IDM - Soutenance 3

Groupe 1 - Les roboticiens

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

29 Janvier 2018

SCIENCES SORBONNE UNIVERSITÉ

Sommaire

SCIENCES SORBONNE UNIVERSITÉ

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 – Exemple de drone

SCIENCES SORBONNE UNIVERSITÉ

Tranches

But des tranches

- 4 Tranches distinctes.
- Associées à des dates de livraison.
- Chacune ayant plusieurs objectifs à remplir.

SCIENCES SORBONNE UNIVERSITÉ

Tranches

But des tranches

- 4 Tranches distinctes.
- Associées à des dates de livraison.
- Chacune ayant plusieurs objectifs à remplir.

Vérification des tranches

- Tests unitaires réalisés par l'équipe.
- Mise en en place de test de validation.
- Détaillés dans le cahier des charges.

SCIENCES SORBONNE UNIVERSITÉ

Tranches 0 et 1

Objectifs de la Tranche 0

- Analyse de la demande par l'équipe.
- Discussion avec le client.
- Rédaction et rendu du cahier des charges.

SCIENCES SORBONNE UNIVERSITÉ

Tranches 0 et 1

Objectifs de la Tranche 0

- Analyse de la demande par l'équipe.
- Discussion avec le client.
- Rédaction et rendu du cahier des charges.

Objectifs de la Tranche 1

- Création d'une grammaire pour DSL.
- Création de l'éditeur associé.
- Vérification des erreurs (syntaxiques, cycles...).

SCIENCES SORBONNE UNIVERSITÉ

Tranches 0 et 1

Objectifs de la Tranche 0

- Analyse de la demande par l'équipe.
- Discussion avec le client.
- Rédaction et rendu du cahier des charges.

Objectifs de la Tranche 1

- Création d'une grammaire pour DSL.
- Création de l'éditeur associé.
- Vérification des erreurs (syntaxiques, cycles...).

Retours du client

- Syntaxe élégante (notamment sur les mouvements parallèles).
- Changement sur l'intégration des bibliothèques de fonction et leur utilisation.
- Changement sur l'appel de fonction dans une fonction (détection de cycle).

Tranche 2

Objectifs

- Génération de code JAVA.
- Création d'une interface de Runtime pour gérer plusieurs API de drone.
- Affichage textuel des commandes qui seront envoyées au drone.

SCIENCES SORBONNE UNIVERSITÉ

Tranche 2

Objectifs

- Génération de code JAVA.
- Création d'une interface de Runtime pour gérer plusieurs API de drone.
- Affichage textuel des commandes qui seront envoyées au drone.

Retours du client

- Tranche satisfaisante.
- Création d'un bouton pour la génération du code (éviter de générer le code trop souvent).

SCIENCES SORBONNE UNIVERSITÉ

Tranche 3

Objectifs

- Liaison de notre interface avec la runtime du Parrot.
- Déployer la solution sur un ordinateur d'exécution (notre ordinateur ou celui du client).
- Test de la solution (recette).

SCIENCES SORBONNE UNIVERSITÉ

Editeur de langage

Description de l'éditeur

Notre éditeur possède les fonctionnalités suivantes :

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

SCIENCES SORBONNE UNIVERSITÉ

Editeur de langage

Description de l'éditeur

Notre éditeur possède les fonctionnalités suivantes :

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

Ecriture de la chorégraphie

- L'utilisateur écrit sa chorégraphie par une suite d'actions séparées par un retour à la ligne.
- Une chorégraphie commence par un décollage et un atterrissage.
- Usage possible de fonctions et d'instructions simples.

SCIENCES SORBONNE UNIVERSITÉ

Editeur de langage

Description de l'éditeur

Notre éditeur possède les fonctionnalités suivantes :

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

Ecriture de la chorégraphie

- L'utilisateur écrit sa chorégraphie par une suite d'actions séparées par un retour à la ligne.
- Une chorégraphie commence par un décollage et un atterrissage.
- Usage possible de fonctions et d'instructions simples.

Conditions de fonctionnement

- Le drone utilisé est un drone à hélices.
- Le drone est allumé, connecté via Wi-Fi à l'ordinateur.
- Java version 1.8.
- Système d'exploitation MacOS et Linux.

