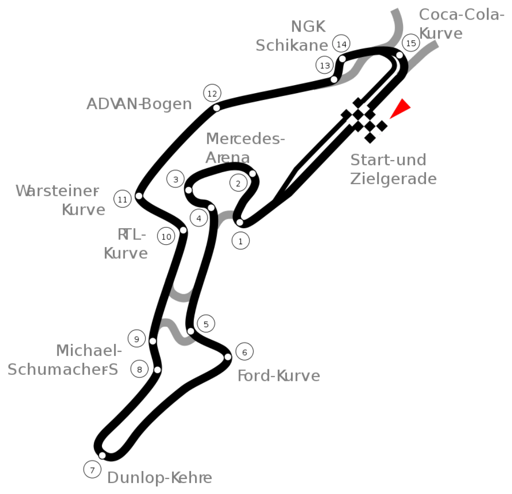
Extracting track sections from coordinates

We want to split a race track into sections, based on the coordinates of the track. A track can be considered of a series of corners and straights. A corner can be connected to another corner, or to a straight. See driver61.com explanation for more details.

The track features to be extracted are:

- the sequence of corners and straights
- the start and end of a sequence in meters on the track
  - the start of a corner is when the track starts to curve
  - the end of a corner is when the track starts to straighten out
- the curvature of a straight. Some straights are not completely straight, but have a slight curve.

Let's look at the Nürburgring track:



The sequence would be:

1. straight "Start und Zielgerade"
2. corner "1"
3. corner "Mercedes Arena"
4. corner "3"
5. corner "4"
6. straight
7. corner "5"
8. corner "Ford Kurve"
9. straight
10. corner "Dunlop Kehre"

11. straight
12. corner "Michael Schumacher S"
13. corner 9
14. straight
15. corner "RTL Kurve"
16. straight
17. corner "Warsteiner Kurve"
18. straight
19. corner "ADVAN Bogen"
20. straight
21. corner "13"
22. corner "NGK Schikane"
23. corner "Coca-Cola Kurve"

Notice that a straight, like number 9, which connects "Ford-Kurve" and "Dunlop Kehre", might not be completely straight, but have a slight curve.

source data

The data is stored in an InfluxDB database. It's collected from sim racing games. The game submits the telemetry data at a sample frequency of 60 Hz. The coordinates are in the world coordinate system, which is different depending on the game. The coordinates are not in meters, but in some game specific unit. The coordinates are the position of the car in the world during a lap. The start and end of a lap data is not 100% connected, since the car does not cross the the start and finish line at the very same coordinates. The field `DistanceOnTrack` is the car position in meters on the track. This field will be used to add a length to each track section.

data preparation

1. collect coordinates for multiple laps on a given track per game
   - since we record data from different cars and different drivers, we want to use many samples to get a statistically significant result.
2. map the coordinates to a universal coordinate system, which is the same for all games
   - every game uses slightly different units and x,y,z coordinates. To make the next steps universal, we need to normalize the data to a universal coordinate system. The coordinates can differ in scale and projection (like being mirrored)
3. remove outliers from laps
   - out of all the collected laps, calculate a single, well connected line, that is the best fit to the actual track. Some drivers have a different racing line on the track, or even spin the car during a lap.
4. split laps into sections, based on the data of all recorded laps

- the extracted features are:
    - the sequence of corners and straights
    - every sequences has a start and end in meters
    - the end of sequence n is the start of sequence n+1

Data exploration

The following is a data exploration of the data. It only shows 3 laps from 2 different games for 2 different tracks. The data for the Nürburgring track shows 2 laps with car spins on the track to visualize the need for removing outliers and normalizing the data for a given track. The data for Spa shows, that the coordinate system can be mirrored for a given game.

The actual data will be provided as SessionId and LapId combinations, for a set of tracks and games. Per track the number of laps varies from 1 to 15. Total it will be approx 4 games with approx 10 tracks per game.

