IDM - Soutenance 1

Groupe 1 - Les roboticiens

N. Salleron Y. Ghigoff K. Vu-Saintonge A. Archambault

14 Décembre 2017

Sommaire

Introduction

Une utilisation de plus en plus populaire

- Utilisation dans le cadre militaire dans les année 1990.
- De plus en plus populaire au sein du grand public.
- Las Vegas, CES 2016, présentation du "drone dance" par la société Parrot
- La télévision avec "La France a un incroyable talent"

FIGURE – Exemple de drone

Introduction

Contexte

- Domaine de la danse.
- Préparation d'une chorégraphie et exécution sur le terrain.
- Création d'un langage de programmation spécifique.

Objectif

- Produire un langage de programmation simple et compréhensible pour l'utilisateur.
- Rendre l'utilisateur autonome.
- Adaptable à plusieurs drones.
- Faire danser le drone.

FIGURE – Autre exemple de drone

Editeur de langage

Description de l'éditeur

Notre éditeur posséde les fonctionnalités suivantes :

- Auto-complétion du langage.
- Détection des fautes syntaxiques.
- Détection des erreurs de cohérence dans les scénarios.

Editeur de langage

Description de l'éditeur

Notre éditeur posséde les fonctionnalités suivantes :

- Auto-complétion du langage.
- Détection des fautes syntaxiques.
- Détection des erreurs de cohérence dans les scénarios.

Écriture de la chorégraphie

- L'utilisateur écrit sa chorégraphie par une suite d'action séparée par un retour à la ligne.
- Une chorégraphie commence par un décollage et un atterrissage.
- Usage possible de fonctions et d'instruction simple.

Editeur de langage

Description de l'éditeur

Notre éditeur posséde les fonctionnalités suivantes :

- Auto-complétion du langage.
- Détection des fautes syntaxiques.
- Détection des erreurs de cohérence dans les scénarios.

Écriture de la chorégraphie

- L'utilisateur écrit sa chorégraphie par une suite d'action séparée par un retour à la ligne.
- Une chorégraphie commence par un décollage et un atterrissage.
- Usage possible de fonctions et d'instruction simple.

Conditions de fonctionnement

- Le drone utilisé est un drone à hélices.
- Le drone est allumé, connecté via Wi-Fi à l'ordinateur.
- Le drone doit être utilisé dans un endroit sans un vent fort.
- Le programme utilisateur réalisé sous Eclipse Oxygen version 4.7.0.
- Java version 1.8.

Fonctionnement du drone

Mouvements sur les axes

Le drone évoluant dans un environnement 3D, ce dernier peut effectuer différentes actions :

- Altitude : évolution sur l'axe Z.
  Correspond aux instructions "monter" et "descendre"
- Roll : évolution sur l'axe X.
  Correspond aux instructions "gauche" et "droite"
- Pitch : évolution sur l'axe Y.
  Correspond aux instructions "avancer" et "reculer"

Fonctionnement du drone

Mouvements sur les axes

Le drone évoluant dans un environnement 3D, ce dernier peut effectuer différentes actions :

- Altitude : évolution sur l'axe Z.
  Correspond aux instructions "monter" et "descendre"
- Roll : évolution sur l'axe X.
  Correspond aux instructions "gauche" et "droite"
- Pitch : évolution sur l'axe Y.
  Correspond aux instructions "avancer" et "reculer"

Éléments particuliers

- Décollage et Attérrissage : correspond aux instructions "decoller" et "atterrir"
- Pause : correspond à l'instruction "pause"
- Rotation : correspond aux instructions "rotationGauche" et "rotationDroite"

Fonctionnement du drone

Mouvements sur les axes

Le drone évoluant dans un environnement 3D, ce dernier peut effectuer différentes actions :

- Altitude : évolution sur l'axe Z.
  Correspond aux instructions "monter" et "descendre"
- Roll : évolution sur l'axe X.
  Correspond aux instructions "gauche" et "droite"
- Pitch : évolution sur l'axe Y.
  Correspond aux instructions "avancer" et "reculer"

Éléments particuliers

- Décollage et Attérrissage : correspond aux instructions "decoller" et "atterrir"
- Pause : correspond à l'instruction "pause"
- Rotation : correspond aux instructions "rotationGauche" et "rotationDroite"

Cas de la caméra

- Caméra : Il n'est pas standard qu'un drone posséde une caméra.

FIGURE – Exemple de drone sans caméra

Prologue

Définir 5 constantes de vol

- define **vitesse_hauteur_max** : vitesse maximale d'élévation du drone pour la chorégraphie.
- define **vitesse_deplacement_max** : vitesse maximale de déplacement sur le plan horizontal du drone pour la chorégraphie.
- define **vitesse_rotation_max** : vitesse maximale de rotation du drone pour la chorégraphie.
- define **hauteur_max** : Cette constante permet de limiter l'altitude maximale du drone en vol.
- define **eloignement_max** : Cette constante permet de contrôler la distance horizontale du drone en vol par rapport au point de décollage.

