国创中心自动驾驶仿真平台

开发者文档

NEVC-SimOne

目录

关于 NEVC-SimOne API

NEVC-SimOne 仿真平台提供多种类型的标准 API 实现外部算法系统与仿真平台的通讯机制，目前开放接口允许只接入感知系统进行目标识别和预测的测试，也可以选择直接跳过感知系统从决策系统输入接入，或者将两者同时接入进行整体测试与训练。仿真平台也提供查询类接口，如高精地图车道级别信息和坐标等。在算法接入后，NEVC-SimOne 允许脚本化的批量创建和运行案例，并自动获取测试报告。仿真平台提供 C++， Python 等多种方式的接入。

Python API reference(1.0)

仿真系统和自动驾驶系统数据通信接口

被测车辆 GPS 和底盘信息

获取主车 GPS 信息

def SoApiSetGpsUpdateCB(cb)

参数说明

入参

- cb

  GPS data update callback function

return(bool)

Get gps data success or not

success:

- SimOne_Data_Gps(class)

  GPS data

  - float posX:Position X on Opendrive (by meter)
  - float posY:Position Y on Opendrive (by meter)
  - float posZ:Position Z on Opendrive (by meter)
  - float oriX:Rotation X on Opendrive (by radian)
  - float oriY:Rotation Y on Opendrive (by radian)
  - float oriZ:Rotation Z on Opendrive (by radian)
  - float velX:MainVehicle Velocity X on Opendrive (by meter)
  - float velY:MainVehicle Velocity Y on Opendrive (by meter)
  - float velZ:MainVehicle Velocity Z on Opendrive (by meter)
  - float throttle:MainVehicle throttle
  - float brake:MainVehicle brake;
  - float steering:MainVehicle Steering angle
  - int gear:MainVehicle gear position
  - float accelX:MainVehicle Acceleration X on Opendrive (by meter)
  - float accelY:MainVehicle Acceleration Y on Opendrive (by meter)
  - float accelZ:MainVehicle Acceleration Z on Opendrive (by meter)

- float angVelX:MainVehicle Angular Velocity X on Opendrive (by meter)
- float angVelY:MainVehicle Angular Velocity Y on Opendrive (by meter)
- float angVelZ:MainVehicle Angular Velocity Z on Opendrive (by meter)
- float wheelSpeedFL:Speed of front left wheel (by meter/sec)
- float wheelSpeedFR:Speed of front right wheel (by meter/sec)
- float wheelSpeedRL:Speed of rear left wheel (by meter/sec)
- float wheelSpeedRR:Speed of rear right wheel (by meter/sec)

def SoGetGps(mainVehicleId, gpsData)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- gpsData(class:SimOne_Data_Gps)

  GPS data

return(bool)

Get gps data success or not

success:

- SimOne_Data_Gps(class)

GPS data

- float posX:Position X on Opendrive (by meter)
- float posY:Position Y on Opendrive (by meter)
- float posZ:Position Z on Opendrive (by meter)
- float oriX:Rotation X on Opendrive (by radian)
- float oriY:Rotation Y on Opendrive (by radian)
- float oriZ:Rotation Z on Opendrive (by radian)
- float velX:MainVehicle Velocity X on Opendrive (by meter)
- float velY:MainVehicle Velocity Y on Opendrive (by meter)
- float velZ:MainVehicle Velocity Z on Opendrive (by meter)
- float throttle:MainVehicle throttle
- float brake:MainVehilce brake;
- float steering:MainVehicle Steering angle
- int gear:MainVehicle gear position
- float accelX:MainVehicle Acceleration X on Opendrive (by meter)
- float accelY:MainVehicle Acceleration Y on Opendrive (by meter)
- float accelZ:MainVehicle Acceleration Z on Opendrive (by meter)
- float angVelX:MainVehicle Angular Velocity X on Opendrive (by meter)
- float angVelY:MainVehicle Angular Velocity Y on Opendrive (by meter)

- float angVelZ:MainVehicle Angular Velocity Z on Opendrive (by meter)
- float wheelSpeedFL:Speed of front left wheel (by meter/sec)
- float wheelSpeedFR:Speed of front right wheel (by meter/sec)
- float wheelSpeedRL:Speed of rear left wheel (by meter/sec)
- float wheelSpeedRR:Speed of rear right wheel (by meter/sec)

获取案例主车路径的终点

获取案例主车的终点

def SoGetTerminalPoint(wayPointsData)

参数说明

入参

- SimOne_Data_WayPoints_Entry(struct)

return(bool)

Success or not

success:

- SimOne_Data_WayPoints_Entry(struct)
  - posX
  - posY

被测车辆周围角色状态

得到仿真场景中的物体的真值

def SoApiSetGroundTruthUpdateCB(cb)

参数说明

入参

- cb

  Obstacle data fetch callback function

return(bool)

Success or not

success:

- SimOne_Data_Obstacle(class)

  Obstacle data

  - int obstacleSize:Obstacle size
  - SimOne_Data_Obstacle_Entry obstacle[SOSM_OBSTACLE_SIZE_MAX]:Obstacle, 100 max
    - int id:Obstacle ID
    - SimOne_Obstacle_Type type:Obstacle Type
      - ESimOne_Obstacle_Type_Unknown = 0
      - ESimOne_Obstacle_Type_Pedestrian = 4
      - ESimOne_Obstacle_Type_Car = 6
      - ESimOne_Obstacle_Type_Static = 7
      - ESimOne_Obstacle_Type_Bicycle = 8

  - ESimOne_Obstacle_Type_RoadMark = 12
  - ESimOne_Obstacle_Type_TrafficSign = 13
  - ESimOne_Obstacle_Type_TrafficLight = 15
  - ESimOne_Obstacle_Type_Rider = 17
  - ESimOne_Obstacle_Type_Truck = 18
  - ESimOne_Obstacle_Type_Bus = 19
  - ESimOne_Obstacle_Type_Train = 20
  - ESimOne_Obstacle_Type_Motorcycle = 21
  - ESimOne_Obstacle_Type_SpeedLimitSign = 26
  - ESimOne_Obstacle_Type_BicycleStatic = 27
  - ESimOne_Obstacle_Type_RoadObstacle = 29
- float theta:Obstacle vertical rotation (by radian)
- float posX:Obstacle Position X on Opendrive (by meter)
- float posY:Obstacle Position Y on Opendrive (by meter)
- float posZ:Obstacle Position Z on Opendrive (by meter)
- float velX:Obstacle Velocity X on Opendrive (by meter)
- float velY:Obstacle Velocity Y on Opendrive (by meter)
- float velZ:Obstacle Velocity Z on Opendrive (by meter)
- float length:Obstacle length
- float width:Obstacle width

- float height:Obstacle height

def SoGetGroundTruth(mainVehicleId, obstacleData)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- obstacleData(class:SimOne_Data_Obstacle)

  Obstacle data

return(bool)

Success or not

success:

- SimOne_Data_Obstacle(class)

  Obstacle data

  - int obstacleSize:Obstacle size
  - SimOne_Data_Obstacle_Entry obstacle[SOSM_OBSTACLE_SIZE_MAX]:Obstacle, 100 max
    - int id:Obstacle ID
    - SimOne_Obstacle_Type type:Obstacle Type
      - ESimOne_Obstacle_Type_Unknown = 0
      - ESimOne_Obstacle_Type_Pedestrian = 4

  - ESimOne_Obstacle_Type_Car = 6
  - ESimOne_Obstacle_Type_Static = 7
  - ESimOne_Obstacle_Type_Bicycle = 8
  - ESimOne_Obstacle_Type_RoadMark = 12
  - ESimOne_Obstacle_Type_TrafficSign = 13
  - ESimOne_Obstacle_Type_TrafficLight = 15
  - ESimOne_Obstacle_Type_Rider = 17
  - ESimOne_Obstacle_Type_Truck = 18
  - ESimOne_Obstacle_Type_Bus = 19
  - ESimOne_Obstacle_Type_Train = 20
  - ESimOne_Obstacle_Type_Motorcycle = 21
  - ESimOne_Obstacle_Type_SpeedLimitSign = 26
  - ESimOne_Obstacle_Type_BicycleStatic = 27
  - ESimOne_Obstacle_Type_RoadObstacle = 29
- float theta:Obstacle vertical rotation (by radian)
- float posX:Obstacle Position X on Opendrive (by meter)
- float posY:Obstacle Position Y on Opendrive (by meter)
- float posZ:Obstacle Position Z on Opendrive (by meter)
- float velX:Obstacle Velocity X on Opendrive (by meter)
- float velY:Obstacle Velocity Y on Opendrive (by meter)

- float velZ:Obstacle Velocity Z on Opendrive (by meter)
- float length:Obstacle length
- float width:Obstacle width
- float height:Obstacle height

被测车辆 Pose 操控

设置主车位置

def SoApiSetPose(mainVehicleId, poseControl)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Pose_Control(class)

  Pose to set

  - float posX:Position X on Opendrive (by meter)
  - float posY:Position Y on Opendrive (by meter)
  - float posZ:Position Z on Opendrive (by meter)
  - float oriX:Rotation X on Opendrive (by radian)
  - float oriY:Rotation Y on Opendrive (by radian)
  - float oriZ:Rotation Z on Opendrive (by radian)
  - bool autoZ = false:Automatically set Z according to scene

return(bool)

Success or not

被测车辆 Drive 操控

主车控制

def SoApiSetDrive(mainVehicleId, driveControl)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Control(class)

  vehicle control data

  - float throttle
  - float brake
  - float steering
  - bool handbrake = false
  - bool isManualGear = false
  - EGearMode gear:gear location
    - EGearMode_Neutral = 0
    - EGearMode_Drive = 1:forward gear for automatic gear
    - EGearMode_Reverse = 2

- EGearMode_Parking = 3
- EGearManualMode_1 = 4:forward gear 1 for manual gear
- EGearManualMode_2 = 5
- EGearManualMode_3 = 6
- EGearManualMode_4 = 7
- EGearManualMode_5 = 8
- EGearManualMode_6 = 9
- EGearManualMode_7 = 10
- EGearManualMode_8 = 11

return(bool)

Success or not

设置主车预警信息

设置主车预警信息

def SoApiSetVehicleEvent(mainVehicleId, vehicleEventInfo)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Vehicle_EventInfo(class)
  - ESimone_Vehicle_EventInfo_Type type

- ESimOne_VehicleEventInfo_Forward_Collision = 0
- ESimOne_VehicleEventInfo_Backward_Collision = 1
- ESimOne_VehicleEventInfo_Left_Turn = 2
- ESimOne_VehicleEventInfo_Right_Turn = 3
- ESimOne_VehicleEventInfo_Forward_Straight = 4
- ESimOne_VehicleEventInfo_Over_Speed = 5

return(bool)

Success or not

获取当前环境相关信息（天气、光照、地面等）

获取当前环境相关信息（天气、光照、地面等）

def SoGetEnvironment(pEnvironment)

参数说明

入参

无

return(bool)

Success or not

success：

- SimOne_Data_Environment(class)
  - timeOfDay(float)

    time of day [0, 2400]

- heightAngle(float)

  height angle [0, 90]
- directionalLight(float)

  light for sun or moon [0, 1]
- ambientLight(float)

  ambient light [0, 1]
- artificialLight(float)

  artificial light [0, 1]
- cloudDensity(float)

  cloud density [0, 1]
- fogDensity(float)

  fog density [0, 1]
- rainDensity(float)

  rain density [0, 1]
- snowDensity(float)

  snow density [0, 1]
- groundHumidityLevel(float)

  ground humidity level [0, 1]
- groundDirtyLevel(float)

  ground dirty level [0, 1]

设置当前环境相关信息（天气、光照、地面等）

设置当前环境相关信息（天气、光照、地面等）

def SoSetEnvironment(pEnvironment)

参数说明

入参

- SimOne_Data_Environment(class)
  - timeOfDay(float)

    time of day [0, 2400]
  - heightAngle(float)

    height angle [0, 90]
  - directionalLight(float)

    light for sun or moon [0, 1]
  - ambientLight(float)

    ambient light [0, 1]
  - artificialLight(float)

    artificial light [0, 1]
  - cloudDensity(float)

    cloud density [0, 1]
  - fogDensity(float)

    fog density [0, 1]

- rainDensity(float)

  rain density [0, 1]

- snowDensity(float)

  snow density [0, 1]

- groundHumidityLevel(float)

  ground humidity level [0, 1]

- groundDirtyLevel(float)

  ground dirty level [0, 1]

return(bool)

Success or not

设置车辆信号灯状态

设置车辆各信号灯的状态

def SoSetSignalLights(mainVehicleId, pSignalLight)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Signal_Lights(struct)

  - SimOne_Signal_Light(struct)

    - ESimOne_Signal_Light_RightBlinker = 1

- ESimOne_Signal_Light_LeftBlinker = (1 << 1)
- ESimOne_Signal_Light_DoubleFlash = (1 << 2)
- ESimOne_Signal_Light_BrakeLight = (1 << 3)
- ESimOne_Signal_Light_FrontLight = (1 << 4)
- ESimOne_Signal_Light_HighBeam = (1 << 5)
- ESimOne_Signal_Light_BackDrive = (1 << 6)

return(bool)

Success or not

获取 SimOneDriver 状态

获取 SimOneDriver 对于主车的控制状态

def SoGetDriverStatus(mainVehicleId, driverStatusData)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

return(bool)

Success or not

success：

- SimOne_Driver_Status(class)
  - ESimOne_Driver_Status_Unknown = 0

- ESimOne_Driver_Status_Controlling = 1
- ESimOne_Driver_Status_Disabled = 2

高精地图运行时接口

获取最临近车道

获取最接近输入点的车道，所属车道优先

def getNearMostLane(pos)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

return

- pyNearMostLane(class)
  - exists(bool)
  - laneId(SimString)

    Lane ID of founded lane. ID with this format roadId_sectionIndex_laneId
  - s, t(double)

    The input point’ s value pair in s-t coordinate system, relative to the found lane
  - s_toCenterLine, t_toCenterLine(double)

is the input point’ s value pair in s-t coordinate system, relative to the found lane’ s owner road’ s center line.

获取临近车道列表

获取临近车道列表

def getNearLanes(pos, distance)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

- distance(double)

  Input range distance

return

- pyNearLanes(class)
  - exists(bool)
  - laneIdList

    Lane IDs of founded lanes. Each ID with this format roadId_sectionIndex_laneId

获取视野范围内所有车道

获取视野范围内所有车道

def getNearLanesWithAngle(pos, distance, headingAngle, angleShift)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

- distance(double)

  Input distance range to search

- headingAngle(double)

  A specified heading direction’ s angle relative to x-axis in 2d-inertial system. headingAngle is defined as radian

- angleShift(double)

  To help define the range of angle as [headingAngle - angleShift, headingAngle + angleShift], and angleShift is defined as radian

return

- pyNearLanes(class)
  - exists(bool)
  - laneIdList

    Lane IDs of founded lanes. Each ID with this format roadId_sectionIndex_laneId

获取离车道左右边缘线的距离

获取离最近车道的左右边缘线的距离信息

def getDistanceToLaneBoundary(pos)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

return

- pyDistanceToLaneBoundary(class)
  - exists(bool)
  - laneId(SimString)

    Lane ID of founded lane. ID with this format roadId_sectionIndex_laneId
  - distToLeft(double)

    The distance to boundaries in 3d space
  - distToRight(double)

    The distance to boundaries in 3d space
  - distToLeft2D(double)

    The distance to boundaries in 3d space
  - distToRight2D(double)

    The distance to boundaries in 3d space

获取车道信息

获取车道信息(包含车道 ID，左右边缘线，虚拟中心线)

def getLaneSample(laneId)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return

- pyLaneSample(class)
  - exists(bool)
  - laneInfo(MLaneInfo)
    - laneName(SimString)

      formatted as roadId_sectionIndex_laneId
    - leftBoundary(SimPoint3DVector)

      sample point list of left boundary
    - rightBoundary(SimPoint3DVector)

      sample point list of right boundary
    - centerLine(SimPoint3DVector)

      sample point list of center line

获取车道连接信息

获取车道连接信息

def getLaneLink(laneId)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return

- pyLaneLink(class)
  - exists(bool)
  - laneLink(MLaneLink)
    - predecessorLaneNameList(SimStringVector)

      lane’s predecessors’ laneName list, formatted as roadId_sectionIndex_laneId
    - successorLaneNameList(SimStringVector)

      lane’s successors’ laneName list, formatted as roadId_sectionIndex_laneId
    - leftNeighborLaneName(SimStringVector)

      lane’s left neighbor’s laneName, formatted as roadId_sectionIndex_laneId
    - rightNeighborLaneName(SimStringVector)

lane’ s right neighbor’ s laneName, formatted as roadId_sectionIndex_laneId

获取车道类型

获取车道类型

def getLaneType(laneId)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return

- pyLaneType(class)
  - exists(bool)
  - laneType(MLaneType)

    Lane type of specified lane
    - none
    - driving
    - stop
    - shoulder
    - biking
    - sidewalk

- border
- restricted
- parking
- bidirectional
- median
- special1
- special2
- special3
- roadWorks
- tram
- rail
- entry
- exit
- offRamp
- onRamp
- mwyEntry
- mwyExit

获取车道宽度

获取车道宽度

def getLaneWidth(laneId, pos)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

- pos(SimPoint3D)

  Input 3d location

return

- pyLaneWidth(class)
  - exists(bool)
  - width(double)

    lane width of specified lane

获取在车道上的 ST 坐标

获取相对于车道虚拟中心线的 ST 坐标

def getLaneST(laneId, pos)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

- pos(SimPoint3D)

Input 3d location

return

- pyST(class)
  - exists(bool)
  - s(double)

    The input point’ s value pair in s-t coordinate system, relative to specified lane.
  - t(double)

    The input point’ s value pair in s-t coordinate system, relative to specified lane.

