Réalisation d'un "bot" pour un jeu

Qualité de Développement - R4.02

Le thème de ce TP est de développer un bot, une IA, pour un jeu en ligne. Le serveur du jeu est le même que pour le TP Android. Pour rappel, vous trouverez sa documentation ici.

> Le projet Rust s'appellera "**game_bot**" dans votre dépôt Git.

Vous procéderez de manière incrémentale et en appliquant la méthode TDD pour développer votre bot.

Les fonctionnalités minimales sont les suivantes

- Connexion au serveur.
- Envoi d'une commande de déplacement aléatoire.
- Envoi d'une commande de tir.
- Prise en compte d'arguments sur la ligne de commande pour spécifier l'adresse du serveur et le port de connexion.

La connexion au serveur se fait via une simple socket. Vous trouverez facilement des exemples de code pour faire ça.

Bot solo

Utilisez ensuite les fonctionnalités offertes par le serveur pour améliorer les chances de victoire de votre bot.

Vous utiliserez notamment les informations sur les autres joueurs (orientation/distance) pour améliorer vos attaques et/ou votre défense.

Vous pouvez également utiliser l'information sur le projectile le plus proche pour votre survie.

Soyez créatif sur les stratégies.

Équipe

Lorsque votre bot solo fonctionne, créez un deuxième binaire `team.rs` dans votre crate (pas un nouveau projet). Pour cela, ajoutez ce fichier dans un le répertoire `src/bin` de votre projet. Vous pouvez également configurer le binaire dans votre Cargo.toml. La documentation se trouve ici.

Ce nouveau programme devra contrôler plusieurs bots via plusieurs connexions au serveur. Vous ajouterez aux arguments de la ligne de commande le nombre de bots dans l'équipe.

Il sera probablement utile de factoriser le code utilisé dans la première partie sur le bot solo pour ne pas ré-écrire du code.

Utilisez à bon escient les informations disponibles sur les différents joueurs pour identifier vos

équipiers et ne pas les attaquer.

Là encore, soyez créatifs sur les stratégies collectives employées pour faire gagner votre équipe.

Bonus

Pour ceux qui se prendraient d'une passion pour le Rust, vous pouvez paralléliser l'exécution des différents bots dans des threads. Vous pouvez utiliser les threads natifs du langage ou utiliser une abstraction de plus haut niveau comme la bibliothèque "*rayon*" dont les exemples sont là.

Évaluation

Les critères d'évaluation sont les suivants :

- **Qualité du code** :
  - Présence de tests pertinents
  - Clarté du code :
    - Structuration du code, découpage en entités logiques
    - Respect des convention de nommage
    - Respect des convention de formatage
- Présence d'un **Readme** présentant :
  - les fonctionnalités réalisées et
  - la structure du code
- Réalisation des **fonctionnalités**
  - Minimales
  - Comportement solo qui tient compte d'informations sur les autres joueurs
  - Comportement collectif