Exemple de define :

```
define vitesse_hauteur_max 50%
```

Instructions basique

Le but est de rendre accessible le pilotage de drone aux chorégraphes.

Les 11 instructions basiques

- decoller
- atterrir
- pause(durée : Seconde)
- monter(durée : Seconde, vitesse_verticale : Pourcentage)
- decendre(durée : Seconde, vitesse_verticale : Pourcentage)
- avancer(durée : Seconde, vitesse_deplacement : Pourcentage)
- reculer(durée : Seconde, vitesse_deplacement : Pourcentage)
- gauche(durée : Seconde, vitesse_deplacement : Pourcentage)
- droite(durée : Seconde, vitesse_deplacement : Pourcentage)
- rotationGauche(durée : Seconde, vitesse_rotation : Pourcentage)
- rotationDroite(durée : Seconde, vitesse_rotation : Pourcentage)

- 1er paramètre est la durée du mouvement, en Seconde.
- 2éme paramètre est la vitesse du mouvement, en pourcentage. Il représente la vitesse du drone par rapport à la vitesse définie dans la section "prologue".

Les Instructions paralléles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèle.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenter par le symbole '&'.

Les Instructions paralléles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèle.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenter par le symbole '&'.
- Disponible que pour certaines instructions comme :
  1. monter
  2. descendre
  3. avancer
  4. reculer
  5. gauche
  6. droite
  7. rotationGauche
  8. rotationDroite

Les Instructions paralléles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèle.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenter par le symbole '&'.
- Disponible que pour certaines instructions comme :
  1. monter
  2. descendre
  3. avancer
  4. reculer
  5. gauche
  6. droite
  7. rotationGauche
  8. rotationDroite
- Maximum de 3 instructions parallélisable.

Exemple :

monter(1,10%) & descendre(4,20%)

Les Instructions paralléles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèle.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenter par le symbole '&'.
- Disponible que pour certaines instructions comme :
  1. monter
  2. descendre
  3. avancer
  4. reculer
  5. gauche
  6. droite
  7. rotationGauche
  8. rotationDroite
- Maximum de 3 instructions parallélisable.

Exemple :

monter(1,10%) & descendre(4,20%)
-> Impossible les deux commandes s'opposent.
gauche(3,25%) & gauche(4,20%)

Les Instructions paralléles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèle.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenter par le symbole '&'.
- Disponible que pour certaines instructions comme :
  1. monter
  2. descendre
  3. avancer
  4. reculer
  5. gauche
  6. droite
  7. rotationGauche
  8. rotationDroite
- Maximum de 3 instructions parallélisable.

Exemple :

monter(1,10%) & descendre(4,20%)
-> Impossible les deux commandes s'opposent.
gauche(3,25%) & gauche(4,20%)
-> Impossible les deux commandes sont de même type.
rotationDroite(2,25%) & avancer(5,20%)

Les Instructions paralléles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèle.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenter par le symbole '&'.
- Disponible que pour certaines instructions comme :
  1. monter
  2. descendre
  3. avancer
  4. reculer
  5. gauche
  6. droite
  7. rotationGauche
  8. rotationDroite
- Maximum de 3 instructions parallélisable.

Exemple :

monter(1,10%) & descendre(4,20%)
-> Impossible les deux commandes s'opposent.
gauche(3,25%) & gauche(4,20%)
-> Impossible les deux commandes sont de même type.
rotationDroite(2,25%) & avancer(5,20%)
-> Ok

Les fonctions

- Le langage permet de définir des fonctions.
- Les fonctions sont une suite d'instructions séquentielles.
- Il n'est pas possible de paralléliser deux fonctions.
- Une fonction ne peut pas s'appeler elle-même.
- Une fonction ne peut pas appeler d'autres fonctions.

Exemple :

```
func maFonction() {
    rotationDroite(2,25%) & avancer(5,20%)
    droite(3,80%)
    avancer(4,10%)
}
```

Le bloc "main" et les bibliothèques de fonctions

Le point d'entrée "main { }" :

- Défini par le mot clé "main"
- Le contenu de ce bloc d'instruction sera exécuté.
- Il est le seul à pouvoir appeler des fonctions.

Le bloc "main" et les bibliothèques de fonctions

Le point d'entrée "main { }" :

- Défini par le mot clé "main"
- Le contenu de ce bloc d'instruction sera exécuté.
- Il est le seul à pouvoir appeler des fonctions.

Exemple :

```
main {
        decoller()
        gauche(1,10%)
        avancer(4,34%)
        maFonction()
        atterrir()
}
func maFonction() {
      rotationDroite(2,25%) & avancer(5,20%)
      droite(3,80%)
      avancer(4,10%)
}
```

Le bloc "main" et les bibliothèques de fonctions

Le point d'entrée "main { }" :

- Défini par le mot clé "main"
- Le contenu de ce bloc d'instruction sera exécuté.
- Il est le seul à pouvoir appeler des fonctions.

