Louis Dhellemmes, Clément Guerin, Charles Hild-Lê
Élèves à l'UPSSTECH en dernière année de la formation SRI

Projet NAAVI

Documentation de l'interface ENVOL

Pour tester le fonctionnement de l'interface, les triggers qui sont normalement renvoyés par le drone peuvent être déclenchés en appuyant sur la touche "m".

1/ Organisation

a) Les scènes

Login	Permet de s'identifier
ENVOL	Principal

b) Les panels

Panels	Description	Scripts associés
Initialiseur	Permet d'initialiser tous les panels et les caméras au lancement du logiciel.	init_scene_home.cs
Container Point Cloud	Permet de charger à l'avance le nuage de point	InitPointCloud.cs
Attribution		
Info Canvas		
Reload Map Canvas		
Map	Affiche la carte suivant les coordonnées en entrer	AbstracMap.cs reloadmapV2.cs
Canvas	Canvas contenant toute la partie 2D de l'interface	
Panneau Latéral	Permet de changer de page	
Home Panel	Page d'accueil de l'interface avec un affichage météo	GetMyWeather.cs

Mission Panel	Regroupe tous les sous panels du déroulement de la mission	LockValider.cs StockerCoord.cs SkipPyloneDetection.cs SendData.cs
Loading Go Back	Cercle de chargement en attendant le drone atterrit	GoingBack.cs
Drone Panel	Paramètres du drone	
History Panel	Historique des missions enregistrées dans une base de données extrêmement perfectionnée et innovante	
Paramètres	Paramètres de l'application	slider.cs
Aide	Fournit une aide sur l'utilisation de l'application	
WarningFlying	Popup de warning permettant d'éviter que l'utilisateur se déconnecte ou éteint la tablette lorsque le drone est en vol	
Canvas 3D	Canvas contenant toute la partie 3D de l'interface	
Panel3D	Permet l'affichage de point du pylône et de son environnement	NuageDePointV5.cs
Panel Way Point	Permet l'affichage des points de passages par lequel passe le drone	WayPoint.cs
Panel En Cours	Texte et bouton lorsque le drone est en train d'analyser le pylône	
Panel Terminée	Texte et bouton lorsque l'analyse est terminée	analyseTerminee.cs
Rotator	Donne la possibilité à l'utilisateur de faire tourner le nuage de points ainsi que les points de passages	Rotate.cs

c) Les caméras de la scène ENVOL

Main Camera	Caméra principale pour l'affichage de l'interface
Main camera 2	Caméra pour l'affichage de la carte
Camera3D	Caméra permettant de gérer l'affichage de toutes les parties 3D

2/ SDK map

Pour la création de la carte nous utilisons le SDK mapbox avec Openstreetmap. Celui-ci nous permet de créer une carte à l'aide de coordonnées GPS ou d'un lieu, dans notre cas seul l'utilisation des coordonnées GPS sera utilisées. Le type de carte affichée peut être changé (satellite, hybride, street) ainsi que l'affichage du relief ou non. Nous avons aussi la possibilité de zoomer, pour le moment cette fonctionnalité a été désactivé car sinon la carte changer de dimension. Nous pouvons choisir une zone de sécurité/atterrissage en récupérant les coordonnées GPS suite à un clic de l'utilisateur.

Pour utiliser le SDK nous devons rentrer une clef dans Unity, Mapbox > setup, cette clef est disponible gratuitement sur leur site après la création d'un compte.

3/ connexion ROS

Afin de récupérer les données du système aptère nous avons mis en place un rosbridge à l'aide de websocket. Celui-ci permet de créer des topic afin de publier des données dessus envoyer des informations, il peut aussi s'abonner à un topic du système aptère afin d'en récupérer les données.

Dans notre cas nous récupérons les données du nuage de point afin de les afficher dans l'interface une fois l'analyse du pylône compléter (topic /rtab/cloud_map). Nous récupérons aussi l'image produite par l'algorithme de reconnaissance du pylône (topic /aptere/approche/image) et le type de pylône détecter (topic /aptere/approche/tp), le retour caméra qui sera affiché lors de la phase d'approche au-dessus du pylône (/zed_node/left_raw/image_raw_color/compressed).

Nous allons aussi envoyer des données (topic /envol) une string contenant 0 si nous voulons relancer l'analyse du type de pylône, 1 si nous voulons nous rapprocher pour relancer l'analyse, 2,B pour changer le type du pylône par B et 3 si nous confirmons.

