AgentObstacles

Rouge = informations à vérifier/tester

Stratégies :

A chaque instant, vérifier le type des prochains objets devant nous.
Si ce sont des obstacles, appeler une méthode qui se charge d'éviter ces obstacles.
Si ce sont des bonus ou des nitros, appeler une méthode qui se charge de se diriger vers ces objets.

Éviter l'obstacle :
On note l'index de l'obstacle qu'on observe.
Si l'item est trop loin et peut être ignoré, ou si on l'a déjà passé (donc il a déjà été géré), on supprime la notation de l'index et on renvoie le dictionnaire d'actions sans rien changer.
Si on a un shield, on renvoie le dictionnaire d'actions sans rien changer.
Sinon, on récupère le steer actuel et on le compare aux coordonnées X du prochain obstacle. Si on se dirige actuellement vers l'obstacle, on réduit ou on augmente le steer d'une valeur fixe (peut être un hyperparamètre). On renvoie ensuite le dictionnaire d'actions modifié.

Se diriger vers les bonus :
On note l'index du bonus qu'on observe.
Si l'item est hors de portée, on supprime la notation de l'index et on renvoie le dictionnaire d'actions sans rien changer.
Si l'item est trop proche de nous en coordonnée z, on abandonne l'idée de se diriger vers lui, on supprime la notation de l'index et on renvoie le dictionnaire d'actions sans rien changer.
Sinon, on récupère le steer actuel et on le compare aux coordonnées X du prochain bonus. Si on se dirige actuellement vers le bonus, on

ne change rien et on renvoie le dictionnaire d'actions. Sinon, la coordonnée X du bonus devient notre steer. On renvoie le dictionnaire d'actions modifié.

Pseudo-codes :

3 méthodes :

- Observation du prochain item(obs, action)
- Éviter obstacle(obs, action, index)
- Dirige vers bonus(obs, action, index)

Ainsi que la méthode choose_action de base corrigée.

Observation du prochain item(obs, action) :

```
tableau des index bonus = [tous les index des bonus dans obs["items_type"]
#les bonus correspondent à 0, 2, 3
tableau des index obstacles = [tous les index des obstacles dans obs["items_type"]
#les obstacles correspondent à 1, 4
pour chaque i dans le tableau des bonus :
    appel à Dirige vers bonus(obs, action, i)
pour chaque i dans le tableau des obstacles :
    appel à Éviter obstacle(obs, action, i)
Renvoie le dictionnaire d'actions
```

NB : On appelle d'abord la méthode pour les bonus avant la méthode pour les obstacles pour que les obstacles soient la dernière modification faite au dictionnaire d'actions, donc la plus importante

Éviter obstacle(obs, action, index) :

```
Si on a aucun obstacle ciblé :
    obstacle ciblé = index
Récupération du vecteur de cet obstacle
```

Si l'obstacle est très proche de nous ou si au contraire il est trop loin :

obstacle ciblé = None

Si on a un obstacle ciblé qui correspond bien à index :

Si on a un shield bubblegum et qu'il reste activé assez de temps :

On renvoie le dictionnaire d'actions sans modification

Si l'obstacle est à notre gauche, on augmente le steer

Si l'obstacle est à notre droite, on réduit le steer

Renvoie le dictionnaire d'actions

NB : Garder l'obstacle ciblé en attribut nous permet d'éviter les obstacles un par un et ne pas avoir de conflit dans le dictionnaire si on a par exemple deux bananes côte à côte (l'agent ne tournera pas à gauche et à droite en même temps).

Dirige vers bonus(obs, action, index) :

Une optimisation de cette méthode est peut-être possible

Si on n'a aucun item ciblé :

item ciblé = index

Récupération du vecteur de cet item

Si l'item est très proche de nous ou si au contraire il est trop loin :

item ciblé = None

Si on a un item ciblé qui correspond bien à index :

Notre steer devient la coordonnée X du vecteur de l'item

Renvoie le dictionnaire d'actions

NB : De la même manière que pour les obstacles, garder l'index de l'item ciblé permet de ne pas viser deux items en même temps pour être sûr d'en avoir au moins 1.

AgentItems

Stratégies

A chaque instant, vérifier si on a un item à disposition.
Si on est très proche d'une boîte à items, en général, activer l'item possédé.
Actions pour chaque item :

Bubblegum :

Si on est très proche d'un obstacle, activer le shield et suivre le chemin de base sans dévier l'obstacle.
Si une balle de basket ou une bombe gâteau nous fonce dessus (si on a l'info), activer le shield.

Cake :

Si on n'est pas premier et si on peut l'activer à n'importe quelle distance du joueur devant nous, alors activer directement.
Sinon, attendre d'atteindre la distance minimale et activer.

Bowling ball :

S'il y a un ennemi en face de nous et que la variation de x par rapport à notre kart n'est pas très élevé, diriger légèrement le kart vers lui et activer.
Vérifier si on peut lancer la balle derrière nous (dans ce cas là on devra s'orienter à l'envers).

