

INSTRUKCJA PROGRAMISTY

1. Link do oficjalnej dokumentacji BeamNG Tech:

Instalacja: https://documentation.beamng.com/beamng_tech/install/

Dokumentacja API: <https://beamngpy.readthedocs.io/en/latest/>

2. Uruchamianie projektu

Należy uzyskać licencję od BeamNG. Otrzymaliśmy link, którego nie możemy tu wkleić, gdzie pobieramy cały projekt BeamNG. Do głównego folderu projektu należy wkleić **klucz** licencji, który również otrzymaliśmy od BeamNG. Następnie otworzyć plik exe, bądź przez przygotowane oprogramowanie API w pythonie, o którym później.

3. Edytor świata

Po załadowaniu mapy, w grze, kliknąć F11.



W edytorze świata, można uzyskać dostęp do większości obiektów w grze, m.in. edycji mapy, oraz nagrywania tras (dla trafficów), pkt 7

4. Dodawanie własnych map

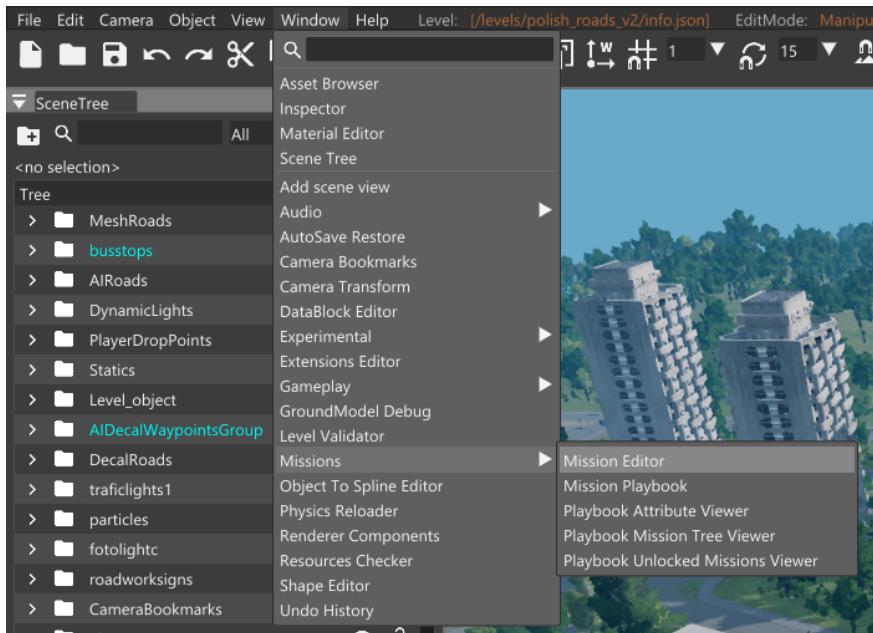
Aby dodać mapę należy pobrać plik .zip i przenieść go do BeamNG\content\levels. My korzystaliśmy z mapy Polski:

<https://www.beamng.com/resources/polish-roads.5846/>

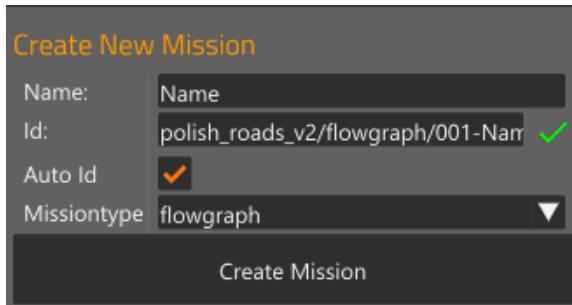
5. Dodawanie scenariuszy przez map editor

Jest to pierwszy z dwóch sposobów dodawania scenariuszy (drugi dotyczy API, opisany później). Umożliwia on dodawanie własnych obiektów dla konkretnego scenariusza (jest to użyteczne, gdy chcemy mieć np. inne znaki albo przeszkody na tej samej mapie dla różnych scenariuszy). Daje też do dyspozycji Flowgraph, czyli system programowania scenariuszy oparty na "node'ach".

