**Track Simulator manual**

1. Preliminary
   a. Python 2.7 !! (ROS only can supports python 2)
   b. The source code is written for ROS kinetic, Gazebo 7 or 8

2. Install
   a. Install **Dataspeedinc, dbw_mkz_simulation** (car model)
   (https://bitbucket.org/DataspeedInc/dbw_mkz_simulation)
   b. Extract "road_models.tar.gz" file in "./gazebo/models"
   c. Place "dbw_runner" folder in "~/catkin_ws/src/"
   d. Build dbw_runner package

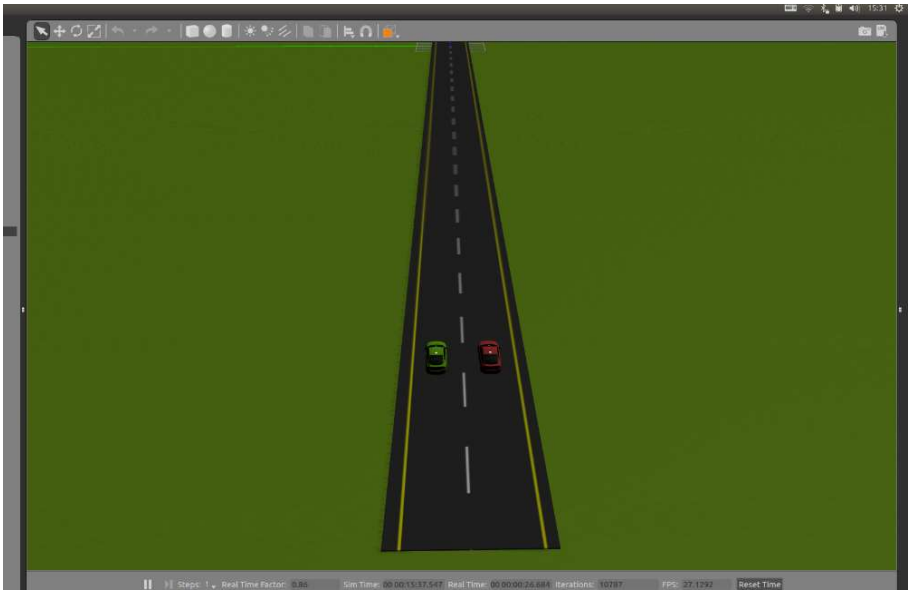
```
$ cd catkin_ws/src
$ catkin_make
```

3. run
   a. 원하는 world의 launch file을 실행하세요

```
$ roslaunch dbw_runner <world name>.launch
```

<world name> = straight_2lane / curved_2lane / straight_4lane / curved_4lane / intersection_2lane NEW

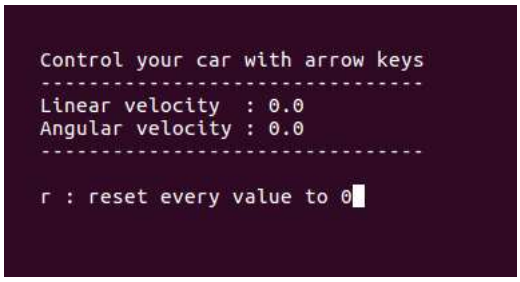
Ex) $ roslaunch dbw_runner straight_2lane.launch



b. car control

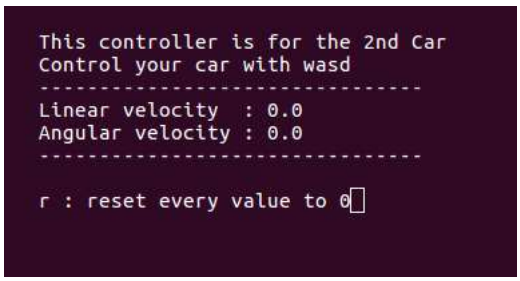
- controller1 : is for the **red car**. Use arrow keys (↑ ↓ →←)

```
$ rosrun dbw_runner keyboard_controller.py
```




- controller2 : is for the **green car**. Use wsad keys.

```
$ rosrun dbw_runner keyboard_controller2.py
```