获取在道路上的 ST 坐标

获取相对于道路参考线的 ST 坐标

def getRoadST(laneId, pos)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId
- pos(SimPoint3D)

  Input 3d location

return

- pySTZ(class)

- exists(bool)
- s(double)

  [s, t] represents the input point’ s value pair in s-t coordinate system
- t(double)

  [s, t] represents the input point’ s value pair in s-t coordinate system,
- z(double)

  z represents the input point’ s height value in localENU.

查询指定车道是否存在于地图之中

查询指定车道是否存在于地图之中

def containsLane(laneId)

参数说明

入参

- laneId(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return(bool)

True if exists, else returns false

获取地图中停车位列表

获取地图中停车位列表

def getParkingSpaceList()

参数说明

入参

- parkingSpaceId(SimString)

  Input parking space ID

return

- parkingSpaceList()
- id
  - pt
    - x
    - y
    - z
  - heading
    - x
    - y
    - z
  - boundaryKnots
    - x
    - y
    - z

- front
  - side
  - type
  - color
  - width
- rear
  - side
  - type
  - color
  - width
- left
  - side
  - type
  - color
  - width
- right
  - side
  - type
  - color
  - width

获取预设规划路径

获取预设规划路径

def getPredefinedRoute

参数说明

入参

无

return(bool)

- route

  Generated route

  - x
  - y
  - z

获取规划路径所途径道路的 ID 列表

获取规划路径所途径道路的 ID 列表

def navigate

参数说明

入参

- inputPts

  Input points that to guide generated route should pass over

return(bool)

- indexOfValidPoints

  Pick valid ones from input points. Valid ones will be used for generting route

- roadIdList

  road id list that are throughed by routing path

根据指定车道 id 和局部坐标获取输入点左右两侧车道标线信息

根据指定车道 id 和局部坐标获取输入点左右两侧车道标线信息

def getRoadMark

参数说明

入参

- pos

  Input 3d location

- id

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return(bool)

- left
  - Left side roadmark
    - sOffset
    - length
    - width

        - type
        - color
    - right

      Right side roadmark
        - sOffset
        - length
        - width
        - type
        - color

获取地图中信号灯列表

获取地图中信号灯列表

def getTrafficLightList

参数说明

入参

无

return

- list

  Traffic light object list
    - id
    - type

- subType
- pt(SimPoint3D)
  - x
  - y
  - z
- heading
- value
- unit
- isDynamic(bool)
- validities(SimVector)

获取地图中交通标志列表

获取地图中交通标志列表

def getTrafficSignList

参数说明

入参

无

return

- list

  Traffic sign object list

  - id

- type
- subType
- pt(SimPoint3D)
  - x
  - y
  - z
- heading
- value
- unit
- isDynamic(bool)
- validities(SimVector)

获取交通灯给定作用车道的关联停止线列表

获取交通灯给定作用车道的关联停止线列表

def getStoplineList

参数说明

入参

- light

  Traffic light object

- laneId

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return

- stoplineList

  Stoplines list that is associated

  - id
  - type
  - pt(SimPoint3D)
    - x
    - y
    - z
  - boundaryKnots(SimPoint3DVector)

获取交通灯给定作用车道的关联人行横道线列表

获取交通灯给定作用车道的关联人行横道线列表

def getCrosswalkList

参数说明

入参

- light

  Traffic light object

- laneId

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return

- crosswalkList

  Crosswalks list that is associated

  - id
  - type
  - pt(SimPoint3D)
    - x
    - y
    - z
  - boundaryKnots(SimPoint3DVector)

获取指定车道所在道路上的网状线列表

获取指定车道所在道路上的网状线列表

def getCrossHatchList

参数说明

入参

- light

  Traffic light object

- laneId

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return

- crossHatchList

Cross hatches list that is associated

- id
- type
- pt(SimPoint3D)
  - x
  - y
  - z
- boundaryKnots(SimPoint3DVector)

获取输入点投影到指定车道中心线上的点和切线方向

获取输入点投影到指定车道中心线上的点和该点处对应中心线的切线方向

def getInertialFromLaneST

参数说明

入参

- inputPt

  Input 3d location

- laneId

  Input lane ID. ID with this format roadId_sectionIndex_laneId

return(bool)

- targetPoint

  The target point that is on specified lane’ s middle line

  - x(double)
  - y(double)
  - z(double)
- dir

  The tangent direction on the target point on lane’ s middle line
  - x
  - y
  - z

得到仿真场景中的交通灯的真值

得到仿真场景中的交通灯的真值

def SoGetTrafficLights

参数说明

入参

- mainVehicleId

  Vehicle index, configure order of web UI, starts from 0
- opendriveLightId

  LightId

return(bool)

- SimOne_Data_TrafficLight(struct)

  light data

- index(int)
- opendriveLightId(int)
- countDown
- status(SimOne_TrafficLight_Status)
    - ESimOne_TrafficLight_Status_Invalid = 0,
    - ESimOne_TrafficLight_Status_Red = 1,
    - ESimOne_TrafficLight_Status_Green = 2,
    - ESimOne_TrafficLight_Status_Yellow = 3,
    - ESimOne_TrafficLight_Status_RedBlink = 4,
    - ESimOne_TrafficLight_Status_GreenBlink = 5,
    - ESimOne_TrafficLight_Status_YellowBlink = 6,
    - ESimOne_TrafficLight_Status_Black = 7

Python Code Recipes(1.0)

仿真系统和自动驾驶系统数据通信接口

被测车辆 GPS 和底盘信息

```
import os
import pySimOneSM
import time
```

```
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

# 方法一：使用回调函数
def SamplegpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

# 方法二：直接调用
def SamplegpsByDirect():
    while(1):
        dataGps = SimOne_Data_Gps()
        if SoGetGps(0, dataGps):
            print(dataGps.posX)
            print(dataGps.posY)
            print(dataGps.posZ)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
```

```
    SoApiSetGpsUpdateCB(SamplegpsCB)
    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            time.sleep(2)

    #SamplegpsByDirect()
```

被测车辆周围角色状态

```
from SimOneSMStruct import *
import time

# 方法一：使用回调函数
def SampleGetGroundTruthByCB():
    def getGroundTruth(mainVehicleId, data):
        # mainVehicleId 默认为 0
        if data:
            ObstacleSize = data[0].obstacleSize
            print(ObstacleSize)

    SoApiStart()
    SoApiSetGroundTruthUpdateCB(getGroundTruth)
    time.sleep(0.1)


# 方法二：直接调用
def SampleGetGroundTruthByByDirect():
```

```
    while (1):
        time.sleep(0.1)
        dataObstacle = SimOne_Data_Obstacle()
        if SoGetGroundTruth(0, dataObstacle):
            print(dataObstacle.obstacleSize)


if __name__ == '__main__':
    SampleGetGroundTruthByCB()
    # SampleGetGroundTruthByByDirect()
    # SampleGetGroundTruthByByDirect()
```

获取案例主车路径的终点

```
from SimOneSMStruct import *

def SampleGetWayPoints():
    while(1):
        wayPoints = SimOne_Data_WayPoints_Entry()
        if SoGetTerminalPoint(wayPoints):
            print(wayPoints.posX)
            print(wayPoints.posY)

if __name__ == '__main__':
    SampleGetWayPoints()
```

被测车辆 Pose 操控

```
import os,sys
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
posX = 0
posY = 0
posZ = 0
timestamp = 0

# 方法一：使用回调函数
def SampleSetGpsUpdateCB():
    def SetGpsUpdateData(mainVehicleId, Data_Gps ):
        global posX, posY, posZ, timestamp
        posX = Data_Gps[0].posX
        posY = Data_Gps[0].posY
        posZ = Data_Gps[0].posZ
        timestamp = Data_Gps[0].timestamp

    SoApiStart()
    SoApiSetGpsUpdateCB(SetGpsUpdateData)

    if posX != 0:
        print("gpsCB:  X:{0} Y:{1} Z:{2}".format(posX, posY, posZ))
```

```
        time.sleep(2)
        pose.timestamp = timestamp
        pose.posX = posX + 10
        pose.posY = posY + 10
        pose.posZ = posZ
        SoApiSetPose(0, pose)


# 方法二：直接调用
def SampleSetGpsByDirect():
    gps = SimOne_Data_Gps()
    pose = SimOne_Data_Pose_Control()
    while (1):
        time.sleep(0.1)
        SoGetGps(0, gps)
        print(gps.posX, gps.posY, gps.posZ)
        pose.timestamp = gps.timestamp
        pose.posX = gps.posX + 10
        pose.posY = gps.posY + 10
        pose.posZ = gps.posZ
        SoApiSetPose(0, pose)
        print(pose.posX, pose.posY, pose.posZ)

if __name__ == '__main__':
    SampleSetGpsUpdateCB()
    #SampleSetGpsByDirect()
```

被测车辆 Drive 操控

```
import os,sys
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
posX = 0
timestamp = 0
control = SimOne_Data_Control()
gps = SimOne_Data_Gps()

# 方法一：使用回调函数
def SampleSetDriveCB():
    def SetGpsUpdateData(mainVehicleId, Data_Gps):
        global posX, timestamp
        posX = Data_Gps[0].posX
        timestamp = Data_Gps[0].timestamp

    SoApiStart()
    SoApiSetGpsUpdateCB(SetGpsUpdateData)

    if posX != 0:
        time.sleep(2)
        control.timestamp = timestamp
        control.throttle = 0.1
```

```
        control.brake = 0.0
        control.steering = 0.0
        control.handbrake = False
        control.isManualGear = False
        control.gear = EGearMode.EGearManualMode_1
        SoApiSetDrive(0, control)

# 方法二：直接调用
def SampleSetDriveByDirect():

    SoApiStart()
    while (1):
        time.sleep(0.1)
        SoGetGps(0, gps)
        control.timestamp = gps.timestamp
        control.throttle = 0.1
        control.brake = 0.0
        control.steering = 0.0
        control.handbrake = False
        control.isManualGear = False
        control.gear = EGearMode.EGearManualMode_1
        SoApiSetDrive(0, control)

if __name__ == '__main__':
    SampleSetDriveCB()
    #SampleSetDriveByDirect()
```

设置主车预警信息

```
import os,sys
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
timestamp = 0

def SampleSetEvent():
    gps = SimOne_Data_Gps()
    vehicleEvent = SimOne_Data_Vehicle_EventInfo()
    SoApiStart()
    while (1):
        SoGetGps(0, gps)
        vehicleEvent.timestamp = gps.timestamp
        vehicleEvent.type = ESimone_Vehicle_EventInfo_Type.ESimOne_VehicleEv
entInfo_Forward_Collision
        SoApiSetVehicleEvent(0, vehicleEvent)

if __name__ == '__main__':
    SampleSetEvent()
```

获取当前环境相关信息（天气、光照、地面等）

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```
from SimOneSMStruct import *
import json


# 获取天气
def SampleGetEnvironment():
    getEnvironmentJson = dict()
    getEnvironment = SimOne_Data_Environment()
    if SoGetEnvironment(getEnvironment):
        getEnvironmentJson.update({'timeOfDay': getEnvironment.timeOfDay})
        getEnvironmentJson.update({'directionalLight': getEnvironment.direction
alLight})
        getEnvironmentJson.update({'artificialLight': getEnvironment.artificialLigh
t})
        getEnvironmentJson.update({'ambientLight': getEnvironment.ambientLig
ht})
        getEnvironmentJson.update({'heightAngle': getEnvironment.heightAngle})
        getEnvironmentJson.update({'cloudDensity': getEnvironment.cloudDensit
y})
        getEnvironmentJson.update({'rainDensity': getEnvironment.rainDensity})
        getEnvironmentJson.update({'snowDensity': getEnvironment.snowDensit
y})
        getEnvironmentJson.update({'groundHumidityLevel': getEnvironment.gro
undHumidityLevel})
        getEnvironmentJson.update({'groundDirtyLevel': getEnvironment.ground
DirtyLevel})
    print(json.dumps(getEnvironmentJson, ensure_ascii=False))
```

```
if __name__ == '__main__':
    SampleGetEnvironment()
```

设置当前环境相关信息（天气、光照、地面等）

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from SimOneSMStruct import *
import json
import time


# 设置天气
def SampleSetEnvironment():
    setEnvironmentJson = dict()
    setEnvironment = SimOne_Data_Environment()

    setEnvironment.timeOfDay = 1200
    setEnvironment.snowDensity = 1.0
    SoSetEnvironment(setEnvironment)

    if SoGetEnvironment(setEnvironment):
        setEnvironmentJson.update({'timeOfDay': setEnvironment.timeOfDay})
        setEnvironmentJson.update({'snowDensity': setEnvironment.snowDensity})

    print(json.dumps(setEnvironmentJson, ensure_ascii=False))
```

```
if __name__ == '__main__':
    SampleSetEnvironment()
```

设置车辆信号灯状态

```
from SimOneSMStruct import *
import time, sys


control = SimOne_Data_Control()
signalLightsData = SimOne_Data_Signal_Lights()
gps = SimOne_Data_Gps()


def sampleSoSetSignalLights():
    if SoGetGps(0, gps):
        control.timestamp = gps.timestamp
    control.throttle = 0.5
    control.gear = EGearMode.EGearManualMode_1

    SoApiSetDrive(0, control)

    signalLightsData.signalLights |= SimOne_Signal_Light.ESimOne_Signal_Light
_LeftBlinker
    SoSetSignalLights(0, signalLightsData)
```

```
if __name__ == '__main__':
    sampleSetSignalLights()
```

获取 SimOneDriver 运行状态

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from SimOneSMStruct import *
import json
import time


def SampleSoGetDriverStatus():
    DriverStatus = SimOne_Data_Driver_Status()
    while(1):
        time.sleep(0.1)
        if SoGetDriverStatus(0, driverStatus):
            status = DriverStatus.driverStatus.value
            print(status)


if __name__ == '__main__':
    SampleSoGetDriverStatus()
```

高精地图运行时接口

查询位置车道

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
```

```
    return info.laneId

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            SampleGetNearMostLane(pos)
            time.sleep(2)
```

获得附近车道

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
```

```
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearLanes(pos, radius):
    print("SampleGetNearLanes:")
    nearLanesInfo = pySimOneSM.getNearLanes(pos, radius)
    if nearLanesInfo.exists == False:
        print("Not exists!")
        return
    lanesInfo = nearLanesInfo.laneIdList
    print("lanesInfo size:", lanesInfo.Size())
    print("lanesInfo id list:")
    for i in range(lanesInfo.Size()):
        element = lanesInfo.GetElement(i)
        print(element.GetString())
        print(",")

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
```

```
        print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
        pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
        SampleGetNearLanes(pos, 5)
        time.sleep(2)
```

获得面前所有车道

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

M_PI = 3.14159265358979323846

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearLanesWithAngle(pos, radius, headingAngle, shiftAngle):
    print("SampleGetNearLanesWithAngle:")
    nearLanesInfo = pySimOneSM.getNearLanesWithAngle(pos, radius, headin
```

```
gAngle, shiftAngle)
    if nearLanesInfo.exists == False:
        print("Not exists!")
        return
    lanesInfo = nearLanesInfo.laneIdList
    print("lanesInfo size:", lanesInfo.Size())
    print("lanesInfo id list:")
    for i in range(lanesInfo.Size()):
        element = lanesInfo.GetElement(i)
        print(element.GetString())
        print(",")

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            headingAngle = 30 / 180 * M_PI
            shiftAngle = 90 / 180 * M_PI
            SampleGetNearLanesWithAngle(pos, 5, headingAngle, shiftAngle)
            time.sleep(2)
```

查询位置两侧车道边缘

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetDistanceToLaneBoundary(pos):
    print("SampleGetDistanceToLaneBoundary:")
    distanceToLaneBoundaryInfo = pySimOneSM.getDistanceToLaneBoundary
(pos);
    if distanceToLaneBoundaryInfo.exists == False:
        print("Not exists!")
        return
    print("laneId:", distanceToLaneBoundaryInfo.laneId.GetString())
    print("distToLeft:", distanceToLaneBoundaryInfo.distToLeft)
```

```
    print("distToRight:", distanceToLaneBoundaryInfo.distToRight)
    print("distToLeft2D:", distanceToLaneBoundaryInfo.distToLeft2D)
    print("distToRight2D:", distanceToLaneBoundaryInfo.distToRight2D)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
      if PosX != 0:
          print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
          pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
          SampleGetDistanceToLaneBoundary(pos)
          time.sleep(2)
```

查询车道信息

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0
```

```
def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetLaneSample(laneId):
    print("SampleGetLaneSample:")
    sampleInfo = pySimOneSM.getLaneSample(laneId)
    if sampleInfo.exists == False:
        print("Not exists!")
        return
    leftBoundary = sampleInfo.laneInfo.leftBoundary
    print("leftBoundary knot size:", leftBoundary.Size())
    print("leftBoundary knot list:")
    for i in range(leftBoundary.Size()):
        element = leftBoundary.GetElement(i)
```

```
        print("(", element.x, ",", element.y, ",", element.z, "),")

    rightBoundary = sampleInfo.laneInfo.rightBoundary
    print("rightBoundary knot size:", rightBoundary.Size())
    print("rightBoundary knot list:")
    for i in range(rightBoundary.Size()):
        element = rightBoundary.GetElement(i)
        print("(", element.x, ",", element.y, ",", element.z, "),")

    centerLine = sampleInfo.laneInfo.centerLine
    print("centerLine knot size:", centerLine.Size())
    print("centerLine knot list:")
    for i in range(centerLine.Size()):
        element = centerLine.GetElement(i)
        print("(", element.x, ",", element.y, ",", element.z, "),")

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
```

```
        SampleGetLaneSample(laneId)
        time.sleep(2)
```