Exemple :

```
main {
        decoller()
        gauche(1,10%)
        avancer(4,34%)
        maFonction()
        atterrir()
}
func maFonction() {
       rotationDroite(2,25%) & avancer(5,20%)
       droite(3,80%)
       avancer(4,10%)
}
```

Les bibliothèques de fonctions :

- Possibilité d'utiliser des fonctions définies dans des fichiers .lib_drone.
- Doivent être dans le même répertoire que le fichier appelant les fonctions.

Le bloc "main" et les bibliothèques de fonctions

Le point d'entrée "main { }" :

- Défini par le mot clé "main"
- Le contenu de ce bloc d'instruction sera exécuté.
- Il est le seul à pouvoir appeler des fonctions.

Exemple :

```
main {
        decoller()
        gauche(1,10%)
        avancer(4,34%)
        maFonction()
        atterrir()
}
func maFonction() {
       rotationDroite(2,25%) & avancer(5,20%)
       droite(3,80%)
       avancer(4,10%)
}
```

Les bibliothèques de fonctions :

- Possibilité d'utiliser des fonctions définies dans des fichiers .lib_drone.
- Doivent être dans le même répertoire que le fichier appelant les fonctions.

Exemple :

```
import <monFichier.lib_drone>
```

Le bloc "main" et les bibliothèques de fonctions

Le point d'entrée "main { }" :

- Défini par le mot clé "main"
- Le contenu de ce bloc d'instruction sera exécuté.
- Il est le seul à pouvoir appeler des fonctions.

Exemple :

```
main {
        decoller()
        gauche(1,10%)
        avancer(4,34%)
        maFonction()
        atterrir()
}
func maFonction() {
      rotationDroite(2,25%) & avancer(5,20%)
      droite(3,80%)
      avancer(4,10%)
}
```

Les bibliothèques de fonctions :

- Possibilité d'utiliser des fonctions définies dans des fichiers .lib_drone.
- Doivent être dans le même répertoire que le fichier appelant les fonctions.

Exemple :

```
import <monFichier.lib_drone>
```

Utilisation du mot-clé "extern" pour référencer la fonction de la bibliothèque que nous souhaitons utiliser.

Exemple :

```
import <monFichier.lib_drone>
extern func toto()
```

Tests de validations

Validation :	TVEDT-01
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	gauche(1,10%)
	atterrir()
	}

Résultat attendu :	Aucune erreur n'est détectée par l'éditeur.

Tests de validations

Validation :	TVEDT-02
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	monter(1,20%)
	avancer(1,20%)
	reculer(1,20%)
	atterrir()
	}

Résultat attendu :	Aucune erreur n'est détectée par l'éditeur.

Tests de validations

Validation :	TVEDT-03
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	blablabla()
	atterrir()
	}

Résultat attendu :	Une erreur est détectée sur la commande *blabalbla*.

Tests de validations

Validation :	TVEDT-04
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	monter("hello")
	atterrir()
	}

Résultat attendu :	Une erreur est détectée sur la commande *monter*.

Tests de validations

Validation :	TVEDT-05
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	atterrir()
	monter(1,20%)
	}

Résultat attendu :	Une erreur est détectée sur la commande *monter*.

Tests de validations

Validation :	TVEDT-06
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	monter(1,20%)
	decoller()
	atterrir()
	}

Résultat attendu :	Une erreur est détectée sur la commande *monter*.

Tests de validations

Validation :	TVEDT-07
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	decoller()
	atterrir()
	}

Résultat attendu :	Une erreur est détectée sur la deuxième commande *decoller*.

Tests de validations

Validation :	TVEDT-08
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	atterrir()
	atterrir()
	}

Résultat attendu :	Une erreur est détectée sur la deuxième commande *atterrir*.

Tests de validations

Validation :	TVEDT-09
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	aller_retour()
	atterrir()
	}
	func aller_retour() {
	avancer(5,20%)
	reculer(5,20%)
	}

Résultat attendu :	Aucune erreur n'est détectée par l'éditeur.

Tests de validations

Validation :	TVEDT-10
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	foo()
	atterrir()
	}
	func bar() {
	avancer(5,20%)
	}

Résultat attendu :	Une erreur est détectée sur l'appel de la fonction *foo*.

Tests de validations

Validation :	TVEDT-11
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	monter(1,20%) & avancer(1,20%)
	atterrir()
	}

Résultat attendu :	Aucune erreur n'est détectée par l'éditeur.

Tests de validations

Validation :	TVEDT-12
Contexte :	L'utilisateur a démarré son éditeur.
Entrée :	aucune

Scénario :	define vitesse_hauteur_max 100%
	define vitesse_deplacement_max 40%
	define vitesse_rotation_max 50%
	define hauteur_max 10
	define eloignement_max 4
	
	main {
	decoller()
	monter(1,20%) & descendre(1,20%)
	atterrir()
	}

Résultat attendu :	Une erreur est détectée sur l'appel de la composition parallèle des fonctions *monter* et *descendre*