SCIENCES SORBONNE UNIVERSITÉ

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"

SCIENCES SORBONNE UNIVERSITÉ

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"

Eléments particuliers

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

SCIENCES SORBONNE UNIVERSITÉ

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"

Eléments 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

SCIENCES SORBONNE UNIVERSITÉ

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%
```

SCIENCES SORBONNE UNIVERSITÉ

Instructions basiques

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

Les 11 instructions basiques

- decoller et 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.
- 2eme paramètre est la vitesse du mouvement, en pourcentage.
  Il représente la vitesse du drone par rapport à la vitesse définie dans la section "prologue".

SCIENCES SORBONNE UNIVERSITÉ

Les Instructions parallèles

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

SCIENCES SORBONNE UNIVERSITÉ

Les Instructions parallèles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèles.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenté 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

SCIENCES SORBONNE UNIVERSITÉ

Les Instructions parallèles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèles.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenté 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 4 instructions parallélisables.

Exemple :
rotationDroite(2,25%) & avancer(5,20%)

SCIENCES SORBONNE UNIVERSITÉ

Les Instructions parallèles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèles.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenté 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 4 instructions parallélisables.

Exemple :
rotationDroite(2,25%) & avancer(5,20%)
-> Ok
monter(1,10%) & descendre(4,20%)

SCIENCES SORBONNE UNIVERSITÉ

Les Instructions parallèles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèles.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenté 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 4 instructions parallélisables.

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

SCIENCES SORBONNE UNIVERSITÉ

Les Instructions parallèles

- Notre langage intègre un mécanisme d'exécution d'instructions parallèles.
- Il est possible d'ordonner au drone de faire plusieurs instructions en même temps.
- Mécanisme implémenté 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 4 instructions parallélisables.

Exemple :
rotationDroite(2,25%) & avancer(5,20%)
-> Ok
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.

SCIENCES SORBONNE UNIVERSITÉ

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 et les cycles ne sont pas autorisés.

Exemple :

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

Exemple ne fonctionnant pas :

```
func A() {
    B() //Erreur sur B()
}
func B() {
    A() //Erreur sur A()
}
```

SCIENCES SORBONNE UNIVERSITÉ

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'instructions sera exécuté.
- Il est le seul à pouvoir appeler des fonctions.

SCIENCES SORBONNE UNIVERSITÉ

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'instructions 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%)
}
```

SCIENCES SORBONNE UNIVERSITÉ

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'instructions 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.

SCIENCES SORBONNE UNIVERSITÉ

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'instructions 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>
```

SCIENCES SORBONNE UNIVERSITÉ

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'instructions 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 nom de fichier (ici : monFichier) pour référencer la fonction de la bibliothèque que nous souhaitons utiliser.

Exemple :