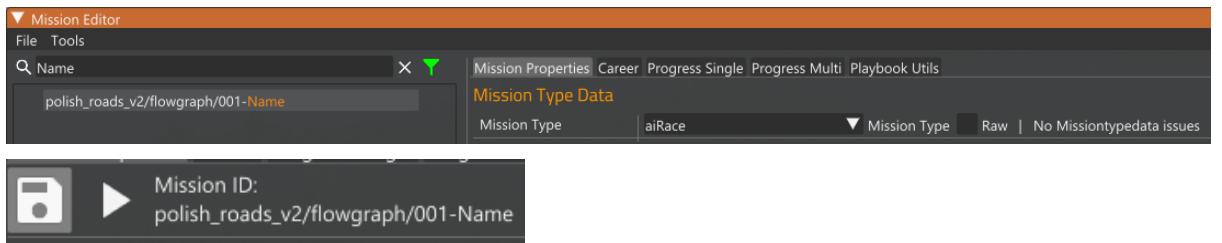
Aby utworzyć nowy scenariusz lub przejrzeć obecne, wejdź w Window -> Missions -> Mission editor



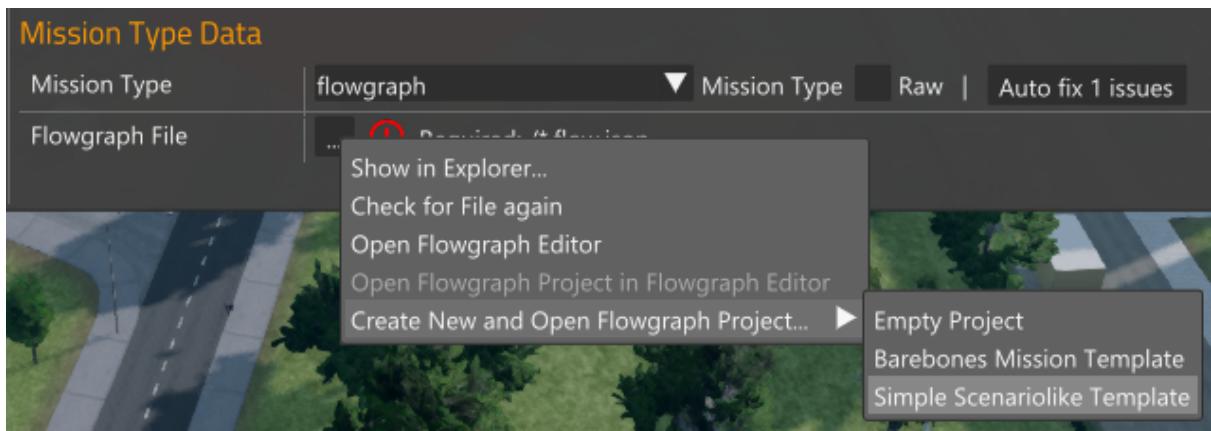
Następnie wybierz File -> New mission i jako typ wybierz flowgraph



Przed błędem, po wybraniu z listy stworzonej misji jej typ będzie ustawiony na aiRace, należy go ponownie zmienić na flowgraph i zapisać ikonką na górze.

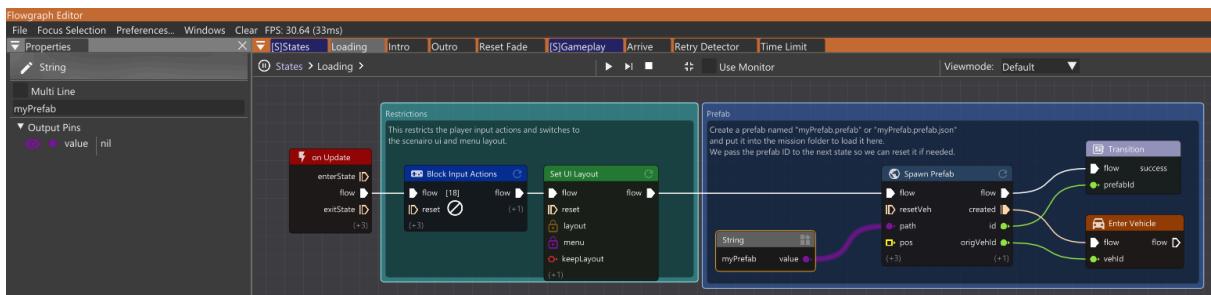


Następnie w Flowgraph File należy wybrać:



Otworzy się Flowgraph editor.

