CARLA
(CAR Learning to ACT)

Taylor Carpenter

CIS 700/002: Topics in Safe Autonomy

Department of Computer and Information Science

School of Engineering and Applied Science

University of Pennsylvania

2/12/2019

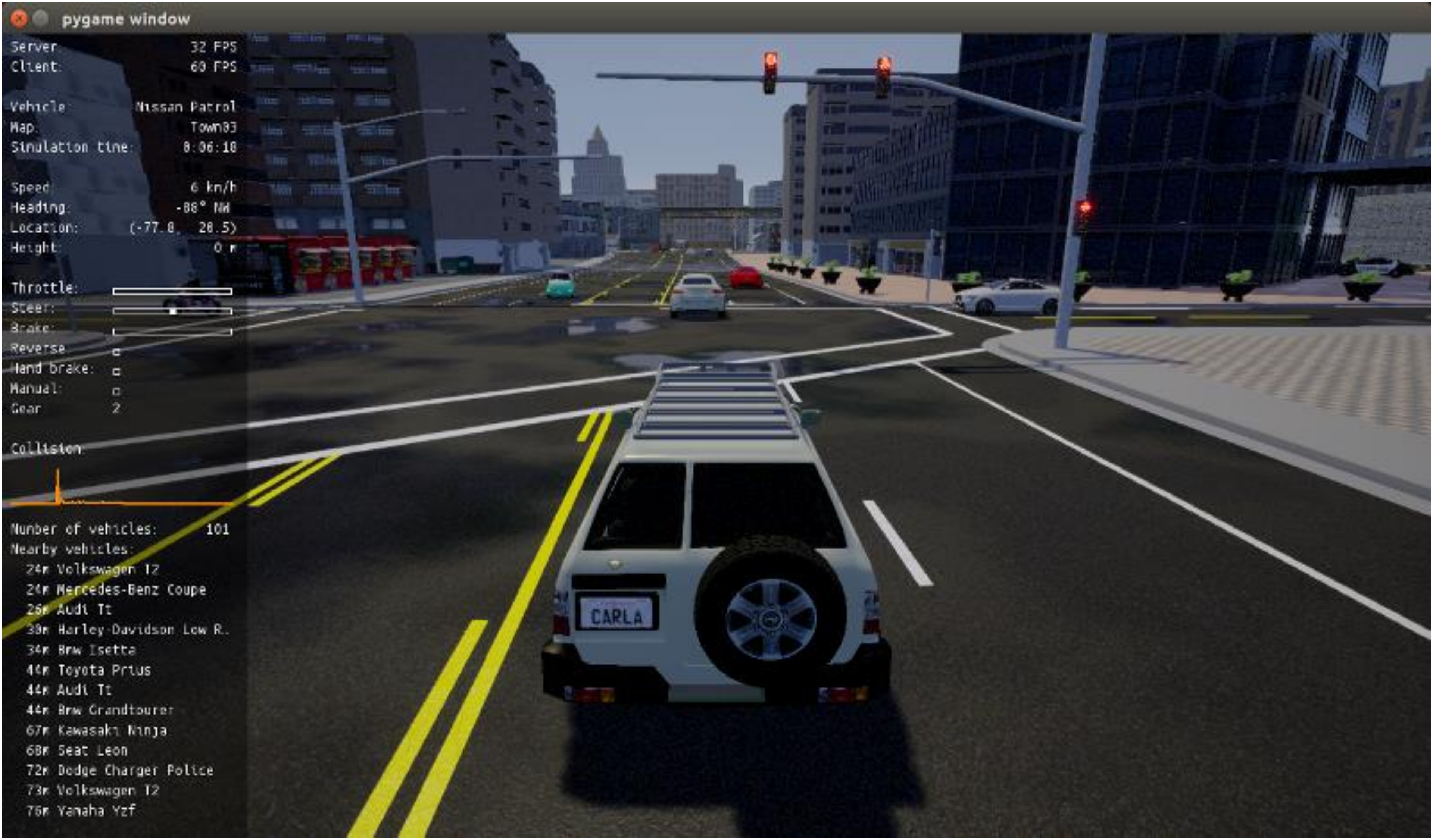
Outline

- Introduction & Background
- Underlying Technologies
- Architecture
- Capabilities
- Use-Cases
- Customizability
- Drawbacks
- Demo

Introduction

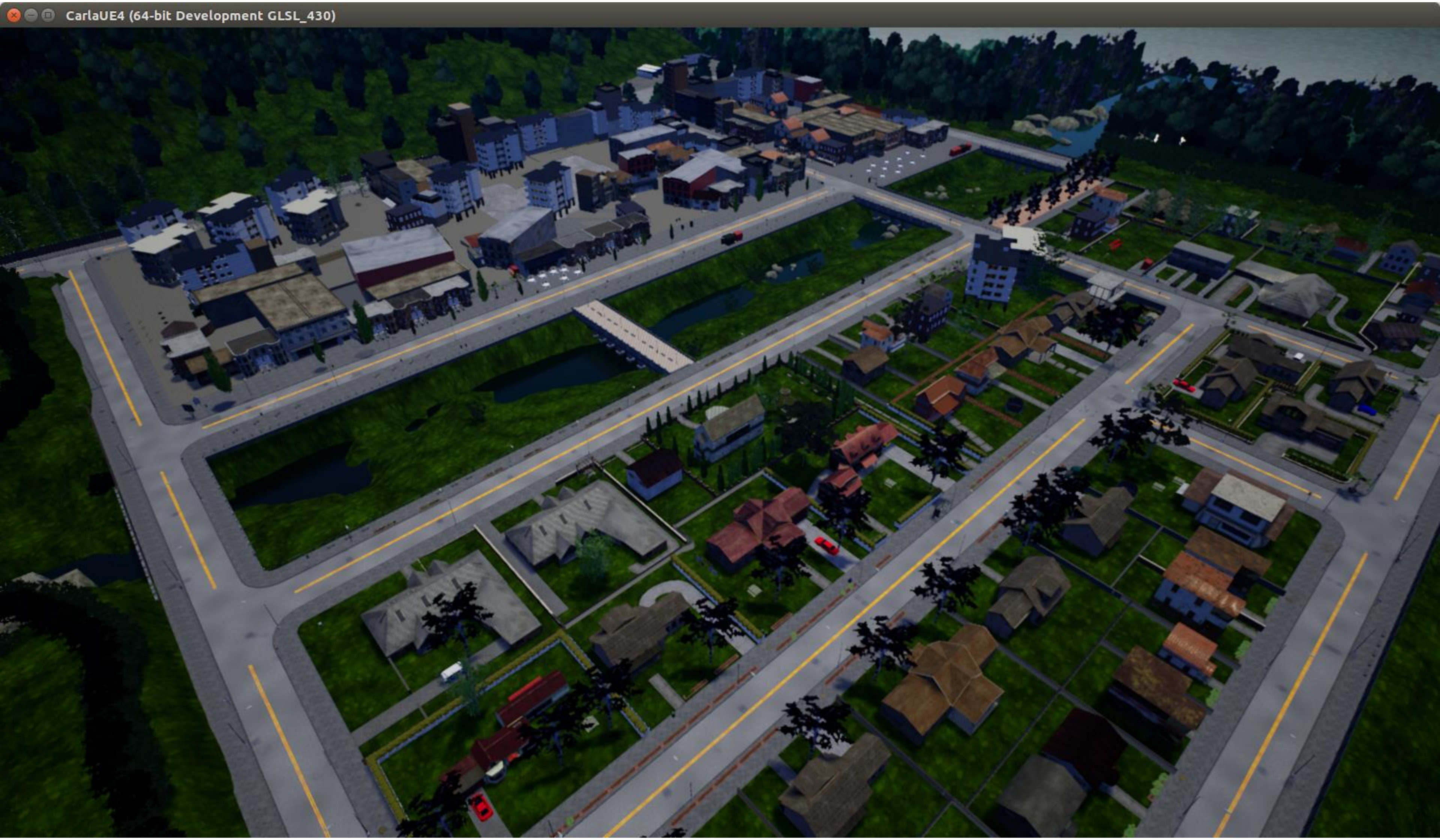
- Open-source simulator for autonomous driving research
- Released in 2017 following a paper at the Conference on Robot Learning (CoRL)
- Created by Computer Vision Center (CVC), Intel
  - Sponsored by Toyota Research Institute, GM Research & Development

https://carla.readthedocs.io/en/latest/



```
Server:                  32 FPS
Client:                  60 FPS

Vehicle:          Nissan Patrol
Map:                     Town03
Simulation time:        0:06:10

Speed:                   6 km/h
Heading:               -88° NW
Location:         (-77.0,  20.5)
Height:                     0 m

Throttle:
Steer:
Brake:
Reverse      □
Hand brake:  □
Manual:      □
Gear         2

Collision:

Number of vehicles:        101
Nearby vehicles:
  24m Volkswagen T2
  24m Mercedes-Benz Coupe
  26m Audi Tt
  30m Harley-Davidson Low R..
  34m Bmw Isetta
  44m Toyota Prius
  44m Audi Tt
  44m Bmw Grandtourer
  67m Kawasaki Ninja
  68m Seat Leon
  72m Dodge Charger Police
  73m Volkswagen T2
  76m Yamaha Yzf
```

https://carla.readthedocs.io/en/latest/


CarlaUE4 (64-bit Development GLSL_430)

Problem to Solve

- Autonomous driving in urban settings is more challenging than other forms of navigation
  - Presence of pedestrians, frequent intersections, road markings, etc.
- Physical development and evaluation of autonomous systems is prohibitively expensive
- Widespread use of ad-hoc simulation, complicating comparisons and benchmarking of techniques

Goals

- “**[d]emocratising** Autonomous Driving via accessible simulation and creating a platform that enables academics and industry members to **share** knowledge and results in the open”
- Operational Goals:
  - Easily create new content (maps, etc.)
  - Create and run complex traffic scenarios
  - Leverage existing sensors
  - Automatically train and assess driving agents
  - Scalable

