

Class Agent Turn:

Rappel:

Pour l'instant grâce a Agent Center le kart cherche en permanence a avancer vers l'avant en maintenant une acceleration de base, tout en restant au centre de la piste.

But:

Agent Turn sert à ameliorer la conduite par rapport a AgentCenter, il va anticiper les virages et adapter la vitesse du kart en fonction de la piste. Grâce a ça on pourra maintenir une trajectoire stable et éviter les sorties de pistes.

Strategie:

Observation de la piste devant le kart:

L'agent va devoir observer un certain nombres de segments devant lui en utilisant les variable **paths_start**, **paths_end** et **paths_distance**.

Grâce a cette vision anticipé l'agent va pouvoir comprendre la forme de la piste. En comparant la direction et la longueur des segments qui arrivent le kart evalue si la piste est droite ou si elle tourne. Toujours en utilisant **paths_start**, **paths_end** et **paths_distance**.

L'agent va classer la portion de piste qui arrivent de maniere suivante:

- Ligne droite : Faible variation de direction
- Virage léger : Pas trop grande variation de direction
- Virage serré : Forte variation de direction

Réactions:

L'agent adapte sa vitesse a l'angle du virage detecté, il accélère si on detecte une ligne droite et ralentit avant les virages, en utilisant l'**angle calculé** et la **variable velocity**.

En meme temps l'agent doit comparer le virage a venir avec la capacité de braquage disponible pour éviter de tourner plus que ce qui est possible a sa vitesse acutelle, e, utilisant **velocity** et **max_steer_angle**.

Les variables utilisés seront definies via OmegaConf et pourrons changer apres les test.

Pseudo Code:

AgentTurn herite d'AgentCenter.

On utilise path_lookahead pour regarder 3 segments devant l'agent

#Analyse des segments:

Pour chaque segment, on:

- Calcule de la direction du segment ($\text{paths_end}[i] - \text{paths_start}[i]$)
- Calcule de l'ecart de direction entre front et direction du segment
- Calcule de la longueur du segment ($\text{paths_distance}[i][1] - \text{paths_distance}[i][0]$)
- Determiner le type de segment (droite, virage léger, virage serré)

#Avec i l'indice du segment que je regarde parmi les 3 prochains:

i = 0 -> segment actuelle, i > 0 segments qui arrivent

#Identifier la situation la plus contraignante:

Si au moins un segment est un virage serré -> on reagit comme un virage serré

Si au moins 1 segment est un virage léger -> on reagit comme un virage léger

Si aucun des deux cas precedent -> on reagit comme une ligne droite

#Reactions:

#Ligne droite

if ligne droite:

velocity max

#Virage léger

if virage léger:

#Est ce que je peux le prendre a ma vitesse actuelle ?

if max steer angle petit (si la capacité de braquage actuelle est limitée):

#Ajustement de la vitesse

velocity -

elif max steer angle moyen:

#Ajustement de la vitesse

maintenir

else:

#Ajustement de la vitesse

velocity++ ++ #forte acceleration

#Virage serré

if virage serré

#Est ce que je peux le prendre a ma vitesse actuelle ?

if max steer angle petit (si la capacité de braquage actuelle est limitée):

#Ajustement de la vitesse

```
velocity- - - #fort ralentissement
elif max steer angle moyen:
    #Ajustement de la vitesse
    maintenir
else:
    #Ajustement de la vitesse
    velocity++ # acceleration plus legere
```