Tu ważnym elementem jest string "myPrefab" w sekcji loading, który określa nazwę prefabu, czyli zbioru zawierającego obiekty tylko dla danego scenariusza. Będzie on załadowany razem ze scenariuszem. (Tworzenie prefabów jest dobrze opisane na YT)



6. Dodawanie obiektów do konkretnych scenariuszy

Aby stworzyć prefab, należy zaznaczyć kilka obiektów z listy i wybrać "Pack into prefab", potem nadać mu nazwę (pamiętać o zmianie stringa we flowgraphie).

Plik prefaba (nazwa.prefab.json) przenieść do [userpath]\

AppData\Local\BeamNG.drive\[wersja]\levels\[nazwa mapy]\scenarios

BeamNG.drive > 0.31 > levels > polish_roads_v2 > scenarios	
Nazwa	Data modyfikacji
Mission 1.json	24.06.2024 22:06
Mission 1.prefab.json	24.06.2024 22:06

7. API

Na samym początku należy ustawić scieżkę home i user w pliku backend/simulator.py

Home - ścieżka do pobranej wersji BeamNG.tech

User - najlepiej ustawić na folder BeamNG.tech w Appdata, który powinien utworzyć się po pierwszym uruchomieniu gry.

```

beamng = BeamNGpy(
    host="localhost",
    port=64256,
    home="D:\BeamNG.tech\BeamNG.tech.v0.31.2.0",
    user="C:\Users\[user]\Appdata\Local\BeamNG.tech",
)
beamng.open()

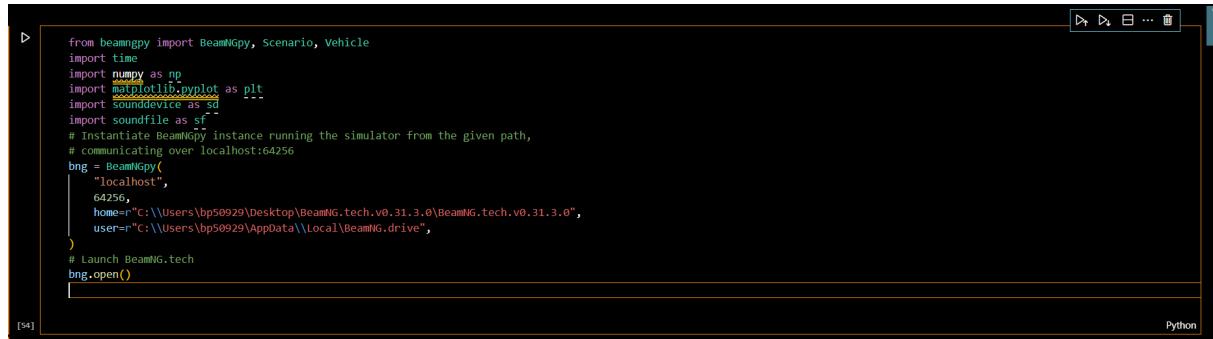
```

Zalecane jest korzystanie z Jupytera do testowania scenariuszy.

Najważniejsza dla API jest dokumentacja BeamNG. Niestety, wiele informacji jest przestarzałych bądź ich po prostu brakuje.

Poprzez API w pythonie możemy m.in.

uruchomić BeamNG:



```

> from beamngpy import BeamNGpy, Scenario, Vehicle
> import time
> import numpy as np
> import matplotlib.pyplot as plt
> import sounddevice as sd
> import soundfile as sf
> # Instantiate BeamNGpy instance running the simulator from the given path,
> # communicating over localhost:64256
> bng = BeamNGpy(
>     "localhost",
>     64256,
>     home=r'C:\\\\Users\\bp50929\\Desktop\\BeamNG.tech.v0.31.3.0\\BeamNG.tech.v0.31.3.0',
>     user=r'C:\\\\Users\\bp50929\\AppData\\Local\\BeamNG.drive',
> )
> # Launch BeamNG.tech
> bng.open()

```

[54] Python

dodawać pojazdy:



```

> vehicle1 = Vehicle("Traffic1", model="etk800", license="Traffic1", color='blue')
> vehicle2 = Vehicle("Traffic2", model="pickup", license="Traffic2", color='purple')
> vehicle3 = Vehicle("Traffic3", model="pessima", license="Traffic3", color='yellow')
> vehicle4 = Vehicle("Traffic4", model="van", license="Traffic4", color='red', color2='black')
> vehicle = Vehicle("ego_vehicle", model="burnside", license="ZUT")
> vehicle5 = Vehicle("Traffic5", model="bluebuck", license="Traffic5", color='green')
> vehicle6 = Vehicle("TESTER", model="bastion", license="TESTER", color='blue')
> #print(bng.vehicles.get_available())