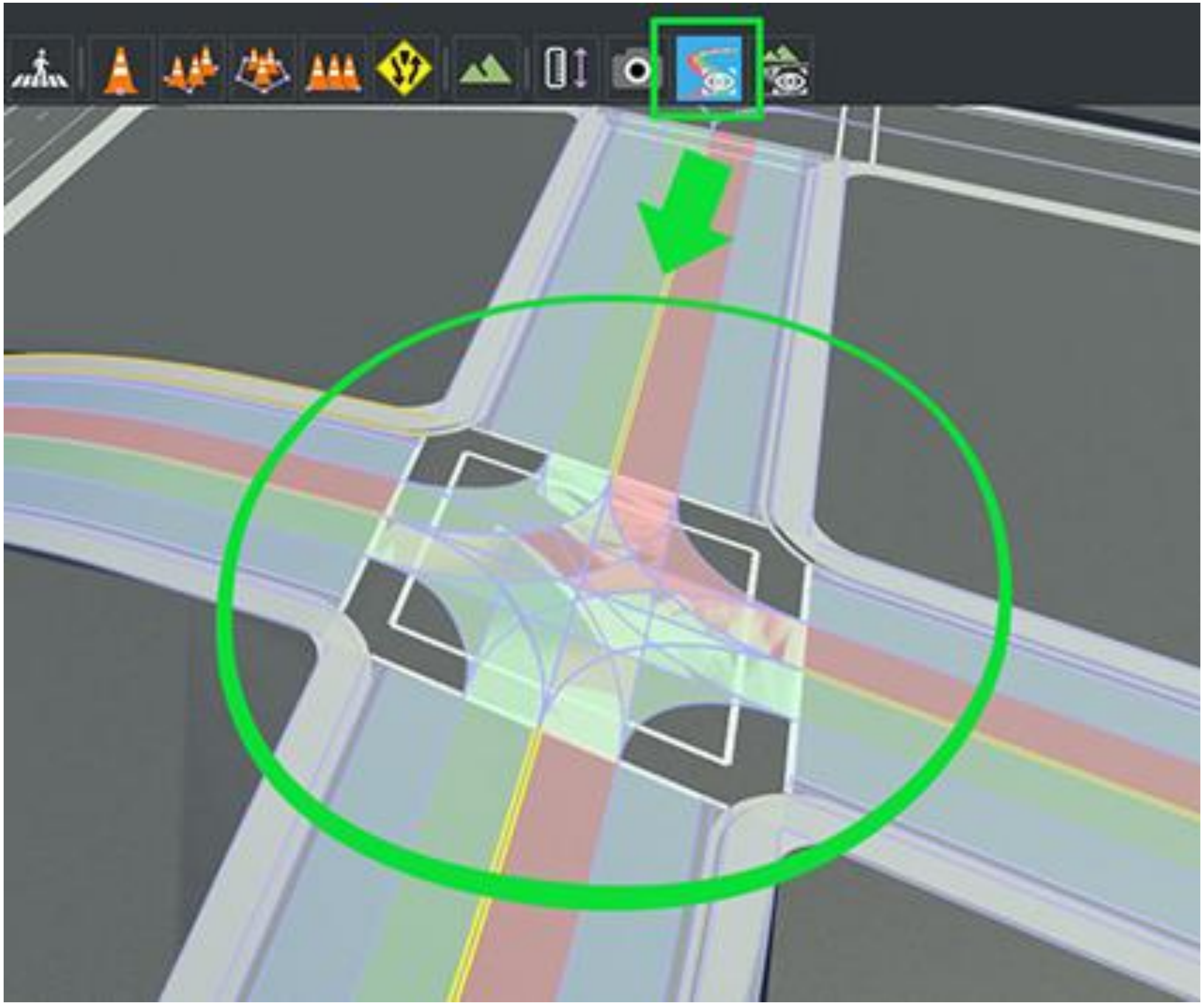
Related Tools

- AirSim
  - Open-source simulator for autonomous vehicles (ground and air)
  - Released 2017
- ROS + Gazebo
  - Open-source simulator + environment for general robotics research
- Autoware
  - Open-source autonomous driving framework(s)
  - Focus on deployment in vehicles

Underlying Technologies

- Unreal Engine 4
  - Free for non-commercial use game engine / physics simulator
  - Used for physics / collisions and rendering
- OpenDRIVE
  - Open XML format for describing road networks
    - Driving lanes, sidewalks, intersections, traffic lights, road signs, pedestrian crossings, etc.
- RoadRunner
  - Map editor by VectorZero
  - Generates OpenDrive and FBX model data

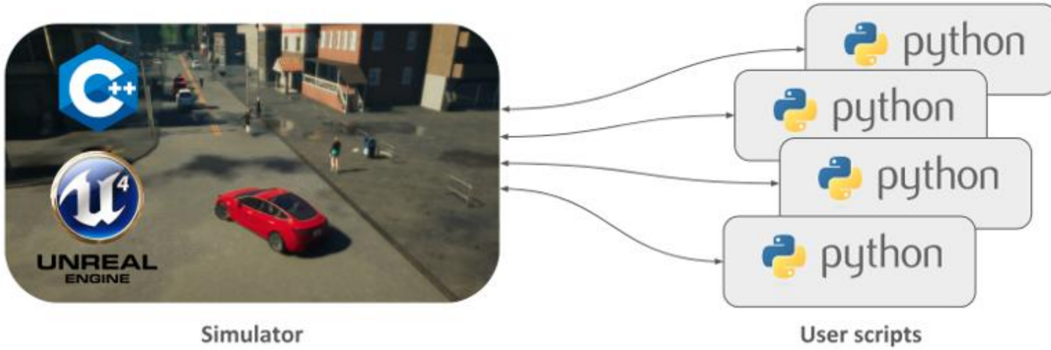
OpenDRIVE / RoadRunner



https://carla.readthedocs.io/en/latest/

Architecture

- Client / Server Architecture
  - Server is the Unreal Engine simulator with own rendering window
  - Clients are python scripts that control or modify simulator
    - Spawn vehicles
    - Control vehicles

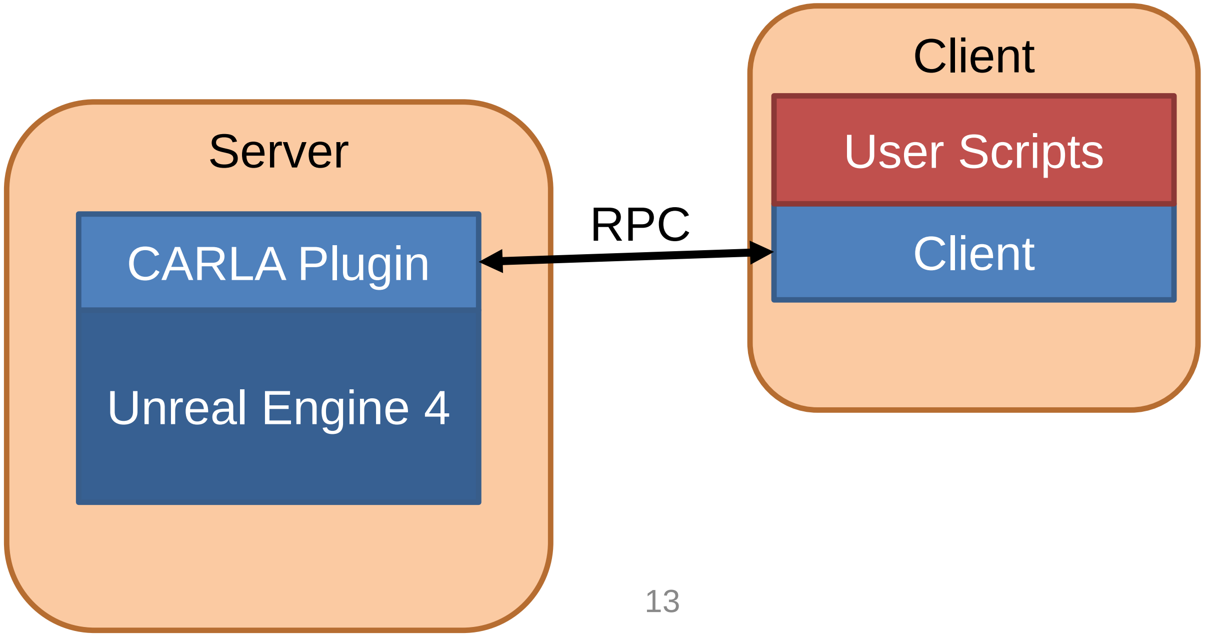


https://carla.readthedocs.io/en/latest/

  - Clients can connect from remote machines
  - Multiple clients can connect at the same time

Architecture – Cont.

- Core is C++ plugin to Unreal Engine 4
- RPC calls for communication
- Sensors implemented on plugin side
- Many tools exist as python scripts