获取车道连接信息

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
```

```
    return info.laneId

def SampleGetLaneLink(laneId):
    print("SampleGetLaneLink:")
    laneLinkInfo = pySimOneSM.getLaneLink(laneId)
    if laneLinkInfo.exists == False:
        print("Not exists!")
        return
    laneLink = laneLinkInfo.laneLink
    predecessorIds = laneLink.predecessorLaneIds
    print("predecessorLaneIds size:", predecessorIds.Size())
    if predecessorIds.Size() > 0:
        print("predecessorLaneIds:")
    for i in range(predecessorIds.Size()):
        element = predecessorIds.GetElement(i)
        print(element.GetString())
    successorIds = laneLink.successorLaneIds
    print("successorLaneIds size:", successorIds.Size())
    if successorIds.Size() > 0:
        print("successorLaneIds:")
    for i in range(successorIds.Size()):
        element = successorIds.GetElement(i)
        print(element.GetString())
    print("leftNeighborLaneId:", laneLink.leftNeighborLaneId.GetString())
    print("rightNeighborLaneId:", laneLink.rightNeighborLaneId.GetString())

if __name__ == '__main__':
```

```
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetLaneLink(laneId)
            time.sleep(2)
```

获取车道类型

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
```

```
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetLaneType(laneId):
    print("SampleGetLaneType:")
    laneType = pySimOneSM.getLaneType(laneId)
    if laneType.exists == False:
        print("Not exists!")
        return
    print("lane type:", laneType.laneType)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
```

```
    if PosX != 0:
        print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
        pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
        laneId = SampleGetNearMostLane(pos)
        SampleGetLaneType(laneId)
        time.sleep(2)
```

获取车道宽度

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
```

```
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetLaneWidth(laneId, pos):
    print("SampleGetLaneWidth:")
    laneWidthInfo = pySimOneSM.getLaneWidth(laneId, pos)
    if laneWidthInfo.exists == False:
        print("Not exists!")
        return
    print("lane width:", laneWidthInfo.width)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetLaneWidth(laneId, pos)
            time.sleep(2)
```

获取在车道上的 ST 坐标

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId
```

```
def SampleGetLaneST(laneId, pos):
    print("SampleGetLaneST:")
    stInfo = pySimOneSM.getLaneST(laneId, pos)
    if stInfo.exists == False:
        print("Not exists!")
        return
    print("[s,t] relative to this lane:", stInfo.s, ",", stInfo.t)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetLaneST(laneId, pos)
            time.sleep(2)
```

获取在道路上的 ST 坐标

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *
```

```
# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetRoadST(laneId, pos):
    print("SampleGetRoadST:")
    stzInfo = pySimOneSM.getRoadST(laneId, pos)
    if stzInfo.exists == False:
        print("Not exists!")
        return
```

```
    print("[s,t] relative to this road:", stzInfo.s, ",", stzInfo.t)
    print("height of input point:", stzInfo.z)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetRoadST(laneId, pos)
            time.sleep(2)
```

根据指定车道 id 和车道上的 ST 坐标获取局部坐标 xyz

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0
```

```
def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetInertialFromLaneST(laneId, s, t):
    print("SampleGetInertialFromLaneST:")
    inertialFromLaneSTInfo = pySimOneSM.getInertialFromLaneST(laneId, s, t)
    if inertialFromLaneSTInfo.exists == False:
        print("Not exists!")
        return
    print("inertial vector: (", inertialFromLaneSTInfo.inertial.x, ",", inertialFromLa
neSTInfo.inertial.y, ",", inertialFromLaneSTInfo.inertial.z, "),")
    print("dir vector: (", inertialFromLaneSTInfo.dir.x, ",", inertialFromLaneSTInfo.
dir.y, ",", inertialFromLaneSTInfo.dir.z, "),")
```

```
if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            s = 0.1
            t = 3.5
            SampleGetInertialFromLaneST(laneId, s, t)
            time.sleep(2)
```

查询指定车道是否存在于地图之中

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0
```

```
def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleContainsLane(laneId):
    print("SampleContainsLane:")
    ret = pySimOneSM.containsLane(laneId)
    print("return state:", ret)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
```

```
    if PosX != 0:
        print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
        pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
        laneId = SampleGetNearMostLane(pos)
        SampleContainsLane(laneId)
        time.sleep(2)
```

获取地图中停车位列表

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

def SampleGetParkingSpaceList():
    print("SampleGetParkingSpaceList:")
    parkingSpaceList = pySimOneSM.getParkingSpaceList()
    print("parkingSpace count:", parkingSpaceList.Size())
    for i in range(parkingSpaceList.Size()):
        parkingSpace = parkingSpaceList.GetElement(i)
        print("parkingSpace id:", parkingSpace.id)
        front = parkingSpace.front
        print("roadMark at which side:", front.side.GetString())
        print("roadMark type:", front.type)
        print("roadMark color:", front.color)
        print("roadMark width:", front.width)
        knots = parkingSpace.boundaryKnots
        print("boundaryKnots count:", knots.Size())
```

```
        knot0 = knots.GetElement(0)
        print("knot0 point: (", knot0.x, ",", knot0.y, ",", knot0.z, ")")

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SampleGetParkingSpaceList():
```

获取预设规划路径

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetPredefinedRoute():
    print("SampleGetPredefinedRoute:")
```

```
    routeInfo = pySimOneSM.getPredefinedRoute()
    if routeInfo.exists == False:
        print("Not exists!")
        return
    print("route point count:", routeInfo.route.Size())
    for i in range(routeInfo.route.Size()):
        pt = routeInfo.route.GetElement(i)
        print("route point: (", pt.x, ",", pt.y, ",", pt.z, "),")

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    #SoApiSetGpsUpdateCB(gpsCB)
    SampleGetPredefinedRoute()
    time.sleep(2)
```

获取规划路径所途径道路的 ID 列表

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0
```

```
def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleNavigate(inputPoints):
    print("SampleNavigate:")
    navigateInfo = pySimOneSM.navigate(inputPoints)
    if navigateInfo.exists == False:
        print("Not exists!")
        return
    print("roadIdList count:", navigateInfo.roadIdList.Size())
    for i in range(navigateInfo.roadIdList.Size()):
        roadId = navigateInfo.roadIdList.GetElement(i)
        print("roadId:", roadId)
    print("indexOfValidPoints count:", navigateInfo.indexOfValidPoints.Size())
```

```
    for i in range(navigateInfo.indexOfValidPoints.Size()):
        index = navigateInfo.indexOfValidPoints.GetElement(i)
        print("index:", index)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    inputPoints = pySimOneSM.pySimPoint3DVector()
    pt1 = pySimOneSM.pySimPoint3D(562.7,531.4,0)
    pt2 = pySimOneSM.pySimPoint3D(837.9,618.4,0)
    inputPoints.AddElement(pt1)
    inputPoints.AddElement(pt2)
    SampleNavigate(inputPoints)
    time.sleep(2)
```

根据指定车道 id 和局部坐标获取局部坐标左右两侧车道标线信息

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
```

```
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetRoadMark(pos, laneId):
    print("SampleGetRoadMark:")
    roadMarkInfo = pySimOneSM.getRoadMark(pos, laneId)
    if roadMarkInfo.exists == False:
        print("Not exists!")
        return
    left = roadMarkInfo.left
    print("left roadMark sOffset:", left.sOffset)
    print("left roadMark type:", left.type)
    print("left roadMark color:", left.color)
```

```
    print("left roadMark width:", left.width)
    right = roadMarkInfo.right
    print("right roadMark sOffset:", right.sOffset)
    print("right roadMark type:", right.type)
    print("right roadMark color:", right.color)
    print("right roadMark width:", right.width)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetRoadMark(pos, laneId)
            time.sleep(2)
```

获取地图中交通灯列表

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *
```

```
# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetTrafficLightList():
    print("SampleGetTrafficLightList:")
    trafficLightList = pySimOneSM.getTrafficLightList()
    print("trafficLight count:", trafficLightList.Size())
    for i in range(trafficLightList.Size()):
        light = trafficLightList.GetElement(i)
        print("light id:", light.id)
        print("light type:", light.type.GetString())
        print("light isDynamic:", light.isDynamic)
        print("validity count:", light.validities.Size())
        for j in range(light.validities.Size()):
            validity = light.validities.GetElement(j)
            print("validity roadId:", validity.roadId)
            print("validity fromLaneId:", validity.fromLaneId)
            print("validity toLaneId:", validity.toLaneId)
```

```
if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    SampleGetTrafficLightList()
    time.sleep(2)
```

获取地图中交通标志牌列表

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetTrafficSignList():
```

```
    print("SampleGetTrafficSignList:")
    trafficSignList = pySimOneSM.getTrafficSignList()
    print("trafficSign count:", trafficSignList.Size())
    for i in range(trafficSignList.Size()):
        sign = trafficSignList.GetElement(i)
        print("sign id:", sign.id)
        print("sign type:", sign.type.GetString())
        print("sign isDynamic:", sign.isDynamic)
        print("validity count:", sign.validities.Size())
        for j in range(sign.validities.Size()):
            validity = sign.validities.GetElement(j)
            print("validity roadId:", validity.roadId)
            print("validity fromLaneId:", validity.fromLaneId)
            print("validity toLaneId:", validity.toLaneId)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    SampleGetTrafficSignList()
    time.sleep(2)
```

获取当前车道所属交通灯管辖的停止线列表

```
import os
import pySimOneSM
```

```
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetStoplineList(light, laneId):
    print("SampleGetStoplineList:")
    stoplineList = pySimOneSM.getStoplineList(light, laneId)
    print("stopline count:", stoplineList.Size())
```

```
    for i in range(stoplineList.Size()):
        stopline = stoplineList.GetElement(i)
        print("stopline id:", stopline.id)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    trafficLightList = pySimOneSM.getTrafficLightList()
    if trafficLightList.Size() < 1:
        print("No traffic light exists!")
        return

    light0 = trafficLightList.GetElement(0)
    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetStoplineList(light0, laneId)
            time.sleep(2)
```

获取当前车道所属交通灯管辖的人行横道线列表

```
import os
import pySimOneSM
```

```
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetCrosswalkList(light, laneId):
    print("SampleGetCrosswalkList:")
    crosswalkList = pySimOneSM.getCrosswalkList(light, laneId)
    print("crosswalk count:", crosswalkList.Size())
```

```
    for i in range(crosswalkList.Size()):
        crosswalk = crosswalkList.GetElement(i)
        print("crosswalk id:", crosswalk.id)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    trafficLightList = pySimOneSM.getTrafficLightList()
    if trafficLightList.Size() < 1:
        print("No traffic light exists!")
        return

    light0 = trafficLightList.GetElement(0)
    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetCrosswalkList(light0, laneId)
            time.sleep(2)
```

获取当前车道所在道路上的网格线列表

```
import os
import pySimOneSM
```

```
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetCrossHatchList(laneId):
    print("SampleGetCrossHatchList:")
    crossHatchList = pySimOneSM.getCrossHatchList(laneId)
    print("crossHatch count:", crossHatchList.Size())
```

```
    for i in range(crossHatchList.Size()):
        crossHatch = crossHatchList.GetElement(i)
        print("crossHatch id:", crossHatch.id)

if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
      if PosX != 0:
          print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
          pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
          laneId = SampleGetNearMostLane(pos)
          SampleGetCrossHatchList(laneId)
          time.sleep(2)
```

根据指定车道 id 和局部坐标获取局部坐标到车道中心线上的投影点信息

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
```

```
PosZ = 0

def gpsCB(mainVehicleId, data):
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetNearMostLane(pos):
    print("SampleGetNearMostLane:")
    info = pySimOneSM.getNearMostLane(pos)
    if info.exists == False:
        print("Not exists!")
        return
    print("lane id:", info.laneId.GetString())
    return info.laneId

def SampleGetLaneMiddlePoint(pos, laneId):
    print("SampleGetLaneMiddlePoint:")
    laneMiddlePointInfo = pySimOneSM.getLaneMiddlePoint(pos, laneId)
    if laneMiddlePointInfo.exists == False:
        print("Not exists!")
        return
    print("targetPoint vector: (", laneMiddlePointInfo.targetPoint.x, ",", laneMidd
lePointInfo.targetPoint.y, ",", laneMiddlePointInfo.targetPoint.z, "),")
    print("dir vector: (", laneMiddlePointInfo.dir.x, ",", laneMiddlePointInfo.dir.y,
",", laneMiddlePointInfo.dir.z, "),")
```

```
if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    while (1):
        if PosX != 0:
            print("gpsCB:  X:{0} Y:{1} Z:{2}".format(PosX, PosY, PosZ))
            pos = pySimOneSM.pySimPoint3D(PosX, PosY, PosZ)
            laneId = SampleGetNearMostLane(pos)
            SampleGetLaneMiddlePoint(pos, laneId)
            time.sleep(2)
```

得到仿真场景中的交通灯的真值

```
import os
import pySimOneSM
import time
from SimOneSMStruct import *

# Global
PosX = 0
PosY = 0
PosZ = 0

def gpsCB(mainVehicleId, data):
```

```
    global PosX, PosY, PosZ
    PosX = data[0].posX
    PosY = data[0].posY
    PosZ = data[0].posZ

def SampleGetTrafficLight():
    print("SampleGetTrafficLightList:")
    trafficLightList = pySimOneSM.getTrafficLightList()
    TrafficLight = SimOne_Data_TrafficLight()
    print("trafficLight count:", trafficLightList.Size())
    for i in range(trafficLightList.Size()):
        light = trafficLightList.GetElement(i)
        print("light id:", light.id)
        if SoGetTrafficLights(0, light.id, TrafficLight):
            print("light status:", TrafficLight.status)


if __name__ == '__main__':
    ret = pySimOneSM.loadHDMap(20)
    print("Load xodr success:", ret)
    SoApiStart()
    SoApiSetGpsUpdateCB(gpsCB)

    SampleGetTrafficLight()
    time.sleep(2)
```

仿真系统和自动驾驶系统数据通信接口

被测车辆 GPS 和底盘信息

获取主车 GPS 信息

SIMONE_SM_API bool SetGpsUpdateCB(void(*cb)(int mainVehicleId, SimOne_Data_Gps* pGps))

参数说明

入参

- cb

  GPS data update callback function

出参

- SimOne_Data_Gps(struct)

  GPS data

  - float posX:Position X on Opendrive (by meter)
  - float posY:Position Y on Opendrive (by meter)
  - float posZ:Position Z on Opendrive (by meter)
  - float oriX:Rotation X on Opendrive (by radian)
  - float oriY:Rotation Y on Opendrive (by radian)
  - float oriZ:Rotation Z on Opendrive (by radian)

- float velX:MainVehicle Velocity X on Opendrive (by meter)
- float velY:MainVehicle Velocity Y on Opendrive (by meter)
- float velZ:MainVehicle Velocity Z on Opendrive (by meter)
- float throttle:MainVehicle throttle
- float brake:MainVehicle brake
- float steering:MainVehicle Steering angle
- int gear:MainVehicle gear position
- float accelX:MainVehicle Acceleration X on Opendrive (by meter)
- float accelY:MainVehicle Acceleration Y on Opendrive (by meter)
- float accelZ:MainVehicle Acceleration Z on Opendrive (by meter)
- float angVelX:MainVehicle Angular Velocity X on Opendrive (by meter)
- float angVelY:MainVehicle Angular Velocity Y on Opendrive (by meter)
- float angVelZ:MainVehicle Angular Velocity Z on Opendrive (by meter)
- float wheelSpeedFL:Speed of front left wheel (by meter/sec)
- float wheelSpeedFR:Speed of front right wheel (by meter/sec)
- float wheelSpeedRL:Speed of rear left wheel (by meter/sec)
- float wheelSpeedRR:Speed of rear right wheel (by meter/sec)

return(bool)

Success or not

SIMONE_SM_API bool GetGps(int mainVehicleId, SimOne_Data_Gps *pGps)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

出参

- SimOne_Data_Gps(struct)

  GPS data

  - float posX:Position X on Opendrive (by meter)
  - float posY:Position Y on Opendrive (by meter)
  - float posZ:Position Z on Opendrive (by meter)
  - float oriX:Rotation X on Opendrive (by radian)
  - float oriY:Rotation Y on Opendrive (by radian)
  - float oriZ:Rotation Z on Opendrive (by radian)
  - float velX:MainVehicle Velocity X on Opendrive (by meter)
  - float velY:MainVehicle Velocity Y on Opendrive (by meter)
  - float velZ:MainVehicle Velocity Z on Opendrive (by meter)
  - float throttle:MainVehicle throttle
  - float brake:MainVehicle brake

- float steering:MainVehicle Steering angle
- int gear:MainVehicle gear position
- float accelX:MainVehicle Acceleration X on Opendrive (by meter)
- float accelY:MainVehicle Acceleration Y on Opendrive (by meter)
- float accelZ:MainVehicle Acceleration Z on Opendrive (by meter)
- float angVelX:MainVehicle Angular Velocity X on Opendrive (by meter)
- float angVelY:MainVehicle Angular Velocity Y on Opendrive (by meter)
- float angVelZ:MainVehicle Angular Velocity Z on Opendrive (by meter)
- float wheelSpeedFL:Speed of front left wheel (by meter/sec)
- float wheelSpeedFR:Speed of front right wheel (by meter/sec)
- float wheelSpeedRL:Speed of rear left wheel (by meter/sec)
- float wheelSpeedRR:Speed of rear right wheel (by meter/sec)

return(bool)