```
import <monFichier.lib_drone>
main {
        decoller()
        monFichier.toto()
        atterrir()
}
```

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Application

- Disponible sur MacOS.
- Disponible sur Linux.

Environnement

Application

- Disponible sur MacOS.
- Disponible sur Linux.

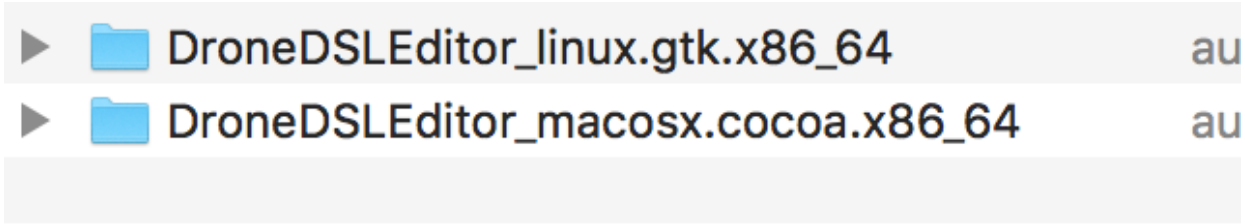


FIGURE – Distributions

Environnement

Lancement de l'environnement : exemple sous MacOS

- Lancement de l'application via le script DroneDSLEditor.sh

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Lancement de l'environnement : exemple sous MacOS

- Lancement de l'application via le script DroneDSLEditor.sh
  1. Vérification des dépendances (seulement la première fois)

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Lancement de l'environnement : exemple sous MacOS

- Lancement de l'application via le script DroneDSLEditor.sh
  1. Vérification des dépendances (seulement la première fois)
  2. Compilation du SDK (seulement la première fois)

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Lancement de l'environnement : exemple sous MacOS

- Lancement de l'application via le script DroneDSLEditor.sh
  1. Vérification des dépendances (seulement la première fois)