```

[6]

tworzyć scenariusze:

```
> scenario2 = Scenario("polish_roads_v2", "Mission 1",)

scenario2.add_vehicle(
    vehicle, pos=(440.91, 754.78, -18.5), rot_quat=(0, 0, 1, 0)
)
scenario2.add_vehicle(
    vehicle3, pos=(430, 754.78, -18.5), rot_quat=(0, 0, 1, 0)
)
scenario2.add_vehicle(
    vehicle4, pos=(420, 754.78, -18.5), rot_quat=(0, 0, 1, 0)
)
scenario2.add_vehicle(
    vehicle5, pos=(410, 754.78, -18.5), rot_quat=(0, 0, 1, 0)
)
scenario2.add_vehicle(
    vehicle6, pos=(400, 754.78, -18.5), rot_quat=(0, 0, 1, 0)
)
scenario2.add_vehicle(
    vehicle1, pos=(462.9, 627.44, -18.6), rot_quat=(0, 0, 1, 0)
)
scenario2.add_vehicle(
    vehicle2, pos=(470, 754.78, -18.5), rot_quat=(0, 0, 1, 0)
)

scenario2.make(bng)
```

i wiele innych.

UWAGA!

Aby wczytać i uruchomić scenariusz należy użyć komend:

```
bng.scenario.load(scenario2)
```

lub F11

Load wczytuje scenariusz. Nie kliknąć “Rozpocznij” w grze, ponieważ to spowoduje bug, który uniemożliwiłąknięcie edycji świata (F11).

Kod **bng.scenario.start()** klikna przycisk “Rozpocznij” za użytkownika (należy ewentualnie chwilkę zaczekać)

8. Aplikacja webowa:

Link do repozytorium: https://github.com/szvbvtk/beamng_tech_driving_simulator/tree/main/

- Dodanie możliwości uruchomienia nowego scenariusza z poziomu przeglądarki: należy uzupełnić plik src/assets/data/scenarios.json nowymi danymi

```
({}) scenarios.json ×
src > assets > data > ({}) scenarios.json > ...
You, 2 weeks ago | 1 author (You)
1 [ ↴
2   [
3     {
4       "id": 1,
5       "name": "Reakcja na znak STOP",
6       "description": "Ustąpienie pierwszeństwa przed znakiem stopu, znalezienie luki w ciągu samochodów i ruszenie z miejsca",
7       "command": "run_scenario_1",
8       "img_name": "scenario_1.jpg"
9     },
10    {
11      "id": 2,
12      "name": "Jazda drogą z pierwszeństwem",
13      "description": "Reakcja na próbę wymuszenia pierwszeństwa przez kierowcę z drogi podporządkowanej",
14      "command": "run_scenario_2",
15      "img_name": "scenario_2.jpg"
16    }
17 ]
18 ]
```

- Dodawanie kodu nowego scenariusza (backend/simulator.py): utworzenie funkcji scenario_{id}_loop, np.:

```
def scenario_1_loop():
    global vehicle_data
    global vehicle
    global isGameRunning
    print("Scenario 1 loop...")
    scenario = Scenario("west_coast_usa", "example")
    # Create an ETK800 with the licence plate 'PYTHON'
    vehicle = Vehicle(vid="kkk", model="etk800", license="PYTHON")
    vehicle.sensors.attach("electrics", Electrics())
    # Add it to our scenario at this position and rotation
    scenario.add_vehicle(
        vehicle, pos=(-717, 101, 118), rot_quat=(0, 0, 0.3826834, 0.9238795)
    )

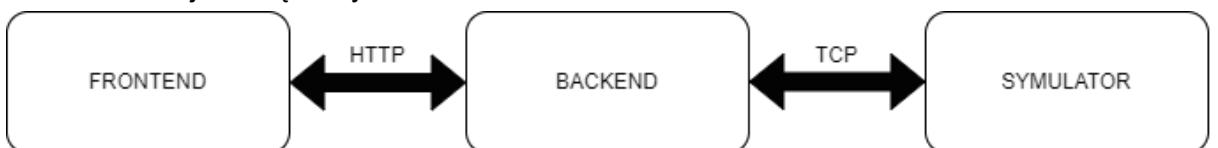
    scenario.make(beamng)

    # Load and start our scenario
    beamng.scenario.load(scenario)
    beamng.scenario.start()

    with isGameRunning_lock:
        isGameRunning = True
        print(isGameRunning)

    vehicle.logging.start("data")
```

