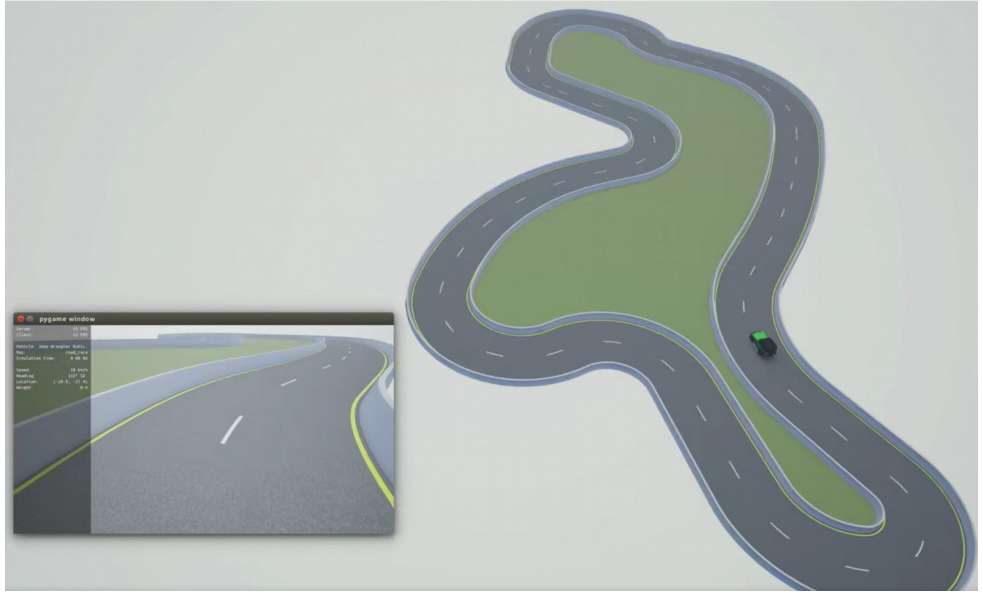
Environment setup and Spawning a vehicle



CARLA courses roadmap

MoCAD Experimental Course Schedule (Carla-python)

	Course title	Course contents	Projects
D1	Environment setup	1. Course introduction 2. Python Environment anaconda 3. Carla quick start installation and linux build 4. Spawning a vehicle in Carla with your own map (RoadRunner) 5. Carla core concepts	
D2	Running a vehicle by keyboard and collecting data	1. Control a vehicle by apply_control method and keyboard 2. Attach a rgb-image sensor on the vehicle 3. Simulation time-step 4. Try different sensors: RGB-camera, Depth-camera, Lidar, Obstacle ...	**Simple: Sensors** Control a vehicle by keyboard and use Carla python API to collect data from different sensors.
D3	Running a vehicle by PID control	1. Mapping and waypoint 2. Global path planning 3. Local planning 4. PID controller	
D4	Running a vehicle by behavior clone	1. Collecting data 2. Supervised learning 3. Training Neural Network 4. Control a vehicle by the trained NN	**Intermediate: Leader-follower instance** Use the keyboard to control the leader (first vehicle) and the second vehicle follows the leader by PID or behavior clone.
D5	Running a vehicle by reinforcement learning I	1. Introduce the reinforcement learning and DQN 2. Create an Carla environment 3. Building a DQN network 4. Python multi-threading 5. Training the network, agent interacts with Carla environment 6. Control a vehicle by the trained NN	
D6	Running a vehicle by reinforcement learning II	1. Continue action 2. Multi-class regression problem 3. Future work	**Complex: Racing** Use all the knowledge you have learned to control the vehicle so that it can complete a lap on the race road as quickly as possible.