  2. Compilation du SDK (seulement la première fois)
  3. Lancement de l'application

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Lancement de l'environnement : exemple sous MacOS

- Lancement de l'application via le script DroneDSLEditor.sh
  1. Vérification des dépendances (seulement la première fois)
  2. Compilation du SDK (seulement la première fois)
  3. Lancement de l'application

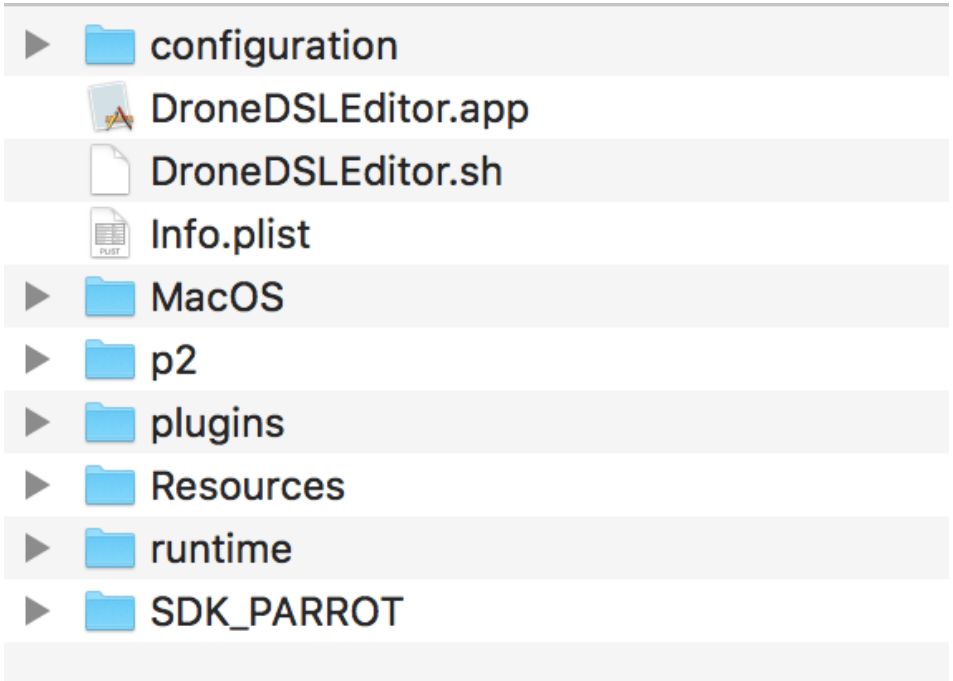


FIGURE – Lancement de l'application

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Création d'un projet

- Via l'interface graphique (file -> new -> project).

Environnement

Création d'un projet

- Via l'interface graphique (file -> new -> project).

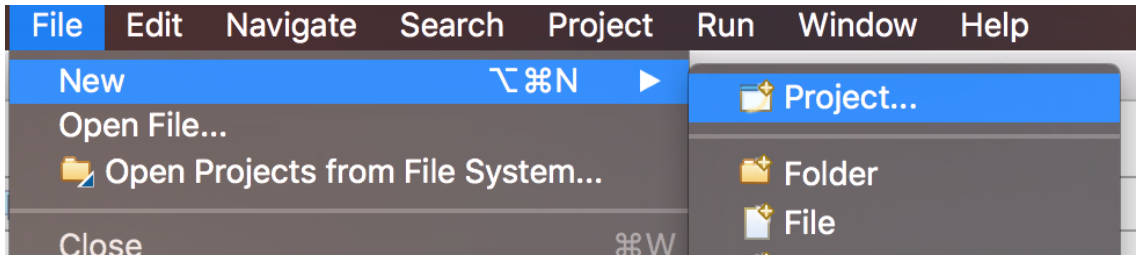


FIGURE – Création d'un nouveau projet

Environnement

Création d'un projet

- Créer un projet Xtext nommé DroneDSL Project.

SCIENCES SORBONNE UNIVERSITÉ

Environnement

Création d'un projet

- Créer un projet Xtext nommé DroneDSL Project.

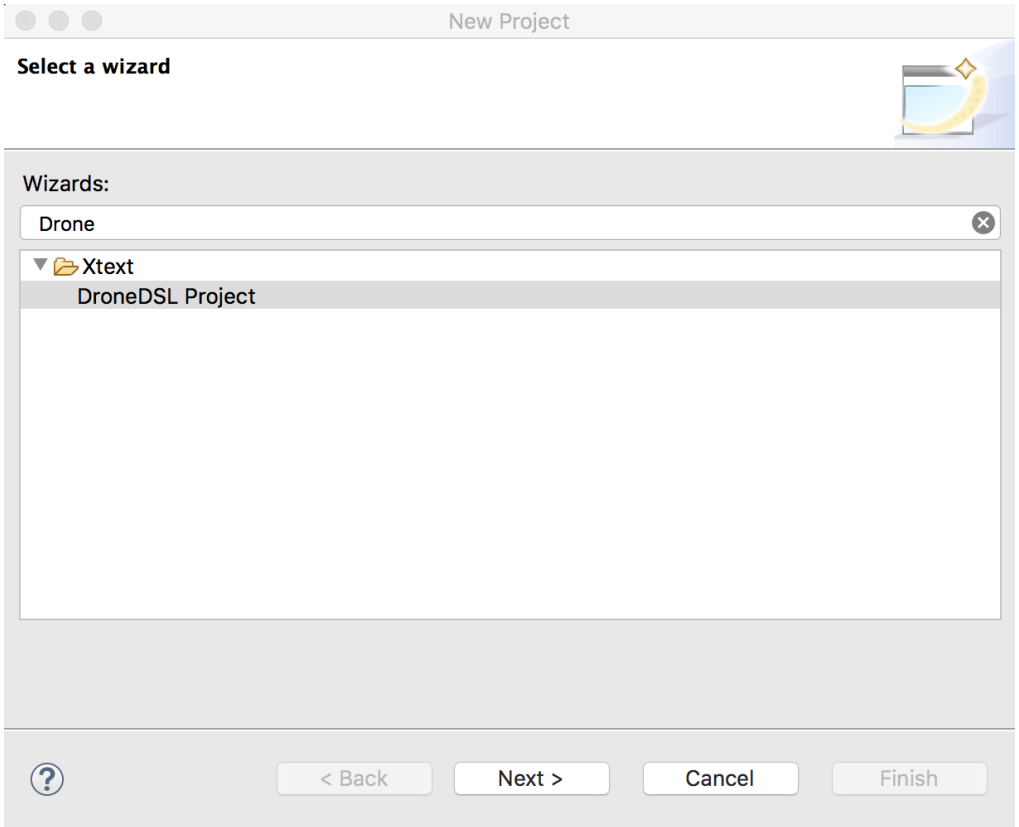


FIGURE – Création d'un nouveau projet Xtext

Environnement

Création d'un projet

- Création du fichier ".main_drone" automatique.

Environnement

Création d'un projet

- Création du fichier ".main_drone" automatique.

