Club informatique de l'Esisar

# Documentation concours I.A.

Version 0.1.0.1

22/10/2024

# I. Sommaire

# II. Introduction

Le concours d'Intelligence artificielle est organisé par le club informatique de l'Esisar. Il consiste au contrôle d'une voiture qui peut connaitre aux maximum 10 distances autour d'elle.

Les participants peuvent créer des I.A. qui contrôlent l'accélération, la direction et le sens des voitures. En entrée, les I.A. reçoivent les mesures des capteurs placés à l'initialisation, la vitesse de la voiture, le sens de la voiture et la direction.

Une itération du serveur correspond à un avancement de 5ms et dure 5ms. Le port de connexion du serveur est le numéro **25566**.

# III. Caractéristiques du véhicule

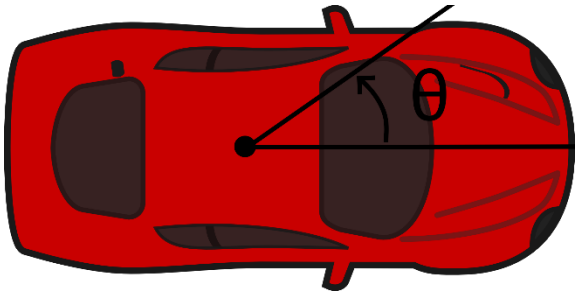
- Le véhicule est une voiture qui possède les caractéristiques suivantes :
  - 5.5 m de longueur
  - 1.8 m de largeur
  - 750 kg
  - 3.6 m d'empattement
  - 3 s pour passer de 0 km/h à 100km/h
  - 20 s pour passer de 100 km/h à 10 km/h si aucune accélération est donnée. (≈frein moteur/force de frottement)
  - 2.7 m/s vitesse max en marche arrière
  - ≈ 94 m/s vitesse max marche avant
  - 2.8 s pour passer de 340 km/h à 100 km/h en capacité de freinage.
  - 8213 $kg.m.s^{-2}$ force maximale d'accélération
  - 12735 $kg.m.s^{-2}$ force maximale de freinage
- Fonction de transfert de la vitesse (m/s) en temps continu :

$$\frac{V(p)}{F(p)} = \frac{1}{m}\frac{1}{p+\frac{k}{m}}, avec\ k = \frac{m \cdot \ln(10)}{20}, m\ la\ masse, F\ la\ force\ d'acceleration$$

- Fonction de transfert de la vitesse (m/s) en temps discret et implémentée dans le serveur :

$$\frac{V(z)}{F(z)} = \alpha\frac{1}{1-p_0 z^{-1}}, où\ \alpha = 6.664748214 \times 10^{-6}, p_0 = 0.99942$$

- Placement des capteurs : la voiture peut avoir au maximum 10 capteurs de distance placés comme on le souhaite avec un angle donné compris entre -180° inclus et 180° exclus.




La valeur retourner par le capteur et la distance entre le bord de la voiture et le mur le plus proche dans la direction et le sens du capteur.

# IV. Communication entre le serveur et l'afficheur

(Toutes les coordonnées suivantes si elles ont une unité, l'unité est le millimètre)

## I) Poignée de main

-Le client envoie : "VISIO'' en ascii c'est-à-dire 86, 73, 83, 73 et 79.

-Le serveur répond ensuite : "VOK'' en ascii i.e. 86, 79 et 75.

## II) Message Client->Serveur

### a) MAP

-Nombre de mur :

Le client envoi **1** sur 1 octet, le serveur répond le nombre de mur sur 2 octets

-Tableau des murs :

Le client envoi **2** sur 1 octet, le serveur répond tous les segments de mur sous la forme suivante dans le même ordre :

4 octets coordonnées x du point A

4 octets coordonnées y du point A

4 octets coordonnées x du point B

4 octets coordonnées y du point B

... Prochain segment

### b) Joueur :

-Nombre de joueur :

Le client envoi **3** sur 1 octet, le serveur répond le nombre de joueur sur 1 octet.

-Position d'un joueur :

Le client envoi **4** sur 1 octet, le serveur répond dans l'ordre des joueurs pour chaque joueur :

4 octets coordonnées x du point au centre du véhicule

4 octets coordonnées y du point au centre du véhicule

2 octets pour l'angle du véhicule. L'angle va de -π à π avec 0 qui vaut -π et 65535 qui vaut π.

(On a : $angle = \frac{data-32768}{10430}$)

-Démarrer la course :

Le client envoi **5** sur 1 octet, le serveur démarre la course.

## III) Fin des messages

-Si le serveur ou le client envoi **255** sur 1 octet la liaison s'arrête

# V. Communication entre le serveur et l'I.A.

(Calcul de $dataDAngleRoue = \frac{angleDegree \cdot 32768}{30} + 32768$ , $dataDAngle = \frac{angleDegree \cdot 32768}{180}$)

## I) Poigné de main

-Le client envoi : "JOU'' en ascii.