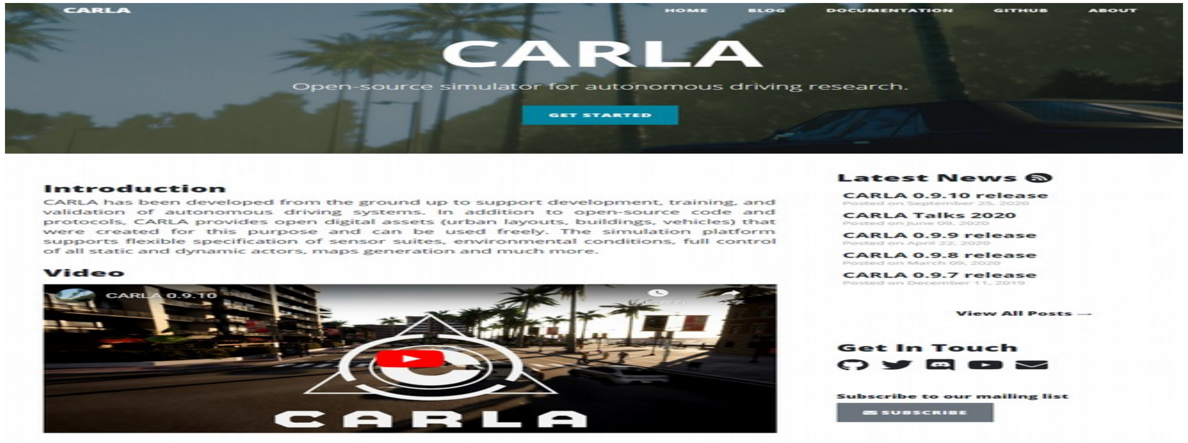
Course contents

1. Introduce Carla
2. Carla installation
3. Carla Linux build
4. Add your own map with Roadrunner
5. CARLA Core concepts
6. Spawning a vehicle in Carla

CARLA simulator

- CARLA is an **open-source** autonomous driving simulator.

  CARLA, 0.9.0 or later



CARLA HOME BLOG DOCUMENTATION GITHUB ABOUT

CARLA

Open-source simulator for autonomous driving research.

GET STARTED

Introduction

CARLA has been developed from the ground up to support development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, maps generation and much more.

Video

CARLA 0.9.10

Latest News

CARLA 0.9.10 release
Posted on September 25, 2020

CARLA Talks 2020
Posted on June 09, 2020

CARLA 0.9.9 release
Posted on April 22, 2020

CARLA 0.9.8 release
Posted on March 09, 2020

CARLA 0.9.7 release
Posted on December 11, 2019

View All Posts →

Get In Touch

Subscribe to our mailing list

SUBSCRIBE

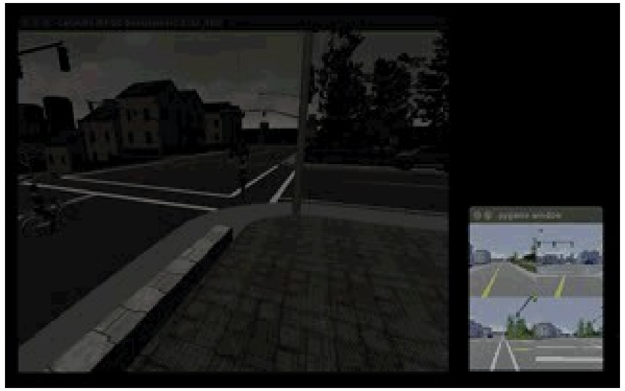
- **What** is Carla ?

  Why we use Carla ?

  How we use Carla ?

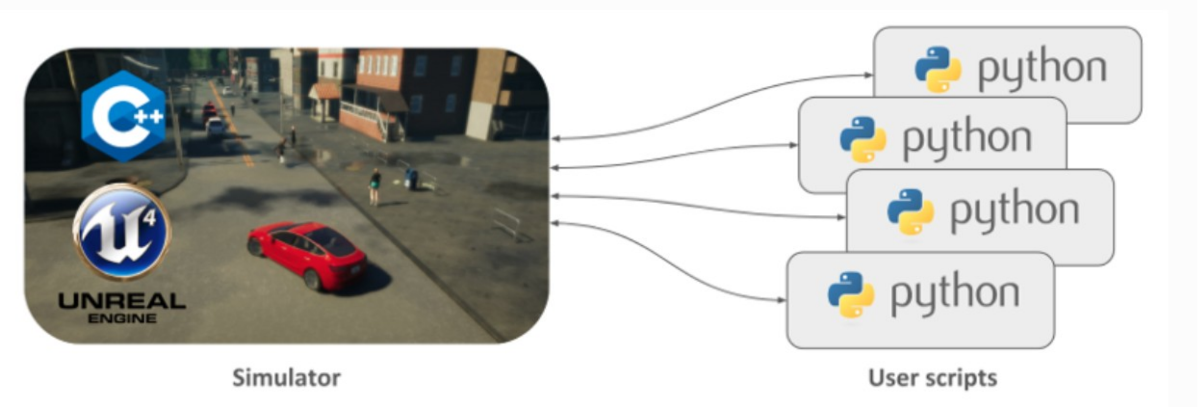
Unreal Engine 4.24

OpenDRIVE 1.4 (roads and urban settings)



The general problem of driving
(e.g. learning driving policies, training perception algorithms, etc.)