In [1]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import influxdb_client
from IPython.display import Image

from influxdb_client.client.write_api import SYNCHRONOUS

import warnings
from influxdb_client.client.warnings import MissingPivotFunction

warnings.simplefilter("ignore", MissingPivotFunction)

# configure influxdb client
ORG = "b4mad"
TOKEN = os.environ.get(
    "INFLUXDB_TOKEN",
    "citqAMr66LLb25hvaaZm2LezOc88k2ocOFJcJDR6QB-RmLJa_-sAr9kYB4vSFYaz8bt26lm
)
URL = "https://telemetry.b4mad.racing/"

client = influxdb_client.InfluxDBClient(url=URL, token=TOKEN, org=ORG)
query_api = client.query_api()
pd.set_option("display.max_columns", None)
```

In [2]:
```
# a function to query the database for a given session and lap
def query_session_lap(session, lap):
    query = f"""
    from(bucket: "racing")
    |> range(start: -10y, stop: now())
    |> filter(fn: (r) => r["_measurement"] == "laps_cc")
    |> filter(fn: (r) => r["SessionId"] == "{session}")
    |> pivot(rowKey: ["_time"], columnKey: ["_field"], valueColumn: "_value"
    |> filter(fn: (r) => r["CurrentLap"] == "{lap}")
    |> sort(columns: ["_time"], desc: false)
    """
    return query_api.query_data_frame(org=ORG, query=query)
```

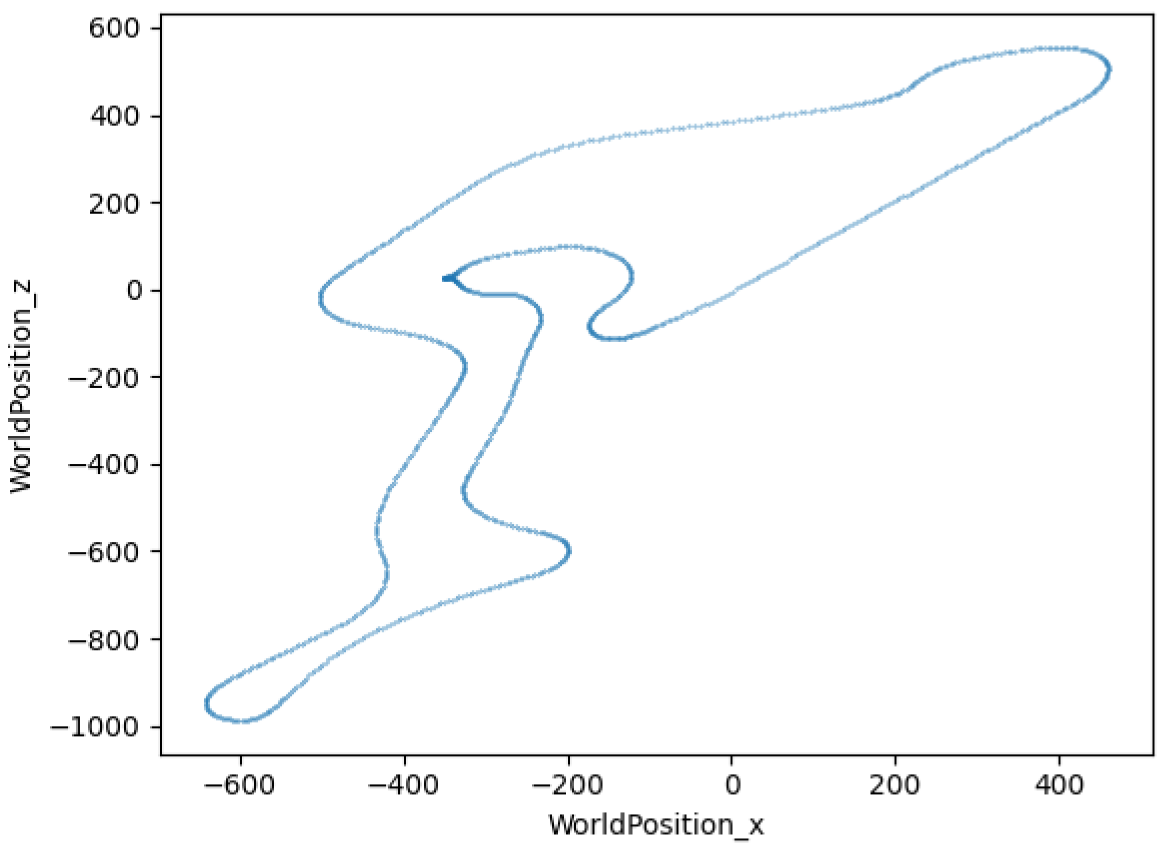
In [3]:

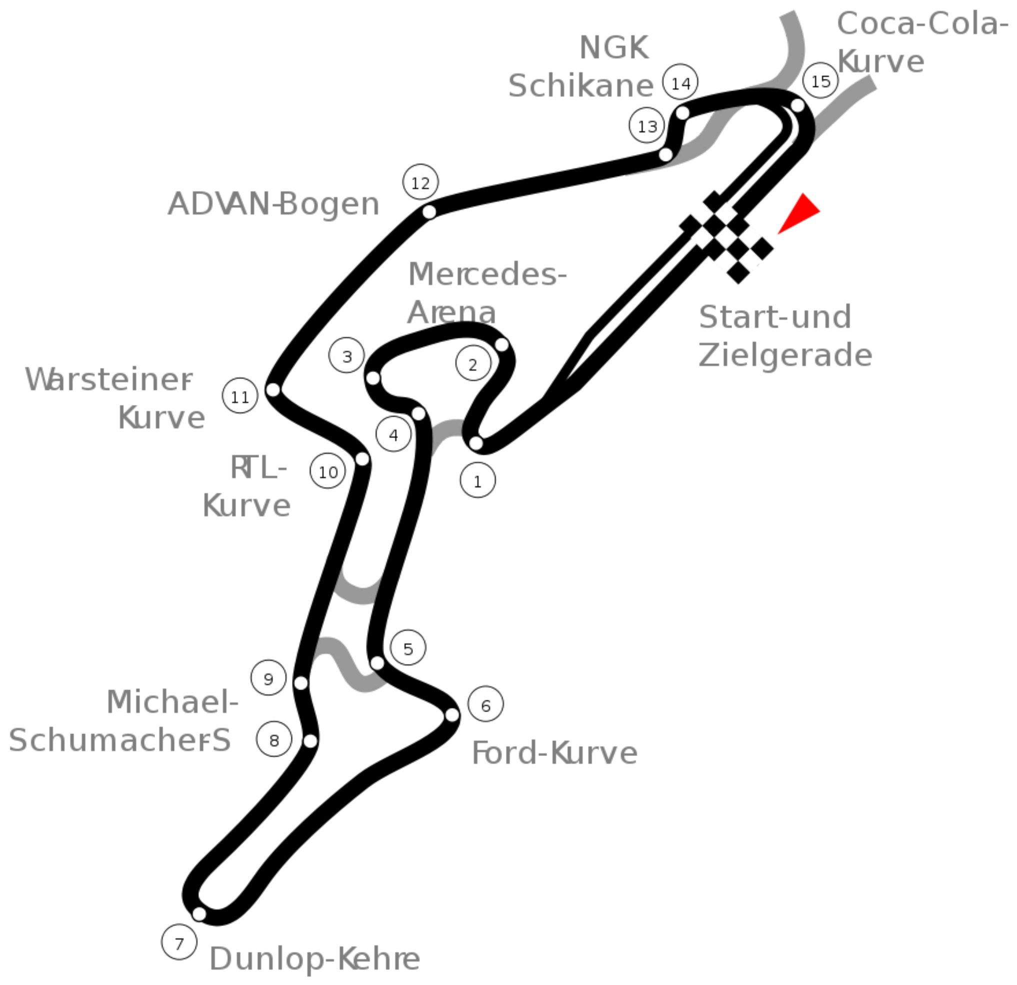
```
tracks = {
    "Nurburgring_2020:Nurb_GP_2020_Veedol-1": {
        "game": "Automobilista 2",
        "session": 1670584820,
        "lap": 2,
        "image": "https://static.wikia.nocookie.net/f1wikia/images/b/b0/Nurb
    },
    "Nurburgring_2020:Nurb_GP_2020_Veedol-2": {
        "game": "Automobilista 2",
        "session": 1670584820,
        "lap": 3,
        "image": "https://static.wikia.nocookie.net/f1wikia/images/b/b0/Nurb
    },
    "Spa:track config": {
        "game": "Assetto Corsa Competizione",
        "session": 1670586823,
        "lap": 8,
        "image": "https://static.wikia.nocookie.net/f1wikia/images/3/3f/Trac
    },
}

