

Class Agent Speed:

Version 1.0:

Rappel:

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

But:

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

Stratégie:

Observation de la piste devant le kart:

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

Grâce à cette vision anticipée l'agent va pouvoir comprendre la forme de la piste. En comparant la direction et la longueur des segments qui arrivent, le kart évalue 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 manière suivante:

- Ligne droite : Faible variation de direction
- Virage serré : Forte variation de direction

Réactions:

L'agent adapte sa vitesse à l'angle du virage détecté, il accélère si on détecte une ligne droite et ralentit pendant les virages, en utilisant l'**angle calculé et la variable acceleration**.

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

Les variables utilisées seront définies via OmegaConf et pourrons changer après les tests.

Pseudo Code:

AgentTurn hérite 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'écart de direction entre front et direction du segment
- Calcule de la longueur du segment ($\text{paths_distance}[i][1] - \text{paths_distance}[i][0]$)
- Déterminer 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 réagit comme un virage serré

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

Si aucun des deux cas précédent -> on réagit comme une ligne droite

#Réactions:

#Ligne droite

if ligne droite:

velocity max

#Virage léger

if virage léger:

#Est ce que je peux le prendre à 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 à 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

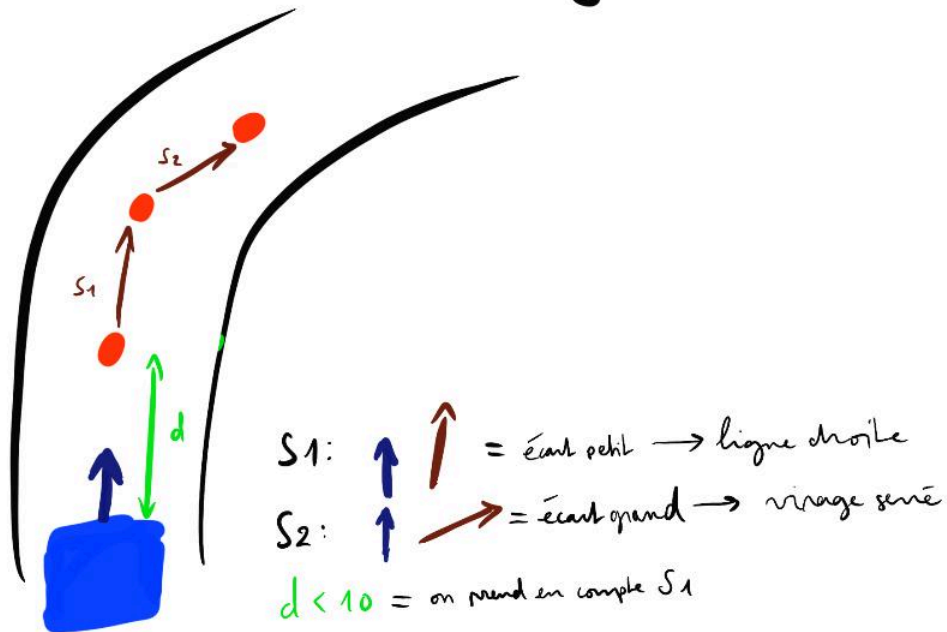
else:

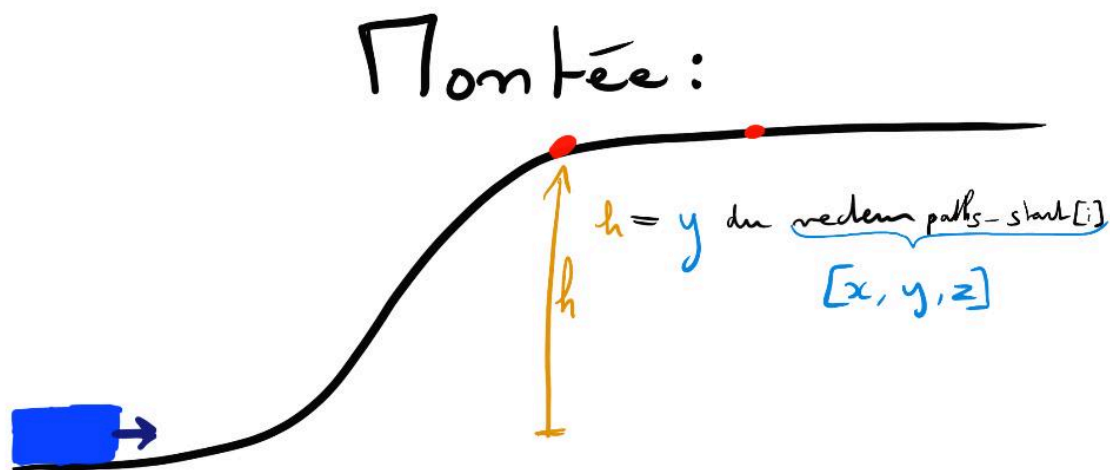
#Ajustement de la vitesse

velocity++ # acceleration plus legere

Version 2.0:

Virage :





Stratégie:

Observation de la piste devant le kart:

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

Grâce à cette vision anticipée l'agent va pouvoir comprendre la forme de la piste. En comparant la direction et la longueur des segments qui arrivent, le kart évalue si la piste est droite ou si elle tourne. La distance du segment permet de mieux anticiper et de catégoriser le segment quand il est assez proche. Toujours en utilisant **paths_start**, **paths_end** et **paths_distance** on calcule l'écart et la distance de chaque segment.

Un virage est considéré comme serré si :

- l'écart directionnel dépasse un seuil "écart grand"
- le segment est à une courte distance du kart

On initialise une variable booléenne **virage_serre** à False pour catégoriser les segments:

- Ligne droite : Faible variation de direction -> **virage_serre** reste False
- Virage serré : Forte variation de direction -> **virage_serre** = True

On retourne alors la variable **virage_serre**.

Réactions:

L'agent adapte sa vitesse à l'angle du virage détecté, il accélère si on détecte une ligne droite et ralentit pendant les virages, en utilisant l'**angle calculé** et la **variable acceleration**. Si **virage_serrer** est False on accélère.

Si non on ralentit.

En même temps, l'agent doit comparer le virage à venir avec la capacité de braquage disponible pour éviter de tourner plus que ce qui est possible à sa vitesse actuelle, en utilisant **accélération et max_steer_angle**.

L'agent detecte les montées en observant la composante vertical (y) du vecteur direction du segment actuel. On accelere légèrement lors d'une montée pour compenser la perte de vitesse.

Les variables utilisées seront définies via OmegaConf et pourrons changer après les tests.