CARLA architecture: Server-Client



CARLA, 0.9.0 or later

- Server (Simulator)

1. Update the world and actor states;
2. **Sensors listen the data;**
3. Simulate real world GPU；

- Client (Python API)

1. Setting the world （scenario， weather， actor）；
2. **Control the actors;**

CARLA installation

- Quick start installation

1. A pre-packaged version of CARLA;
2. With Python API;
3. Without advanced customization, development options;

Demo:

# Linux: > ./CarlaUE4.sh
# Windows: > CarlaUE4.exe *python spawn_npc.py*

- Server side. A **4GB minimum GPU** will be needed to run a highly realistic environment. A dedicated GPU is highly advised for machine learning.
- Client side. **Python** is necessary to access the API via command line. Also, a good internet connection and two **TCP ports** (2000 and 2001 by default).
- System requirements. Any **64-bits OS** should run CARLA. However, since release 0.9.9, CARLA cannot run in 16.04 Linux systems with default compilers. These should be upgraded to work with CARLA.
- Other requirements. Two Python modules: **Pygame** to create graphics directly with Python, and **Numpy** for great calculus.

- Linux build

1. Ubuntu 18.04 Dependencies;
2. Unreal Engine 4.24 **only;**
3. CARLA build;

Demo:

# Linux: > make launch
python spawn_npc.py

- **Ubuntu 18.04**. CARLA provides support for previous Ubuntu versions up to 16.04. However proper compilers are needed for UE to work properly. Dependencies for Ubuntu 18.04 and previous versions are listed separately below. Make sure to install the ones corresponding to your system.
- **30GB disk space**. The complete build will require quite a lot of space, especially Unreal Engine. Make sure to have around 30/50GB of free disk space.
- An adequate GPU. CARLA aims for realistic simulations, so the server needs at least a **4GB GPU**. A dedicated GPU is highly recommended for machine learning.
- **Two TCP ports** and good internet connection. 2000 and 2001 by default. Be sure neither the firewall nor any other application block these.

CPU: i7-9700
RAM: 16G
GPU: 1080 Ti
Disk: 1T ¥ 8000

Python installation

- Anaconda: python management

1. conda create –n env-name python=3.7
2. conda activate env-name
3. conda install / pip install package-name

- Pytorch & pygame

1. conda install pytorch torchvision cudatoolkit=10.2 -c pytorch
2. Pip install pygame