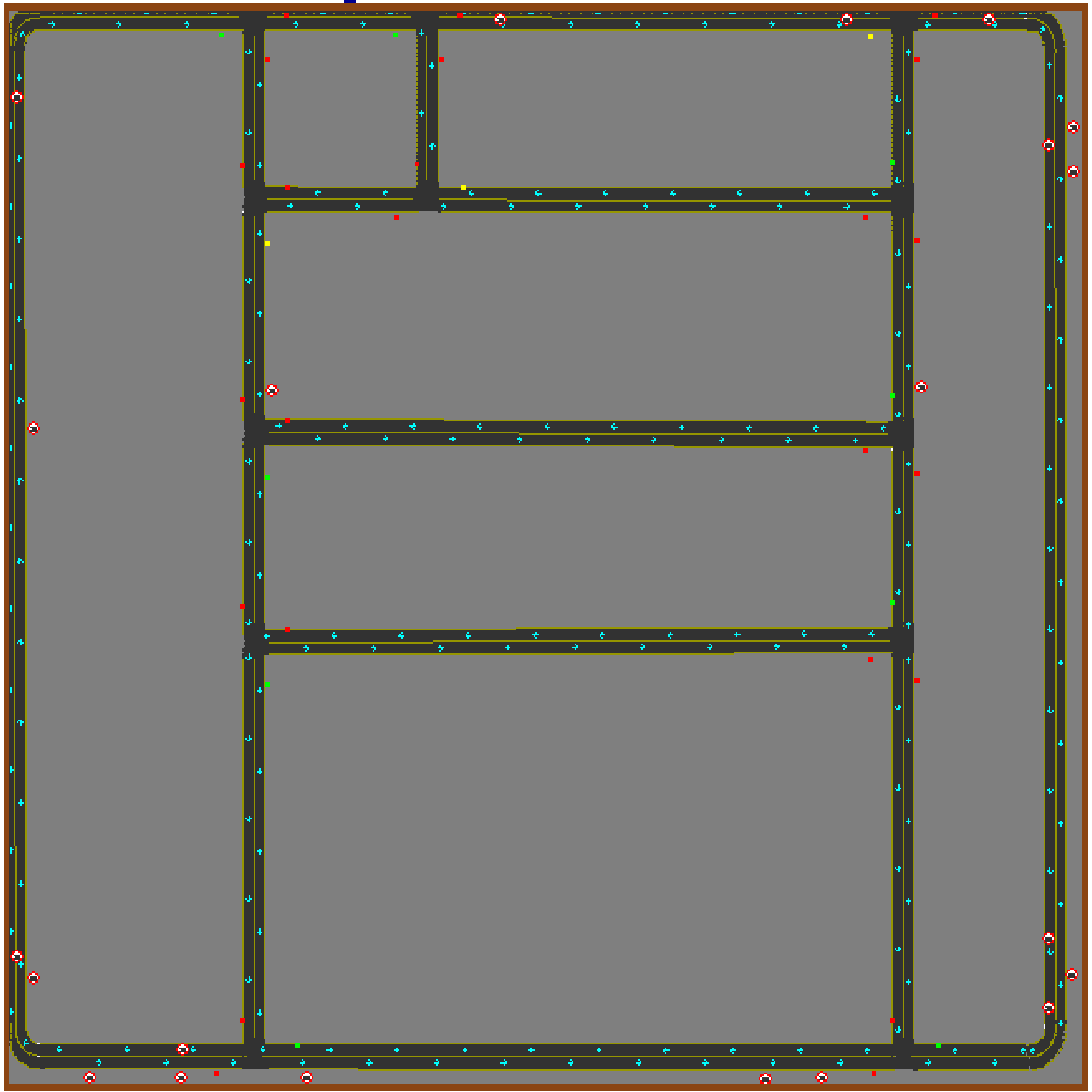
Key Components

- World
  - Provides access to map, weather conditions, actors, method for spawning new actors
- Blueprint
  - Template for spawning an acting, including customizable attributes
  - Can include “suggested values” for attributes for ease of randomized spawning
- Actor
  - Anything dynamic instantiated in the simulation, including: vehicles, pedestrians, sensors, traffic lights, etc.
  - Have physics related attributes such as *location*, *acceleration*, *velocity*, etc.
  - Additional attributes and controls depend on the subtype

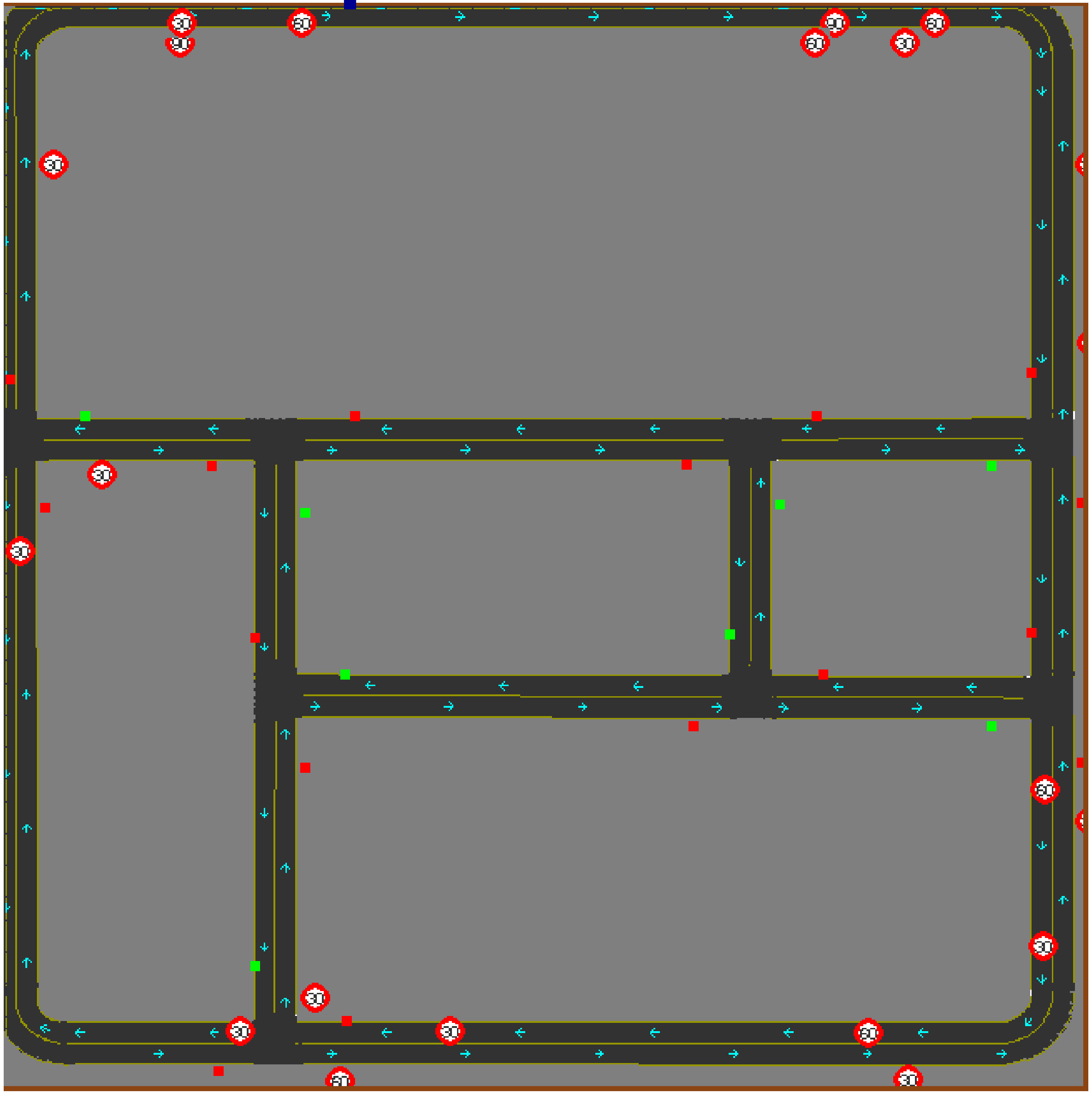
Maps

- OpenDrive format provides map annotations
- *Map* object provides logical, node-based topology of the road network
  - Allows querying of nearest waypoint to a vehicle, or selecting a “next” waypoint
- Currently 5 maps are provided of various sizes and including various traffic components

