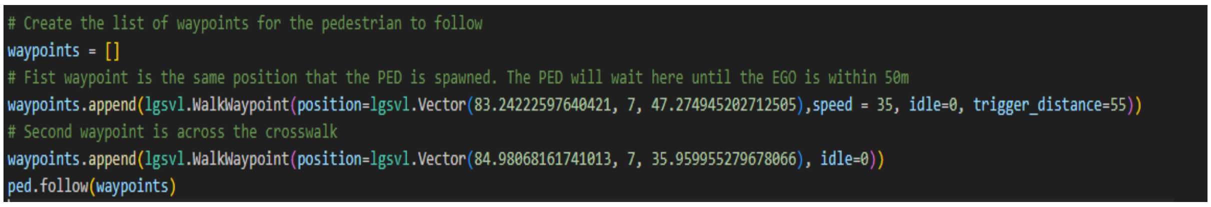
```
pedState = lgsvl.AgentState()
pedState.transform.position = spawns[0].position + 160 * forward
ped = sim.add_agent("Bob", lgsvl.AgentType.PEDESTRIAN, pedState)
```

Pedestrians have the following additional functionality:

1. walk_randomly - method to make pedestrian walk randomly on the sidewalk.
2. follow - method to make pedestrian follow specific waypoints.

You can control the movement of pedestrians either by manually specifying state, or instructing them to follow waypoints or walk randomly.

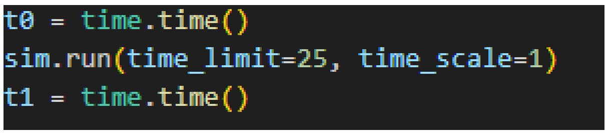
To make pedestrians follow waypoints prepare a list of Walk Waypoint objects and call the follow method for pedestrians:



```
# Create the list of waypoints for the pedestrian to follow
waypoints = []
# Fist waypoint is the same position that the PED is spawned. The PED will wait here until the EGO is within 50m
waypoints.append(lgsvl.WalkWaypoint(position=lgsvl.Vector(83.24222597640421, 7, 47.274945202712505),speed = 35, idle=0, trigger_distance=55))
# Second waypoint is across the crosswalk
waypoints.append(lgsvl.WalkWaypoint(position=lgsvl.Vector(84.98068161741013, 7, 35.959955279678066), idle=0))
ped.follow(waypoints)
```

9. Stop simulation

For stooping simulation, you can simply define time like this **sim.run (20)** or define as per shown in image



```
t0 = time.time()
sim.run(time_limit=25, time_scale=1)
t1 = time.time()
```

Learn More

For further insights and references, explore the provided links:

PythonAPI