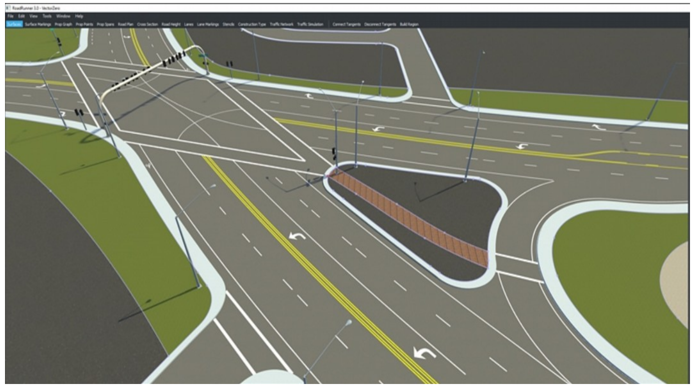
- Pycharm: python IDE

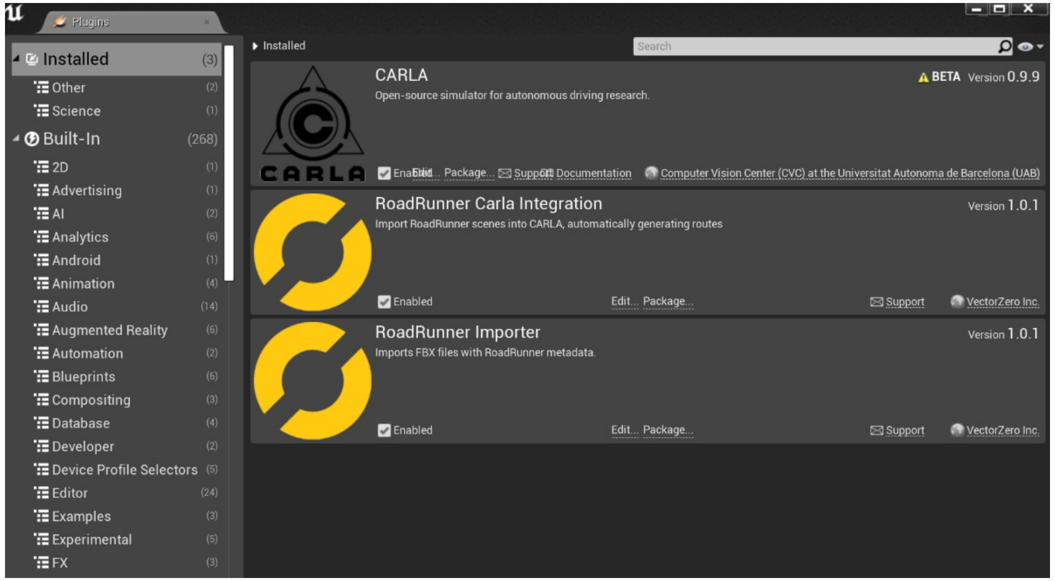
Import conda interpretation

Add your own map in Unreal Engine

- Create a map with RoadRunner (Matlab)

1. The resulting map should consist of a **.fbx** and a **.xodr** with the mesh and road network informtion respectively
2. Create a map with RoadRunner (**.fbx** and a **.xodr**)
3. Importing into CARLA/Unreal (**RoadRunner plugin**)
4. Creating Map Packages for Distribution (without building)



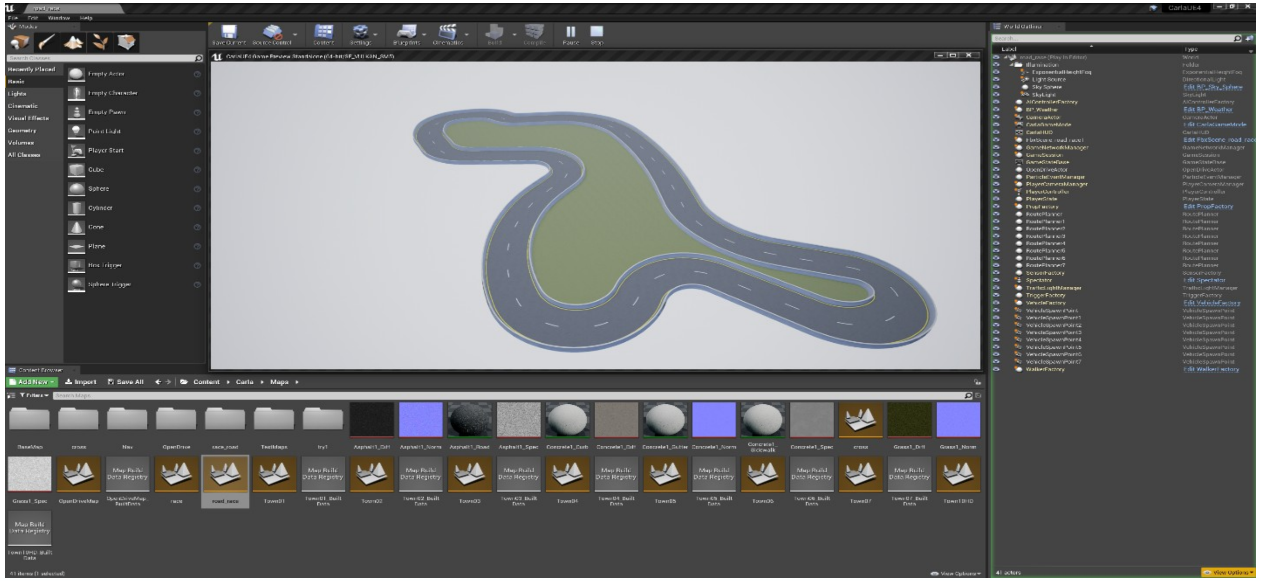


Category	Count
Installed	(3)
Other	(2)
Science	(1)
Built-In	(268)
2D	(1)
Advertising	(1)
AI	(2)
Analytics	(6)
Android	(1)
Animation	(4)
Audio	(14)
Augmented Reality	(6)
Automation	(2)
Blueprints	(6)
Compositing	(3)
Database	(4)
Developer	(2)
Device Profile Selectors	(5)
Editor	(24)
Examples	(3)
Experimental	(5)
FX	(3)

CARLA — BETA Version 0.9.9
Open-source simulator for autonomous driving research.
☑ Enabled Edit... Package... Support Documentation Computer Vision Center (CVC) at the Universitat Autonoma de Barcelona (UAB)

RoadRunner Carla Integration — Version 1.0.1
Import RoadRunner scenes into CARLA, automatically generating routes
☑ Enabled Edit... Package... Support VectorZero Inc.