```
/*
 * Paramètres d'exécution
 */
define eloignement_max 2
define hauteur_max 2
define vitesse_deplacement_max 100%
define vitesse_hauteur_max 100%
define vitesse_rotation_max 100%

main {
    decoller()
    // main auto-généré
    atterrir()
}
```

FIGURE – Fichiers auto-générés

Environnement

Lancer le projet

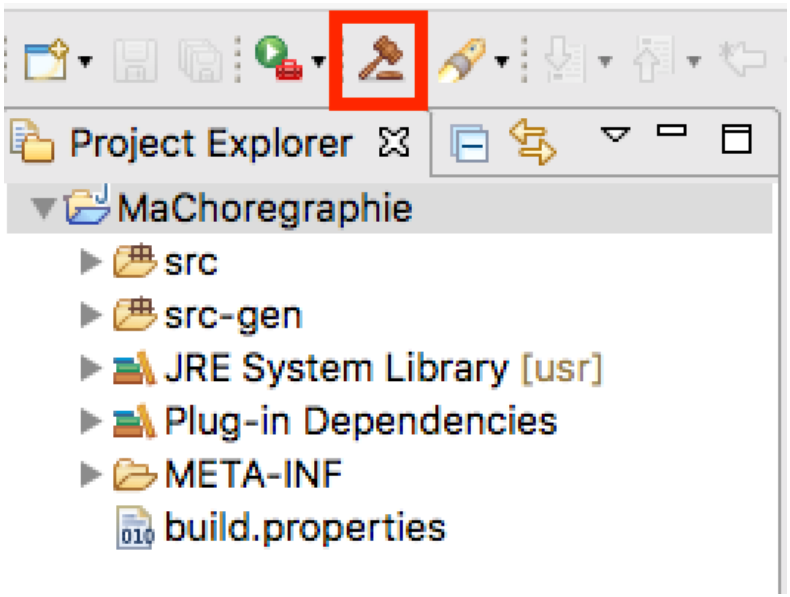
- Cliquer sur le bouton "génération" pour chaque fichier que l'utilisateur à modifier. Il peut ensuite faire un sélectionner le dossier "src" puis cliquer sur le bouton Run (Bouton play avec un fond vert) une fois qu'il n'y a plus d'erreur syntaxique.

SCIENCES SORBONNE UNIVERSITÉ

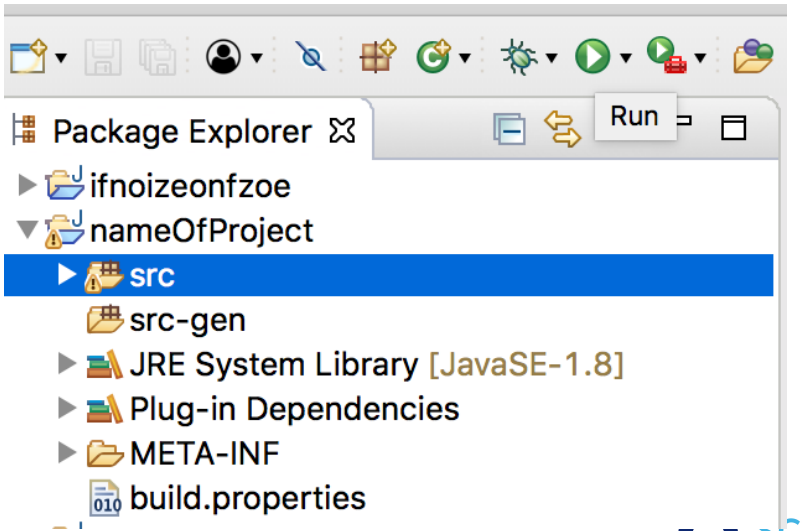
Environnement

Lancer le projet

- Cliquer sur le bouton "génération" pour chaque fichier que l'utilisateur à modifier. Il peut ensuite faire un sélectionner le dossier "src" puis cliquer sur le bouton Run (Bouton play avec un fond vert) une fois qu'il n'y a plus d'erreur syntaxique.



(c) Bouton génération



(d) Lancement

Fonctionnement

Général

- Chorégraphie écrite par l'utilisateur.

Fonctionnement

Général

- Chorégraphie écrite par l'utilisateur.
- Chorégraphie interprétée pour générer des classes JAVA (dont la principale nommée Main).

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement

Général

- Chorégraphie écrite par l'utilisateur.
- Chorégraphie interprétée pour générer des classes JAVA (dont la principale nommée Main).
- La classe Main instancie les mouvements de l'utilisateur et les différents appels de fonctions.

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement

Général

- Chorégraphie écrite par l'utilisateur.
- Chorégraphie interprétée pour générer des classes JAVA (dont la principale nommée Main).
- La classe Main instancie les mouvements de l'utilisateur et les différents appels de fonctions.
- La classe Runtime (JAVA) se charge de l'ouverture d'un flux de communication en direction de notre programme runtime C (déjà compilé).

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement

Général

- Chorégraphie écrite par l'utilisateur.
- Chorégraphie interprétée pour générer des classes JAVA (dont la principale nommée Main).
- La classe Main instancie les mouvements de l'utilisateur et les différents appels de fonctions.
- La classe Runtime (JAVA) se charge de l'ouverture d'un flux de communication en direction de notre programme runtime C (déjà compilé).
- Chaque mouvements de notre programme est envoyé au drone via notre Runtime (JAVA vers C).

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement

Général

- Chorégraphie écrite par l'utilisateur.
- Chorégraphie interprétée pour générer des classes JAVA (dont la principale nommée Main).
- La classe Main instancie les mouvements de l'utilisateur et les différents appels de fonctions.
- La classe Runtime (JAVA) se charge de l'ouverture d'un flux de communication en direction de notre programme runtime C (déjà compilé).
- Chaque mouvements de notre programme est envoyé au drone via notre Runtime (JAVA vers C).
- Notre runtime C interprète ces informations et envoi des commandes au drone via l'API du constructeur.

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement

Détails

- Mode de fonctionnement Client/Serveur.
- Une commande est associée à un Thread.
- Pour les commandes parallèles, une liste de thread est associée à ces dernières.
- Un Thread attend le temps renseigné par l'utilisateur dans sa chorégraphie avant de demander à la runtime C de d'arrêter le mouvement.

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement : quelques images