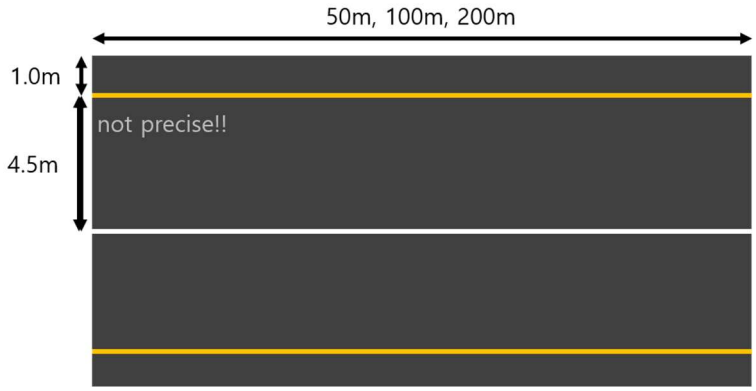

c. data는 launch file을 실행한 순간부터 자동으로 저장. – main.py 따로 실행할 필요X

- Every data is based on the **red car.**
- Each data will be automatically updated every time the car moves over a certain distance

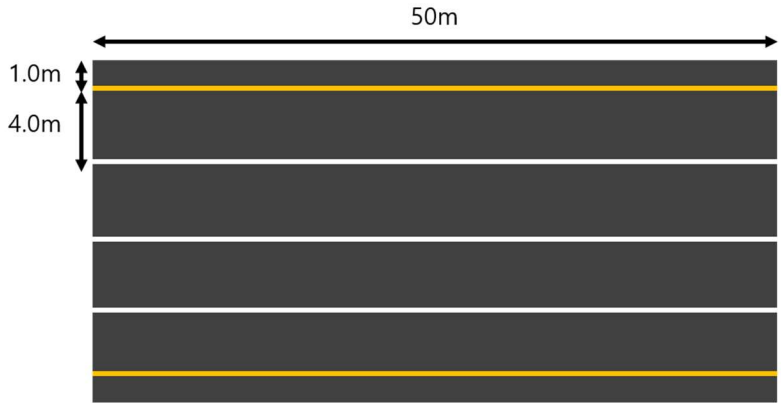
| Pose | Velocity | Dist2obstacle, left | Dist2obstacle middle | Dist2obstacle right | Deviation from side lines | Camera image |
|---|---|---|---|---|---|---|

A. model

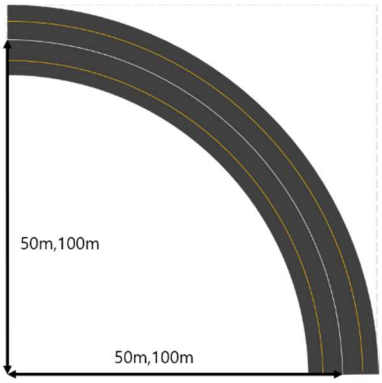
a. straight road 50m (100m, 200m) – Dataspeed inc

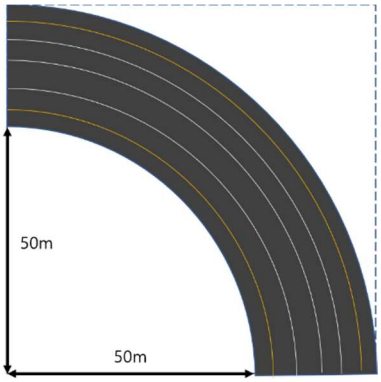



b. straight road 4lane – obin




c. curved - 도로 내부는 straight와 동일



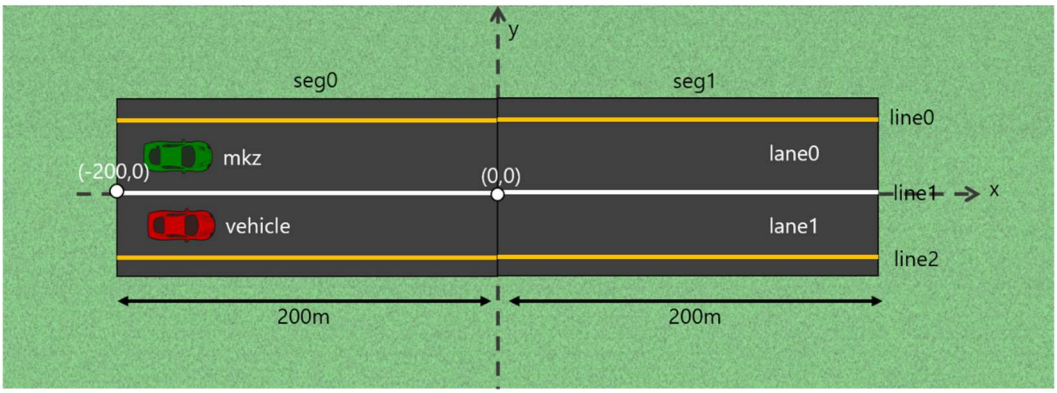

d. about car

e. more roads..!