Success or not

被测车辆周围角色状态

得到仿真场景中的物体的真值

SIMONE_SM_API bool SetGroundTruthUpdateCB(void(*cb)(int mainVehicleId, SimOne_Data_Obstacle* pObstacle))

参数说明

入参

- cb

  Obstacle data fetch callback function

出参

- SimOne_Data_Obstacle(struct)

  Obstacle data

  - int obstacleSize:Obstacle size
  - SimOne_Data_Obstacle_Entry obstacle[SOSM_OBSTACLE_SIZE_MAX]:Obstacle, 100 max
    - int id:Obstacle ID
    - SimOne_Obstacle_Type type:Obstacle Type
      - ESimOne_Obstacle_Type_Unknown = 0
      - ESimOne_Obstacle_Type_Pedestrian = 4
      - ESimOne_Obstacle_Type_Car = 6
      - ESimOne_Obstacle_Type_Static = 7
      - ESimOne_Obstacle_Type_Bicycle = 8
      - ESimOne_Obstacle_Type_RoadMark = 12
      - ESimOne_Obstacle_Type_TrafficSign = 13

  - ESimOne_Obstacle_Type_TrafficLight = 15
  - ESimOne_Obstacle_Type_Rider = 17
  - ESimOne_Obstacle_Type_Truck = 18
  - ESimOne_Obstacle_Type_Bus = 19
  - ESimOne_Obstacle_Type_Train = 20
  - ESimOne_Obstacle_Type_Motorcycle = 21
  - ESimOne_Obstacle_Type_SpeedLimitSign = 26
  - ESimOne_Obstacle_Type_BicycleStatic = 27
  - ESimOne_Obstacle_Type_RoadObstacle = 29
- float theta:Obstacle vertical rotation (by radian)
- float posX:Obstacle Position X on Opendrive (by meter)
- float posY:Obstacle Position Y on Opendrive (by meter)
- float posZ:Obstacle Position Z on Opendrive (by meter)
- float velX:Obstacle Velocity X on Opendrive (by meter)
- float velY:Obstacle Velocity Y on Opendrive (by meter)
- float velZ:Obstacle Velocity Z on Opendrive (by meter)
- float length:Obstacle length
- float width:Obstacle width
- float height:Obstacle height

return(bool)

Success or not

SIMONE_SM_API bool GetGroundTruth(int mainVehicleId, SimOne_Data_Obstacle *pObstacle)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

出参

- SimOne_Data_Obstacle(struct)

  Obstacle data

  - int obstacleSize:Obstacle size
  - SimOne_Data_Obstacle_Entry obstacle[SOSM_OBSTACLE_SIZE_MAX]:Obstacle, 100 max
    - int id:Obstacle ID
    - SimOne_Obstacle_Type type:Obstacle Type
      - ESimOne_Obstacle_Type_Unknown = 0
      - ESimOne_Obstacle_Type_Pedestrian = 4
      - ESimOne_Obstacle_Type_Car = 6
      - ESimOne_Obstacle_Type_Static = 7
      - ESimOne_Obstacle_Type_Bicycle = 8
      - ESimOne_Obstacle_Type_RoadMark = 12

  - ESimOne_Obstacle_Type_TrafficSign = 13
  - ESimOne_Obstacle_Type_TrafficLight = 15
  - ESimOne_Obstacle_Type_Rider = 17
  - ESimOne_Obstacle_Type_Truck = 18
  - ESimOne_Obstacle_Type_Bus = 19
  - ESimOne_Obstacle_Type_Train = 20
  - ESimOne_Obstacle_Type_Motorcycle = 21
  - ESimOne_Obstacle_Type_SpeedLimitSign = 26
  - ESimOne_Obstacle_Type_BicycleStatic = 27
  - ESimOne_Obstacle_Type_RoadObstacle = 29
- float theta:Obstacle vertical rotation (by radian)
- float posX:Obstacle Position X on Opendrive (by meter)
- float posY:Obstacle Position Y on Opendrive (by meter)
- float posZ:Obstacle Position Z on Opendrive (by meter)
- float velX:Obstacle Velocity X on Opendrive (by meter)
- float velY:Obstacle Velocity Y on Opendrive (by meter)
- float velZ:Obstacle Velocity Z on Opendrive (by meter)
- float length:Obstacle length
- float width:Obstacle width
- float height:Obstacle height

return(bool)

Success or not

获取案例主车路径的终点

获取案例主车的终点

SIMONE_SM_API bool GetTerminalPoint(SimOne_Data_WayPoints_Entry* terminal)

参数说明

入参

- SimOne_Data_WayPoints_Entry(Pointer)

出参

- SimOne_Data_WayPoints_Entry(struct)
  - posX
  - posY

return(bool)

Success or not

被测车辆 Pose 操控

设置主车位置

SIMONE_SM_API bool SetPose(int mainVehicleId, SimOne_Data_Pose_Control *pPose)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Pose_Control(struct)

  Pose to set

  - float posX:Position X on Opendrive (by meter)
  - float posY:Position Y on Opendrive (by meter)
  - float posZ:Position Z on Opendrive (by meter)
  - float oriX:Rotation X on Opendrive (by radian)
  - float oriY:Rotation Y on Opendrive (by radian)
  - float oriZ:Rotation Z on Opendrive (by radian)
  - bool autoZ = false;// Automatically set Z according to scene

出参

无

return(bool)

Success or not

被测车辆 Drive 操控

主车控制

SIMONE_SM_API bool SetDrive(int mainVehicleId, SimOne_Data_Control *pControl)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Control(struct)

  vehicle control data

  - float throttle
  - float brake
  - float steering
  - bool handbrake = false
  - bool isManualGear = false
  - EGearMode gear:gear location
    - EGearMode_Neutral = 0
    - EGearMode_Drive = 1:forward gear for automatic gear
    - EGearMode_Reverse = 2
    - EGearMode_Parking = 3
    - EGearManualMode_1 = 4:forward gear 1 for manual gear
    - EGearManualMode_2 = 5

- EGearManualMode_3 = 6
- EGearManualMode_4 = 7
- EGearManualMode_5 = 8
- EGearManualMode_6 = 9
- EGearManualMode_7 = 10
- EGearManualMode_8 = 11

出参

无

return(bool)

Success or not

设置主车预警消息

主车预警消息设置

SIMONE_SM_API bool SetVehicleEvent(int mainVehicleId, SimOne_Data_Vehicle_EventInfo *pVehicleEvent);

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

- SimOne_Data_Vehicle_EventInfo(struct)
  - ESimone_Vehicle_EventInfo_Type(struct)

- ESimOne_VehicleEventInfo_Forward_Collision = 0
- ESimOne_VehicleEventInfo_Backward_Collision = 1
- ESimOne_VehicleEventInfo_Left_Turn = 2
- ESimOne_VehicleEventInfo_Right_Turn = 3
- ESimOne_VehicleEventInfo_Forward_Straight = 4
- ESimOne_VehicleEventInfo_Over_Speed = 5

出参

无

return(bool)

Success or not

获取当前环境相关信息（天气、光照、地面等）

获取当前环境相关信息（天气、光照、地面等）

SIMONE_SM_API bool GetEnvironment(SimOne_Data_Environment *pEnvironment)

参数说明

入参

无

出参

- SimOne_Data_Environment(struct)
  - timeOfDay(float)

time of day [0, 2400]

- heightAngle(float)

  height angle [0, 90]

- directionalLight(float)

  light for sun or moon [0, 1]

- ambientLight(float)

  ambient light [0, 1]

- artificialLight(float)

  artificial light [0, 1]

- cloudDensity(float)

  cloud density [0, 1]

- fogDensity(float)

  fog density [0, 1]

- rainDensity(float)

  rain density [0, 1]

- snowDensity(float)

  snow density [0, 1]

- groundHumidityLevel(float)

  ground humidity level [0, 1]

- groundDirtyLevel(float)

ground dirty level [0, 1]

return(bool)

Success or not

设置当前环境相关信息（天气、光照、地面等）

设置当前环境相关信息（天气、光照、地面等）

SIMONE_SM_API bool SetEnvironment(SimOne_Data_Environment *pEnvironment)

参数说明

入参

- SimOne_Data_Environment(struct)
  - timeOfDay(float)

    time of day [0, 2400]
  - heightAngle(float)

    height angle [0, 90]
  - directionalLight(float)

    light for sun or moon [0, 1]
  - ambientLight(float)

    ambient light [0, 1]
  - artificialLight(float)

    artificial light [0, 1]

- cloudDensity(float)

  cloud density [0, 1]
- fogDensity(float)

  fog density [0, 1]
- rainDensity(float)

  rain density [0, 1]
- snowDensity(float)

  snow density [0, 1]
- groundHumidityLevel(float)

  ground humidity level [0, 1]
- groundDirtyLevel(float)

  ground dirty level [0, 1]

出参

无

return(bool)

Success or not

设置车辆信号灯状态

设置车辆各信号灯的状态

SIMONE_SM_API bool SetSignalLights(const int mainVehicleId, SimOne_Data_Signal_Lights* pSignalLights)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

出参

- SimOne_Data_Signal_Lights(struct)
  - SimOne_Signal_Light(struct)
    - ESimOne_Signal_Light_RightBlinker = 1
    - ESimOne_Signal_Light_LeftBlinker = (1 << 1)
    - ESimOne_Signal_Light_DoubleFlash = (1 << 2)
      - ESimOne_Signal_Light_BrakeLight = (1 << 3)
      - ESimOne_Signal_Light_FrontLight = (1 << 4)
      - ESimOne_Signal_Light_HighBeam = (1 << 5)
      - ESimOne_Signal_Light_BackDrive = (1 << 6)

return(bool)

Success or not

获取 SimOneDriver 状态

获取 SimOneDriver 对于主车的控制状态

SIMONE_SM_API bool GetDriverStatus(const int mainVehicleId, SimOne_Data_Driver_Status* pDriverStatus)

参数说明

入参

- mainVehicleId(int)

  Vehicle index, configuration order from scenario editing, starting from 0

出参

- SimOne_Data_Driver_Status(struct)
  - SimOne_Driver_Status(struct)
    - ESimOne_Driver_Status_Unknown = 0
    - ESimOne_Driver_Status_Controlling = 1
    - ESimOne_Driver_Status_Disabled = 2

return(bool)

Success or not

高精地图运行时接口

获取最临近车道

获取最接近输入点的车道，所属车道优先

SIMONE_SM_API bool GetNearMostLane(const SSD::SimPoint3D& pos, SSD::SimString& id, double& s, double& t, double& s_toCenterLine, double& t_toCenterLine)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

出参

- id(SimString)

  Lane ID of founded lane. ID with this format roadId_sectionIndex_laneId

- s, t(double)

  The input point’ s value pair in s-t coordinate system, relative to the found lane

- s_toCenterLine, t_toCenterLine(double)

  is the input point’ s value pair in s-t coordinate system, relative to the found lane’ s owner road’ s center line.

return(bool)

True when any lane is found, else returns false

获取临近车道列表

获取临近车道列表

SIMONE_SM_API bool GetNearLanes(const SSD::SimPoint3D& pos, const double& distance, SSD::SimStringVector& nearLanes)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

- distance(double)

  Input range distance

出参

- nearLanes(SimStringVector)

  Lane IDs of founded lanes. Each ID with this format roadId_sectionIndex_laneId

return(bool)

True when any lane(lanes) is(are) found, else returns false

获取视野范围内所有车道

获取视野范围内所有车道

SIMONE_SM_API bool GetNearLanesWithAngle(const SSD::SimPoint3D& pos, const double& distance,const double& headingAngle, const double& angleShift, SSD::SimStringVector& nearLanes)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

- distance(double)

  Input distance range to search

- headingAngle(double)

  A specified heading direction’ s angle relative to x-axis in 2d-inertial system. headingAngle is defined as radian

- angleShift(double)

  To help define the range of angle as [headingAngle - angleShift, headingAngle + angleShift], and angleShift is defined as radian

出参

- nearLanes(SimStringVector)

  Lane IDs of founded lanes. Each ID with this format roadId_sectionIndex_laneId

return(bool)

True when any lane(lanes) is(are) found, else returns false

获取离车道左右边缘线的距离

获取输入点离最近车道的左右边缘线的距离信息

SIMONE_SM_API bool GetDistanceToLaneBoundary(const SSD::SimPoint3D& pos, SSD::SimString& id, double& distToLeft, double& distToRight, double& distToLeft2D, double& distToRight2D)

参数说明

入参

- pos(SimPoint3D)

  Input 3d location

出参

- id(SimString)

  Lane ID of founded lane. ID with this format roadId_sectionIndex_laneId

- distToLeft(double)

  The distance to boundaries in 3d space

- distToRight(double)

  The distance to boundaries in 3d space

- distToLeft2D(double)

  The distance to boundaries in 3d space

- distToRight2D(double)

  The distance to boundaries in 2d space(ignore height)

return(bool)

True if near most lane is found, else returns false

获取车道信息

获取车道信息(包含车道 ID，左右边缘线，虚拟中心线)

SIMONE_SM_API bool GetLaneSample(const SSD::SimString &id, HDMapStandalone::MLaneInfo& info)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- info(MLaneInfo)

  Lane information(HDMapStandalone::MLaneInfo) of specified lane

  - laneName(SimString)

    formatted as roadId_sectionIndex_laneId

  - leftBoundary(SimPoint3DVector)

    sample point list of left boundary

  - rightBoundary(SimPoint3DVector)

    sample point list of right boundary

  - centerLine(SimPoint3DVector)

    sample point list of center line

return(bool)

True if specified lane exists in the map, else returns false

获取车道连接信息

获取车道连接信息

SIMONE_SM_API bool GetLaneLink(const SSD::SimString& id, HDMapStandalone::MLaneLink& laneLink)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- laneLink(MLaneLink)

  Lane link information(HDMapStandalone::MLaneLink) of specified lane

  - predecessorLaneNameList(SimStringVector)

    lane’s predecessors’ laneName list, formatted as roadId_sectionIndex_laneId

  - successorLaneNameList(SimStringVector)

    lane’s successors’ laneName list, formatted as roadId_sectionIndex_laneId

  - leftNeighborLaneName(SimStringVector)

    lane’s left neighbor’s laneName, formatted as roadId_sectionIndex_laneId

- rightNeighborLaneName(SimStringVector)

  lane’ s right neighbor’ s laneName, formatted as roadId_sectionIndex_laneId

return(bool)

True if specified lane exists in the map, else returns false

获取车道类型

获取车道类型

SIMONE_SM_API bool GetLaneType(const SSD::SimString& id, HDMapStandalone::MLaneType& laneType)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- laneType(MLaneType)

  Lane type of specified lane

  - none
  - driving
  - stop
  - shoulder
  - biking

- sidewalk
- border
- restricted
- parking
- bidirectional
- median
- special1
- special2
- special3
- roadWorks
- tram
- rail
- entry
- exit
- offRamp
- onRamp
- mwyEntry
- mwyExit

return(bool)

True if specified lane exists in the map, else returns false

获取车道宽度

获取车道宽度

SIMONE_SM_API bool GetLaneWidth(const SSD::SimString& id, const SSD::SimPoint3D& pos, double& width)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

- pos(SimPoint3D)

  Input 3d location

出参

- width(double)

  lane width of specified lane

return(bool)

True if specified lane exists in the map, else returns false

获取在车道上的 ST 坐标

获取相对于车道虚拟中心线的 ST 坐标

SIMONE_SM_API bool GetLaneST(const SSD::SimString& id, const SSD::SimPoint3D& pos, double& s, double& t)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

- pos(SimPoint3D)

  Input 3d location

出参

- s(double)

  The input point’ s value pair in s-t coordinate system, relative to specified lane.

- t(double)

  The input point’ s value pair in s-t coordinate system, relative to specified lane.

return(bool)

True if specified lane exists in the map, else returns false

获取在道路上的 ST 坐标

获取相对于道路参考线的 ST 坐标

SIMONE_SM_API bool GetRoadST(const SSD::SimString& id, const SSD::SimPoint3D& pos, double& s, double& t, double& z)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

- pos(SimPoint3D)

  Input 3d location

出参

- s, t, z(double)

  [s, t] represents the input point’ s value pair in s-t coordinate system, and z represents the input point’ s height value in localENU.