```
public class Main {
    static DroneRuntime runtime = new DroneRuntimePrint();
    static Prologue prologue_1981352351 = new Prologue(new Pourcent(10),new Pourcent(10), new Pourcent(20), 20,30);
    public static void main(String[] args) {

        runtime.execPrologue(prologue_1981352351);
        new Decoller().execute(runtime);
        Parallele p2_258366224 = new Parallele();
              p2_258366224.addCommande(      new Gauche(new Seconde(10),new Pourcent(30)));
              p2_258366224.addCommande(      new Avancer(new Seconde(10),new Pourcent(30)));
        p2_258366224.execute(runtime);
        Parallele p2_1987299330 = new Parallele();
              p2_1987299330.addCommande(     new Droite(new Seconde(10),new Pourcent(30)));
              p2_1987299330.addCommande(     new Avancer(new Seconde(10),new Pourcent(30)));
        p2_1987299330.execute(runtime);
        Parallele p3_1008265679 = new Parallele();
              p3_1008265679.addCommande(     new Droite(new Seconde(10),new Pourcent(30)));
              p3_1008265679.addCommande(     new Avancer(new Seconde(10),new Pourcent(30)));
              p3_1008265679.addCommande(     new RotationGauche(new Seconde(3),new Pourcent(10)));
        p3_1008265679.execute(runtime);
        new Gauche(new Seconde(10),new Pourcent(10)).execute(runtime);
        A();
        new Droite(new Seconde(10),new Pourcent(20)).execute(runtime);
        new Atterrir().execute(runtime);
    }

    public static void A() {
        new Gauche(new Seconde(10),new Pourcent(10)).execute(runtime);
        B();
    }
    public static void B() {
        new Droite(new Seconde(10),new Pourcent(5)).execute(runtime);
    }

}
```

FIGURE – Classe Main générée

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement : quelques images

```
@Override
public void execAvancer(Avancer a) {
        try {
                writeToSubProcessStdin(AVANCER_INPUT_CODE, a.getVitesse().getValue(), true);
                Thread.sleep(a.getDuree().getValue() * 1000);
                writeToSubProcessStdin(AVANCER_INPUT_CODE, 0, true);
        } catch (Exception e) {
                e.printStackTrace();
                process.destroy();
                System.exit(-1);
        }
}
```

FIGURE – Exemple de la fonction Avancer de la Runtime Parrot

SCIENCES SORBONNE UNIVERSITÉ

Fonctionnement : quelques images

```
void handle_avancer(int pourcent_vitesse,void *customData)
{
  ARCONTROLLER_Device_t *deviceController = (ARCONTROLLER_Device_t *)customData;
  deviceController->aRDrone3->setPilotingPCMDFlag(deviceController->aRDrone3, 1);
  deviceController->aRDrone3->setPilotingPCMDPitch(deviceController->aRDrone3, pourcent_vitesse);
  ARSAL_PRINT(ARSAL_PRINT_ERROR, TAG, "Received avancer(%d) from stdin\n", pourcent_vitesse);
}
```

FIGURE – Exemple de la fonction Avancer de notre programme C pour Parrot

SCIENCES SORBONNE UNIVERSITÉ

Demonstration

Démonstration

SCIENCES SORBONNE UNIVERSITÉ