for track, data in tracks.items():
    tracks[track]["df"] = query_session_lap(data["session"], data["lap"])
```

In [4]:

```
# iterate over all tracks
for track, data in tracks.items():
    df = data["df"]
    display(track)
    # scatter plot with a point size of 0.1
    if data["game"] == "Automobilista 2":
        df.plot.scatter(x="WorldPosition_x", y="WorldPosition_z", s=0.1)
    if data["game"] == "Assetto Corsa Competizione":
        df.plot.scatter(x="WorldPosition_x", y="WorldPosition_z", s=0.1)
    plt.show()
    display(Image(url=data["image"], width=800, unconfined=True))
    # display the first and last row of the dataframe
    display(pd.concat([df.head(1), df.tail(1)]))
```

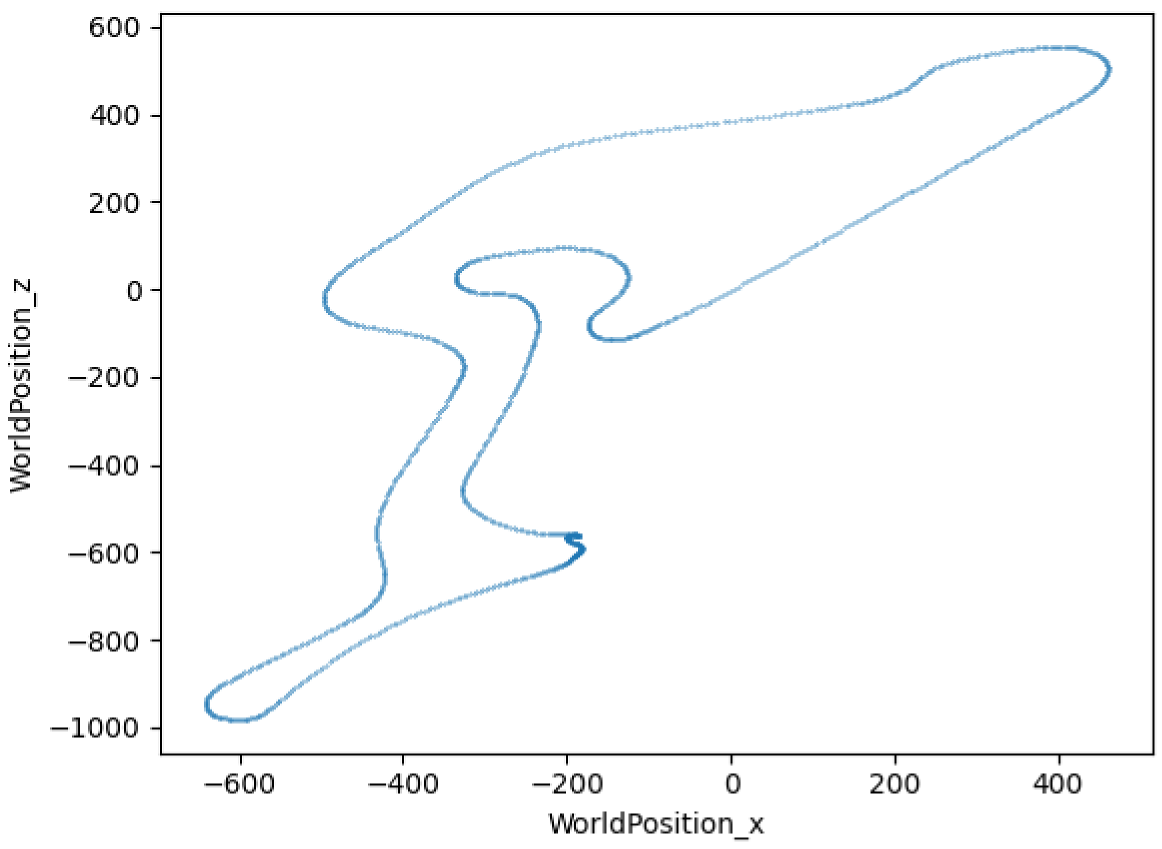
```
'Nurburgring_2020:Nurb_GP_2020_Veedol-1'
```


600
400
200
0
−200
−400
−600
−800
−1000
WorldPosition_z
−600
−400
−200
0
200
400
WorldPosition_x

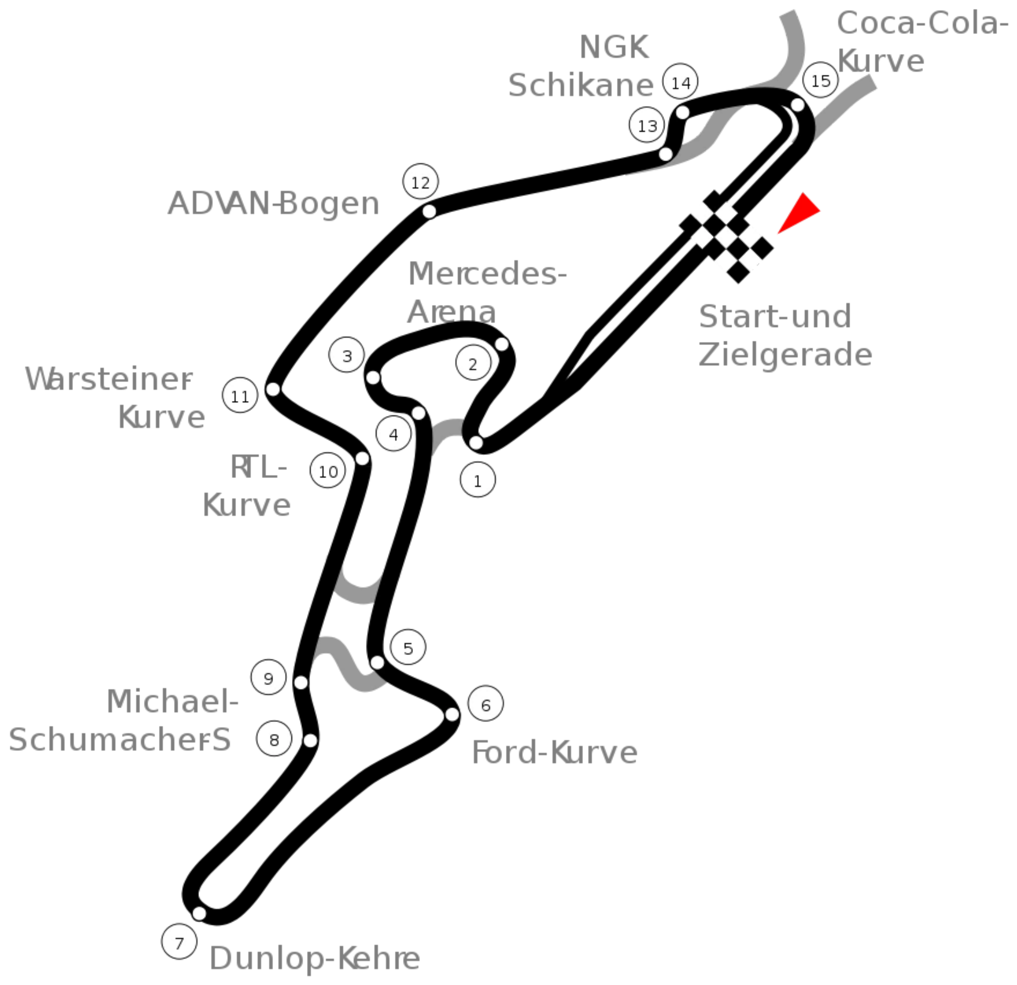


	result	table	_start	_stop	_time
0	_result	0	2012-12-10 03:29:48.720656+00:00	2022-12-10 15:29:48.720656+00:00	2022-12-09 12:25:28.196479+00:00
1217	_result	0	2012-12-10 03:29:48.720656+00:00	2022-12-10 15:29:48.720656+00:00	2022-12-09 12:27:31.084774+00:00