'simulator_manual_v1_2_0.pdf' 참조!!

B. World

a. straight_2lane.world

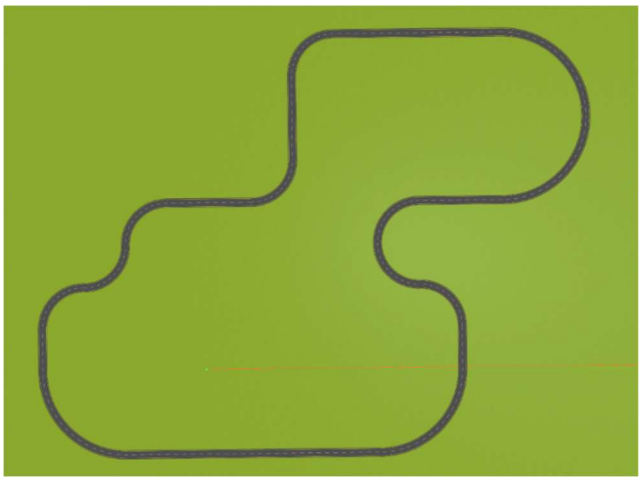



Track : 200m-2lane road를 두개 붙여 총 400m 길이의 직선도로

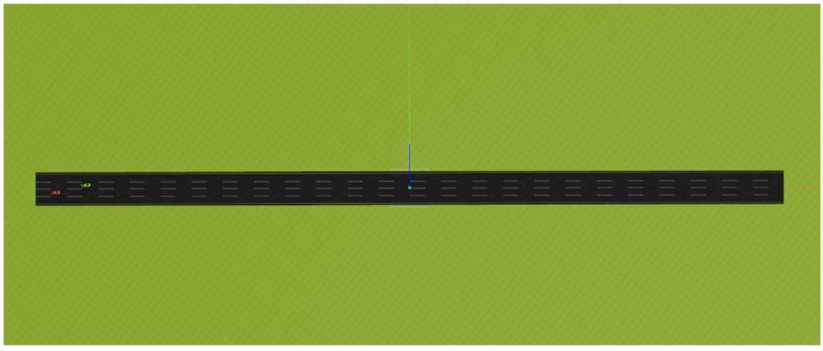
Cars : 1. **vehicle** : will be spawn at (-180,-2.3), 2. **mkz** : will be spawn at (-180,2.3.)