- Backend komunikuje się z symulatorem przez gniazdo typu strumieniowego (TCP), zapewniając dwukierunkowe połączenie. Port oraz adres ip, na którym nasłuchuje symulator można zmienić w pliku backend/tcp_config.json.
- Frontend komunikuje się z backendem żądań http (POST), mechanizm CORS jest włączony.



oraz dodanie nowego warunku:

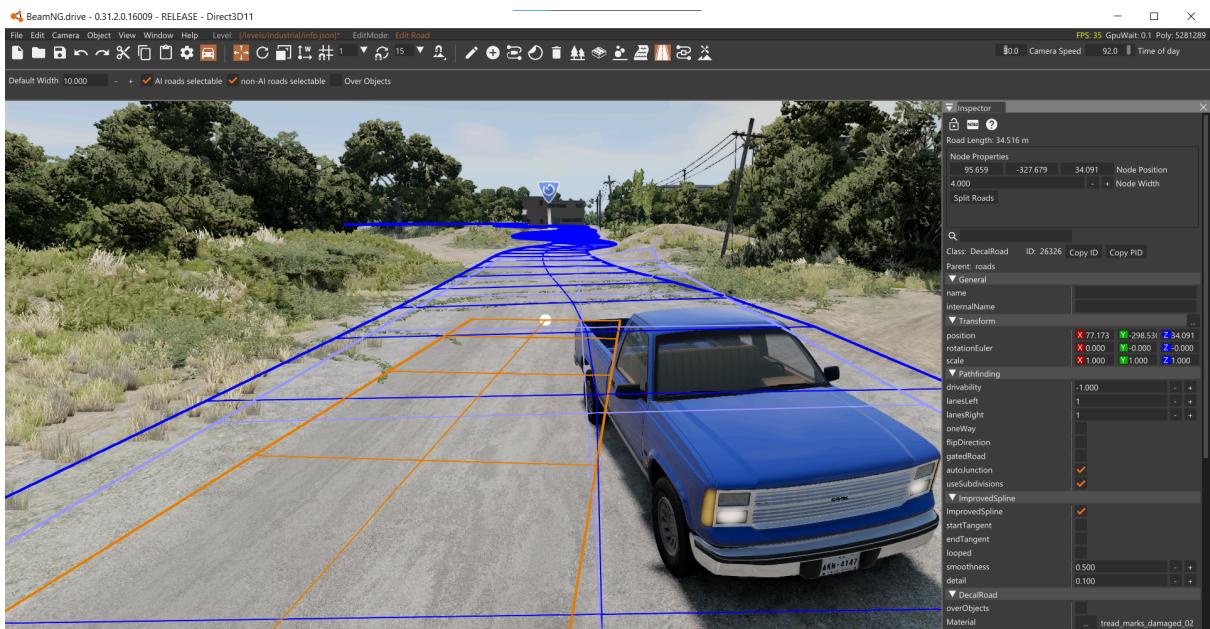
```
def main_loop(exit_event):
    global isGameRunning
    global vehicle
    while not exit_event.is_set():
        print("Main loop...")
        data = data_queue.get()

        command = data["command"]
        payload = data["payload"]

        if command == "exit":      You, last month + logging
            conn.send("zwrot".encode())
            exit_event.set()
            break
        elif command == "start-scenario":
            if payload['scenarioId'] == 1:
                scenario_thread = threading.Thread(target=scenario_1_loop)
            elif payload['scenarioId'] == 2:
                scenario_thread = threading.Thread(target=scenario_2_loop)
```