查询指定车道是否存在于地图之中

查询指定车道是否存在于地图之中

SIMONE_SM_API bool ContainsLane(const SSD::SimString& id)

参数说明

入参

- id(SimString)

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

无

return(bool)

True if exists, else returns false

获取地图中停车位列表

获取地图中停车位列表

SIMONE_SM_API void GetParkingSpaceList(SSD::SimVector<HDMapStandalone::MParkingSpace>& parkingSpaceList)

参数说明

入参

- 无

出参

- parkingSpaceList(SimVector)
- id
  - pt
    - x
    - y
    - z
  - heading
    - x
    - y
    - z
  - boundaryKnots
    - x
    - y
    - z

- front
  - side
  - type
  - color
  - width
- rear
  - side
  - type
  - color
  - width
- left
  - side
  - type
  - color
  - width
- right
  - side
  - type
  - color
  - width

True if specified parking space exists in the map, else returns false

获取预设规划路径

获取预设规划路径

SIMONE_SM_API bool GetPredefinedRoute(SSD::SimPoint3DVector& route);

参数说明

入参

无

出参

- route

  Generated route

  - x
  - y
  - z

return(bool)

True if exists, else returns false

获取规划路径所途径道路的 ID 列表

获取规划路径所途径道路的 ID 列表

SIMONE_SM_API bool Navigate(const SSD::SimPoint3DVector& inputPoints, SSD::SimVector<int>& indexOfValidPoints, SSD::SimVector<long>& roadIdList)

参数说明

入参

- inputPoints

  Input points that to guide generated route should pass over

出参

- indexOfValidPoints

  Pick valid ones from input points. Valid ones will be used for generting route

- roadIdList(SimVector)

  road id list that are throughed by routing path

return(bool)

True if exists, else returns false

根据指定车道 id 和局部坐标获取输入点左右两侧车道标线信息

根据指定车道 id 和局部坐标获取输入点左右两侧车道标线信息

SIMONE_SM_API bool GetRoadMark(const SSD::SimPoint3D& pos, const SSD::SimString& id, HDMapStandalone::MRoadMark& left, HDMapStandalone::MRoadMark& right)

参数说明

入参

- pos

  Input 3d location

- id

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- left

  Left side roadmark

  - sOffset
  - length
  - width
  - type
  - color

- right

  Right side roadmark

  - sOffset
  - length
  - width
  - type
  - color

return(bool)

True if exists, else returns false

获取地图中信号灯列表

获取地图中信号灯列表

SIMONE_SM_API void GetTrafficLightList(SSD::SimVector<HDMapStandalone::MSignal>& list)

参数说明

入参

无

出参

- list

  Traffic light object list

  - id
  - type
  - subType
  - pt(SimPoint3D)
    - x
    - y
    - z
  - heading

  - value
  - unit
  - isDynamic(bool)
  - validities(SimVector)

获取地图中交通标志列表

获取地图中交通标志列表

SIMONE_SM_API void GetTrafficSignList(SSD::SimVector<HDMapStandalone::MSignal>& list)

参数说明

入参

无

出参

- list

  Traffic sign object list
  - id
  - type
  - subType
  - pt(SimPoint3D)
    - x
    - y

        - z
    - heading
    - value
    - unit
    - isDynamic(bool)
    - validities(SimVector)

获取交通灯给定作用车道的关联停止线列表

获取交通灯给定作用车道的关联停止线列表

SIMONE_SM_API void GetStoplineList(const HDMapStandalone::MSignal& light, const SSD::SimString& id, SSD::SimVector<HDMapStandalone::MObject>& stoplineList)

参数说明

入参

- light

  Traffic light object

- id

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- stoplineList

  Stoplines list that is associated

    - id

  - type
  - pt(SimPoint3D)
    - x
    - y
    - z
  - boundaryKnots(SimPoint3DVector)

获取交通灯给定作用车道的关联人行横道线列表

获取交通灯给定作用车道的关联人行横道线列表

SIMONE_SM_API void GetCrosswalkList(const HDMapStandalone::MSignal& light, const SSD::SimString& id, SSD::SimVector<HDMapStandalone::MObject>& crosswalkList)

参数说明

入参

- light

  Traffic light object

- id

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- crosswalkList

  Crosswalks list that is associated

  - id

- type
- pt(SimPoint3D)
  - x
  - y
  - z
- boundaryKnots(SimPoint3DVector)

获取指定车道所在道路上的网状线列表

获取指定车道所在道路上的网状线列表

SIMONE_SM_API void GetCrossHatchList(const SSD::SimString& id, SSD::SimVector<HDMapStandalone::MObject>& crossHatchList)

参数说明

入参

- light

  Traffic light object

- id

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- crossHatchList

  Cross hatches list that is associated

  - id

- type
- pt(SimPoint3D)
  - x
  - y
  - z
- boundaryKnots(SimPoint3DVector)

获取输入点投影到指定车道中心线上的点和切线方向

获取输入点投影到指定车道中心线上的点和该点处对应中心线的切线方向

SIMONE_SM_API bool GetLaneMiddlePoint(const SSD::SimPoint3D& inputPt, const SSD::SimString& id, SSD::SimPoint3D& targetPoint, SSD::SimPoint3D& dir)

参数说明

入参

- inputPt

  Input 3d location

- id

  Input lane ID. ID with this format roadId_sectionIndex_laneId

出参

- targetPoint

  The target point that is on specified lane’ s middle line

  - x(double)

  - y(double)
  - z(double)
- dir

  The tangent direction on the target point on lane’ s middle line
  - x
  - y
  - z

return(bool)

True if exists, else returns false

得到仿真场景中的交通灯的真值

得到仿真场景中的交通灯的真值

SIMONE_SM_API bool GetTrafficLights(int mainVehicleId, int opendriveLightId, SimOne_Data_TrafficLight* pTrafficLight)

参数说明

入参

- mainVehicleId

  Vehicle index, configure order of web UI, starts from 0
- opendriveLightId

  LightId

出参

- SimOne_Data_TrafficLight(struct)

  light data

  - index(int)
  - opendriveLightId(int)
  - countDown
  - status(SimOne_TrafficLight_Status)
    - ESimOne_TrafficLight_Status_Invalid = 0,
    - ESimOne_TrafficLight_Status_Red = 1,
    - ESimOne_TrafficLight_Status_Green = 2,
    - ESimOne_TrafficLight_Status_Yellow = 3,
    - ESimOne_TrafficLight_Status_RedBlink = 4,
    - ESimOne_TrafficLight_Status_GreenBlink = 5,
    - ESimOne_TrafficLight_Status_YellowBlink = 6,
    - ESimOne_TrafficLight_Status_Black = 7

return(bool)

True if exists, else returns false

仿真系统和自动驾驶系统数据通信接口

被测车辆 GPS 和底盘信息

```
#include "SimOneSMAPI.h"
#include <vector>
#include <chrono>
#include <thread>

int main(int argc, char* argv[])
{
    // 方法一：通过回调方法获得数据更新
    SimOneSM::Start();
    SimOneSM::SetGpsUpdateCB([](int mainVehicleId, SimOne_Data_Gps *pGps)
{
        printf("gpsCB: V:%d GPS:%d X:%.2f Y:%.2f\n", mainVehicleId, pGps->time
stamp, pGps->posX, pGps->posY);
    });
    while (1)
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;

    // 方法二：直接获取最新的数据
```

```
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, pGps.get()))
        {
            printf("Fetch GPS failed\n");
        }
        printf("gps: GPS timestamp:%d X:%.2f Y:%.2f\n", pGps->timestamp, pGps
->posX, pGps->posY);
    }
    return 0;
}
```

被测车辆周围角色状态

```
#include "SimOneSMAPI.h"
#include <vector>
#include <chrono>
#include <thread>

int main(int argc, char* argv[])
{
    // 方法一：通过回调方法获得数据更新
    SimOneSM::Start();
    SimOneSM::SetGroundTruthUpdateCB([](int mainVehicleId, SimOne_Data_O
bstacle *pObstacle){
        printf("GroundTruthUpdateCB: V:%d OBS:%d N:%d\n", mainVehicleId, pO
```

```
bstacle->obstacleSize);
    });

    while (1)
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;

    // 方法二：直接获取最新的数据
    std::unique_ptr<SimOne_Data_Obstacle> pObstacle = std::make_unique<Si
mOne_Data_Obstacle>();
    while (1)
    {
        if (!SimOneSM::GetGroundTruth(0/*vehicleId*/, pObstacle.get()))
        {
            printf("Get obstacle failed\n");
        }
        printf("GetGroundTruth success: OBS:%d N:%d\n", pObstacle->obstacleSi
ze);
    }
    return 0;
}
```

获取案例主车路径的终点

```
#include "SimOneSMAPI.h"
#include <iostream>
```

```
SimOne_Data_WayPoints_Entry terminal;

int main(int argc, char* argv[])
{
    if (SimOneSM::GetTerminalPoint(&terminal))
    {
        std::cout<<terminal.posX<<std::endl;
        std::cout<<terminal.posY<<std::endl;
    }

}
```

被测车辆 Pose 操控

```
#include "SimOneSMAPI.h"
#include <vector>
#include <chrono>
#include <thread>

int main(int argc, char* argv[])
{
    // 方法一：通过回调方法获得数据更新
    SimOneSM::Start();
    SimOneSM::SetGpsUpdateCB([](int mainVehicleId, SimOne_Data_Gps *pGps)
{
        printf("gpsCB: V:%d GPS:%d X:%.2f Y:%.2f\n", mainVehicleId, pGps->time
stamp, pGps->posX, pGps->posY);
```

```
        std::unique_ptr<SimOne_Data_Pose_Control> posePtr = std::make_uniqu
e<SimOne_Data_Pose_Control>();
        // must need
        posePtr->timestamp = pGps->timestamp;
        // Shift along X axis
        posePtr->posX = pGps->posX + 0.01;
        posePtr->posY = pGps->posY;
        posePtr->posZ = pGps->posZ;
        if (!SimOneSM::SetPose(0, posePtr.get()))
        {
            printf("Set Pose failed\n");
        }
    });
    while (1)
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;

    // 方法二：直接获取最新的数据
    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
```

```
        }

        // Shift along X axis
        std::unique_ptr<SimOne_Data_Pose_Control> posePtr = std::make_uniqu
e<SimOne_Data_Pose_Control>();
        // must need
        posePtr->timestamp = gpsPtr->timestamp;
        posePtr->posX = gpsPtr->posX + 0.01;
        posePtr->posY = gpsPtr->posY;
        posePtr->posZ = gpsPtr->posZ;
        if (!SimOneSM::SetPose(0/*vehicleId*/, posePtr.get()))
        {
            printf("Set Pose failed\n");
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;
}
```

被测车辆 Drive 操控

```
#include "SimOneSMAPI.h"
#include <vector>
#include <chrono>
#include <thread>

int main(int argc, char* argv[])
{
```

```
    // 方法一：通过回调方法获得数据更新
    SimOneSM::Start();
    SimOneSM::SetGpsUpdateCB([](int mainVehicleId, SimOne_Data_Gps *pGps)
{
        printf("gpsCB: V:%d GPS:%d X:%.2f Y:%.2f\n", mainVehicleId, pGps->time
stamp, pGps->posX, pGps->posY);
        std::unique_ptr<SimOne_Data_Control> controlPtr = std::make_unique<S
imOne_Data_Control>();
        // must need
        controlPtr->timestamp = pGps->timestamp;
        controlPtr->throttle = 0.5;
        controlPtr->steering = 0.0f;
        controlPtr->gear = EGearMode::EGearManualMode_1;

        if (!SimOneSM::SetDrive(0/*vehicleId*/, controlPtr.get()))
        {
            printf("SetDrive failed\n");
        }
    });
    while (1)
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;
```

```
    // 方法二：直接获取最新的数据
    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_Data_Gps>();

    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
        }

        std::unique_ptr<SimOne_Data_Control> controlPtr = std::make_unique<SimOne_Data_Control>();
        // must need
        controlPtr->timestamp = gpsPtr->timestamp;
        controlPtr->throttle = 0.5;
        controlPtr->steering = 0.0f;
        controlPtr->gear = EGearMode::EGearManualMode_1;
        if (!SimOneSM::SetDrive(0/*vehicleId*/, controlPtr.get()))
        {
            printf("SetDrive failed\n");
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;
}
```

设置主车预警信息

```
#include "SimOneSMAPI.h"
#include <memory>

int main(int argc, char* argv[]) {

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_Data_Gps>();

    std::unique_ptr<SimOne_Data_Vehicle_EventInfo> vehicleEventPtr = std::make_unique<SimOne_Data_Vehicle_EventInfo>();

    while(1){
        if (SimOneSM::GetGps(0, gpsPtr.get())) {
            vehicleEventPtr->timestamp = gpsPtr->timestamp;
        }
        else {
                printf("Fetch GPS failed\n");
        }
        vehicleEventPtr->type = ESimone_Vehicle_EventInfo_Type::ESimOne_VehicleEventInfo_Forward_Collision;
        SimOneSM::SetVehicleEvent(0, vehicleEventPtr.get());
    }
    return 0;
}
```

获取当前环境相关信息（天气、光照、地面等）

```
#include "SimOneSMAPI.h"
#include <vector>
#include <chrono>
#include <thread>

int main(int argc, char* argv[])
{
    std::unique_ptr<SimOne_Data_Environment > environmentPtr = std::make_
unique<SimOne_Data_Environment >();

    while (1)
    {
        if (!SimOneSM::GetEnvironment(environmentPtr.get()))
        {
            std::cout << "Get Environment failed" << std::endl;
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }

        std::cout << "timeOfDay:" << environmentPtr->timeOfDay << " heightA
ngle:" << environmentPtr->heightAngle
            << " directionalLight:" << environmentPtr->directionalLight << " ambi
entLight:" << environmentPtr->ambientLight
            << " artificialLight:" << environmentPtr->artificialLight << " cloudDen
sity:" << environmentPtr->cloudDensity
```

```
        << " fogDensity:" << environmentPtr->fogDensity << " rainDensity:"
<< environmentPtr->rainDensity
        << " snowDensity:" << environmentPtr->snowDensity << " groundHu
midityLevel:" << environmentPtr->groundHumidityLevel
        << " groundDirtyLevel:" << environmentPtr->groundDirtyLevel << st
d::endl;

    std::this_thread::sleep_for(std::chrono::milliseconds(100));
  }
  return 0;
}
```

设置当前环境相关信息（天气、光照、地面等）

```
#include "SimOneSMAPI.h"
#include <vector>
#include <chrono>
#include <thread>

int main(int argc, char* argv[])
{
  std::unique_ptr<SimOne_Data_Environment > environmentPtr = std::make_
unique<SimOne_Data_Environment >();

  while (1)
  {
    if (!SimOneSM::GetEnvironment(environmentPtr.get()))
    {
```

```
            std::cout << "Get Environment failed << std::endl;
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }

        // 设置时间
        environmentPtr->directionalLight += 0.05;
        if (environmentPtr->directionalLight > 1)
        {
            environmentPtr->directionalLight = 0;
        }

        SimOneSM::SetEnvironment(environmentPtr.get()

        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;
}
```

设置车辆信号灯状态

```
#include "SimOneSMAPI.h"
#include <iostream>

int main(int argc, char* argv[])
{
    SimOneSM::Start();
```

```
    std::unique_ptr<SimOne_Data_Signal_Lights> pSignalLights = std::make_unique<SimOne_Data_Signal_Lights>();
    std::unique_ptr<SimOne_Data_Control> pdriver = std::make_unique<SimOne_Data_Control>();
    std::unique_ptr<SimOne_Data_Gps> pGPS = std::make_unique<SimOne_Data_Gps>();

    SimOneSM::GetGps(0, pGPS.get());
    pdriver->timestamp = pGPS->timestamp;

    pdriver->gear = EGearMode::EGearManualMode_1;
    pdriver->throttle = 0.5;
    SimOneSM::SetDrive(0, pdriver.get());

    while (1) {
      //pSignalLights->signalLights |= SimOne_Signal_Light::ESimOne_Signal_Light_DoubleFlash;
      //pSignalLights->signalLights |= SimOne_Signal_Light::ESimOne_Signal_Light_FrontLight;
      //pSignalLights->signalLights |= SimOne_Signal_Light::ESimOne_Signal_Light_HighBeam;
      //pSignalLights->signalLights |= SimOne_Signal_Light::ESimOne_Signal_Light_BackDrive;
      //pSignalLights->signalLights |= SimOne_Signal_Light::ESimOne_Signal_Light_BrakeLight;
      //pSignalLights->signalLights |= SimOne_Signal_Light::ESimOne_Signal_Light_LeftBlinker;
```

```
        pSignalLights->signalLights = SimOne_Signal_Light::ESimOne_Signal_Light_RightBlinker;
        SimOneSM::SetSignalLights(0, pSignalLights.get());
    }


}
```

获取 SimOneDriver 运行状态

```
#include "SimOneSMAPI.h"
#include <thread>
#include <iostream>
#include <string>

int main(int argc, char* argv[]) {
    SimOneSM::Start();
    std::unique_ptr<SimOne_Data_Driver_Status> pDriverStatus = std::make_unique<SimOne_Data_Driver_Status>();

    while (1) {
        if (SimOneSM::GetDriverStatus(0, pDriverStatus.get())) {
            std::cout << "get simoneDriver status" << std::endl;
            std::cout << pDriverStatus->driverStatus<<std::endl;
            std::this_thread::sleep_for(std::chrono::microseconds(2000));
        }
    }
```

```
}
```

高精地图运行时接口

查询位置车道

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
#include <chrono>
#include <thread>
#include <iostream>

using namespace SSD;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
```

```
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
    return laneId;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }


    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
```

```
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);

        SSD::SimString laneId = SampleGetNearMostLane(pos);
        if (laneId.GetString() != nullptr)
        {
            std::cout << "Get NearMostLane success, laneId:" << laneId.GetString()
 << std::endl;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获得附近车道

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
#include <chrono>
```

```cpp
#include <thread>
#include <iostream>

using namespace SSD;

void SampleGetNearLanes(const SimPoint3D& pos, const double& distance)
{
    SimStringVector nearLanes;
    if (!SimOneSM::GetNearLanes(pos, distance, nearLanes))
    {
        std::cout << "No lanes(lane) are(is) found." << std::endl;
        return;
    }
    std::cout << "lane id list:";
    for (unsigned int i = 0; i < nearLanes.size(); i++)
    {
        const SimString& laneId = nearLanes[i];
        std::cout << laneId.GetString() << ",";
    }
    std::cout << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
```