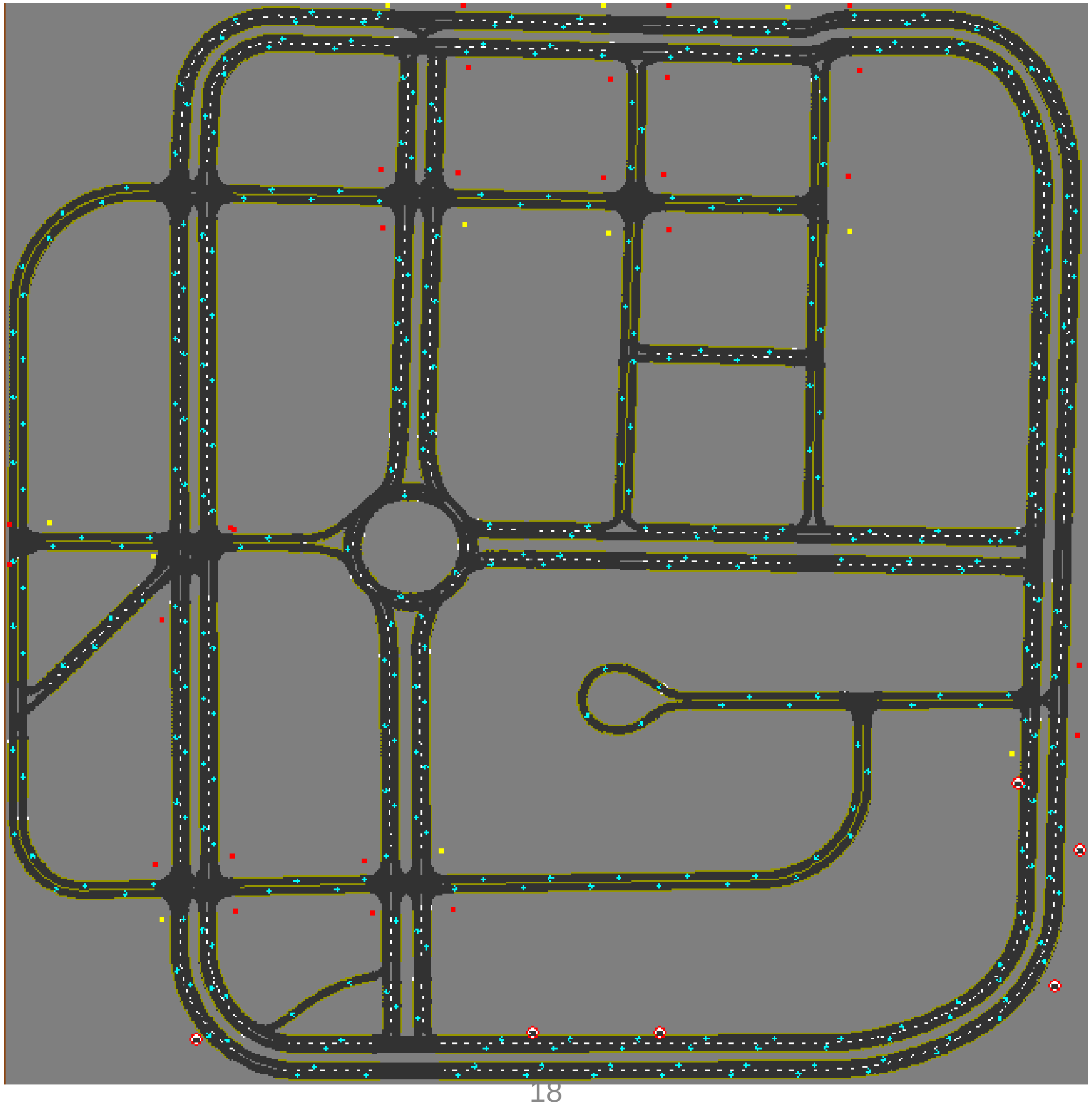
Maps – Town01



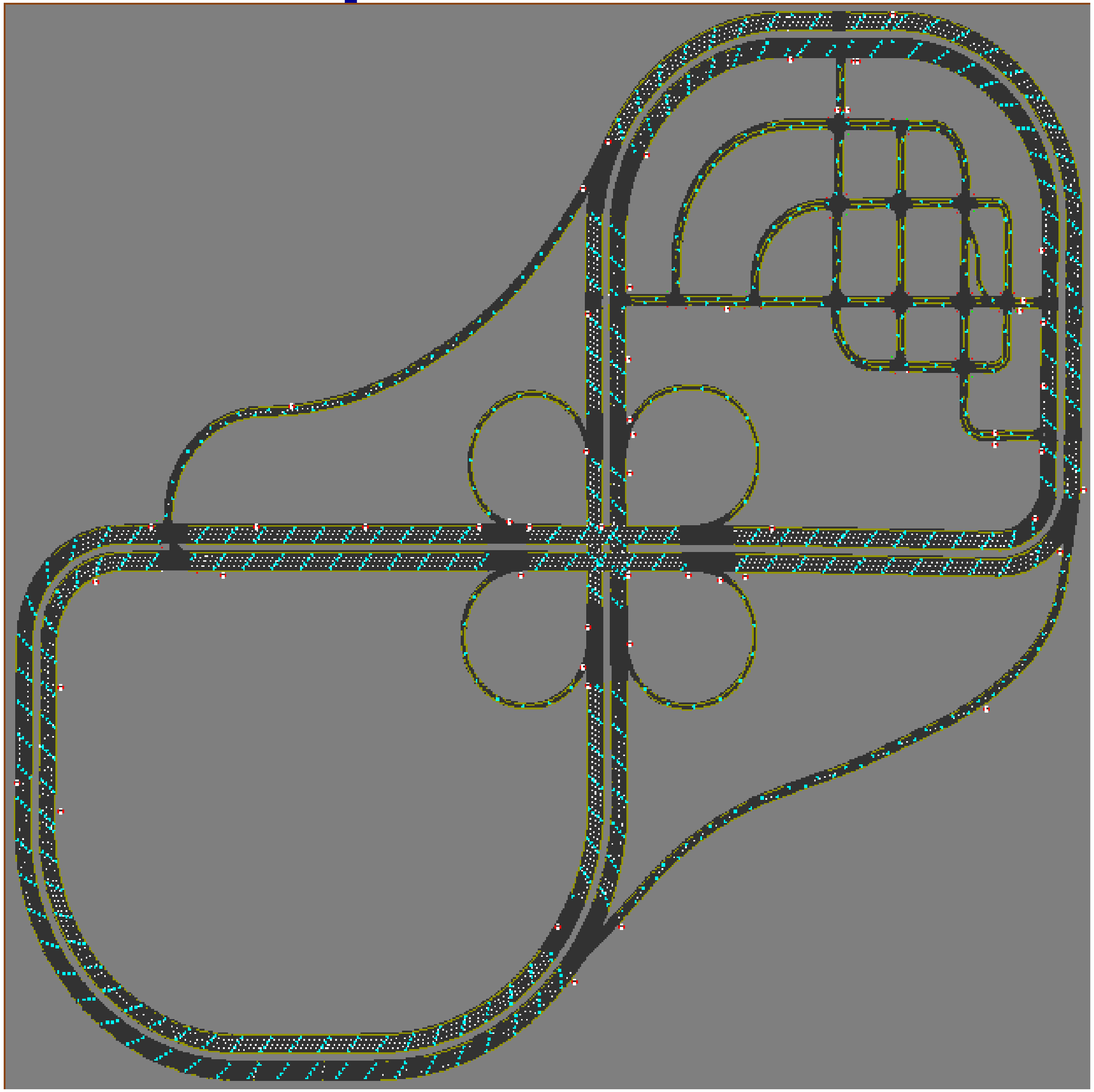
Maps – Town02



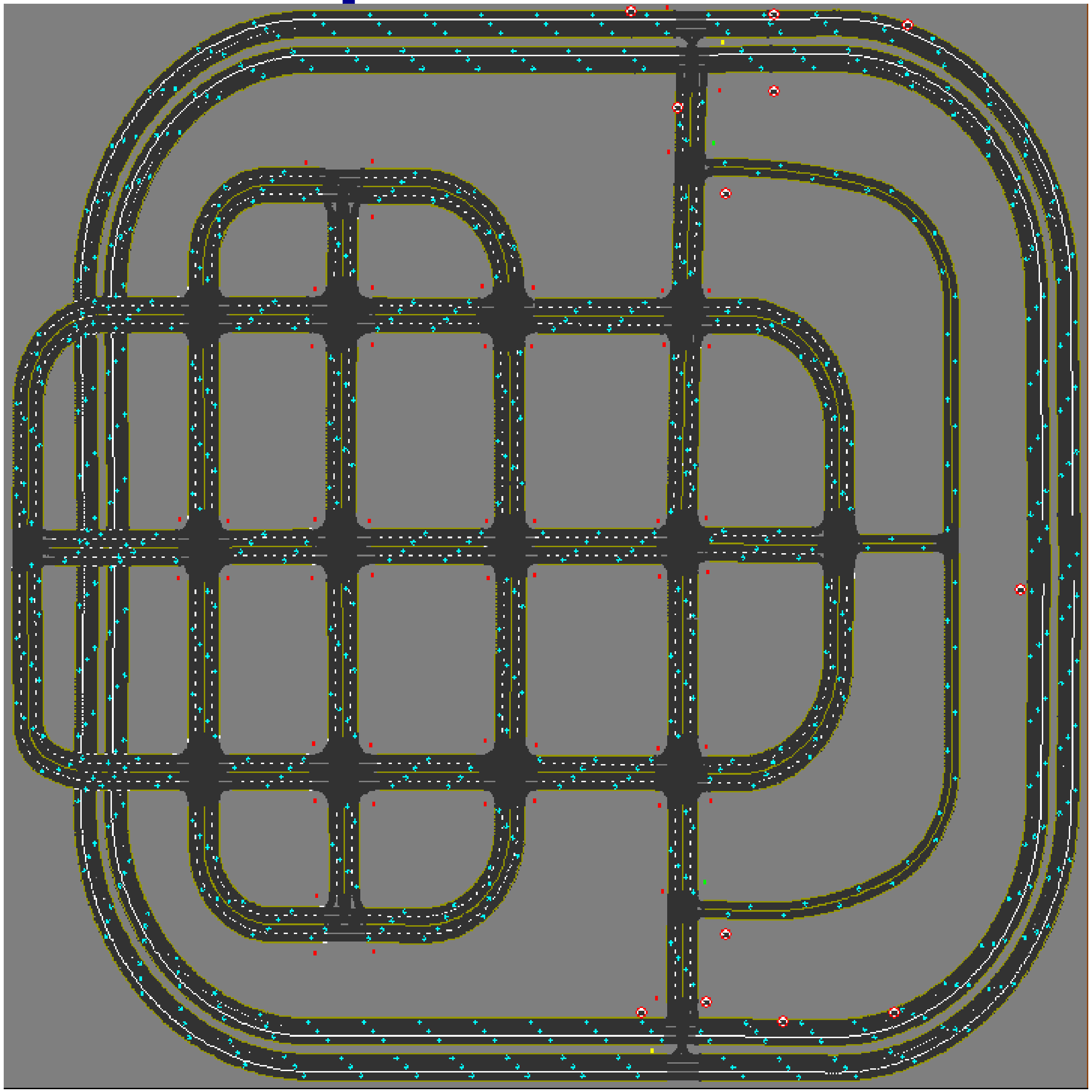
Maps – Town03



Maps – Town04



Maps – Town05



Vehicles

- Special actors that can be controlled through the application of *throttle*, *steer,* and *brake* values
- Includes both cars and motorcycles
- 16+ vehicle models included

https://carla.readthedocs.io/en/latest/

Sensors

- Special actors that can be attached to vehicles and produce data streams
- Available Sensors:
  - RGB Camera
  - Depth Camera
  - Semantic Segmentation Camera
  - Rotating LIDAR
  - Collision Detector
  - Lane Detector
  - Obstacle Detector
  - GNSS

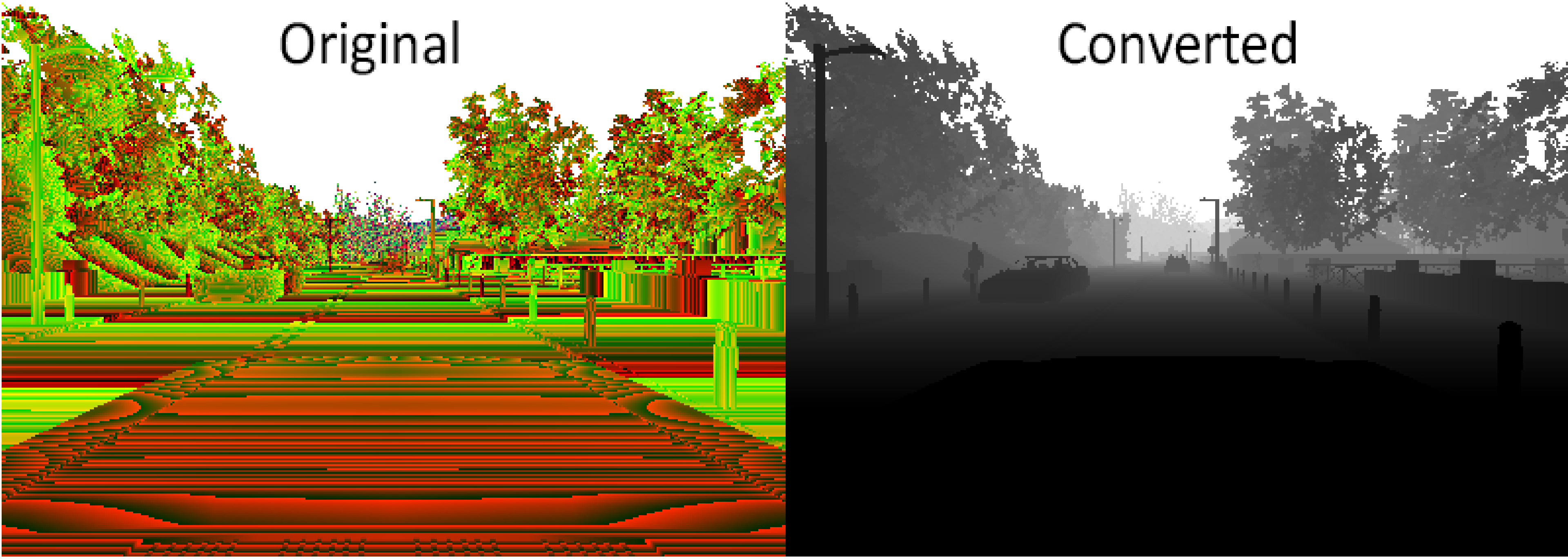
RGB Camera

- Produces RGB images
- Can include post-process effects such as *grain jitter, bloom, lens flare,* etc.