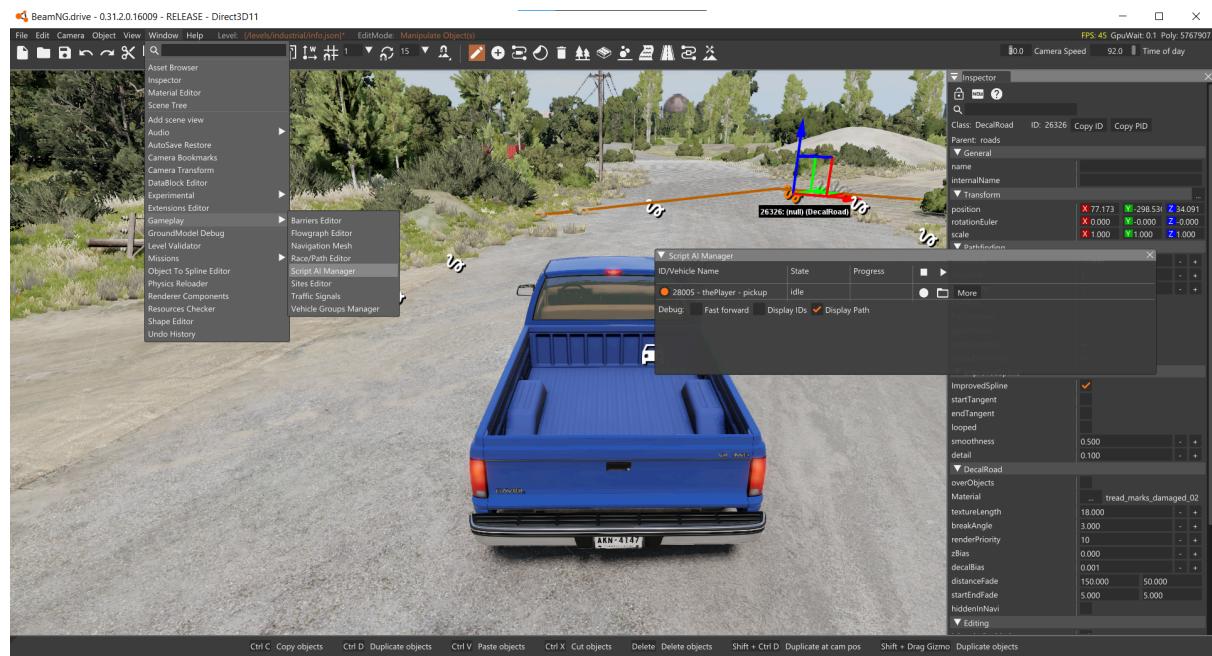
9. Tworzenie tras trafficów

Trasy trafficów można stworzyć na wiele sposobów, między innymi poprzez edytor świata. Jest to jednak dość trudne, podczas tworzenia naszego projektu właściwie to nie działało, a dokumentacja niestety nie zawierała specjalnych informacji o tym. Kolejnym problemem jest dokładne zaprojektowanie wymaganej trasy ręcznie poprzez edytor świata.