-Le serveur répond ensuite : "JOK'' en ascii.

-Le client envoi le nombre de caractère sur 1 octet du nom du joueur.

-Le client envoi ensuite en ascii le nom du joueur sur le nombre d'octet dit précédemment.

-Le client indique ensuite le nombre de capteur sur un octet qui doit être compris entre 1 et 10 inclus.

-Le client envoi sur 2 octets chaque angle ($dataDAngle$) de chaque capteur.

-Le serveur renvoi 0 sur un octet une fois qu'il a tous reçu.

-Le serveur envoi 1 sur un octet pour commencer la course.

## II) Message Serveur->Client

A chaque avancement du système le serveur envoi toutes les informations mise à jour de la manière suivante :

-La valeur de chaque capteur sur 2 octets (0 représente 0 mètre et 65535 représente la distance max qu'un capteur peut capter soit 20 mètres).

-Sur 1 octet si la voiture est en marche arrière ou avant, ‘0’ pour la marche avant et ‘1’ pour la marche arrière.

-La vitesse de la voiture sur 2 octets (vitesse en $cm/s$).

-L’angle des roues de la voiture sur 2 octets ($dataDAngleRoue$).

-Un dernier octet s’il vaut ‘0’ la course continu, si ‘1’ la course est en pause, si ‘2’ la course est finie.

Si le système est en pause le serveur enverra sur un octet soit ‘0’ pour que la course reprenne soit ‘1’ pour dire que la course est finie.

## III) Message Client->Serveur

Le client doit répondre au serveur à chaque fois que le serveur envoi des informations sur le système.

Le client envoi :

-Sur 1 octet si le joueur veut être en marche avant ou arrière.

-Sur 2 octets l’accélération/frein voulu ‘0’ ni frein ni accélération, > 0 accélération, <0 frein.

-Sur 2 octets l’angle des roues ($dataDAngleRoue$).

Remarque : si le client ne répond pas les informations précédentes seront utilisées par le système.

# VI. Description d’un fichier de carte

Toute valeur donnée dans ce fichier est un entier en millimètre.

## I) Extension et emplacement

L’extension du fichier sera ‘m’, on trouvera ce type de fichier dans le dossier data/map.

Le répertoire data/map contiendra alors des fichiers du type “carte0.m’’

## II) En tête

Le fichier commencera avec une en tête qui contiendra le nom de la map à la première ligne. A la seconde ligne nous avons le nombre de voiture maximum autorisée sur le circuit.

Exemple :

```
Monde_0
2
```

## III) Liste des points

Les points sont définis avec le mot clé **points** et sur les lignes suivantes (une ligne par point) par sa coordonnée x et sa coordonnée y. Le nom du point est son numéro d'apparition par rapport aux autres points (commençant par 1).

Exemple :

```
points
  0 0
  100 0
```

## IV) Liste de lignes

Les lignes sont définies avec le mot clé **lignes** et sur les lignes suivantes (une ligne par segment) par le premier point de la ligne et le second point.

Exemple :

```
lignes
  1 2
```

## V)Liste des checkpoints

Les checkpoints sont définis avec le mot clé checkpoints et sur les lignes suivantes (une ligne par checkpoint) le premier point du checkpoint et le second point qui forment une ligne par laquelle les joueurs doivent passer.

Exemple :

```
checkpoints
  1 4
  2 3
```

## VI) Emplacements de départ

Les emplacements de départ sont définis avec le mot clé emplacements et sur les lignes suivantes (une ligne par emplacement) un point qui est le centre de la voiture et un angle en degré.

Exemple :

```
emplacements
  5 0
  6 0
```

## VII) Ligne d’arrivée

La ligne d’arrivée est définie avec le mot clé arrivee suivit de deux points.

Exemple :

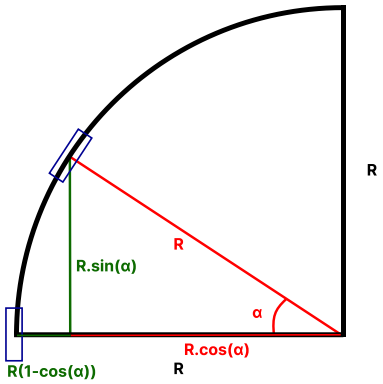
```
arrivee 2 3
```

## VIII) Exemple de résultat final

```
Monde_0
2
points
  0 0
  100 0
  100 100
  0 100
  12 50
  37 50

lignes
  1 2
  2 3
  3 4
  4 1

checkpoints
  1 4
  2 3

emplacements
  5 0
  6 0

arrivee 2 3
```

# VII. Comportement physique

## I) Calcul de la trajectoire




Soit $d$ la distance parcourue par la voiture. Soit $R$ le rayon de braquage ($R := \frac{L}{\sin(\theta)}$, $L$ l'empattement et $\theta$ l'angle des roues). Or $d = R\alpha$ et $R = \frac{L}{\sin(\theta)}$ donc $\alpha = \frac{d \cdot \sin(\theta)}{L}$. Soit $\Omega$, l'angle de la voiture par rapport à l'axe de sa hauteur.