➔ yaml 폴더의 straight_2lane.yaml 파일에서 spawn pose 변경가능

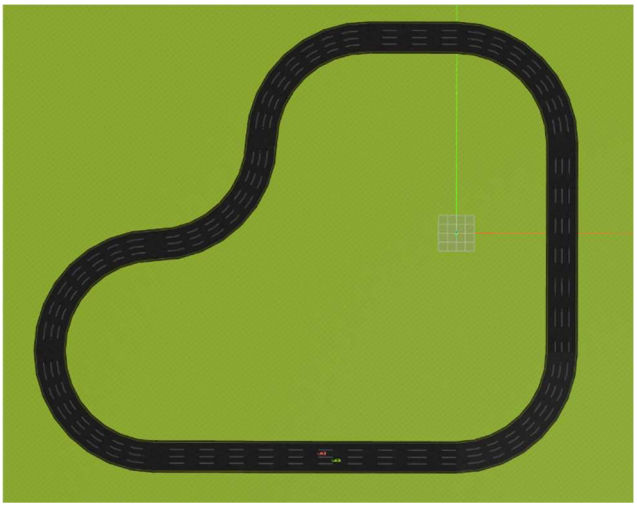
b. curved_2lane.world



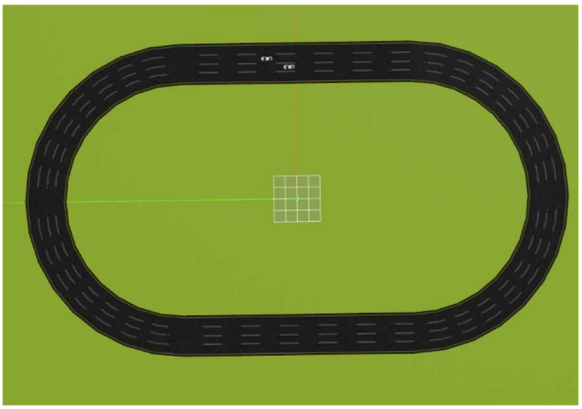
c. straight_4lane.world



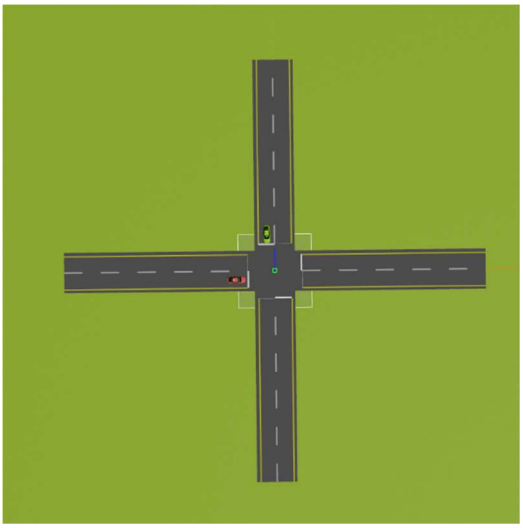
d. curved_4lane.world



e. round_4lane.world



f. intersection_2lane.world NEW



## C. source code

| File name | |
|---|---|
| main.py | Track setting, Spin rosnode, Update data |
| Trackinfo.py | Calculate & Hold data about Track and Car |
| Carinfo.py | Hold information about car |
| TrackSegment.py | Hold information about each track segment |
| Recorder.py | Save data into csv, png file |

### a. main.py

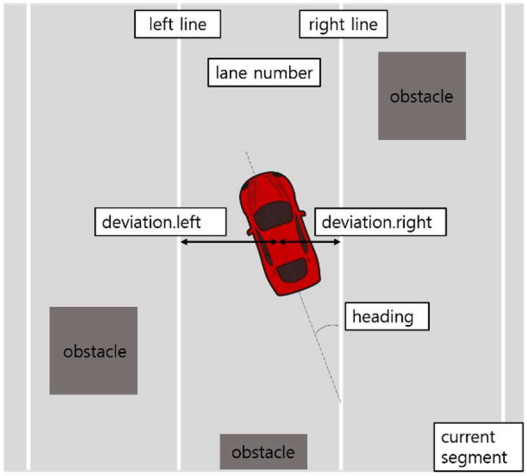
- Track을 만들고, 노드를 실행. Data save명령을 내림.

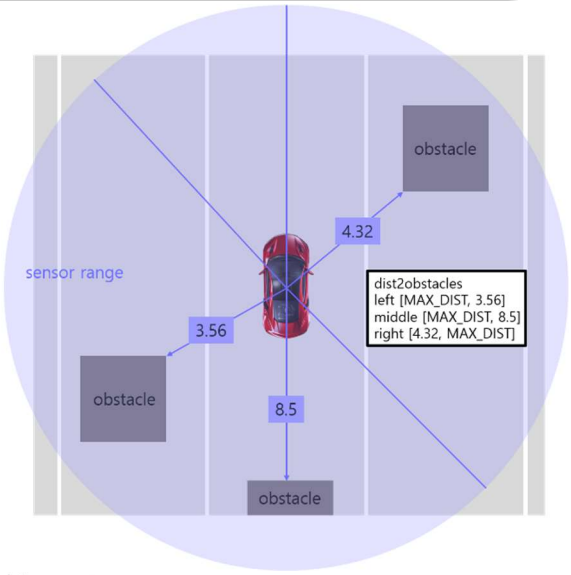
### b. Trackinfo.py

**Trackinfo**

- **variable**
  - my_car: 자동차에 대한 정보(type: Carinfo)
  - Track : 트랙을 구성하는 segment들(type : list)
  - deviation : 양 옆 line으로부터 떨어진 거리
  - heading : 양 옆 line에 대한 자동차의 각도
  - distance : 자동차가 시작 지점부터 달려온 거리
  - current segment : 현재 자동차가 있는 segment num
  - lane_number : 몇 lane에 있는지
  - left line : 자동차 왼쪽의 line 양 끝 점
  - right line : 자동차 오른쪽 line 양 끝 점
  - obstacles : 장애물이 되는 모든 point의 list
  - dist2obstalces : 가장 가까운 장애물까지의 거리
- **function**
  - sensorcallback : sensor data 들어올때마다 실행
  - positioncallback : position data 들어올때마다 실행
  - findDeviation (position data)
  - findHeading(position data)
  - findSegment(position data)
  - findLane(position data)
  - findSidelines(position data)
  - findObstacles(sensor data)
  - finMinDist : list 내에서 최소 값 도출