https://carla.readthedocs.io/en/latest/

Depth Camera

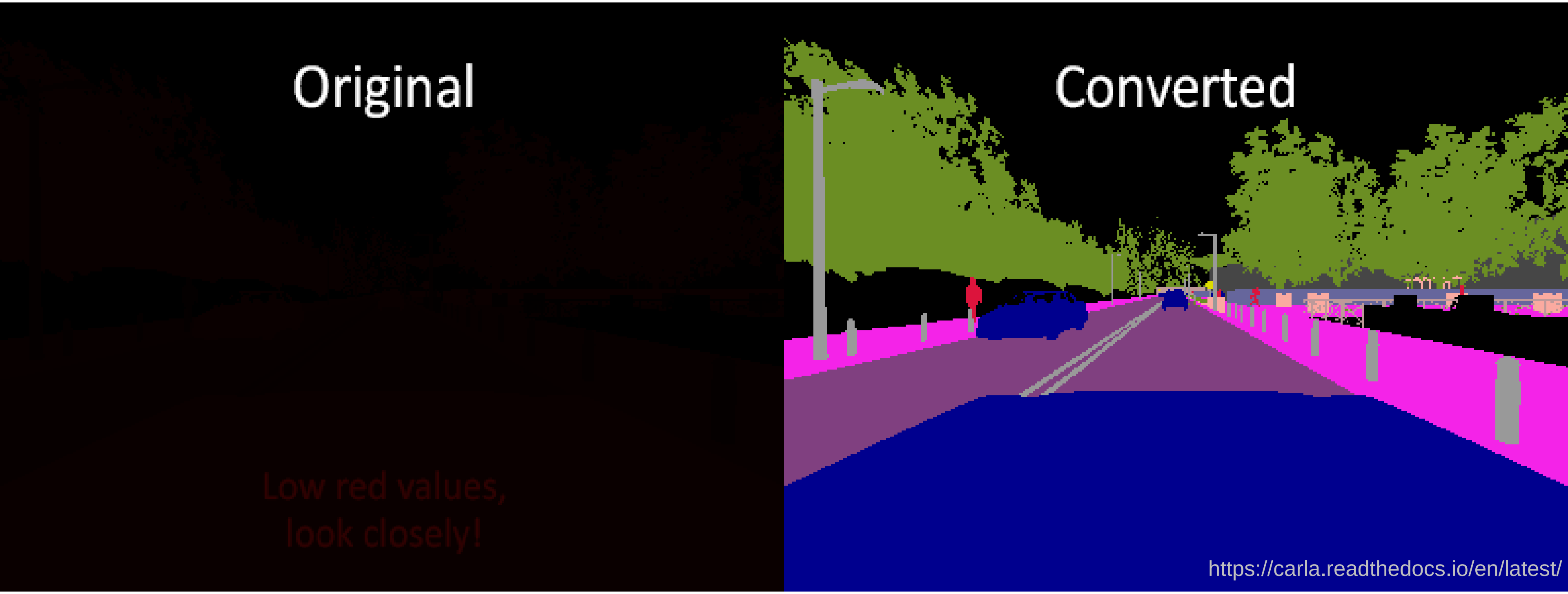
- Produces “images” where depth is encoded in pixel value



https://carla.readthedocs.io/en/latest/

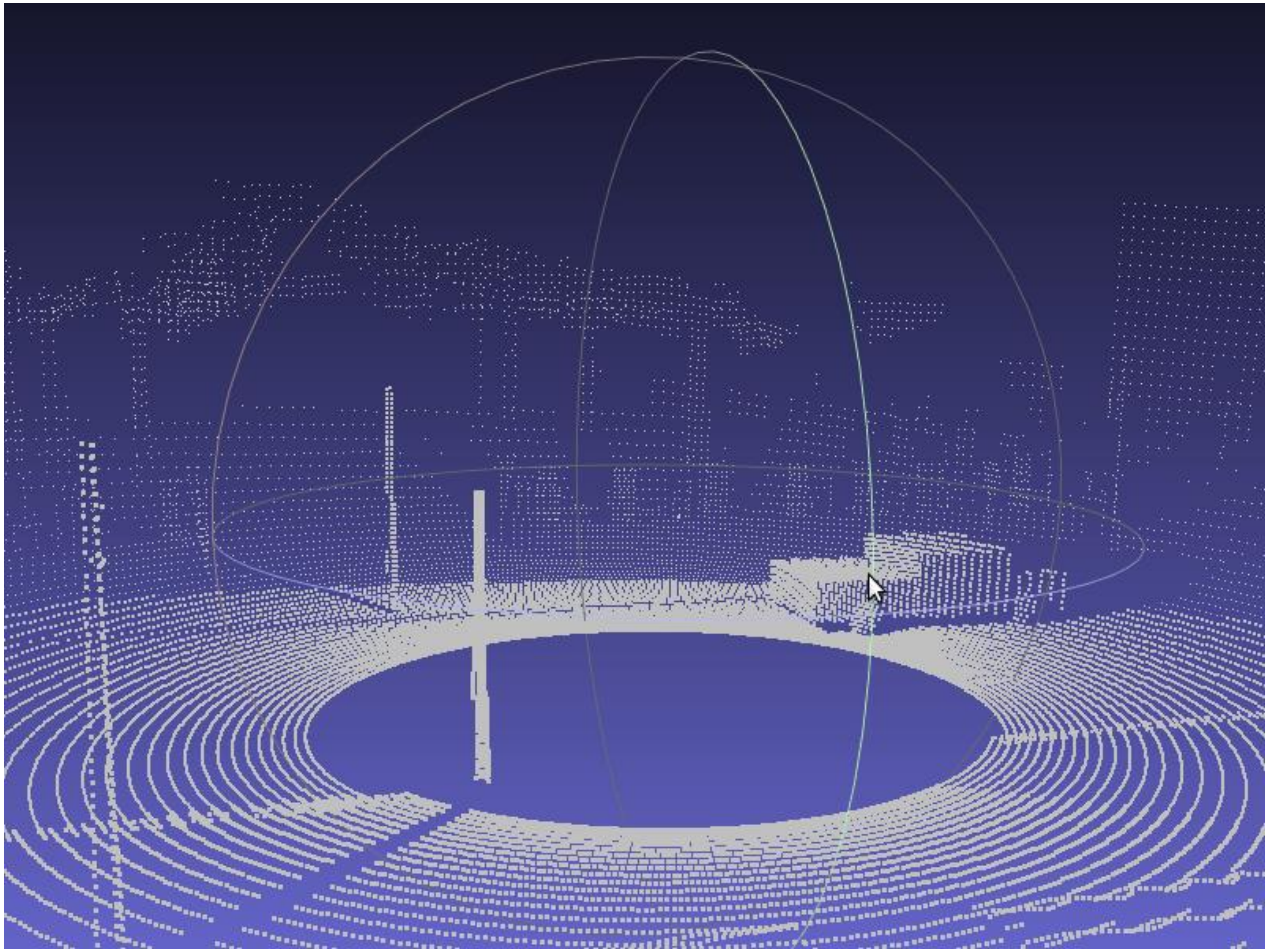
Semantic Segmentation Camera

- Produces “images” where tagging information from ground truth is encoded in the red channel



https://carla.readthedocs.io/en/latest/

Rotating LIDAR



https://carla.readthedocs.io/en/latest/

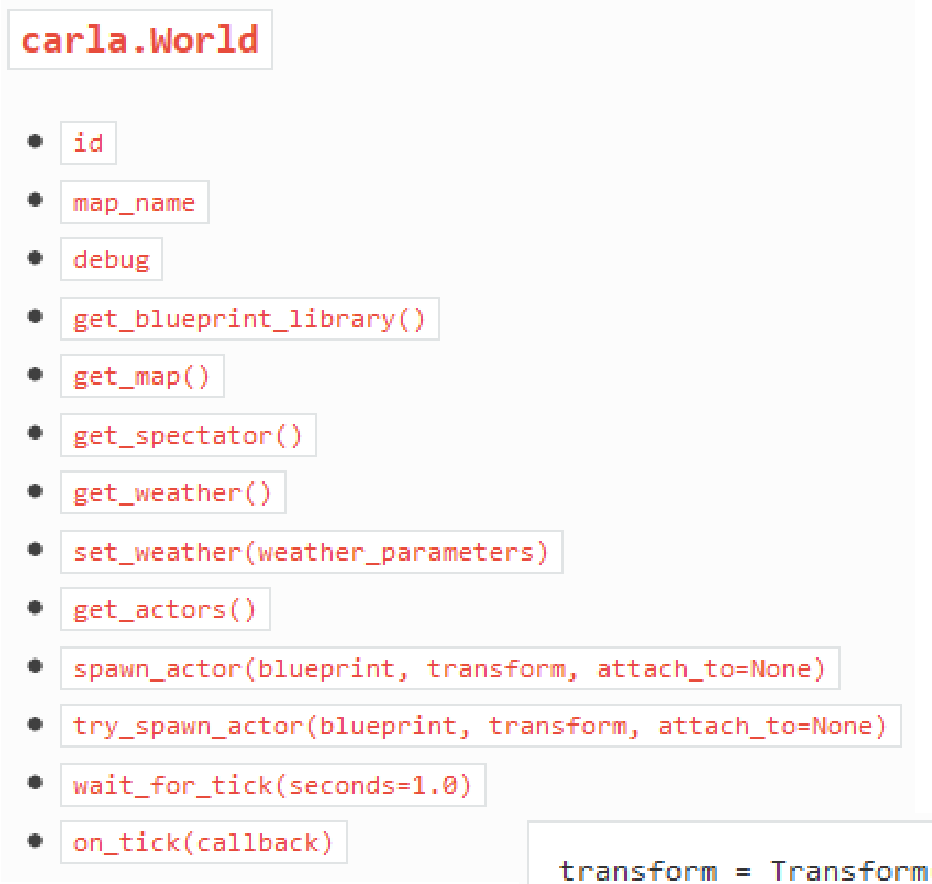
Misc. Sensors

- Collision Detector – produces collision events that record what the collision was with and the energy of the collision
- Lane Detector – produces events identifying the lane markers that were crossed
- Obstacle Detector – Reports if / distance to obstacle
- GNSS – Current GPS location with respect to the geo reference within the OpenDRIVE map

Vehicle Interaction Mechanisms

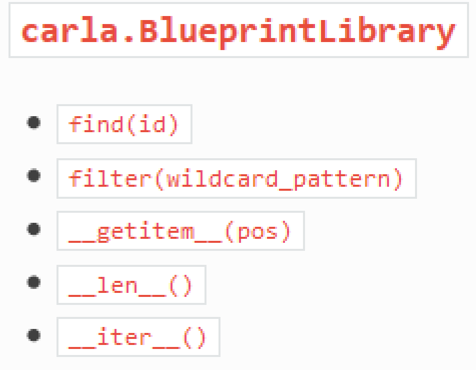
- Autopilot – Vehicles drive using the ground truth of the simulation and a prescribed set of rules
  - Normally drives straight and is easily broken
- Manual Driving – User of script manually controls speed and steering of vehicle
  - Can also be used to control pedestrians
- Scripts / Autonomous Driving – Script sends control signals to vehicle, usually in response to an autonomous driving agent