```
'Nurburgring_2020:Nurb_GP_2020_Veedol-2'
```

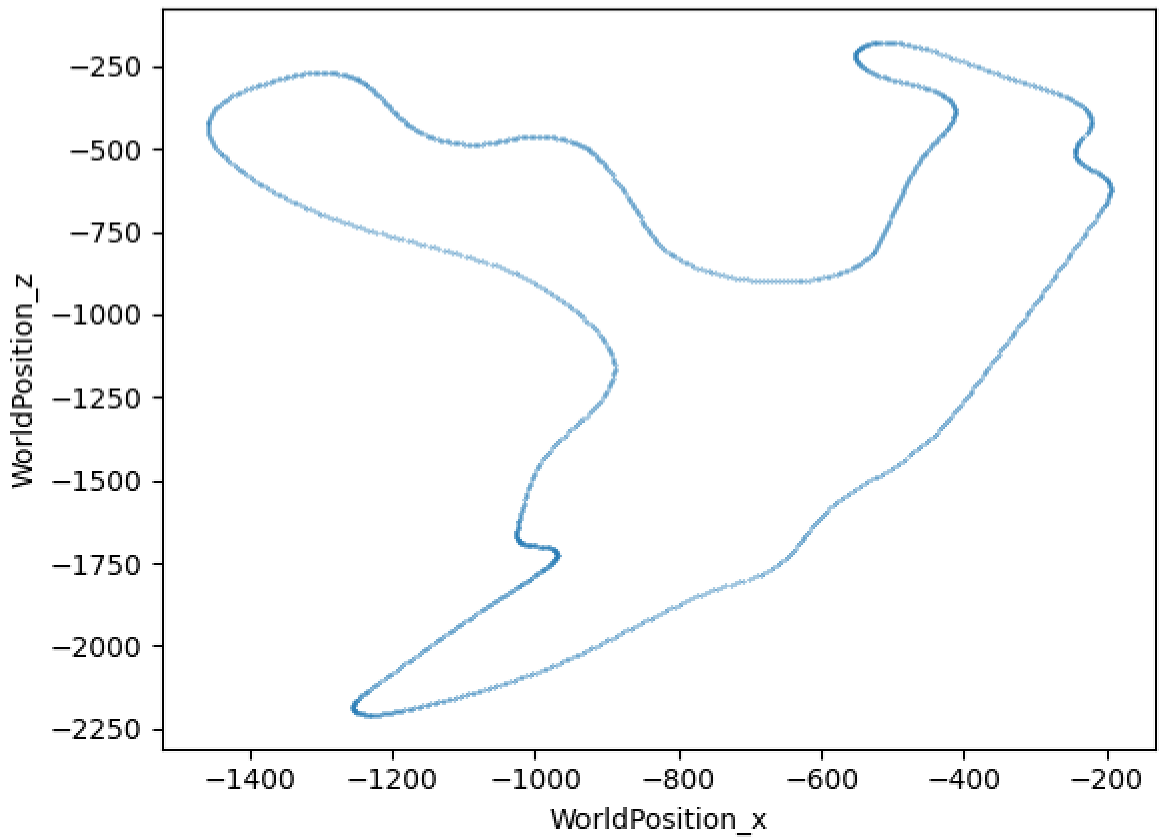


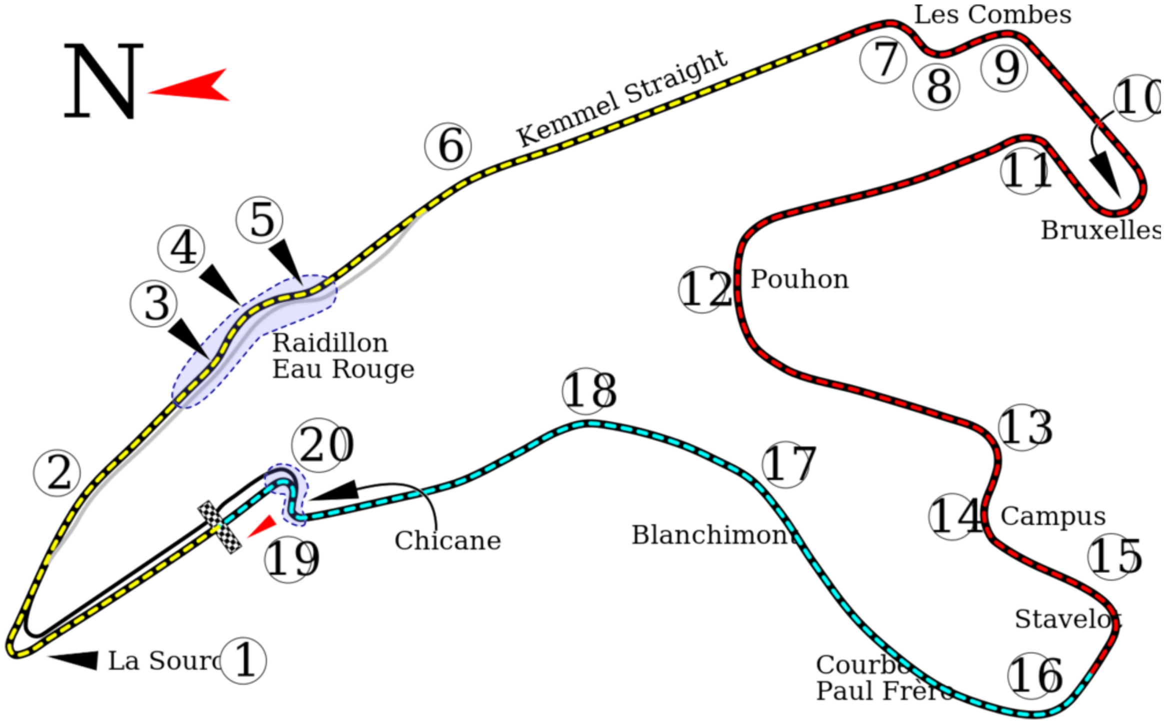
WorldPosition_z
WorldPosition_x
600
400
200
0
−200
−400
−600
−800
−1000
−600
−400
−200
0
200
400



	result	table	_start	_stop	_time
0	_result	0	2012-12-10 03:29:49.920884+00:00	2022-12-10 15:29:49.920884+00:00	2022-12-09 12:27:31.193622+00:00
1358	_result	0	2012-12-10 03:29:49.920884+00:00	2022-12-10 15:29:49.920884+00:00	2022-12-09 12:29:48.316888+00:00

```
'Spa:track config'
```





	result	table	_start	_stop	_time
0	_result	0	2012-12-10 03:29:50.924227+00:00	2022-12-10 15:29:50.924227+00:00	2022-12-09 13:10:01.655171+00:00
1274	_result	0	2012-12-10 03:29:50.924227+00:00	2022-12-10 15:29:50.924227+00:00	2022-12-09 13:12:21.624334+00:00