RoadRunner Importer — Version 1.0.1
Imports FBX files with RoadRunner metadata.
☑ Enabled Edit... Package... Support VectorZero Inc.



- Environment

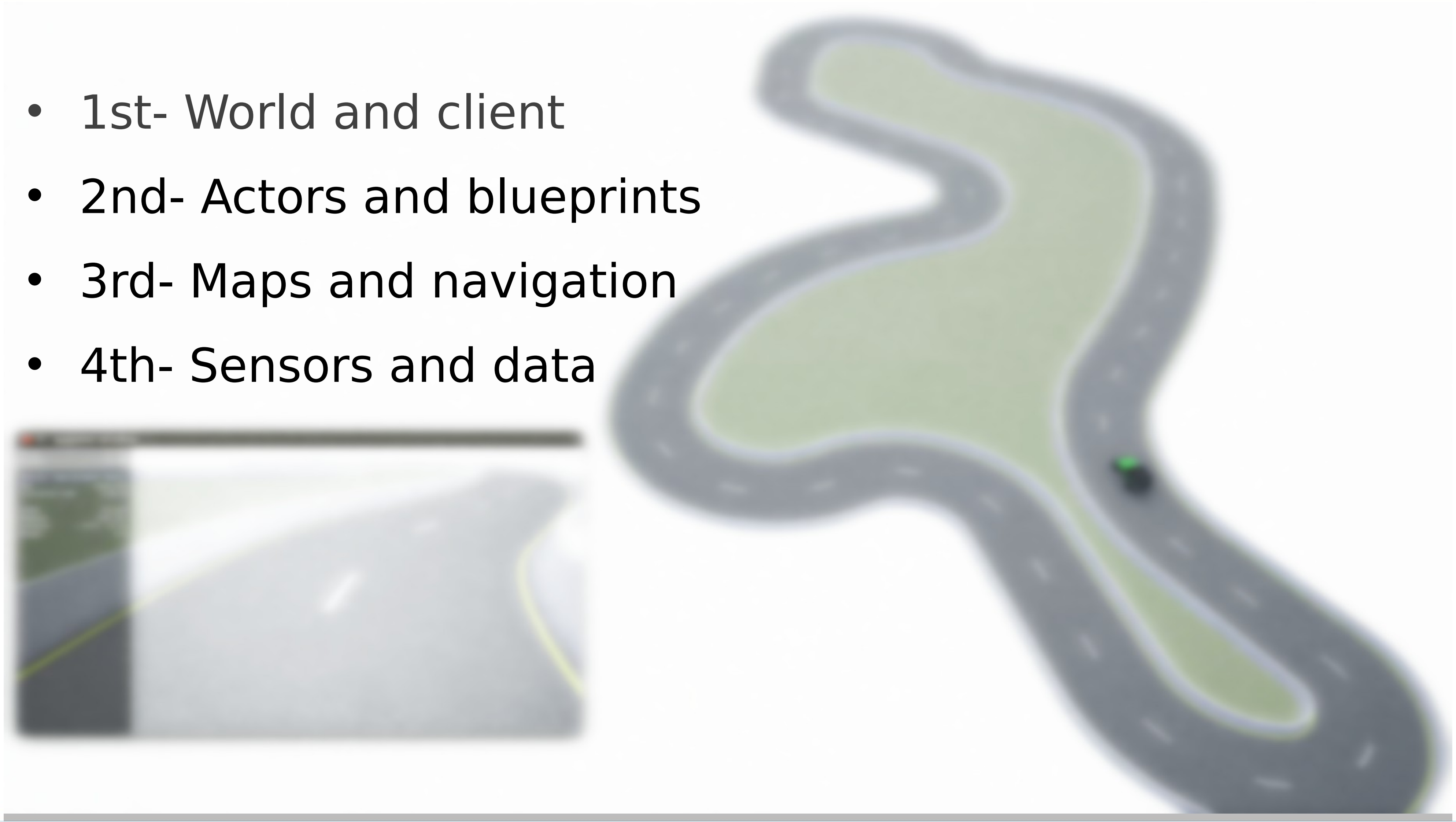
UE4： 4.24
Carla： 0.9.9.4
Python： 3.5
Pytorch: 1.5.0