World API



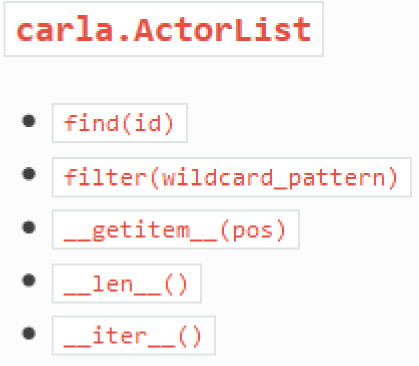
carla.World

- id
- map_name
- debug
- get_blueprint_library()
- get_map()
- get_spectator()
- get_weather()
- set_weather(weather_parameters)
- get_actors()
- spawn_actor(blueprint, transform, attach_to=None)
- try_spawn_actor(blueprint, transform, attach_to=None)
- wait_for_tick(seconds=1.0)
- on_tick(callback)



carla.BlueprintLibrary

- find(id)
- filter(wildcard_pattern)
- __getitem__(pos)
- __len__()
- __iter__()

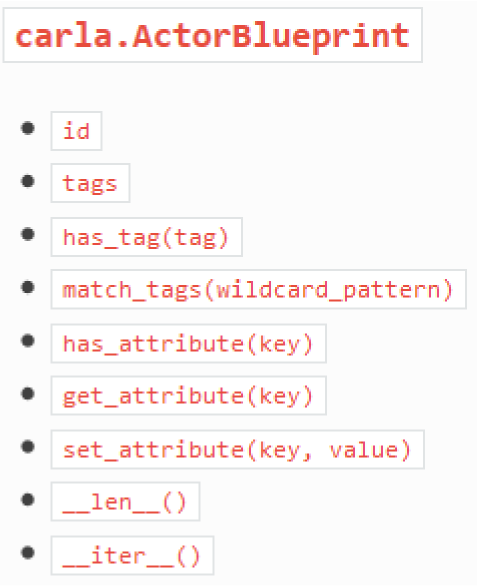


carla.ActorList

- find(id)
- filter(wildcard_pattern)
- __getitem__(pos)
- __len__()
- __iter__()

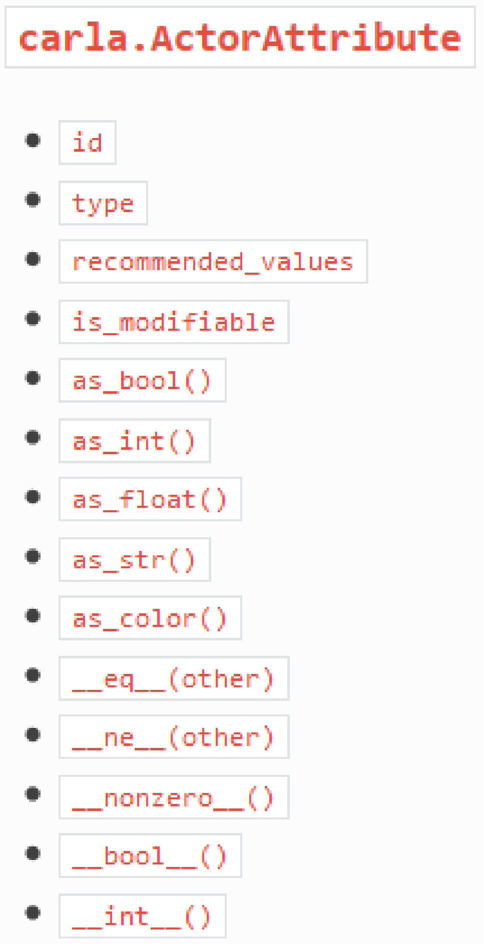
```
transform = Transform(Location(x=230, y=195, z=40), Rotation(yaw=180))
actor = world.spawn_actor(blueprint, transform)
```

Blueprint API



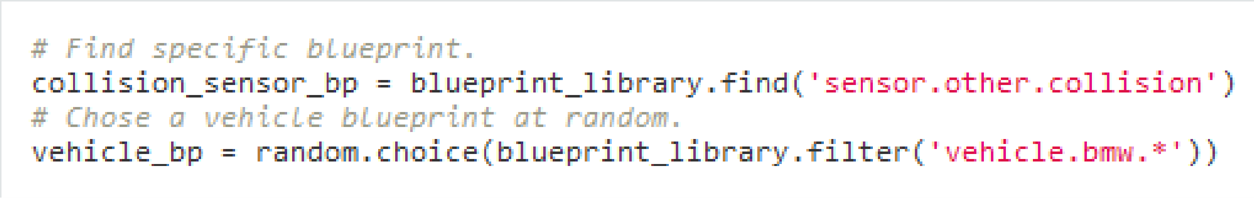
carla.ActorBlueprint

- id
- tags
- has_tag(tag)
- match_tags(wildcard_pattern)
- has_attribute(key)
- get_attribute(key)
- set_attribute(key, value)
- __len__()
- __iter__()

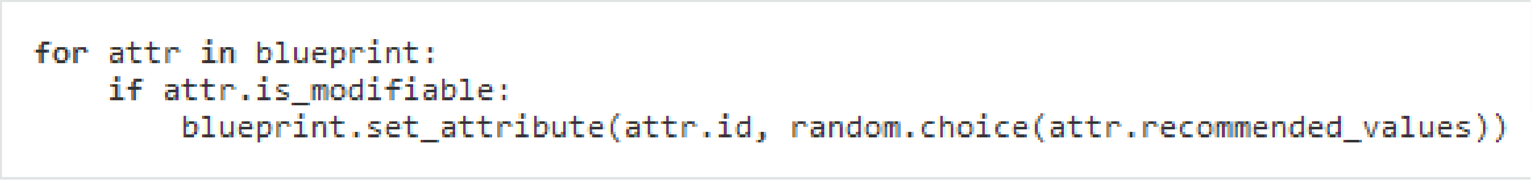


carla.ActorAttribute

- id
- type
- recommended_values
- is_modifiable
- as_bool()
- as_int()
- as_float()
- as_str()
- as_color()
- __eq__(other)
- __ne__(other)
- __nonzero__()
- __bool__()
- __int__()

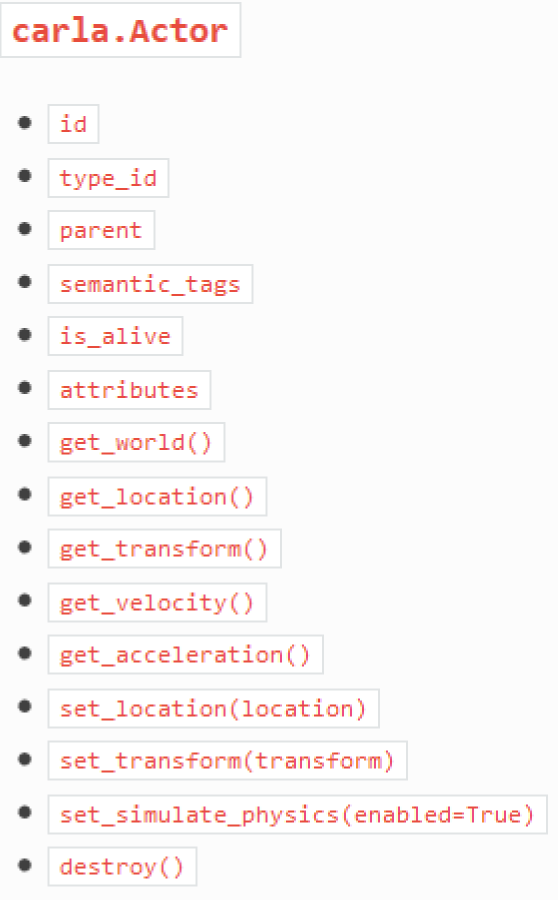


```
# Find specific blueprint.
collision_sensor_bp = blueprint_library.find('sensor.other.collision')
# Chose a vehicle blueprint at random.
vehicle_bp = random.choice(blueprint_library.filter('vehicle.bmw.*'))
```



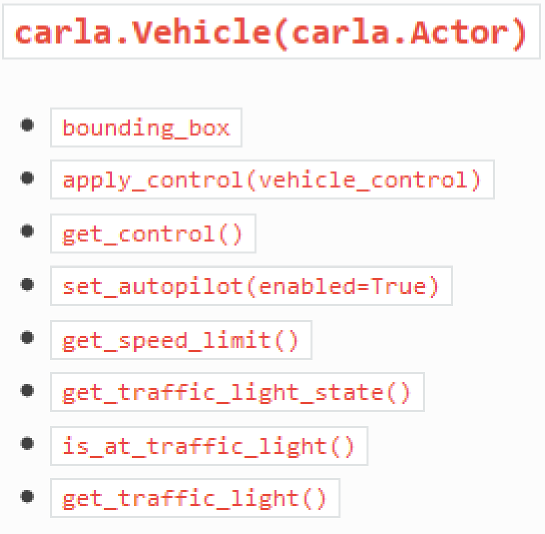
```
for attr in blueprint:
    if attr.is_modifiable:
        blueprint.set_attribute(attr.id, random.choice(attr.recommended_values))
```

Actor API



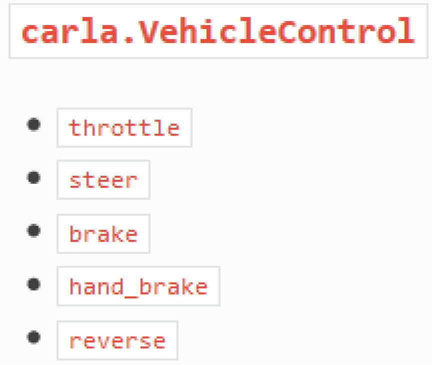
`carla.Actor`

- `id`
- `type_id`
- `parent`
- `semantic_tags`
- `is_alive`
- `attributes`
- `get_world()`
- `get_location()`
- `get_transform()`
- `get_velocity()`
- `get_acceleration()`
- `set_location(location)`
- `set_transform(transform)`
- `set_simulate_physics(enabled=True)`
- `destroy()`



