Rapport Semaine 8
27 au 31 mai 2024

Lundi 27 mai

Pour avoir un référentiel cohérent dans la grille, j'ai décidé de récupérer les dimensions de la carte. J'ai réalisé un script python qui récupère cela puis dans mon script principal j'envoie les données de position.
Ce que je n'avais pas relevé plus tôt c'est que le problème de position vient des nombres négatifs, et je n'avais pas encore compris comment la grille se construisait (en fonction des positions du nœuds max et min). J'ai compris qu'il faudrait que j'adapte le référentiel.

Mardi 28 mai

J'ai changer mon approche pour convertir toutes les données en positions positives : l'idée étant de récupérer le point minimal x,y,z sur toutes les coordonnées enregistrées dans le premier référentiel, puis de soustraire chaque point par son représentant minimum et on obtient une dimension spatiale cohérente. J'ai donc appliquer ça à mon script C++.

Ensuite j'ai intégrer un second nœud mobile pour tester avec plusieurs véhicules. J'ai également créé un script python qui récupère les données du nombre de voiture simulées, ce script sera utile quand je créerais des nœuds en fonction du nombre d'objets existants.

```cpp
// Find the minimum coordinates
double minX = *std::min_element(xCoordinates.begin(), xCoordinates.end());
double minY = *std::min_element(yCoordinates.begin(), yCoordinates.end());

for (size_t i = 0; i < xCoordinates.size(); i++)
{
  // Adjust the coordinates to make them all positive
  double adjustedX = xCoordinates[i] - minX;
  double adjustedY = yCoordinates[i] - minY;

  Waypoint waypoint (Seconds (i), Vector (adjustedX, adjustedY, 0));
  waypointMobility->AddWaypoint (waypoint);
}
```

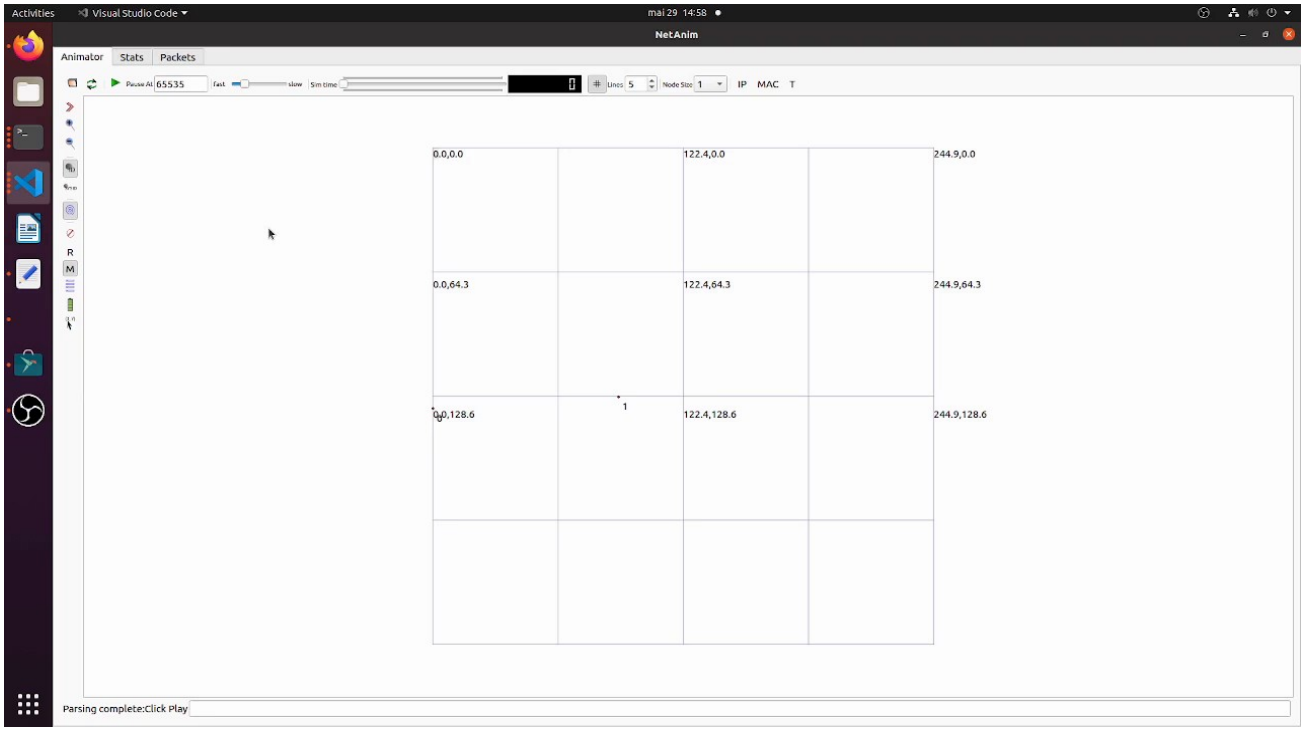
Mercredi 29 mai

J’ai changer la structure de mon script C++ avec du « split » au lieu de « regex » pour avoir une structure plus cohérente. J’ai également modifier le script pour qu’il crée des nœuds en fonction du nombre de véhicules avec une gestion de liste de liste. Mon script python qui publie les positions indique les ID du véhicule et sa position pour que le bon véhicule soit bougé.

J’ai été confronté à un problème qui m’empêche de donner l’ID que je veux à un objet (exemple « vehicle_1 », « vehicle_2 », etc.) mais je suis obligé de mettre pour tous « ego_vehicle ». Donc pour différentier les véhicules j’ai utiliser une méthode de séquençage ou je fais le modulos du nombre de véhicule par la séquence, alors j’obtiens toujours l’index voulu.

```
vehicle_index = data.header.seq % self.num_vehicles
print(f"Vehicle index: {vehicle_index}")
self.publisher_odo(data, vehicle_index)
```

Enfin j’ai fais plusieurs simulations avec des positions différentes, dont une de 15min pour avoir un aperçu de la cohérence (sachant que les positions sont ajustées pour tenir compte du référentiel). Voici une de mes simulations :



Jeudi 29 mai

J’ai affiné mes scripts restants, réorganisé mes fichiers et fait des simulations. J’ai commencé la création du poster.

Vendredi 30 mai

J’ai continué le début du poster.