```
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_Data_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SampleGetNearLanes(pos, 5);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;
}
```

获得面前所有车道

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
#include <iostream>
#include <chrono>
#include <thread>

using SSD::SimPoint3D;
using SSD::SimString;
using SSD::SimStringVector;

void SampleGetNearLanesWithAngle(const SimPoint3D& pos,
    const double& distance,
    const double& headingAngle,
    const double& shiftAngle)
{
    SimStringVector nearLanes;
    if (!SimOneSM::GetNearLanesWithAngle(pos, distance, headingAngle, shiftA
ngle, nearLanes))
    {
        std::cout << "No lanes(lane) are(is) found." << std::endl;
        return;
    }
    std::cout << "lane id list:";
```

```
    for (unsigned int i = 0; i < nearLanes.size(); i++)
    {
        const SimString& laneId = nearLanes[i];
        std::cout << laneId.GetString() << ",";
    }
    std::cout << std::endl;
}


int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
```

```
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        double headingAngle = M_PI / 180 * 30;
        double shiftAngle = M_PI / 180 * 90;
        SampleGetNearLanesWithAngle(pos, 5, headingAngle, shiftAngle);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }
    return 0;
}
```

查询位置两侧车道边缘

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
#include <chrono>
#include <iostream>
#include <thread>

using SSD::SimPoint3D;
```

```
using SSD::SimString;

void SampleGetDistanceToLaneBoundary(const SimPoint3D& pos)
{
    SimString laneId;
    double distToLeft, distToRight, distToLeft2D, distToRight2D;
    if (!SimOneSM::GetDistanceToLaneBoundary(pos, laneId, distToLeft, distToR
ight, distToLeft2D, distToRight2D))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[distToLeft, distToLeft]:" << distToLeft << "," << distToLeft <
< std::endl;
    std::cout << "[distToLeft2D, distToRight2D]:" << distToLeft2D << "," << dis
tToRight2D << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }
```

```
    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SampleGetDistanceToLaneBoundary(pos);
    }
    return 0;
}
```

查询车道信息

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
```

```
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include <vector>
#include <iostream>
#include <chrono>
#include <thread>

using namespace HDMapStandalone;
using SSD::SimPoint3D;
using SSD::SimString;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
    return laneId;
}
```

```
void SampleGetLaneSample(const SimString& laneId)
{
    double s, t, s_toCenterLine, t_toCenterLine;
    MLaneInfo info;
    if (!SimOneSM::GetLaneSample(laneId, info))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::cout << "left boundary knot size:" << info.leftBoundary.size() << std::e
ndl;
    std::cout << "left boundary knot list:" << std::endl;
    for (unsigned int i = 0; i < info.leftBoundary.size(); i++)
    {
        const SimPoint3D& knot = info.leftBoundary[i];
        std::cout << "(" << knot.x << ","
            << knot.y << ","
            << knot.z << "),";
    }
    std::cout << std::endl;
    std::cout << "right boundary knot size:" << info.rightBoundary.size() << st
d::endl;
    std::cout << "right boundary knot list:" << std::endl;
    for (unsigned int i = 0; i < info.rightBoundary.size(); i++)
    {
        const SimPoint3D& knot = info.rightBoundary[i];
```

```
            std::cout << "(" << knot.x << ","
                << knot.y << ","
                << knot.z << "),";
        }
        std::cout << std::endl;
        std::cout << "center line knot size:" << info.centerLine.size() << std::endl;
        std::cout << "center line knot list:" << std::endl;
        for (unsigned int i = 0; i < info.centerLine.size(); i++)
        {
            const SimPoint3D& knot = info.centerLine[i];
            std::cout << "(" << knot.x << ","
                << knot.y << ","
                << knot.z << "),";
        }
        std::cout << std::endl;
    }

    int main(int argc, char* argv[])
    {
        int timeout = 20;
        if (!SimOneSM::LoadHDMap(timeout))
        {
            std::cout << "Failed to load hdmap!" << std::endl;
            return 0;
        }

        std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
```

```
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleGetLaneSample(laneId);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获取车道连接信息

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/MDataStruct.h"
#include <vector>
#include <chrono>
#include <iostream>
#include <thread>

using SSD::SimPoint3D;
using SSD::SimString;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
```

```
< t_toCenterLine << std::endl;
    return laneId;
}

void SampleGetLaneLink(const SimString& laneId)
{
    MLaneLink link;
    if (!SimOneSM::GetLaneLink(laneId, link))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::cout << "predecessor lane Id list:" << std::endl;
    for (unsigned int i = 0; i < link.predecessorLaneNameList.size(); i++)
    {
        const SimString& laneId = link.predecessorLaneNameList[i];
        std::cout << laneId.GetString() << ",";
    }
    std::cout << std::endl;
    std::cout << "successor lane Id list:" << std::endl;
    for (unsigned int i = 0; i < link.successorLaneNameList.size(); i++)
    {
        const SimString& laneId = link.successorLaneNameList[i];
        std::cout << laneId.GetString() << ",";
    }
    std::cout << std::endl;
    std::cout << "left neighbor Id:" << link.leftNeighborLaneName.GetString()
```

```
<< std::endl;
    std::cout << "right neighbor Id:" << link.rightNeighborLaneName.GetString
() << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
```

```
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleGetLaneLink(laneId);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获取车道类型

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
#include <chrono>
#include <iostream>
#include <thread>

using SSD::SimPoint3D;
using SSD::SimString;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
```

```
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
    return laneId;
}

void SampleGetLaneType(const SimString& laneId)
{
    ESimOneLaneType type;
    if (!SimOneSM::GetLaneType(laneId, type))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::string typeStr;
    switch (type)
    {
```

```
case ESimOneLaneType_none:
{
    typeStr = "ESimOneLaneType_none";
}
break;
case ESimOneLaneType_driving:
{
    typeStr = "ESimOneLaneType_driving";
}
break;
case ESimOneLaneType_stop:
{
    typeStr = "ESimOneLaneType_stop";
}
break;
case ESimOneLaneType_shoulder:
{
    typeStr = "ESimOneLaneType_shoulder";
}
break;
case ESimOneLaneType_biking:
{
    typeStr = "ESimOneLaneType_biking";
}
break;
case ESimOneLaneType_sidewalk:
{
```

```
        typeStr = "ESimOneLaneType_sidewalk";
    }
    break;
    case ESimOneLaneType_border:
    {
        typeStr = "ESimOneLaneType_border";
    }
    break;
    case ESimOneLaneType_restricted:
    {
        typeStr = "ESimOneLaneType_restricted";
    }
    break;
    case ESimOneLaneType_parking:
    {
        typeStr = "ESimOneLaneType_parking";
    }
    break;
    case ESimOneLaneType_bidirectional:
    {
        typeStr = "ESimOneLaneType_bidirectional";
    }
    break;
    case ESimOneLaneType_median:
    {
        typeStr = "ESimOneLaneType_median";
    }
```

```
        break;
        case ESimOneLaneType_special1:
        {
            typeStr = "ESimOneLaneType_special1";
        }
        break;
        case ESimOneLaneType_special2:
        {
            typeStr = "ESimOneLaneType_special2";
        }
        break;
        case ESimOneLaneType_special3:
        {
            typeStr = "ESimOneLaneType_special3";
        }
        break;
        case ESimOneLaneType_roadWorks:
        {
            typeStr = "ESimOneLaneType_roadWorks";
        }
        break;
        case ESimOneLaneType_tram:
        {
            typeStr = "ESimOneLaneType_tram";
        }
        break;
        case ESimOneLaneType_rail:
```

```
{
    typeStr = "ESimOneLaneType_rail";
}
break;
case ESimOneLaneType_entry:
{
    typeStr = "ESimOneLaneType_entry";
}
break;
case ESimOneLaneType_exit:
{
    typeStr = "ESimOneLaneType_exit";
}
break;
case ESimOneLaneType_offRamp:
{
    typeStr = "ESimOneLaneType_offRamp";
}
break;
case ESimOneLaneType_onRamp:
{
    typeStr = "ESimOneLaneType_onRamp";
}
break;
case ESimOneLaneType_mwyEntry:
{
    typeStr = "ESimOneLaneType_mwyEntry";
```

```
        }
        break;
        case ESimOneLaneType_mwyExit:
        {
            typeStr = "ESimOneLaneType_mwyExit";
        }
        break;
        default:
            typeStr = "ESimOneLaneType_none";
    }
    std::cout << "lane type:" << typeStr.c_str() << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
```

```
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleGetLaneType(laneId);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获取车道宽度

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
```

```
#include <chrono>
#include <iostream>
#include <thread>

using SSD::SimString;
using SSD::SimPoint3D;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
    return laneId;
}

void SampleGetLaneWidth(const SimString& laneId, const SimPoint3D& pos)
{
    double width;
```

```
    if (!SimOneSM::GetLaneWidth(laneId, pos, width))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::cout << "lane width at this location(" << pos.x
        << "," << pos.y
        << "," << pos.z
        << ") is:"
        << width
        << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
```

```
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleGetLaneWidth(laneId, pos);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获取在车道上的 ST 坐标

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
```

```
#include <chrono>
#include <thread>
#include <iostream>

using SSD::SimString;
using SSD::SimPoint3D;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
    return laneId;
}

void SampleGetLaneST(const SimString& laneId, const SimPoint3D& pos)
{
    double s, t;
```

```
    if (!SimOneSM::GetLaneST(laneId, pos, s, t))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::cout << "relative to this lane, this location(" << pos.x
        << "," << pos.y
        << "," << pos.z
        << ") "
        << "'s s-t coordidate position [s, t]:"
        << s
        << "," << "t"
        << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
```

```
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleGetLaneST(laneId, pos);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获取在道路上的 ST 坐标

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
```

```
#include "SSD/SimString.h"
#include <vector>
#include <chrono>
#include <iostream>
#include <thread>

using SSD::SimString;
using SSD::SimPoint3D;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
    return laneId;
}

void SampleGetRoadST(const SimString& laneId, const SimPoint3D& pos)
```

```
{
    double s, t, z;
    if (!SimOneSM::GetRoadST(laneId, pos, s, t, z))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::cout << "relative to this lane's owner road, this location(" << pos.x
        << "," << pos.y
        << "," << pos.z
        << ") "
        << "'s s-t coordidate position [s, t]:"
        << s
        << "," << t
        << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
```

```
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleGetRoadST(laneId, pos);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

查询指定车道是否存在于地图之中

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
#include <chrono>
#include <iostream>
#include <thread>

using SSD::SimString;
using SSD::SimPoint3D;

SimString SampleGetNearMostLane(const SimPoint3D& pos)
{
    SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;
    if (!SimOneSM::GetNearMostLane(pos, laneId, s, t, s_toCenterLine, t_toCente
rLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
        return laneId;
    }
    std::cout << "lane id:" << laneId.GetString() << std::endl;
    std::cout << "[s, t]:" << s << "," << t << std::endl;
    std::cout << "[s_toCenterLine, t_toCenterLine]:" << s_toCenterLine << "," <
< t_toCenterLine << std::endl;
```

```
    return laneId;
}

void SampleContainsLane(const SimString& laneId)
{
    if (!SimOneSM::ContainsLane(laneId))
    {
        std::cout << "Error: lane does not eixst in the map." << std::endl;
        return;
    }
    std::cout << "lane exists in the map." << std::endl;
}

int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    std::unique_ptr<SimOne_Data_Gps> gpsPtr = std::make_unique<SimOne_D
ata_Gps>();
    while (1)
    {
        if (!SimOneSM::GetGps(0/*vehicleId*/, gpsPtr.get()))
```

```
        {
            printf("Fetch GPS failed\n");
            std::this_thread::sleep_for(std::chrono::milliseconds(100));
            continue;
        }
        std::cout << "GPS:" << gpsPtr->timestamp
            << " pos:(" << gpsPtr->posX
            << "," << gpsPtr->posY
            << "," << gpsPtr->posZ
            << ")"
            << std::endl;

        //HDMap samples based on gps location
        SSD::SimPoint3D pos(gpsPtr->posX, gpsPtr->posY, gpsPtr->posZ);
        SSD::SimString laneId = SampleGetNearMostLane(pos);
        SampleContainsLane(laneId);
        std::this_thread::sleep_for(std::chrono::milliseconds(100));
    }

    return 0;
}
```

获取地图中停车位列表

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include <vector>
```

```cpp
#include <chrono>
#include <thread>
#include <iostream>

using SSD::SimPoint3D;
using SSD::SimPoint3DVector;
using SSD::SimString;
using SSD::SimStringVector;


int main(int argc, char* argv[])
{
    int timeout = 20;
    if (!SimOneSM::LoadHDMap(timeout))
    {
        std::cout << "Failed to load hdmap!" << std::endl;
        return 0;
    }

    SSD::SimVector<HDMapStandalone::MParkingSpace> parkingSpaceList;
    SimOneSM::GetParkingSpaceList(parkingSpaceList);
    std::cout << "GetParkingSpaceIds returns count:" << parkingSpaceList.size()
 << std::endl;
}
```

获取预设规划路径

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"
#include "util/UtilDriver.h"
#include <iostream>
#include <vector>

using namespace HDMapStandalone;

int main()
{
    int timeout = 20;
    if (SimOneSM::LoadHDMap(timeout))
    {
        SSD::SimPoint3DVector route;
        if (SimOneSM::GetPredefinedRoute(route)) {
            std::cout << "route path size" << route.size() << std::endl;
        }
    }
}
```

获取规划路径所途径道路的 ID 列表

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    SSD::SimPoint3D startPt, endPt;
    SimOne_Data_WayPoints_Entry terminal;
    SSD::SimPoint3DVector ptList;
    SSD::SimVector<long> naviRoadIdList;
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();
    SSD::SimVector<int> indexOfValidPoints;
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);


    if (SimOneSM::GetGps(0, pGps.get()))
    {
        startPt.x = pGps->posX;
        startPt.y = pGps->posY;
```

```
        startPt.z = pGps->posZ;
    }

    if (SimOneSM::GetTerminalPoint(&terminal))
    {
        endPt.x = terminal.posX;
        endPt.y = terminal.posY;
        endPt.z = 0;
    }

    ptList.push_back(startPt);
    ptList.push_back(endPt);

    if (SimOneSM::Navigate(ptList, indexOfValidPoints, naviRoadIdList)) {
        std::cout <<"naviRoadIdList size:"<<naviRoadIdList.size() << std::endl;
        for (auto & pt: naviRoadIdList) {
            std::cout << pt << std::endl;
        }
    }

}
```

根据指定车道 id 和局部坐标获取局部坐标左右两侧车道标线信息

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
```

```
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();

    SimOneSM::GetGps(0, pGps.get());
    SSD::SimPoint3D MainVehiclePos(pGps->posX, pGps->posY, pGps->posZ);
    SSD::SimString laneId;

    HDMapStandalone::MRoadMark left;
    HDMapStandalone::MRoadMark right;
    double s, t, s_toCenterLine, t_toCenterLine;

    if (!SimOneSM::GetNearMostLane(MainVehiclePos, laneId, s, t, s_toCenterLi
ne, t_toCenterLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
    }

    if (SimOneSM::GetRoadMark(MainVehiclePos, laneId, left, right)) {
        std::cout << "Left side roadmark: " << left.length << std::endl;
```

```
        std::cout << "right side roadmark: " << right.length << std::endl;
    }

}
```

获取地图中信号灯列表

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);

    SSD::SimVector<HDMapStandalone::MSignal> lightList;
    SimOneSM::GetTrafficLightList(lightList);
    std::cout << "lightList size" << lightList.size() << std::endl;
}
```

获取地图中交通标志列表

```
#include <iostream>
#include <vector>
```

```
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);

    SSD::SimVector<HDMapStandalone::MSignal> TrafficSignList;
    SimOneSM::GetTrafficSignList(TrafficSignList);
    std::cout << "TrafficSignList size" << TrafficSignList.size() << std::endl;
}
```

获取交通灯给定作用车道的关联停止线列表

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
```

```
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();

    SimOneSM::GetGps(0, pGps.get());
    SSD::SimPoint3D MainVehiclePos(pGps->posX, pGps->posY, pGps->posZ);
    SSD::SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;

    if (!SimOneSM::GetNearMostLane(MainVehiclePos, laneId, s, t, s_toCenterLi
ne, t_toCenterLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
    }

    SSD::SimVector<HDMapStandalone::MObject> stoplineList;
    SSD::SimVector<HDMapStandalone::MSignal> TrafficSignList;
    SimOneSM::GetTrafficSignList(TrafficSignList);

    for (auto& pt : TrafficSignList) {
        SimOneSM::GetStoplineList(pt, laneId, stoplineList);
        std::cout << "stoplineList size" << stoplineList.size() << std::endl;
    }
}
```

获取交通灯给定作用车道的关联人行横道线列表

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();

    SimOneSM::GetGps(0, pGps.get());
    SSD::SimPoint3D MainVehiclePos(pGps->posX, pGps->posY, pGps->posZ);
    SSD::SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;

    if (!SimOneSM::GetNearMostLane(MainVehiclePos, laneId, s, t, s_toCenterLi
ne, t_toCenterLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
    }
```