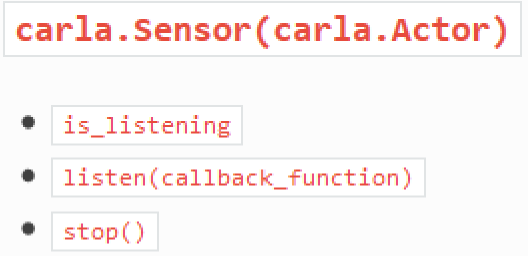
`carla.Vehicle(carla.Actor)`

- `bounding_box`
- `apply_control(vehicle_control)`
- `get_control()`
- `set_autopilot(enabled=True)`
- `get_speed_limit()`
- `get_traffic_light_state()`
- `is_at_traffic_light()`
- `get_traffic_light()`



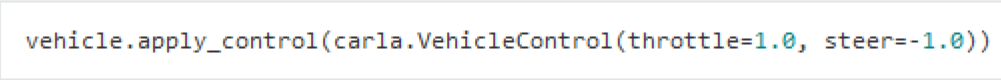
`carla.VehicleControl`

- `throttle`
- `steer`
- `brake`
- `hand_brake`
- `reverse`

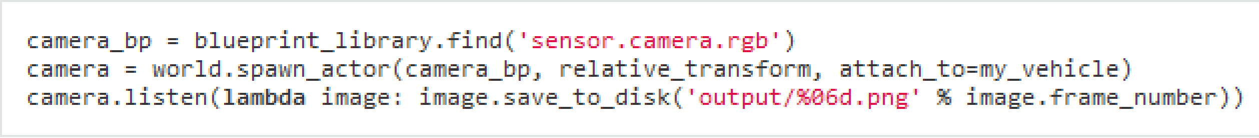


`carla.Sensor(carla.Actor)`

- `is_listening`
- `listen(callback_function)`
- `stop()`



```python
vehicle.apply_control(carla.VehicleControl(throttle=1.0, steer=-1.0))
```

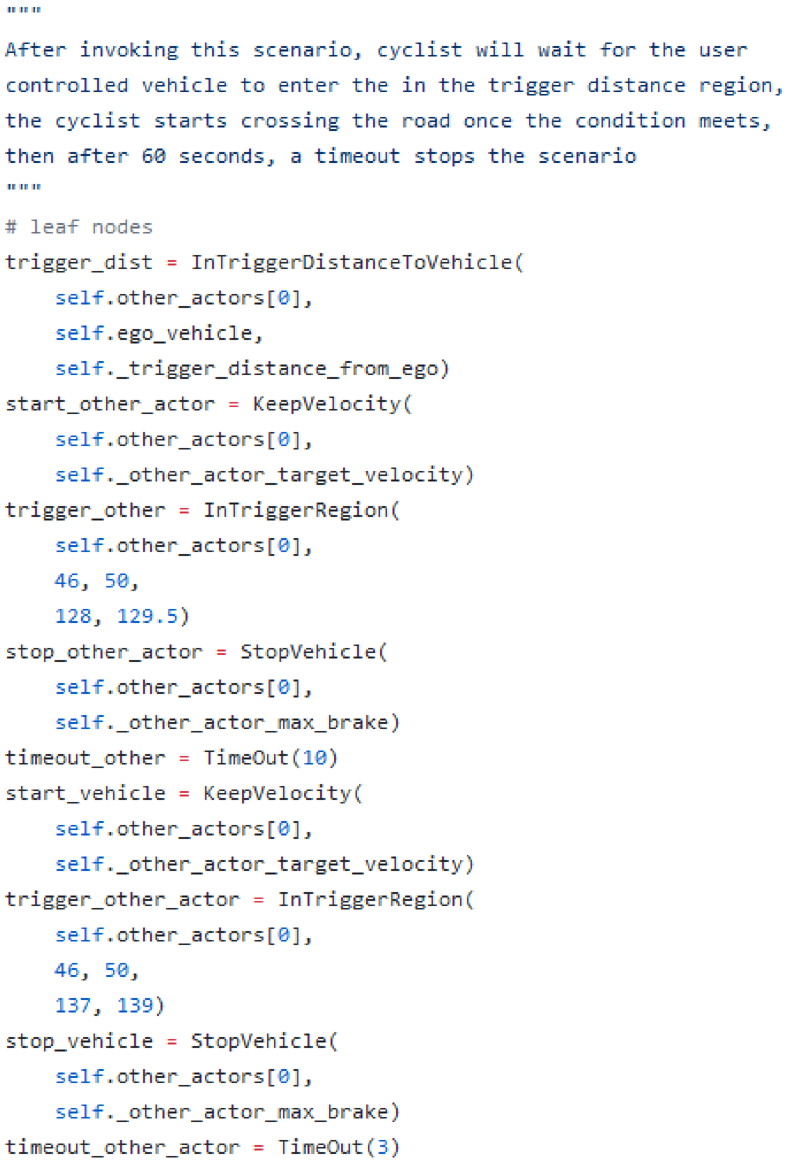


```python
camera_bp = blueprint_library.find('sensor.camera.rgb')
camera = world.spawn_actor(camera_bp, relative_transform, attach_to=my_vehicle)
camera.listen(lambda image: image.save_to_disk('output/%06d.png' % image.frame_number))
```

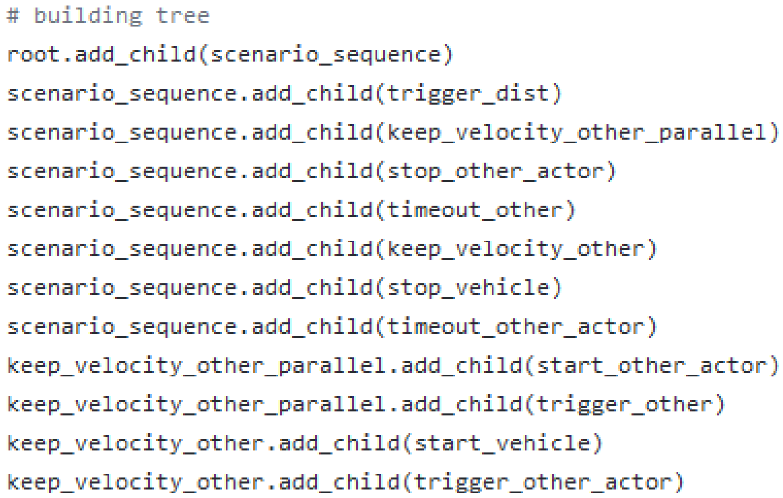
Scenarios

- Scenario Manager allows for defining and running traffic scenarios
- Scenarios are built up from atomic behaviors in trees
  - E.g. following lead vehicle that slows and then stops, obstacle in the road, loss of control
- Criteria can be specified for success / failure
  - E.g. distance driven, average velocity, lane kept

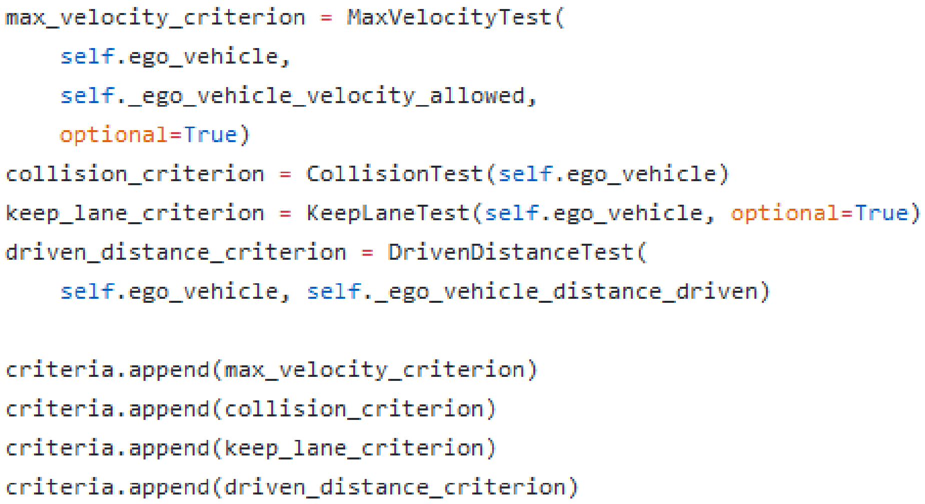
Scenarios - Example



```
"""
After invoking this scenario, cyclist will wait for the user
controlled vehicle to enter the in the trigger distance region,
the cyclist starts crossing the road once the condition meets,
then after 60 seconds, a timeout stops the scenario
"""
# leaf nodes
trigger_dist = InTriggerDistanceToVehicle(
    self.other_actors[0],
    self.ego_vehicle,
    self._trigger_distance_from_ego)
start_other_actor = KeepVelocity(
    self.other_actors[0],
    self._other_actor_target_velocity)
trigger_other = InTriggerRegion(
    self.other_actors[0],
    46, 50,
    128, 129.5)
stop_other_actor = StopVehicle(
    self.other_actors[0],
    self._other_actor_max_brake)
timeout_other = TimeOut(10)
start_vehicle = KeepVelocity(
    self.other_actors[0],
    self._other_actor_target_velocity)
trigger_other_actor = InTriggerRegion(
    self.other_actors[0],
    46, 50,
    137, 139)
stop_vehicle = StopVehicle(
    self.other_actors[0],
    self._other_actor_max_brake)
timeout_other_actor = TimeOut(3)
```



```
# building tree
root.add_child(scenario_sequence)
scenario_sequence.add_child(trigger_dist)
scenario_sequence.add_child(keep_velocity_other_parallel)
scenario_sequence.add_child(stop_other_actor)
scenario_sequence.add_child(timeout_other)
scenario_sequence.add_child(keep_velocity_other)
scenario_sequence.add_child(stop_vehicle)
scenario_sequence.add_child(timeout_other_actor)
keep_velocity_other_parallel.add_child(start_other_actor)
keep_velocity_other_parallel.add_child(trigger_other)
keep_velocity_other.add_child(start_vehicle)
keep_velocity_other.add_child(trigger_other_actor)
```