data가 들어올때마다 이 함수들을 실행하여 data 업데이트




c. **Carinfo.py**

**Carinfo**

- **variable**

- pose = ( x, y, theta)
- velocity = linear velocity, angular velocity

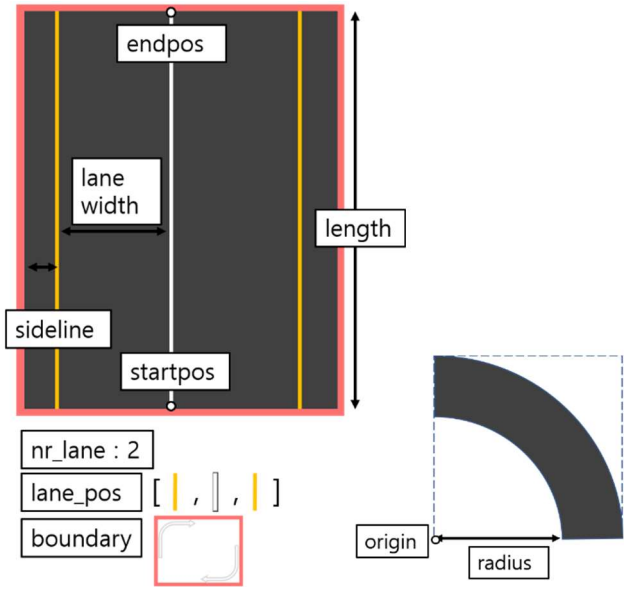
---

- boundary
- width / height / mass

d. TrackSegment.py

**TrackSegment**

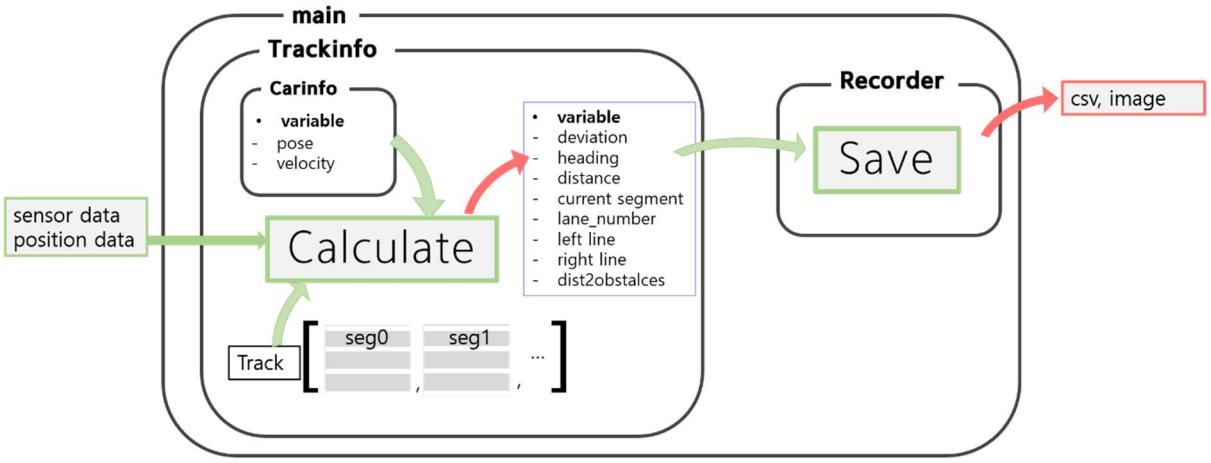
- **variable**
  - segnum : 이 segment가 몇번째 segment인지
  - startpos/endpos : seg의 시작위치/끝위치
  - origin : 'curved'일 경우 원의 중심
  - radius : 'curved'일 경우의 반지름
  - length : segment의 길이
  - boundary : segment의 boundary points
  - lane_width : 한 lane의 너비
  - nr_lane : lane의 개수
  - lane_pos : segment를 이루는 모든 line 모음
  - sideline : lane이 아닌 가장자리의 너비




e. Recorder.py

**Recorder**

- **variable**
  - MX_NR_DATA : data 최대 개수
  - dist_th , deg_th : data를 저장하기 위한 최소 변화량 (threshold)
- **function**
  - update(trackinfo)
    : pose의 변화량이 dist_th나 deg_th보다 클 때
    현재 data 저장 (csv에 줄 추가 + camera image 저장)


main
Trackinfo
Carinfo
• variable
- pose
- velocity
sensor data
position data
Calculate
Track
seg0
seg1
...
• variable
- deviation
- heading
- distance
- current segment
- lane_number
- left line
- right line
- dist2obstalces
Recorder
Save
csv, image