- Bug

UE4: github, running UE4 edit
Carla： python – client, anaconda python
Comment anaconda
Update about 12G
CarlaUE4 project slowly

CARLA Core concepts

- 1st- World and client
- 2nd- Actors and blueprints
- 3rd- Maps and navigation
- 4th- Sensors and data



CARLA Core concepts

- 1st- World and client

  1. Client: an IP and a specific port ;
  2. only one world per simulation ;
  3. World: **spawn actors,** change the weather, get the current state of the world ;

Note:

1. Client and server have different *libcarla* modules. If the versions differ, issues may arise. *get_client_version()* and *get_server_version()*
2. Changes in the weather do *not affect physics*. They are only visuals that can be captured by the *camera sensors*.

Demo(**carla.Client**):

1. Client creation

```
client = carla.Client('localhost', 2000)
client.set_timeout(10.0) # seconds
```

2. World connection

```
world = client.get_world()
world = client.load_world('Town01')
```

Demo(carla.World):

1. Actors

2. Weather

world.set_weather(weather)

3. Lights

4. Debugging

debug = world.debug

5. World snapshots

world_snapshot = world.get_snapshot()

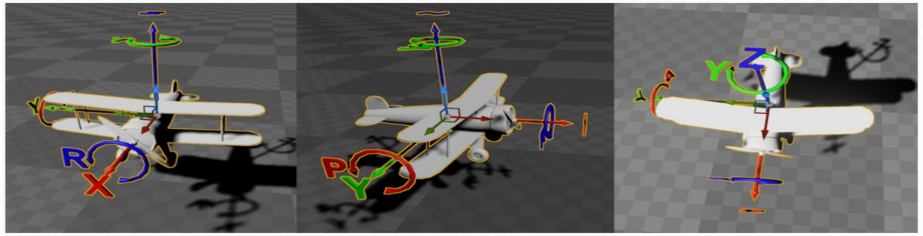
6. World settings

simulation time-steps
synchrony between clients and server

CARLA Core concepts

- 2nd- Actors and blueprints

1. An **actor** is anything that plays a role in the simulation ;
2. **Blueprints** are already-made actor layouts



Unreal Engine's coordinates system.

Note:

1. Some of the attributes *cannot be modified*. Check it out in the blueprint library.

2. CARLA uses the Unreal Engine coordinates system. Remember that carla.Rotation constructor is defined as (pitch, yaw, roll), that differs from Unreal Engine Editor (roll, pitch, yaw).

Demo(carla.ActorBlueprint):

1. Managing the blueprint library

```
blueprint_library = world.get_blueprint_library()
# Choose a vehicle blueprint at random.
vehicle_bp =
random.choice(blueprint_library.filter('vehicle.*.*'))
vehicle_bp.set_attribute('color', '255,0,0')
```

2. Spawning

```
actor = world.spawn_actor(blueprint, transform)
```

Demo(carla.Transform):

1. Stating a location and rotation for the actor

```
transform = Transform(Location(x=230, y=195, z=40),
Rotation(yaw=180))
```

Demo(carla.Actor):

1. get() and set()

```
actor.get_location()/ actor.get_velocity()
actor.set_location(location)
```

2. Destruction

```
destroyed_sucessfully = actor.destroy() # Returns True if
successful
```

CARLA Core concepts

- 2nd- Actors and blueprints

Types of actors： Sensors， Spectator， Traffic signs and traffic lights， **Vehicles**， Walkers

Demo(**carla.Vehicle**):

1. **carla.VehicleControl**

vehicle.apply_control(carla.VehicleControl(throttle=1.
0, steer=-1.0))

