Projet Mario Kart IRL

Alexandre Le Lann – Lény Neurrisse – Camille Sicot

Partie 5 : Documentation

Voici la documentation de l'implémentation de notre fonctionnalité de pilote automatique du drone.

Objectif

Permettre aux joueurs de Mario Kart de se déplacer sur les circuits en utilisant une assistance automatique qui prend le contrôle de leur drone.

Détails

Lors du lancement d'une partie, une popup apparaît et demande au joueur s'il souhaite utiliser l'autopilot. Si l'autopilot est activé, un bouton est ajouté à l'interface utilisateur pendant la course pour permettre au joueur de basculer entre le mode manuel et l'autopilot. Les mouvements effectués par le drone en mode manuel sont enregistrés dans un fichier XML pour être utilisés en mode autopilot. Une classe *MovementRegistry* est créée pour gérer l'enregistrement et la récupération des mouvements, et une classe *AutoPilot* est créée pour gérer le pilote automatique.

Choix techniques

Les choix techniques effectués pour implémenter cette fonctionnalité incluent l'utilisation d'un fichier XML pour stocker les mouvements effectués en mode manuel, la création de classes pour gérer l'enregistrement et la récupération des mouvements ainsi que la gestion de l'autopilot. L'utilisation d'un fichier XML permet une flexibilité pour stocker les mouvements effectués, la création de classes pour gérer les mouvements et l'autopilot permet une séparation claire de la logique de chaque fonctionnalité et facilite la maintenance et la modification future du code.

Nous avons donc créé 2 classes : *AutoPilot* et *MovementRegistry*, et modifié 3 classes : *GUIGame*, *GUIWelcome*, *DroneController*.

Détail des classes et modifications

On peut retrouver facilement le code que l'on a modifié dans les classes déjà existantes par l'ajout de commentaires */* MODIFICATIONS */* et */* MODIFICATIONS FIN */* entre les bouts de code.

Classe *AutoPilot*

La classe *AutoPilot* est utilisée pour contrôler le drone en utilisant une liste de mouvements enregistrés. Elle contient des méthodes pour activer et désactiver l'autopilote, démarrer et arrêter l'autopilote, et exécuter les mouvements enregistrés.

Les méthodes suivantes sont utilisées pour activer et désactiver l'autopilote :

- *enable()* : Active l'autopilote
- *disable()* : Désactive l'autopilote
- *isEnable()* : Renvoie vrai si l'autopilote est activé

Les méthodes suivantes sont utilisées pour démarrer et arrêter l'autopilote :

- *start()* : Démarre l'autopilote
- *stop()* : Arrête l'autopilote
- *toggleAutoPilot()* : Inverse l'état de l'autopilote (démarre s'il est arrêté et arrête s'il est démarré)
- *isRun()* : Renvoie vrai si l'autopilote est en cours d'exécution

La méthode *executeMovements()* est utilisée pour contrôler le drone en fonction de la liste de mouvements enregistrés. Elle lit la liste de commandes à partir du fichier XML, attend un certain temps entre chaque commande et exécute la commande en utilisant les méthodes de *DroneController*. Ce temps est calculé en fonction du temps de pression d'un mouvement qui est enregistré dans le XML.

Les variables suivantes sont utilisées pour stocker l'état de l'autopilote :

- *enableStatus* : Indique si l'autopilote est activé
- *runStatus* : Indique si l'autopilote est en cours d'exécution
- *ctrlrAutopilot* : Contrôleur utilisé pour contrôler le drone

Classe *MovementRegistry*

La classe *MovementRegistry* permet d'enregistrer et de récupérer les mouvements qui ont été activés pendant la course. Elle contient des méthodes pour démarrer et arrêter l'enregistrement, définir le nom de fichier où les mouvements seront enregistrés, créer un fichier XML pour stocker les mouvements, et récupérer la liste des mouvements enregistrés.