```
    std::cout << "laneId: " << laneId.GetString() << std::endl;
    SSD::SimVector<HDMapStandalone::MObject> crosswalkList;
    SSD::SimVector<HDMapStandalone::MSignal> TrafficLightList;
    SimOneSM::GetTrafficLightList(TrafficLightList);

    for (auto& pt : TrafficLightList) {
        SimOneSM::GetCrosswalkList(pt, laneId, crosswalkList);
        std::cout << "stoplineList size" << crosswalkList.size() << std::endl;
    }
}
```

获取指定车道所在道路上的网状线列表

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();
```

```
    SimOneSM::GetGps(0, pGps.get());
    SSD::SimPoint3D MainVehiclePos(pGps->posX, pGps->posY, pGps->posZ);
    SSD::SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;

    if (!SimOneSM::GetNearMostLane(MainVehiclePos, laneId, s, t, s_toCenterLi
ne, t_toCenterLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
    }
    SSD::SimVector<HDMapStandalone::MObject> crossHatchList;
    SimOneSM::GetCrossHatchList(laneId, crossHatchList);
    std::cout << "crossHatchList size:  " << crossHatchList.size() << std::endl;
}
```

获取输入点投影到指定车道中心线上的点和切线方向

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
```

```
    SimOneSM::LoadHDMap(timeout);
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Data_Gps>();

    SimOneSM::GetGps(0, pGps.get());
    SSD::SimPoint3D MainVehiclePos(pGps->posX, pGps->posY, pGps->posZ);
    SSD::SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;

    if (!SimOneSM::GetNearMostLane(MainVehiclePos, laneId, s, t, s_toCenterLine, t_toCenterLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
    }

    SSD::SimPoint3D changeToPoint;
    SSD::SimPoint3D dir;

    if (SimOneSM::GetLaneMiddlePoint(MainVehiclePos, laneId, changeToPoint, dir))
    {
        std::cout << "changeToPoint x:" << changeToPoint.x << "changeToPoint y:" << changeToPoint.y << "z:" << changeToPoint.z << std::endl;
    }
}
```

得到仿真场景中的交通灯的真值

```
#include <iostream>
#include <vector>
#include "SimOneSMAPI.h"
#include "SSD/SimPoint3D.h"
#include "SSD/SimString.h"
#include "public/common/MLaneInfo.h"
#include "public/common/MLaneId.h"


int main() {
    int timeout = 20;
    SimOneSM::LoadHDMap(timeout);
    std::unique_ptr<SimOne_Data_Gps> pGps = std::make_unique<SimOne_Da
ta_Gps>();
    SimOne_Data_TrafficLight trafficLight;

    SimOneSM::GetGps(0, pGps.get());
    SSD::SimPoint3D MainVehiclePos(pGps->posX, pGps->posY, pGps->posZ);
    SSD::SimString laneId;
    double s, t, s_toCenterLine, t_toCenterLine;

    if (!SimOneSM::GetNearMostLane(MainVehiclePos, laneId, s, t, s_toCenterLi
ne, t_toCenterLine))
    {
        std::cout << "Error: lane is not found." << std::endl;
```

```
    }

    SSD::SimVector<HDMapStandalone::MObject> stoplineList;
    SSD::SimVector<HDMapStandalone::MSignal> TrafficLightList;
    SimOneSM::GetTrafficLightList(TrafficLightList);

    for (auto& pt : TrafficLightList) {

        if (SimOneSM::GetTrafficLights(0, pt.id, &trafficLight)) {
            std::cout << "trafficLight.status:" << trafficLight.status << std::endl;
        }
    }
}
```

ROS BRIDGE(1.0)

标准 ROS Bridge 接口描述

TaskRosControl

仿真引擎通过从无人驾驶系统获取的控制消息驱动主车行驶，基本控制信息包括油门、刹车和转向

数据来源

- 订阅 ROS Topic: “/control/control_command”
- 接收消息数据类型:cybertrondrive_msgs::ControlCmd
- 接收消息内容:油门、刹车、转向

数据发送

向主车节点发送油门、刹车和转向信息

- 发送消息数据类型:cybertron::proto::sensor::DataVehicleControlState
- 发送频率:60Hz

TaskRosGps

主车位置消息，包括主车当前位置，速度和朝向信息

数据来源 1

订阅主车节点车辆底盘信息

- 接收消息数据类型:cybertron::proto::sensor::DataVehicleChassisState

数据来源 2

- 从热区获取主车位置和朝向信息(mPos/mRot)

数据发送 1

- 发布 ROS Topic: “/localization/odometry”
- 发送消息数据类型:cybertrondrive_msgs::OdometryExt
- 发送消息内容:gps/utm 坐标、heading、euler_angle、linear_velocity、linear_acceleration、angular_velocity
- 发送频率:由网页端设置

数据发送 2

- 发布 ROS Topic: “/tf”
- 发送消息数据类型:tf2_msgs::TFMessage

- 发送消息内容:主车 utm 坐标
- 发送频率:由网页端设置

TaskRosImageBase

摄像头，输出相机规格参数

数据来源

订阅摄像头传感器节点图像数据

- 接收消息数据类型:cybertron::proto::sensor::ImageWidthGroundTruth
- 接收消息内容:图像部分数据.image()

数据发送

- 发送 ROS Topic:由网页端配置
- 发送消息数据类型:sensor_msgs::Image
- 发送消息内容:图像规格参数(width/height/step)
- 发送频率:有网页端设置

TaskRosObstacles

障碍物消息包括障碍物类型(车、行人等)、位置、朝向、速度和包围盒等信息(仿真系统可绕过感知，直接向无人驾驶系统发送障碍物和红绿灯信息)

数据来源 1

- 摄像头设置参数信息:由网页端配置
- 接收数据类型:BridgeConfig::ObstacleSetting

数据来源 2

订阅障碍物感知传感器节点探测数据

- 接收数据类型:cybertron::proto::sensor::ObstacleDetect
- 接收消息内容:车辆、行人、交通指示灯及交通道路标识信息

数据发送

- 发送 ROS Topic: “/perception_object/object_detect_results”
- 发送消息数据类型:cybertrondrive_msgs::ObjectResults
- 发送消息内容:
  - 车俩:位置坐标、landmark、label(car/truck/bus)
  - 行人:位置坐标、landmark、行为信息(umbrella/call/phone/wave/stading)
  - 交通指示等:位置坐标
  - 交通道路标识:位置坐标
  - 发送频率:30Hz

TaskRosPerfectPerception

将障碍物坐标信息转换为标准 ROS 可视化格式发布

数据来源

订阅完美传感器节点探测数据

- 接收数据类型:cybertron::proto::sensor::ObstacleDetect

数据发送

- 发送 ROS Topic:由网页端设置

- 发送消息数据类型:visualization_msgs::MarkerArray
- 发送消息内容:车辆、行人、自行车、交通指示灯及交通道路标识信息
- 发送频率:由网页端配置

TaskRosPointCloud

激光雷达消息包括点云中每个点的位置和强度信息

数据来源

订阅激光雷达传感器节点数据

- 接收数据类型:cybertron::proto::sensor::PointCloudWithGroundTruth

数据发送 1

将激光雷达点云数据转换为标准 ROS 可视化格式发布

- 发布 ROS Topic:有网页端配置
- 发送消息数据类型:sensor_msgs::PointCloud2
- 发送消息内容:point_step、row_step、height、width
- 发送频率:由网页端配置

数据发送 2

将地面真值数据（障碍物位置、朝向及规格信息）转换为标准 ROS 可视化格式发布

- 发布 ROS Topic:网页端配置 + “groundtruth”
- 发送消息数据类型:visualization_msgs::MarkerArray
- 发送消息内容:position、orientation、scale、color
- 发送频率:由网页端配置

TaskRosPrediction

预测信息包括车辆和行人在预测时间段内的位置轨迹

数据来源

从热区获取车辆/行人预测轨迹坐标、速度及朝向数据

数据发送 1

- 发布 ROS Topic: “/action_prediction/prediction_results”
- 发送消息数据类型:cybertrondrive_msgs::PredictionResults
- 发送消息内容:障碍物感知传感器探测到的车辆/行人预测轨迹(每组十个点)
- 发送频率:60H

数据发送 2

- 发布 ROS Topic: “/action_prediction/auxiliary_prediction_results”
- 发送消息数据类型:cybertrondrive_msgs::TrafficInfoResult
- 发送消息内容:障碍物感知传感器探测到的车辆/行人当前位置坐标、速度及朝向
- 发送频率:60Hz

TaskRosReplayPos

向 UE 端发送主车的运动姿态（车体）数据来驱动主车行驶，实现案例回放

数据来源

- 订阅 ROS Topic: “/localization/odometry”
- 接收数据类型:cybertrondrive_msgs::OdometryExt

数据发送

向主车节点发送位置、朝向信息

- 发送消息数据类型:cybertron::proto::sensor::DataVehicleBodyState
- 发送消息内容:local 坐标位置，转向坐标
- 发送频率:60Hz

TaskRosRoutingReq

路由信息包括主车起点、途径点和终点，发送给无人驾驶系统规划主车的全局路径

数据来源

网页端发送，经 BridgeConfig 解析

- 接收数据类型:std::pair<float, float>

数据发送

- 发布 ROS Topic: “/hmi/routing_request”
- 发送消息数据类型:cybertrondrive_msgs::WaypointArray
- 发送消息内容:路由点坐标值
- 发送频率:30Hz

TaskRosTrafficLight

交通灯指示状态标记

数据来源

从热区获取交通灯状态消息

数据发送

- 发布 ROS Topic: “/perception/traffic_light”

- 发送消息数据类型:cybertrondrive_msgs::TrafficLightDetection
- 发送消息内容:id、confidence、color
- 发送频率:10Hz

TaskRosVehicleInfo

车辆当前的底盘状态，包括驾驶模式、车速和发动机转速等信息

数据来源

向主车节点订阅底盘状态和车轮状态消息

- 接收数据类型:cybertron::proto::sensor::DataVehicleChassisState、Cybertron::proto::sensor::DataVehicleWheelSpeedState

数据发送

- 发布 ROS Topic: “/canbus/vehicle_info”
- 发送消息数据类型:cybertrondrive_msgs::VehicleInfo
- 发送消息内容:车速、刹车、油门、加速度、转角、偏航角、车轮转速
- 发送频率:60Hz

ROS 文本形式输出验证工具

outputControlInfo

主车控制消息调试工具

- 订阅 ROS Topic: “/control/control_command”
- 接收数据类型:cybertrondrive_msgs::ControlCmd
- 输出文件名称格式:cmdInfo_year-mond-day_hour-min-sec

- 输出内容
  - throttle
  - steering angle
  - brake

outputGpsInfo

GPS 消息调试工具

- 订阅 ROS Topic: “/localization/odometry”
- 接收数据类型:cybertrondrive_msgs::OdometryExt
- 输出文件名称格式:gpsInfo_year-mond-day_hour-min-sec
- 输出内容
  - odometry.wgs84: x/y/z
  - odometry.utm: x/y/z
  - odometry.heading:
  - odometry.euler_angle: x/y/z
  - odometry.linear_velocity: x/y/z
  - odometry.linear_acceleration: x/y/z
  - odometry.angular_velocity: x/y/z
  - odometry.header.frame_id
  - odometry.header.stamp:

outputVehicleInfo

主车状态消息调试工具

- 订阅 ROS Topic: “/canbus/vehicle_info”
- 接收数据类型:cybertrondrive_msgs::VehicleInfo
- 输出文件名称格式:vehInfo_year-mond-day_hour-min-sec
- 输出内容
  - speed
  - brake
  - throttle
  - steering angle
  - accel lat
  - accel lon
  - accel vert
  - angular roll
  - angular yaw

ROS 可视化格式输出工具

obstacle_box_to_marker

Obstacle 轨迹调试工具

- 订阅 ROS Topic: “/perception_object/object_detect_results”
- 接收数据类型:cybertrondrive_msgs::ObjectResults

- 接收消息内容:车辆、行人、自行车、交通指示灯、交通标识
- 发布 ROS Topic:“obstacle_bbox_marker”
- 发送数据类型:visualization_msgs::MarkerArray
- Rviz 设置
  - Fixed Frame:“obstacle_box_frame”
  - 添加 visualization:“Add” -> “By topic” -> “obstacle bbox marker”
  - 图示:车辆(蓝色)、行人(红色)、自行车(黄色)

perception_to_marker

障碍物感知 BBox 调试工具

- 订阅 ROS Topic:“/perception_object/object_detect_results”
- 接收数据类型:cybertrondrive_msgs::ObjectResults
- 发布 ROS Topic:“/perception_object/vis_marker”
- 发送数据类型:visualization_msgs::MarkerArray
- Rviz 设置
  - Fixed Frame:“perception_frame”
  - 添加 visualization:“Add” -> “By topic” -> “perception_object vis_marker”
  - 图示:车辆(蓝色)、行人(红色)、交通指示灯(紫色)、交通标识(黄色)

perception_to_image

障碍物感知 BBox 合成调试工具

将感知障碍物 BBox 合成到摄像头图像数据中

- 订阅 1 ROS Topic: “/perception_object/object_detect_results”
- 接收数据 1 类型:cybertrondrive_msgs::ObjectResults
- 订阅 2 ROS Topic: “/camera_identifier”
- 接收数据 2 类型:sensor_msgs::Image
- 发布 ROS Topic: “/perception_object/vis_image”
- 发送数据类型:sensor_msgs::Image
- Rqt_image_view 设置
  - 消息订阅选项: “perception_object/vis_image”

prediction_to_marker

车辆/行人预测轨迹调试工具

- 订阅 ROS Topic: “/action_prediction/prediction_results”
- 接收数据类型:cybertrondrive_msgs::PredictionResults
- 发布 ROS Topic: “visualization_marker”
- 发送数据类型:visualization_msgs::MarkerArray
- Rviz 设置
  - Fixed Frame: “prediction_frame”
  - 添加 visualization: “Add” -> “By topic” -> “visualization_marker”
  - 图示:车辆(蓝色)、行人(红色)

Restful API(1.0)

Terminonlogy

该文档使用 python(3.6+)介绍了 Sim-One 目前可调用的主要 API 信息，及 API 调用的请求参，返回参

SimOneLauncher

定义

基于分布式运行方式，用于配置用户个人属性的操作界面

操作

默认 Api 调用默认 URL 为 http://localhost:8088(以下样例使用默认端口号，简称 Sim-One host)

其中端口号 8088，可通过修改 CZ_WEB_PORT 实现

token

定义

Api 调用过程中，向服务端请求认证的凭证

案例

用户基于内置资源可自主选择编辑用于测试的交通流

任务

运行案例即创建一个任务

Test Case

How to create a test case

创建默认案例仅包含案例标准设置,其他参数需调用编辑案例实现

request

URL:为 Sim-One host 与创建案例方法组合，如下

http://localhost:8088/api/cases

Methods:Post

Headers:Content-Type,cookie

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- mapId(string)

  element is set to be the name (without file extention) of the OpenDrive file for

  which we use together with this test case.

- categoryId(string)

  element is a uuid for a case group, which we use to group a couple of related

  cases

- opponent(string)

element is to identify the type of the case, which can have the following values:

  - case (opponent is expressed with points in caseData object)
  - trafficflow (opponent is controlled by traffic simulation)
  - record (opponent is from recordings)

  set it to “case” if your data is waypoints

- name(string)

  element is the display name of the test case

- standards(list)

  element is to define several validation rules for a case run. For now, we support “timeout”, “collision”, “crossLane”, “speedLimitation” and “stopSign”.

response

返回新创建案例的 caseid 信息

How to edit a test case

对创建的案例进行参数编辑,或修改已配置的案例信息

request

URL：为 Sim-One host 与编辑案例方法组合，如下

http://localhost:8088/api/cases/update

Methods：Post

Headers：Content-Type,cookie

Content-Type: application/json

Cookie: siderFlex=flex; token=token

Body:

The SIMONE case file mainly has two parts, a “caseDef” object to define some high-level information for this case, and a “caseData” object to define the case data.

- caseDef object
  - schema:element is fixed to be “casedef”
  - id: element is a uuid for this case, need to be different for different test cases.
  - opponent:element is to identify the type of the case, which can have the following values:
    - case (opponent is expressed with points in caseData object) o trafficflow (opponent is controlled by traffic simulation)
    - record (opponent is from recordings)

      set it to “case” if your data is waypoints
  - userId:element is to identify the user who own this test case, set it to “default” for single pc case.
  - mapId:element is set to be the name (without file extention) of the OpenDrive file for which we use together with this test case.
  - categoryId:element is a uuid for a case group, which we use to group a couple of related cases.

- name:element is the display name of the test case.
- tags:element is an array of string tags, which we can use to further filter the cases.
- created:element is the timestamp (UNIX time) to identify the creation time of this test case.
- lastModified:element is the timestamp (UNIX time) to identify the last modification time of this test case.
- selected:element is for internal use only, can be simply set to be false.

- caseData object
  - schema:element is fixed to be “casedata”.
  - caseId:element is a uuid for this case, need to be the same as id element in caseDef object.
  - mapId:element is set to be the name (without file extention) of the OpenDrive file for which we use together with this test case. It needs to be the same as mapId element in caseDef object.
  - loop:element is deprecated, simply set it to false, please.
  - timeout:element is deprecated, simply set it to 300000.0, please.
  - speed:element is deprecated, simply set it to 1.0, please
  - observer:element is deprecated, simply set it to “none”, please
  - opponent:element is to identify the type of the case, which can have following values:

- case (opponent is expressed with points in caseData object) o trafficflow (opponent is controlled by traffic simulation)
- record (opponent is from recordings)

  set it to “case” if your data is waypoints. It needs to be same as opponent element in caseDef object.

– oncollision:element is deprecated, simply set it to be “stop”, please.

– ongoal:elemet is deprecated, simply set it to be “stop”, please.

– routes element is to describe the ego car data. At higher level, it has two parts, a “byId” object to contain all the ego car objects’ detail information, and an “allIds” array to hold all the ego car ids only. Each element within “byId” object has a uuid name, which is used as the identity of an ego car instance, each ego car instance has these details:

  - id:element is the uuid for this ego car instance, it must be the same as the ego car object name.
  - vehicleId:element is also a uuid, it’s used to reference the pre-defined ego car model in NEVC-SimOne simulator, for that model, we define the car model, vehicle dynamics, controllers and sensor setups.
  - name:element is the display name for this ego car instance.
  - points:element is the waypoints for this ego car instance, each waypoint has these details:

- position:element for position of this ego car instance, in local ENU space, unit(m).
- heading:element for heading of this ego car instance, in quaternion.
- interval:element for time interval from previous waypoint, unit(s).
- distance:element for distance from previous waypoint, unit(m). It will be calculated automatically internally if set to 0.0.
- speed:element for speed of this ego car instance, unit (km/h).
- accel:element for acceleration of this ego car instance, unit (m/s^2). It will be calculated automatically internally if set to 0.0.
- driver:element is used to define the ego behavior, for now it can be:
    - ai:after setting to “ai”, the behavior of ego car instance is totally controlled by external controllers, like an AD package.
    - maneuver:after setting to “maneuver”, NEVC-SimOne is in charge of controlling the movement of this ego car instance. One thing to note is that different waypoints can be set to have different driving behaviors, this is normally

used to implement a way to switch control. E.g. we can follow the waypoints using "maneuver" behavior first, and then pass the control to an AD package at some point.

- level:element is used to identify the driving levels, for now "L2" and "L4" are supported:
  - L2:To use maneuver driver mode, or to mix maneuver driver mode with ai driver mode.
  - L4:The driver mode for each waypoint is ignored and set default to be "ai". One thing to note is that when using an AD package, how to use these waypoints are totally decided by the AD package, for Apollo case, these waypoints are used as routing request, so sometimes only two waypoints is enough for an Apollo ego car to drive.

– elements:element is to describe the non-ego car data, at higher level, it has two parts, a "byId" object to contain all the non-ego car object, and an "allIds" array to hold all the non-ego car ids only. Each element within "byId" object has a uuid name, which is used as the identity of a non-ego car instance, each non-ego car instance has these details:

- id:element is the uuid for this non-ego car instance, it must be the same as the non-ego car object name.

- category:element is to describe the category for this non-ego car instance, it can be set to “vehicle”, “bicycle”, “pedestrian” and “static”.
- type:element is to define the type of this non-ego car instance, for now set it to “universal/obstalce”.
- name:element is the display name for this non-ego car instance.
- style:element is used to reference the pre-defined non-ego car preset, each preset includes car models, vehicle dynamics, etc.
- interop:element is used to define the interpolation method between waypoints, for now set it to “clothoid”.
- points:element is the waypoints for this non-ego car instance, each waypoint has these details:
  - position:element for position of this ego car instance, in local ENU space, unit(m).
  - heading:element for heading of this ego car instance, in quaternion.
  - interval:element for time interval from previous waypoint, unit(s).
  - distance:element for distance from previous waypoint, unit(m). It will be calculated automatically if set to 0.0.

- speed:element for speed of this ego car instance, unit (km/h).
- accel:element for acceleration of this ego car, unit (m/s^2). It will be calculated automatically if set to 0.0.
- reversed:element is for reverse case, for now set it to be "false"

- trigger element is to reference the trigger object (defines later) which links to this non-ego element.

- triggers:element is to define the trigger object, at higher level, it has two parts, a "byId" object to contain all the trigger object, and an "allIds" array to hold all the trigger object ids only. Each element within "byId" object has a uuid name, which is used as the identity of a trigger object instance, each trigger object instance has these details:
  - id:element is the uuid for this trigger object instance, it must be the same as the trigger object name.
  - name:element is the display name for this trigger object instance.
  - type:element is to define the type of this trigger object, for now set it to "general".
  - triggerBox:element is to define the shape of this trigger object. It has these details:
    - position:element for position of this trigger object, in local ENU space, unit(m).

    - heading:element for heading of this trigger object, in quaternion.
    - size:element for the size of this trigger object, unit(m)

- environment:element is to define several environment conditions, including time, weather conditions, etc.
    - timeOfDay:element is to describe the time of day, in “hhmm”.
    - lightIntensity:element is to describe the intensity of the day light, rage from 0.0 to 1.0.
    - heightAngle:element is to describe the height angle of the sun, range from 0.0 to 1.0.
    - cloudDensity:element is to describe the density of the cloud, range from 0.0 to 1.0.
    - rainDensity:element is to describe the density of the rain, range from 0.0 to 1.0.
    - fogDensity:element is to describe the density of the fog, range from 0.0 to 1.0.
    - snowDensity:element is to describe the density of the snow, range from 0.0 to 1.0.
    - ground:element is to describe the ground information, set it to {“enable”: false} for now.

- passStandard:element is to define several validation rules for a case run. For now, we support “timeout”, “collision”, “crossLane”, “speedLimitation” and “stopSign”. Each rule has these details:
  - checked:element. Whether this rule is activated.
  - action:element. What action to take after violating the rule.
  - message:element. Message level.
  - minute:element. For timeout element only, use to describe the timeout value, in minutes.

json sample:


```
{
    "caseDef": {
        "schema": "casedef",
        "id": "438d1810-f09e-11e9-bca5-c981f86e1c6e",
        "opponent": "case",
        "userId": "default",
        "categoryId": "93848190-72e6-11e9-90db-5fc67123c3b1",
        "mapId": "FourLanes",
        "tags": [],
        "name": "test",
        "lastModified": 1571289910095,
        "created": 1571289814034
    },
    "caseData": {
        "schema": "casedata",
        "caseId": "438d1810-f09e-11e9-bca5-c981f86e1c6e",
```

```
"mapId": "FourLanes",
"loop": false,
"timeout": 300000,
"speed": 1,
"observer": "default",
"opponent": "case",
"oncollision": "stop",
"ongoal": "stop",
"routes": {
  "byId": {
    "631af1c0-f09e-11e9-8a14-cfd25f6811c5": {
      "id": "631af1c0-f09e-11e9-8a14-cfd25f6811c5",
      "vehicleId": "apollo",
      "name": "Apollo 决策",
      "points": [
        {
          "position": {
            "x": -941.2,
            "y": -11.852581688921054,
            "z": 0
          },
          "heading": {
            "x": 0,
            "y": 0,
            "z": 0,
            "w": 1
          },
```

```
            "speed": 25,
            "accel": 0,
            "interval": 0,
            "driver": "ai"
        },
        {
            "position": {
                "x": -786.3999999999999,
                "y": -11.912263592034197,
                "z": 0
            },
            "heading": {
                "x": 0,
                "y": 0,
                "z": 0,
                "w": 1
            },
            "speed": 25,
            "accel": 0,
            "interval": 22.291201946164218,
            "driver": "ai",
            "distance": 154.8000135150293
        }
    ],
    "level": "L4"
  }
},
```

```
        "allIds": [
          "631af1c0-f09e-11e9-8a14-cfd25f6811c5"
        ]
      },
      "elements": {
        "byId": {
          "70131ab0-f09e-11e9-8a14-cfd25f6811c5": {
            "id": "70131ab0-f09e-11e9-8a14-cfd25f6811c5",
            "category": "vehicle",
            "name": "ObsJeep_1",
            "type": "universal/obstacle",
            "style": "ObsJeep_1",
            "bp": "/Game/WIP/Shiqiang/Vehicles/Vehicle002BP_Obstacle.Vehi
cle002BP_Obstacle_C",
            "interpo": "clothoid",
            "points": [
              {
                "position": {
                  "x": -923.2,
                  "y": -15.32514241443737,
                  "z": 0
                },
                "heading": {
                  "x": 0,
                  "y": 0,
                  "z": 0,
                  "w": 1
```

```
    },
    "reversed": false,
    "distance": 0,
    "interval": 0,
    "speed": 25,
    "accel": 0
},
{
    "position": {
        "x": -859.9999999999998,
        "y": -15.51136125415851,
        "z": 0
    },
    "heading": {
        "x": 0,
        "y": 0,
        "z": -0.001060358667964879,
        "w": 0.9999994378195897
    },
    "reversed": false,
    "distance": 63.200319803845915,
    "interval": 9.100846051753813,
    "speed": 25,
    "accel": 0
},
{
    "position": {
```

```
                "x": -792.4,
                "y": -15.511361254158691,
                "z": 0
            },
            "heading": {
                "x": 0,
                "y": 0,
                "z": 0,
                "w": 1
            },
            "reversed": false,
            "distance": 67.60002689371615,
            "interval": 9.734403872695125,
            "speed": 25,
            "accel": 0
        }
    ],
    "trigger": "77738830-f09e-11e9-8a14-cfd25f6811c5"
  }
 },
 "allIds": [
  "70131ab0-f09e-11e9-8a14-cfd25f6811c5"
 ]
},
"triggers": {
 "byId": {
  "77738830-f09e-11e9-8a14-cfd25f6811c5": {
```

```
        "id": "77738830-f09e-11e9-8a14-cfd25f6811c5",
        "name": "通用触发器",
        "type": "general",
        "triggerBox": {
          "type": "box",
          "position": {
            "x": -933.2,
            "y": -12.05,
            "z": 0
          },
          "heading": {
            "x": 0,
            "y": 0,
            "z": 0,
            "w": 1
          },
          "size": {
            "x": 3.5999999999999996,
            "y": 17.908203071594585,
            "z": 50
          }
        }
      }
    },
    "allIds": [
      "77738830-f09e-11e9-8a14-cfd25f6811c5"
    ]
```

```
},
"environment": {
    "timeOfDay": 1000,
    "lightIntensity": 0.5,
    "heightAngle": 0.5,
    "cloudDensity": 0.2,
    "rainDensity": 0,
    "fogDensity": 0,
    "snowDensity": 0,
    "ground": {
      "enable": false
    }
},
"sumoConfig": {
    "Config": "${SUMO_ROOT}/FourLanes/fourlanes.sumocfg",
    "RemotePort": 58289,
    "StepLength": 0.03,
    "Resolution": 0.8,
    "AdditionalRoutes": "${SUMO_ROOT}/FourLanes/fourlanes.rou.xml"
},
"geoReference": {
    "originAlt": 0,
    "vendor": "",
    "RnkMinorVersion": "1",
    "RnkMajorVersion": "1",
    "originLat": 0,
    "originHdg": 0,
```

```
    "localEnuExt": "6378137,0,0;0,1,0;0,0,1;1,0,0",
    "originLong": 0
  },
  "passStandard": {
    "timeout": {
      "checked": true,
      "action": "stop",
      "message": "error",
      "minute": 10
    },
    "stopSign": {
      "checked": true,
      "action": "stop",
      "message": "error"
    },
    "speedLimitation": {
      "checked": true,
      "action": "stop",
      "message": "error"
    },
    "crossLane": {
      "checked": true,
      "action": "stop",
      "message": "error"
    },
    "collision": {
      "checked": true,
```

```
            "action": "stop",
            "message": "error"
        }
    }
  }
}
```


response

返回创建该案例更新的具体信息

How to import test cases

导入案例根据本地是否已上传案例文件，分别调用不同的 api，共使用三个 api 完成

judge whether it has been imported

request

URL：为 Sim-One host + 判断文件是否已导入(/api/files/) + 文件的 md5 值

http://localhost:8088/api/files/文件 md5 值

Methods：Get

Headers：Cookie

Cookie:siderFlex=flex;token=token

response

返回该文件的 md5 值，以及文件大小

根据以上请求返回码，再进行第二步 api 调用

import .sim files

若 **judge whether it has been imported** 返回状态码为 200 且 size 等于 0，说明该文件未曾导入，先进行导入文件 api 调用

request

URL：为 Sim-One host +判断文件是否已导入(/api/files/)+文件的 md5 值+导入方法(/upload)

http://localhost:8088/api/files/文件 md5 值/upload

Methods:Post

Headers:Content-Type,cookie

Content-Type:application/json
Cookie:siderFlex=flex; token=token 信息

Body:

- filename(string)

  target file name
- catagory(string)

  category name
- index(string)

  file index
- chunks(int)

  chunk about file
- filesize(int)

  size of file

- uploadId(string)

  uuid about target file

- files(binary)

  target file

response

返回码：code，成功 0，其他不成功

upload to case library

将文件导入后，再调用 import 方法上传至导入的案例库中

并且若 **judge whether it has been imported** 返回状态码为 200 且 size 大于 0，说明该文件已导入，则直接调用 import 方法 api

request

URL:为 Sim-One host +上传方法(/import)

http://localhost:8088/api/cases/import

Methods:Post

Headers:Content-Type，cookie

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- id(string)

  uuid about upload file

- category(string)

uuid about category

- file(string)

  Md5 of the file

- includes(list)

  caseid

- includesVehicle(bool)

  include vehicle informatione or not

response

返回导入文件的 id 信息，若请求时未传，则返回为空

How to create a category

新建案例库类别

request

URL:为 Sim-One host + 创建方法(api/categories)

http://localhost:8088/api/categories

Methods:使用 Post 方法

Headers:Content-Type，以及 cookie 信息

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- parentId(string)

uuid about parent case

- name(string)

  case name

response

返回创建的案例类的 id 信息，以及该案例名称

How to create a test suite

新建测试集

request

URL：为 Sim-One host + api/suites

http://localhost:8088/api/suites

Methods:Post

Headers:Content-Type，以及 cookie 信息

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- parentId(string)

  uuid about parent case

- name(string)

  case suits name

response

返回新建测试集的 id 以及名称

How to delete a test case

删除测试案例

request

URL:为 Sim-One host +api/cases/delete

http://localhost:8088/api/cases/delete

Methods:Post

Headers:Content-Type,cookie

Content-Type:application/json

Cookie:siderFlex=flex;token=token

Body:

- caseIds(list)

  caseid

response

返回删除任务 taskid 信息

How to delete a category

删除指定的案例库

request

URL:为 Sim-One host +api/categories/案例库 id

http://localhost:8088/api/categories/案例库 id

Methods:DELETE

Headers:cookie

Cookie:siderFlex=flex;token=token

response

返回删除的案例库 id

How to delete a test suite

删除指定的测试集

request

URL:为 Sim-One host +api/suites/测试集 id

http://localhost:8088/api/suites/测试集 id

Methods:DELETE

Headers:cookie

Cookie:siderFlex=flex;token=token

response

返回删除的案例库 id

Test task

How to creat a task

运行选中的案例，创建任务

requests

URL:为 Sim-One host+api/tasks

http://localhost:8088/api/tasks

Methods:Post

Headers:Content-Type，cookie

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- caseIds(list)

  caseid

- taskName(string)

  task name

response

返回创建该任务的 sessionId 信息

How to pause a task

暂停正在运行的用例

request

URL:为 Sim-One host +api/sessions/sessionid/pause

http://localhost:8088/api/sessions/sessionid/pause

Methods:Post

Headers:cookie

Cookie:siderFlex=flex;token=token

response

返回为空

How to resume a task

继续已暂停的用例

request

URL:为 Sim-One host +api/sessions/sessionid/resume

http://localhost:8088/api/sessions/sessionid/resume

Methods:Post

Headers:cookie

Cookie:siderFlex=flex; token=token 信息

response

返回为空

How to stop a task

停止任务，分为停止正在运行的一个案例，与停止整个任务，二者仅入参不同

stop the running case

request

URL：为 Sim-One host +停止方法(/api/tasks/stop)

http://localhost:8088/api/tasks/stop

Methods:使用 Post 方法

Headers:Content-Type，cookie

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- taskIds(string)

  uuid about this task

response

返回为空

stop the session

request

URL：为 Sim-One host +停止方法(/api/sessions/stop)

http://localhost:8088/api/sessions/stop

Methods:Post

Headers:Content-Type,cookie

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body

- sessionId(string)

  uuid about this session

response

返回为空

How to delete a task

删除已/未完成的任务分为两个 api 请求，删除 taskid，以及删除 sessionid

delete taskid

request

URL:为 Sim-One host +api/tasks/delete

http://localhost:8088/api/tasks/delete

Methods:Post

Headers:Content-Type,cookie

Content-Type:application/json
Cookie:siderFlex=flex;token=token

Body:

- taskIds(list)

  taskid

response

返回删除任务 taskid 信息

delete sessionid

request

URL：为 Sim-One host +api/sessions/delete

http://localhost:8088/api/sessions/delete

Methods：Post

Headers：Content-Type,cookie

Content-Type:application/json

Cookie:siderFlex=flex;token=token

Body:

- taskSessionIds(list)

  sessionid

response

返回删除任务 sessionid 信息

Test analysis

How to export the report

导出报告查看运行结果

request

URL:为 Sim-One host +api/sessions/+该任务的 sessionid+/report

http://localhost:8088/api/sessions/sessionid/report

Methods:Get

Headers:Cookie

Cookie:siderFlex=flex;token=token

response

返回该任务执行的测试结果

How to export the csv file

针对已运行后的案例，到处关于主车运行轨迹的 csv 文件

request

URL：为 Sim-One host +api/tasks/taskid 信息/data

http://localhost:8088/api/tasks/taskid/data

Methods:Get

Headers:cookie

Cookie:siderFlex=flex; token=token

response

返回请求结果信息

How to replay a case

已完成任务中重新测试该案例

request

URL：为 Sim-One host +api/tasks/taskid 信息/replay

http://localhost:8088/api/tasks/taskid/replay

Methods:Post

Headers:cookie

Cookie:siderFlex=flex;token=token

response

返回 sessionid 信息