2. **carla.VehiclePhysicsControl**

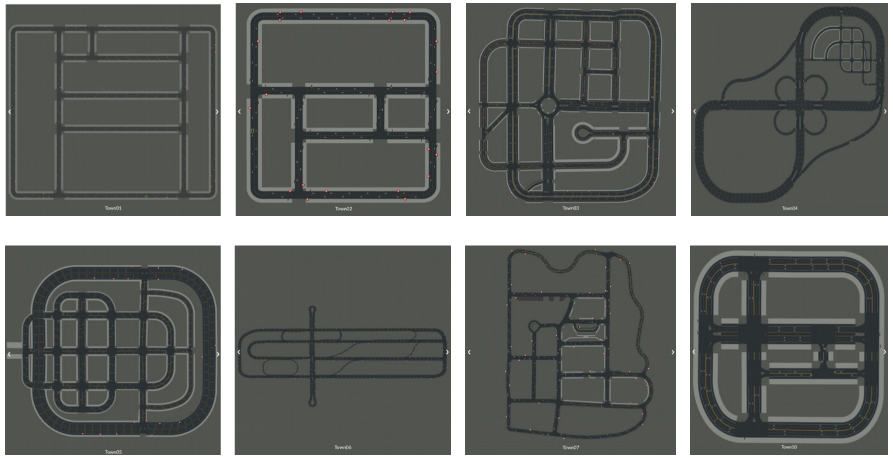
3. **carla.VehicleLightState**

4. vehicle.set_autopilot(**True**)
5. vehicle.bounding_box

CARLA Core concepts

- 3rd- Maps and navigation

1. A map includes both the **3D model of a town** and its **road definition** ;
2. The **OpenDRIVE** defines roads, lanes, junctions, etc. is extremely important ;



Demo(Carla.Client):

1. Changing the map

world = client.load_world('Town01')

Demo(Landmarks):

1. carla.Landmark

All the information defining a landmark in OpenDRIVE

2. carla.LandmarkOrientation

The orientation of a landmark in the road

3. carla.LandmarkType

A set of commonly used landmark types as defined by the default country code

4. carla.Waypoint

A carla.Transform and road information

5. carla.Map

The road information and waypoint managing

Demo(**Waypoint**):

1. a carla.Transform: location on the map and the orientation of the lane
2. The variables road_id,section_id,lane_id

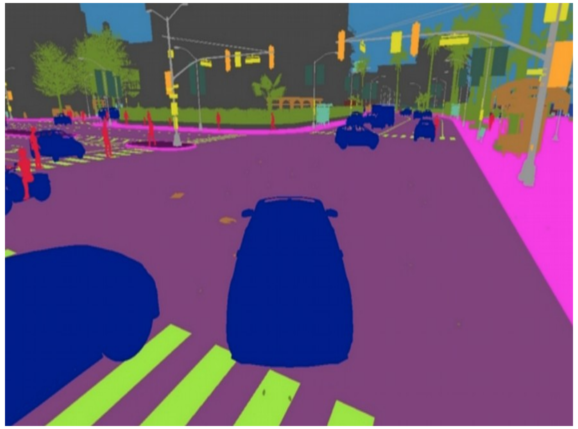
Demo(Navigating through waypoints):

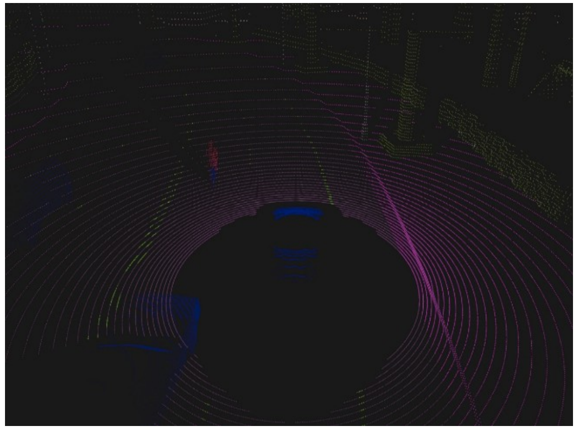
1. Waypoints create a road flow

CARLA Core concepts

- 4th- Sensors and data

1. Gather data from the simulation . The waypoint class to provide vehicles with a navigation path；
2. Different types of sensor data；
3. A sensor is an actor attached to a parent vehicle；
4. Cameras， Collision， IMU, Obstacle … ;





Reference

la Documentation

/carla.readthedocs.io/en/latest/start_introduction/

esome CARLA

//github.com/Amin-Tgz/awesome-CARLA

dRunner map

//tracetransit.atlassian.net/wiki/spaces/VS/pages/752779356/Exporting+to+CARLA