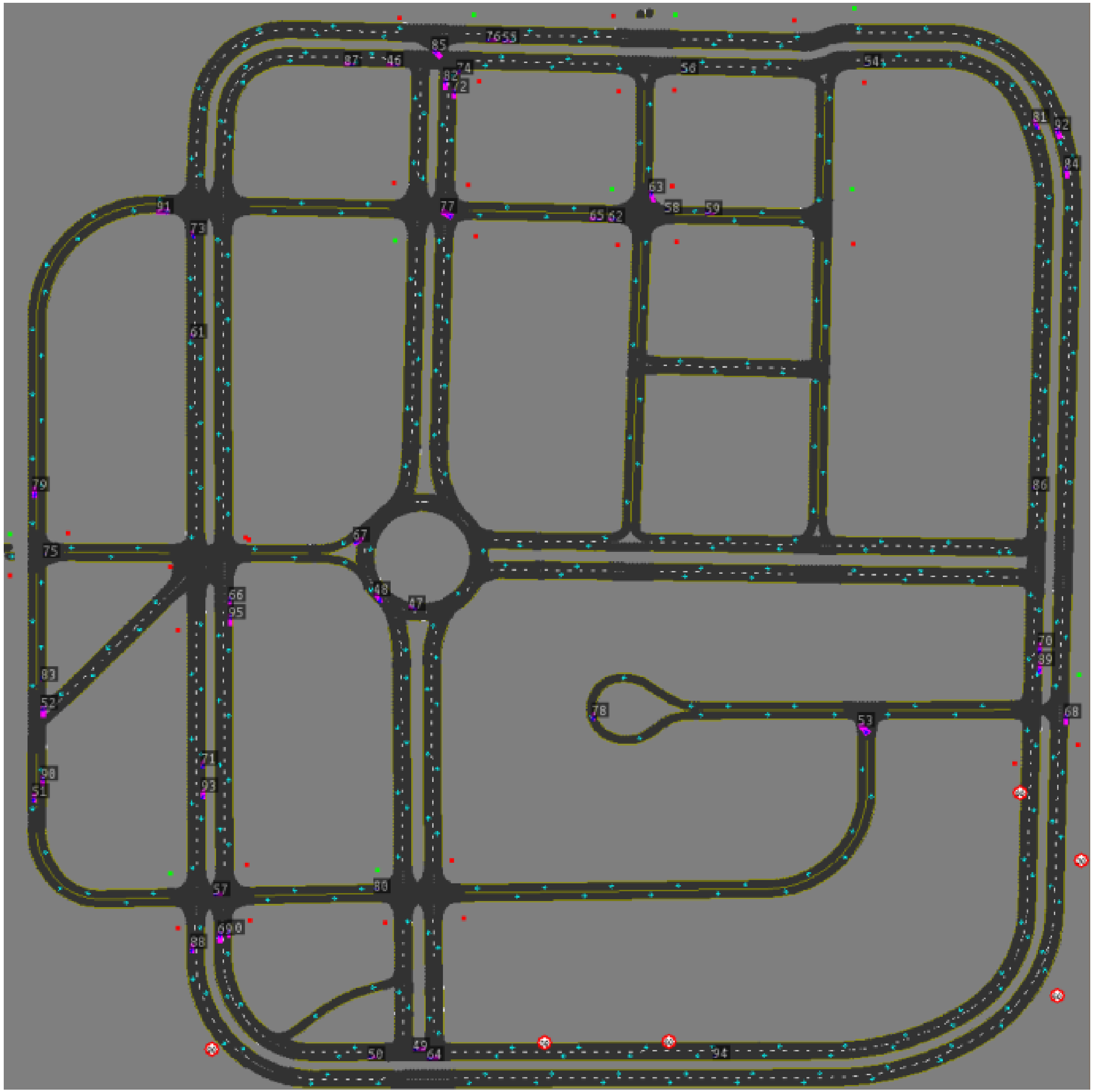
```
max_velocity_criterion = MaxVelocityTest(
    self.ego_vehicle,
    self._ego_vehicle_velocity_allowed,
    optional=True)
collision_criterion = CollisionTest(self.ego_vehicle)
keep_lane_criterion = KeepLaneTest(self.ego_vehicle, optional=True)
driven_distance_criterion = DrivenDistanceTest(
    self.ego_vehicle, self._ego_vehicle_distance_driven)

criteria.append(max_velocity_criterion)
criteria.append(collision_criterion)
criteria.append(keep_lane_criterion)
criteria.append(driven_distance_criterion)
```

Misc. - Weather

- Simulates different illumination and precipitation conditions
- Time of day / Lighting: midday and sunset
- Weather Conditions: cloud cover, precipitation levels, puddles

https://carla.readthedocs.io/en/latest/

Misc. – Overhead View



https://carla.readthedocs.io/en/latest/

Misc. – ROS Bridge

- Allows message passing between simulator and ROS
  - Vehicles publish transform information
  - Sensors publish data stream
  - Publish control messages from ROS
- Ego vehicle is separated from other vehicles

Misc. – Benchmarks

- Runs a set of experiments and produces performance metrics
  - Experiments for: going straight, making a turn, going to a position, going to a position while avoiding traffic
  - Performance based on: percentage success, off road intersection, other lane intersection, collisions, etc.
- Allows for comparison on a common set of experiments

Use-cases

- Testing self-driving algorithms
  - Modular Pipeline
  - Imitation Learning End-to-End
  - Reinforcement Learning End-to-End
- Testing of sensor algorithms
- Testing of semi-autonomous algorithms
- Traffic behavior simulations

Customizability

- Add maps
  - Create new maps with RoadRunner editor, including importing real-world data
- Add vehicles
  - Provide models, textures, and rigs
  - Drag/drop into folder structure

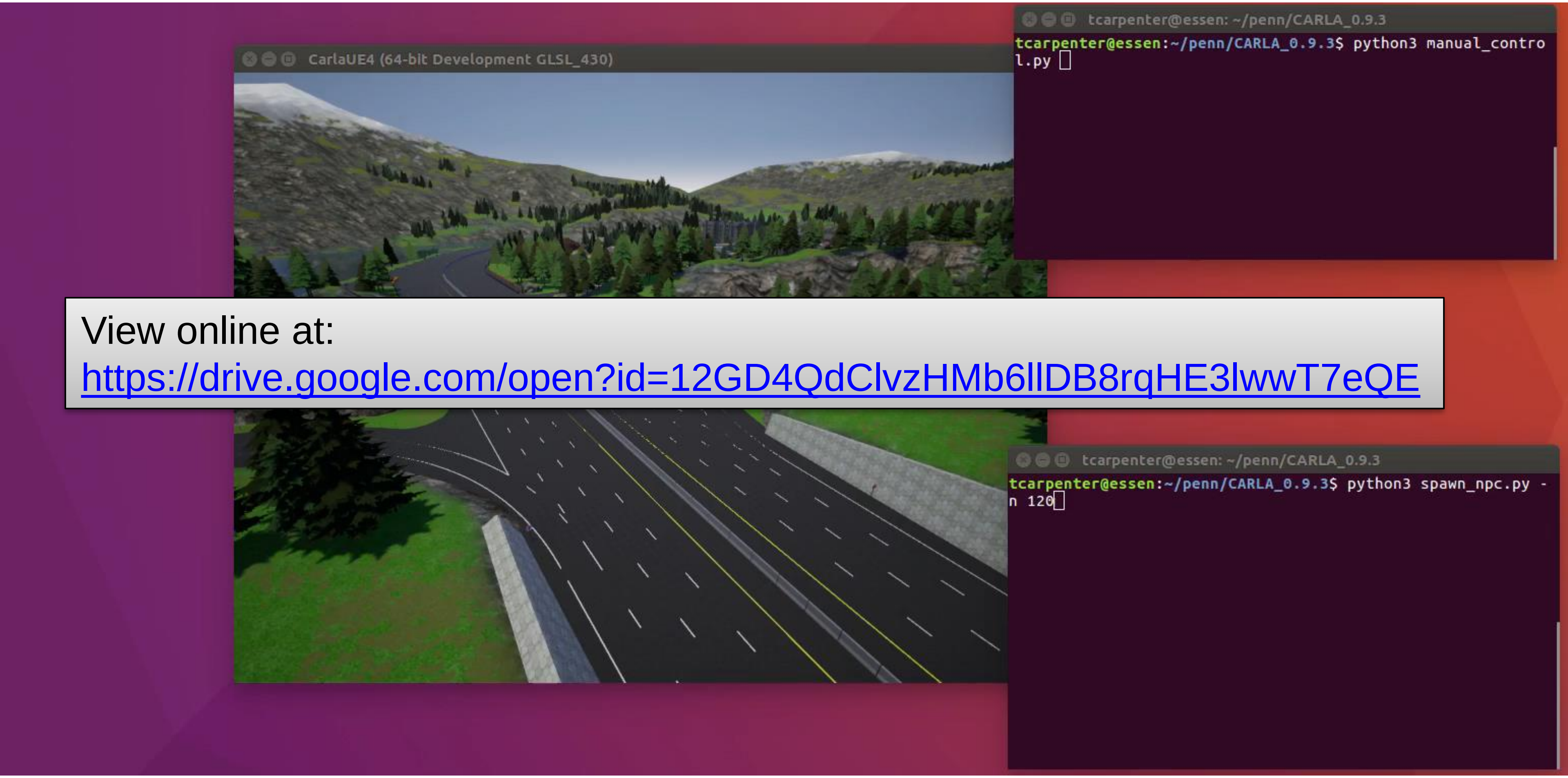
Extensions

- Anomaly / sensor failure models
- Autonomous car coordination (platooning)
- Smart traffic lights

Drawbacks

- Heavy active development
  - Unstable platform due to rapid changes
  - Poor documentation of newer features
- Intersections only allow one green light at a time
- Autopilot cars run through stopsigns
- Resource intensive
- Requires dedicated GPU

Demo


CarlaUE4 (64-bit Development GLSL_430)
tcarpenter@essen: ~/penn/CARLA_0.9.3
tcarpenter@essen: ~/penn/CARLA_0.9.3

```
tcarpenter@essen:~/penn/CARLA_0.9.3$ python3 manual_contro
l.py
```

```
tcarpenter@essen:~/penn/CARLA_0.9.3$ python3 spawn_npc.py -
n 120
```

View online at:
https://drive.google.com/open?id=12GD4QdClvzHMb6lIDB8rqHE3lwwT7eQE

References

- CARLA - http://carla.org/
- RoadRunner - https://www.vectorzero.io/
- Unreal Engine 4 - https://www.unrealengine.com
- AirSim - https://github.com/Microsoft/AirSim
- ROS - http://www.ros.org/
- Autoware - https://www.autoware.ai/
- OpenDRIVE - http://www.opendrive.org/
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V.. (2017). CARLA: An Open Urban Driving Simulator. *Proceedings of the 1st Annual Conference on Robot Learning, in PMLR* 78:1-16
- Kiran, B., Roldao, L., Irastorza, B., Verastegui, R., Suss, S., Yogamani, S., Talpaert, V., Lepoutre, A. & Trehard, G. (2018). Real-time Dynamic Object Detection for Autonomous Driving using Prior 3D-Maps*.*