Soient $V_x$ et $V_y$ les coordonnées de la voiture, on a :

- Si $\theta \neq 0$, $x = R(1 - \cos(\alpha)$ et $y = R \cdot \sin(\alpha)$
- Sinon $x = 0$ et $y = d$

On a $V_x = V_x + x \cdot \cos(\Omega) - y \cdot \sin(\Omega)$, $V_y = V_y + y \cdot \cos(\Omega) + x \cdot \sin(\Omega)$

Et $\Omega = \Omega - 2 \arctan\left(\frac{x}{y}\right)$, si $y \neq 0$ sinon $\Omega$ garde sa valeur

## II) Gestion des collisions

A chaque itération le serveur calcule la position de chaque voiture sans les collisions. Ensuite le serveur regarde si l'intersection entre une voiture et au moins un mur est non nulle, si l'intersection est non nulle alors la position est interdite, la voiture récupère sa position précédente et la vitesse de la voiture est mise à 0.

Une voiture fait 5.5m de long et 1.8m de large. Supposons que par une bizarrerie inexplicable la voiture se déplace sur son plus petit segment soit 1.8m quelle vitesse doit-elle atteindre pour passer à travers un mur pour que la collision ne soit pas détectée ?

La durée que simule une itération est de 5ms donc il faut que la vitesse soit supérieure à $\frac{1.8}{5 \cdot 10^{-3}} = 360\ m/s$. La vitesse maximale de la voiture est d'environ (arrondi supérieur) $95\ m/s$ donc il n'est pas possible de passer à travers un mur.

# VIII. Fonctionnement du serveur

## I) Segments de mémoires partagées

Fichier $informationCarte$ :

1. Informations concernent le nombre d'élément sur la carte. $\boldsymbol{infoBaseMap}$
2. Tableau de points. $\boldsymbol{Point}$[]
3. Tableau de lignes. $\boldsymbol{Ligne}$[]
4. Tableau de checkpoints. $\boldsymbol{Checkpoint}$[]
5. Tableau d'emplacements de départs. $\boldsymbol{Emplacement}$[]
6. Ligne d'arrivée. $\boldsymbol{Arrivee}$

Fichier $informationJoueur$ :

1. Nombre maximum d'I.A. dans la partie. $\boldsymbol{int}$
2. Nombre d'I.A. actuellement dans la partie. $\boldsymbol{int}$
3. Tableau des informations de chaque I.A. $\boldsymbol{Voiture}$[]

Fichier $informationCourse$ :

1. Contient l'information si la course a commencé ou pas. $\boldsymbol{int}$

## II) Le main

Le main commence par chargé le fichier carte, initialisé les segments de mémoires partagées et de se diviser en deux processus, le processus accueil et jeu.

## III) Accueil

L'accueil gère toutes les connexions TCP. A chaque nouvelle connexion l'accueil crée un nouveau processus qui détermine s'il est en connexion avec un afficheur ou une I.A.

Le processus qui est en connexion avec une I.A. récupère toutes les informations envoyées par le client et les ranges dans l'emplacement correspond à son I.A (ceci en boucle, jusqu'à que le serveur se ferme). Aussi quand il reçoit le signal SIGUSR1, il envoi les informations correspondant à son I.A. au client.

## IV) Jeu

Dans un premier temps ce processus attend que tout soit bien initialisé et de recevoir le signal start pour commencer à lancer la simulation.

Une fois la simulation lancée il va répéter toute les 5ms environ pour chaque I.A. présente :

- Calcule de la trajectoire
- Collision avec un mur
- Mesure des capteurs
- Envoi du signal SIGUSR1 au processus gérant la connexion avec l'I.A. courante
- Vérification des checkpoints

Si toutes les voitures ont fini alors arrêter la course.

# IX. Exemple de code d'une I.A.

Sur le git vous retrouverez un client d'exemple nommé Joueur.

Le client prend en argument l'adresse I.P. du serveur.

Le client commence par initialiser l'emplacement des capteurs et ensuite se connecte au serveur. (Le nom de l'I.A. est défini dans le fichier $\boldsymbol{codeAModifier.h}$)

Ensuite le client attend que la course commence.

Une fois que la course commence il répète en boucle :

- Récupération des données du serveur
- Conversion des données
- Choix de commande
- Conversion de la commande
- Envoi des données

Jusqu'à que le serveur s'arrête.

La fonction qui gère la commande se trouve dans le fichier $\boldsymbol{codeAModifier.c}$ et dans notre cas elle donne comme commande d'aller à la vitesse de $100\ m/s$ et de garder la même distance a gauche et à droite de la voiture.