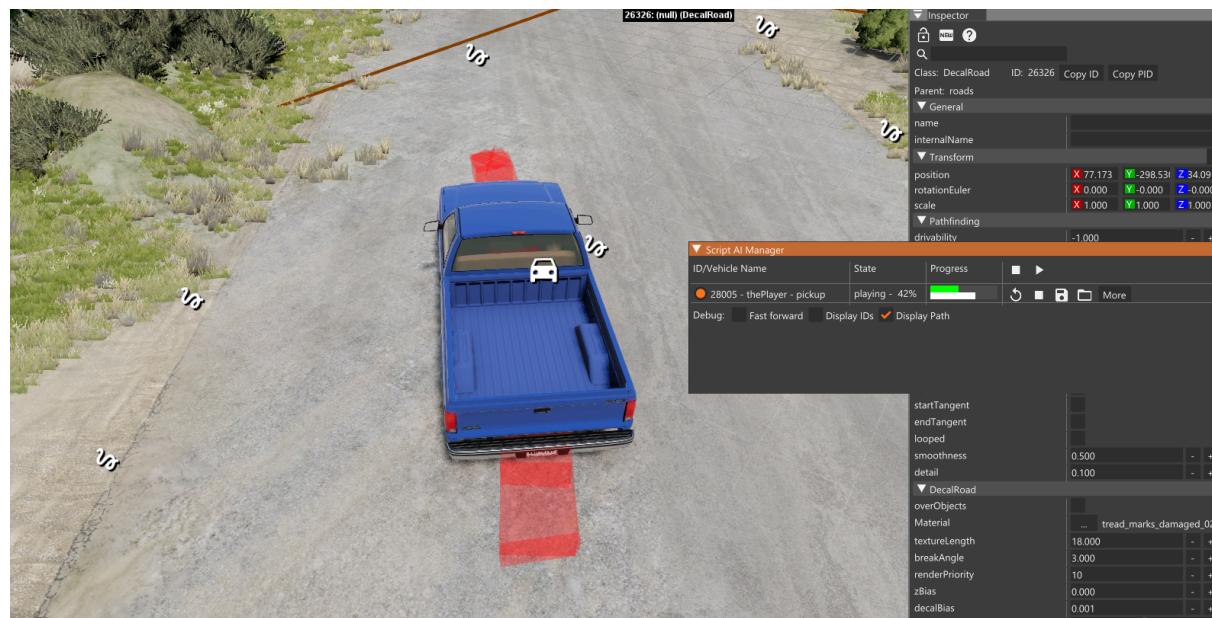


Znacznie prostszym rozwiązaniem jest nagrywanie trasy tj. jazda samochodem i nagrywanie położenia pojazdu w danej sekundzie.

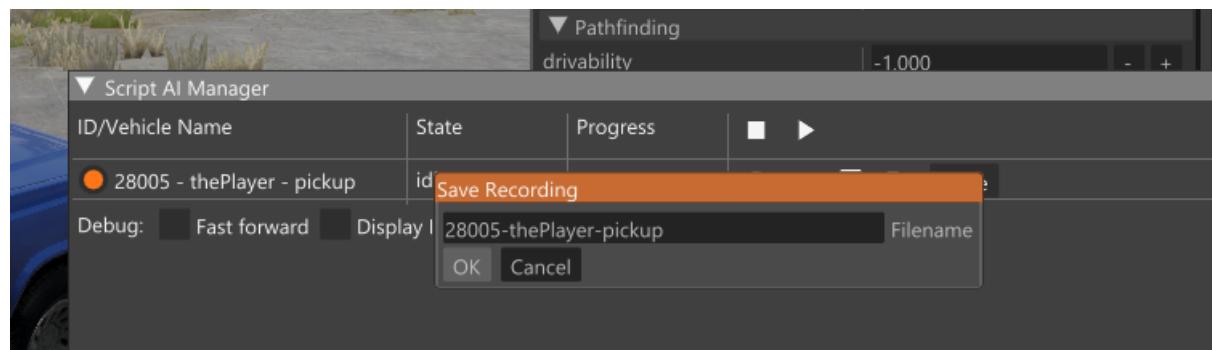
1. Window ->Gameplay -> Script Ai Manager