Les méthodes sont :

- *startRecord()* : Démarre l'enregistrement des mouvements
- *stopRecord()* : Stoppe l'enregistrement des mouvements
- *getRecordingStatus()* : Vérifie si l'enregistrement est en cours ou non
- *setFileName(String fn)* : Défini le nom du fichier où les mouvements seront enregistrés
- *createFileXML()* : Créer un fichier XML vide pour enregistrer les mouvements
- *saveMovement(String mvmt)* : Enregistre un mouvement dans le fichier XML. Prend en paramètre le mouvement à enregistrer
- *getMovementList()* : Lis tous les mouvements enregistrés dans le fichier XML et les retourne sous forme d'une liste de chaînes de caractères

Les variables de classe suivantes définies sont :

- *isRecording* : un booléen qui indique si l'enregistrement est en cours ou non
- *fileName* : le nom du fichier où les mouvements seront enregistrés

Modifications de la classe *DroneController*

Cette classe gère les commandes envoyées au drone, ainsi que les informations reçues de celui-ci. Elle possède des méthodes pour démarrer et arrêter le contrôleur, ainsi que pour envoyer des commandes de déplacement au drone

Dans cette classe on a rajouté une méthode autopilot qui appelle la méthode *toggleAutoPilot()* afin de pouvoir activer ou non le pilote automatique.

Cette méthode est appelée dans la classe *GUIGame* lorsqu'on appuie sur le bouton.

Modifications de la classe *GUIWelcome*

La classe *GUIWelcome* définit l'interface de bienvenue de l'application, où l'on peut lancer une partie, se connecter au drone, créer des circuits.

Nous avons rajouté dans cette classe la possibilité d'activer le pilote automatique pendant la partie. Pour cela une boîte de dialogue d'alerte est créée en utilisant la classe *AlertDialog.Builder* lorsque l'on clique sur le bouton pour commencer une course (*startRaceBtnAction*).

La boîte de dialogue contient un message demandant à l'utilisateur s'il souhaite utiliser l'autopilote. Il y a deux boutons, "Oui" et "Non". Si l'utilisateur clique sur "Oui", plusieurs méthodes sont appelées de la classe *MovementRegistry* et *AutoPilot*:

- *MovementRegistry.setFileName(fn)*: Défini le nom de fichier XML
- *MovementRegistry.createFileXML()* : Créer le fichier XML pour enregistrer les mouvements
- *MovementRegistry.startRecord()*, : Démarre l'enregistrement des mouvements
- *AutoPilot.enable()* pour activer l'autopilote

Si l'utilisateur clique sur "Non", le code lance l'activité avec l'autopilote désactivé, en appelant la méthode *AutoPilot.disable()*. Le joueur ne verra alors pas le bouton du pilote automatique durant la partie.

Modifications de la classe *GUIGame*

La classe *GUIGame* s'occupe de la visualisation et de la détection des éléments sur les images de la caméra. En même temps, elle affiche les boutons de contrôle du drone et l'état du joueur, en envoyant des messages via le gestionnaire aux différents contrôleurs de l'application pour donner des informations sur le jeu en cours et l'état de la course.

Ici on s'intéresse surtout aux boutons de déplacement, puisque ceux-ci doivent être enregistrés dans le fichier XML pour être réutilisés, *turnLeftBtn*, *turnRightBtn*, *moveForwardBtn* et *moveBackwardBtn*. On ajoute une condition dans chacun des boutons : si l'enregistrement est activé (*if (MovementRegistry.getRecordingStatus()*), alors appelle la méthode *MovementRegistry.saveMovement* afin de stocker les mouvements dans le XML avec le nom du mouvement qui a été utilisé.

On crée aussi le bouton *autoPilotBtn* qui appelle la méthode *AutoPilot* dans la classe *DroneController*. C'est géré avec la méthode *setOnClickListener()*.

Il y a aussi la gestion de l'affichage du bouton, si le pilote est activé ou non.

De plus, lorsque l'on clique sur le bouton, les autres boutons de déplacement deviennent invisibles, pour une meilleure immersion. Il suffit à l'utilisateur de cliquer à nouveau sur le bouton autopilot pour réactiver le mode manuel et faire réapparaître les boutons.