Ainsi un script permettant d'établir la connexion grâce aux websocket nommée Websocket Jetson est présent. Nous avons deux scripts de création des différents publishers et subscribers.

L'implémentation de divers subscriber spécifique pour récupérer la pose (/localization_pose) du drone afin d'afficher la trajectoire effectuée, pour récupérer un point cloud et un dernier pour avoir le retour de la caméra ZED.

4/ Nuage de points

Pendant l'analyse du pylône électrique, nous recevons un nuage de point de l'environnement et du pylône.

En fonction de type de nuage reçu nous avons développé plusieurs solutions possibles de récupérations de ces données.

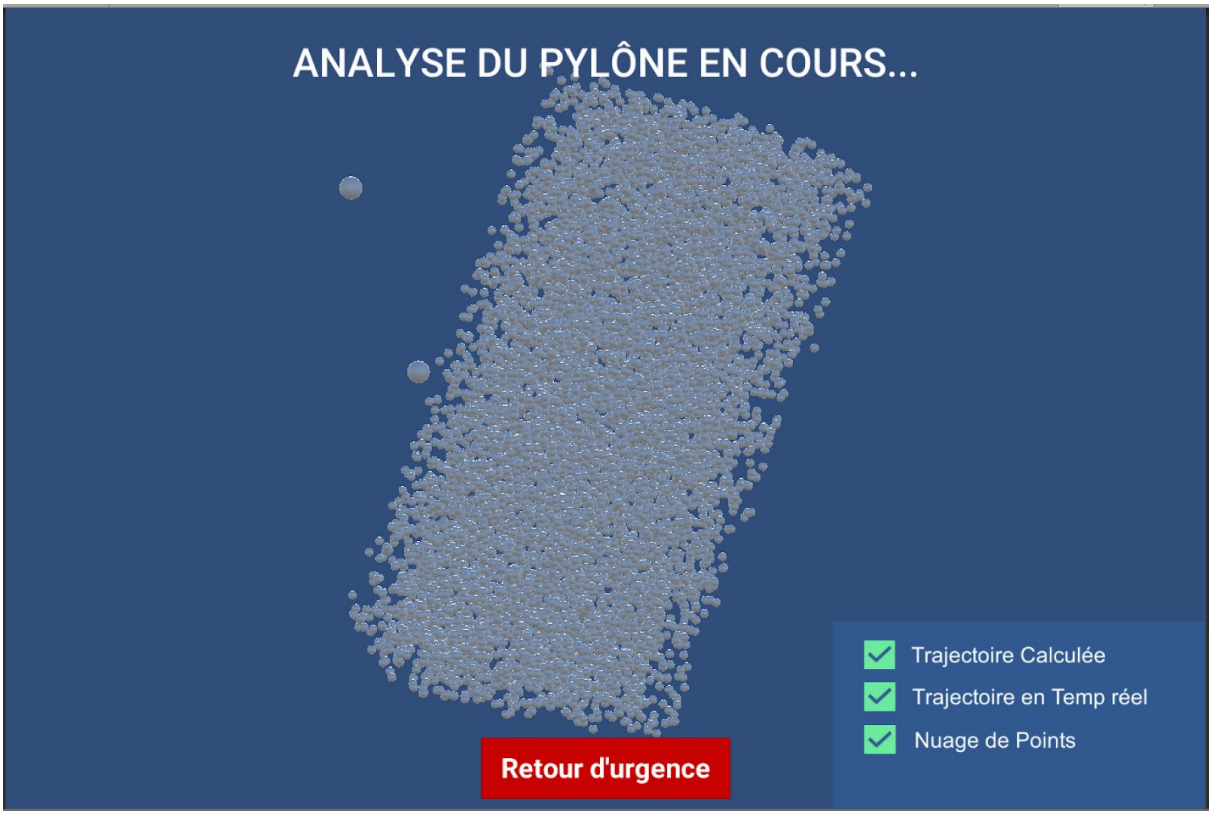
a) Réception des données grâce à la position de chaque point (en temps réel ou non)

Cette solution est utilisée dans le script nommé "NuageDePointV5.cs".
Chaque point est une sphère préconfigurée à l'avance dans un préfab.
Nous chargeons à l'avance tous ces points lors du lancement de l'application pour permettre de ne pas ralentir l'affichage du nuage de point lors de la phase d'analyse du pylône.

Lors du lancement de l'affichage du nuage de points, à chaque point que l'on reçoit on déplace une des sphères préchargées en tant qu'enfant du panel lié à l'affichage. On lui applique ainsi sa position en fonction des coordonnées que l'on reçoit ainsi qu'un matériel.