Zipper :

Si on est sur une ligne droite et qu'il n'y a aucun obstacle, activer.
Essayer de foncer sur les karts ennemis.
Peut-être plus tard optimiser ça avec des raccourcis ?

Plunger :

Si on a un virage devant nous à la distance d'activation, activer.

Si on a un joueur devant nous à la distance d'activation, diriger légèrement le kart vers le joueur en face et activer.

Switch :

Si on a un avantage (boîte d'item ou nitro) devant nous, NE PAS ACTIVER.
Sinon, activer directement.
Peut-être attendre qu'un joueur ennemi soit en face d'un avantage pour qu'il ne puisse pas l'esquiver ?

Swatter :

Si un ennemi est à moins de 10m de nous, activer.

Rubberball :

Si on n'est pas premier, activer directement.

Parachute :

Si on n'est pas premier, activer directement.
Peut-être attendre qu'on soit troisième ou quatrième pour l'activer...

Pseudo-codes :

Prend en paramètres : self, obs, action

Si obs["powerup"] égal 0 ou 10 (car l'anvil n'est ni ramassable, ni utilisable), action["fire"] = 0, fin de la fonction.
Sinon :

```
si obs["powerup"] = 1 :
#BUBBLEGUM
    si obs["items_type"][0] == 1 ou 4
    #1 : BANANA, 4 : BUBBLEGUM
        vec = obs["items_position"][0]
        si z du vecteur < 5 mètres
            action["fire"] = 1
```

```
                    return action
          <Si on a l'info qu'un item nous fonce dessus>
          si obs["items_type"][0] == 0 :
          #0 : BONUS BOX
               vec = obs["items_position"][0]
               si z du vecteur < 5 mètres
                    action["fire"] = 1
                    return action

     si obs["powerup"] = 2 :
     #CAKE
          premier_kart = obs["karts_position"][0]
          si premier_kart[2] (sa coordonnée z) > 0 :
          #si le premier kart est devant nous, ajouter condition
de la distance du kart devant, voir si le CAKE peut se lancer de
n'importe où
               action["fire"] = 1
               return action

     si obs["powerup"] = 3 :
     #BOWLING
          pour kart dans obs["karts_position"]
               si kart[z] < 15 et kart[z] > 0 et abs(kart[x]) < 20
                    Diriger légèrement le kart vers l'ennemi
                    action["fire"] = 1
                    return action
          Regarder si on peut lancer derrière, à remplir ici

     si obs["powerup"] = 4 :
     #ZIPPER
          react = #appel fonction analyse d'AgentTurn
          si react = "ligne droite" :
               si obs["items_type"][0] == 1 ou 4
               #1 : BANANA, 4 : BUBBLEGUM
                    vec = obs["items_position"][0]
                    si z du vecteur < 15 mètres
                         action["fire"] = 0
```

```
                        return action
            action[“fire”] = 1
            return action

si obs[“powerup”] = 5 :
#PLUNGER
      pour kart dans obs[“karts_position”]
            si kart[z] < RANGE_PLUNGER et abs(kart[x]) < 20
                  Diriger légèrement le kart vers l’ennemi
                  action[“fire”] = 1
                  return action
      #Appel de la fonction d’AgentTurn pour voir s’il y a un
      #virage
      #S’il y a un virage et que distance[z] < RANGE_PLUNGER
            action[“fire”] = 1
            return action
      si obs[“items_type”][0] == 0 :
      #0 : BONUS BOX
            vec = obs[“items_position”][0]
            si z du vecteur < 5 mètres
                  action[“fire”] = 1
                  return action

si obs[“powerup”] = 6 :
#SWITCH
      si obs[“items_type”][0] = 0 ou 2 ou 3
      #0 : BONUS BOX, 2 : NITRO BIG, 3 : NITRO SMALL
            vec = obs[“items_position”][0]
            si z du vecteur < 10 mètres :
                  action[“fire”] = 0
                  return action
      action[“fire”] = 1
      return action

si obs[“powerup”] = 7 :
#SWATTER
      pour kart dans obs[“karts_position”]
```

```
            si kart[z] < 20 et kart[z] > 0
                  action[“fire”] = 1
                  return action
      si obs[“items_type”][0] == 0 :
      #0 : BONUS BOX
            vec = obs[“items_position”][0]
            si z du vecteur < 5 mètres
                  action[“fire”] = 1
                  return action

si obs[“powerup”] = 8 :
#RUBBERBALL
      premier_kart = obs[“karts_position”][0]
      si premier_kart[2] (sa coordonnée z) > 0 :
      #si le premier kart est devant nous
            action[“fire”] = 1
            return action

si obs[“powerup”] = 9 :
#PARACHUTE
      premier_kart = obs[“karts_position”][0]
      si premier_kart[2] (sa coordonnée z) > 0 :
      #si le premier kart est devant nous
            action[“fire”] = 1
            return action

action[“fire”] = 0
return action
```