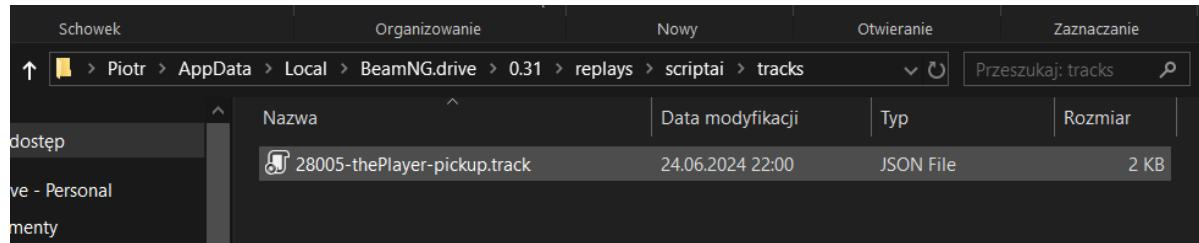
2. Record



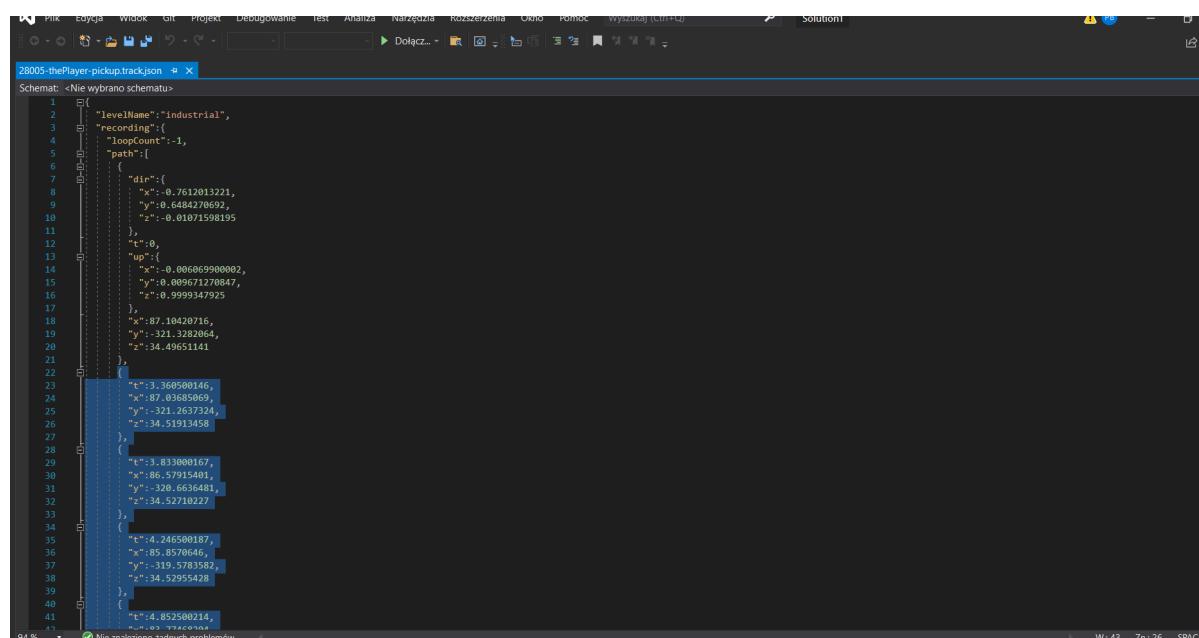
Save record.



Nagranie znajduje się jako plik .json w appdacie.



Następnie w tym pliku należy skopiować do API współrzędne opisane w czasie t.



A następnie wczytać do skryptu.

```
skrypt = [ { "t": 2.18150012, "x": 161.8789999, "y": 1629.2753332, "z": -17.83815008 }, { "t": 2.684000155, "x": 1461.6548465, "y": 1629.8794334, "z": -17.75922883 }, { "t": 3.738500213, "x": 1459.9725994, "y": 1634.4698079, "z": -17.34934193 } ]
```

Na koniec przypisujemy skrypt do pojazdu .

```
vehicle2.ai.execute_script(skrypt)
```

7. Tryby Trafficów

Oficjalna dokumentacja ma niepełne informacje. Niektóre tryby nie działają, a o niektórych nie ma napisane. Przykładowo

```
set_mode(mode: str)→ None
```

Sets the desired mode of the simulator's built-in AI for this vehicle. Possible values are:

- `disabled` : Turn the AI off (default state)
- `random` : Drive from random points to random points on the map
- `span` : Drive along the entire road network of the map
- `manual` : Drive to a specific waypoint, target set separately
- `chase` : Chase a target vehicle, target set separately
- `flee` : Flee from a vehicle, target set separately
- `stopping` : Make the vehicle come to a halt (AI disables itself once the vehicle stopped.)

tryb default i span działają identycznie. Tryb manual nie działał (może niepoprawnie użyliśmy?)

Tryb chase jest agresywny, powoduje stłuczki (zmiana agresywności nie zadziałała)
Odkryliśmy tryb follow

```
vehicle6.ai.set_target("ego_vehicle","follow")
vehicle6.ai.set_mode('follow')
```

8. Triggery

Można próbować je ustawiać we flowgraph (tryb do ustawiania zdarzeń w scenariuszu) jednak tryb ten jest trudny do opanowania i aby działał poprawnie. Można ustawiać proste triggery lub inne wydarzenia kodem w API w pythonie.

Przykładowy kod, jeśli położenie x samochodu przekroczy 17, wówczas skrypt drugiego samochodu zostaje uruchomiony

```
▶
while(True):
    vehicle.sensors.poll()
    current_pos=vehicle.sensors['state'][['pos']]
    print(current_pos)
    if current_pos[0]<17:
        vehicle1.ai.execute_script(BUMBUMBUMBUM)
        break
[6]
...
[31.993524480314136, -225.00097935172107, -22.552987376888268]
[31.99352182568009, -225.00097969775356, -22.552989302069477]
[31.993518833481403, -225.00098118907317, -22.55299268185604]
```

`vehicle.sensors` pozwala na ciągłe sprawdzanie położenia pojazdu. `current_pos[0]` to x, `current_pos[1]` to y, `current_pos[2]` to z.