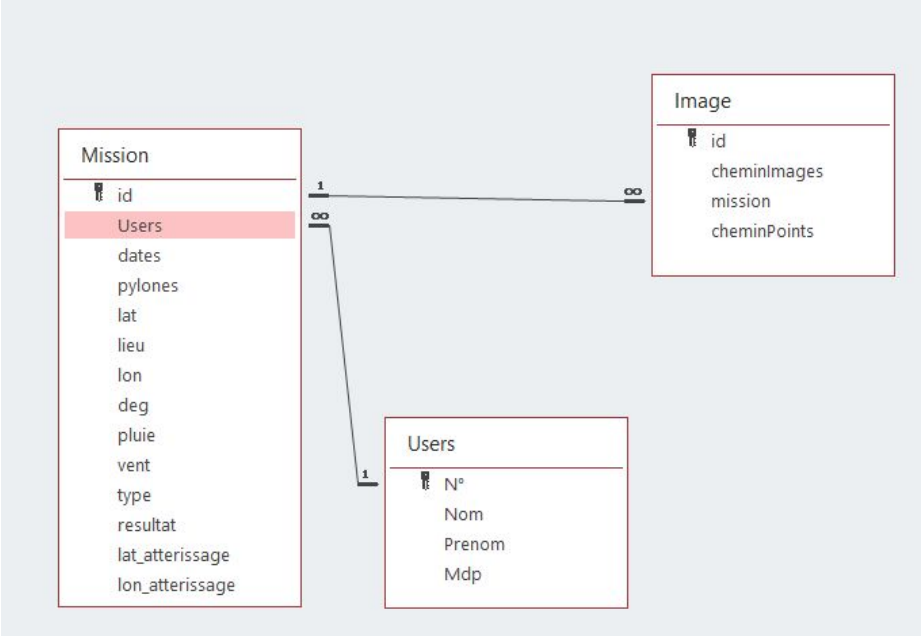
b) Réception des données à la fin sous forme de mech

Cette solution est utilisée dans le script "NuageDePointV2.cs".
Elle permet de récupérer un fichier .obj dans lequel les mesh ont été directement enregistrés, et sont alors générés grâce à l'asset "OBJ importer".
En cas d'envoi de fichier .ply l'asset PCX peut aussi être utilisé.



Capture d'écran du nuage de points ainsi que des points de passages lors de l'analyse du pylône (le nuage de points est simulé ici)

5/ Base de données



Louis Dhellemmes, Clément Guerin, Charles Hild-Lê
Élèves à l'UPSSTECH en dernière année de la formation SRI

Nom : naavi_pge.
Système de gestion : mySQL.
Langage informatique : C#, php.
Web Serveur : Xampp.

Tables :

mission :

Users : L'utilisateur qui a lancé la mission.
dates : La date de la mission.
pylone : Le type de pylône détecté.
lat : Latitude du pylône.
lon : Longitude du pylône.
lieu : Lieu de l'opération.
deg : Degré Celsius le jour de l'opération.
pluie : %Humidité le jour de l'opération.
vent : Km/h du vent le jour de l'opération.
type : condition météo
resultat : Si le vol c'est bien passé ou non.
lat_atterissage: Latitude zone d'atterrissage.
lon_atterissage: longitude zone d'atterrissage.

Code php :

nom	description	code associé
save.php	Sauvegarde le contenu de DBManager.cs dans la base de données	DBManager.cs
histo.php	Récupère les contenus de toutes les missions pour générer l'historique	historique.cs
login.php	Insert le mot de passe dans la table de hashage + salt et vérifie s'il correspond au hash enregistré dans la BDD et vérifie également les identifiants.	login.cs
register.php	Permet de créer un compte dans la base de données.	registration.cs

Code Unity :

nom	description	Panel associé
DBManager.cs	Stocke toutes les informations relatives à la mission	Projet Envol Scène Envol Mission panel
sendData.cs	Requête http pour stocker les données d'une mission.	Projet Envol Scène Envol Mission panel
historique.cs	Appelle login.php et génère-les buttons du menu historique.	Projet Envol Scène Envol History panel
login.cs	Appel le code login.php et change de scène si succès ou affiche une notification en cas d'erreur.	

PS. Il existe une petite interface pour la création de compte nommé DataBaseManagement.
Elle permet de créer un utilisateur en entrant son nom, prénom, identifiant et mdp.