

BTS CIEL SESSION 2025 - IR

E6 – PROJET TECHNIQUE

Dossier Personnel

Lycée : Touchard-Washington		Ville : LE MANS
Nom du projet :	Simulation de pilotage de drone	
N° du projet :	TW1	

Projet nouveau :	Oui <input checked="" type="checkbox"/> Non <input type="checkbox"/>	Projet interne :	Oui <input checked="" type="checkbox"/> Non <input type="checkbox"/>
Délai de réalisation :	150 heures	Statut des étudiants :	Formation initiale <input checked="" type="checkbox"/> Apprentissage <input type="checkbox"/>
Spécialité des étudiants :	ER <input type="checkbox"/> IR <input checked="" type="checkbox"/> Mixte <input type="checkbox"/>	Nombre d'étudiants :	4 étudiants
Professeurs responsables :	Didier BERNARD , Jilali KHAMLACH		



Simulation de Pilotage de Drone VR

ETU-3 GERARD Lenny

GERARD Lenny ETU-3	Simulation de pilotage de drone	1
--------------------	---------------------------------	---

Diagramme de cas d'utilisation “Lancer la simulation” :	3
Affichage de l'environnement 3D dans le casque VR :	5
Interaction avec la manette ou l'application mobile :	5
Tables utilisées pour le cas d'utilisation :	6
Choix du protocole de communication :	7
Diagramme de séquence du serveur	11
Diagramme de classe du serveur	12
Choix logiciel Environnement 3D	13
Brève explication des blueprints	14
Qu'est-ce qu'un Blueprint ?.....	14
Comment ça fonctionne ?.....	14
Types de Blueprint :.....	14
Plugins Unreal Engine nécessaire	16
Importation des modèles de drones sous Unreal Engine	17
Fiche de tests unitaire	18
Versionning	21
Apport personnel du projet	22
Difficultés rencontrées et solutions	23
Durant ce projet plusieurs problèmes ont été rencontrés	23
Axes d'améliorations	24

Diagramme de cas d'utilisation “Lancer la simulation” :

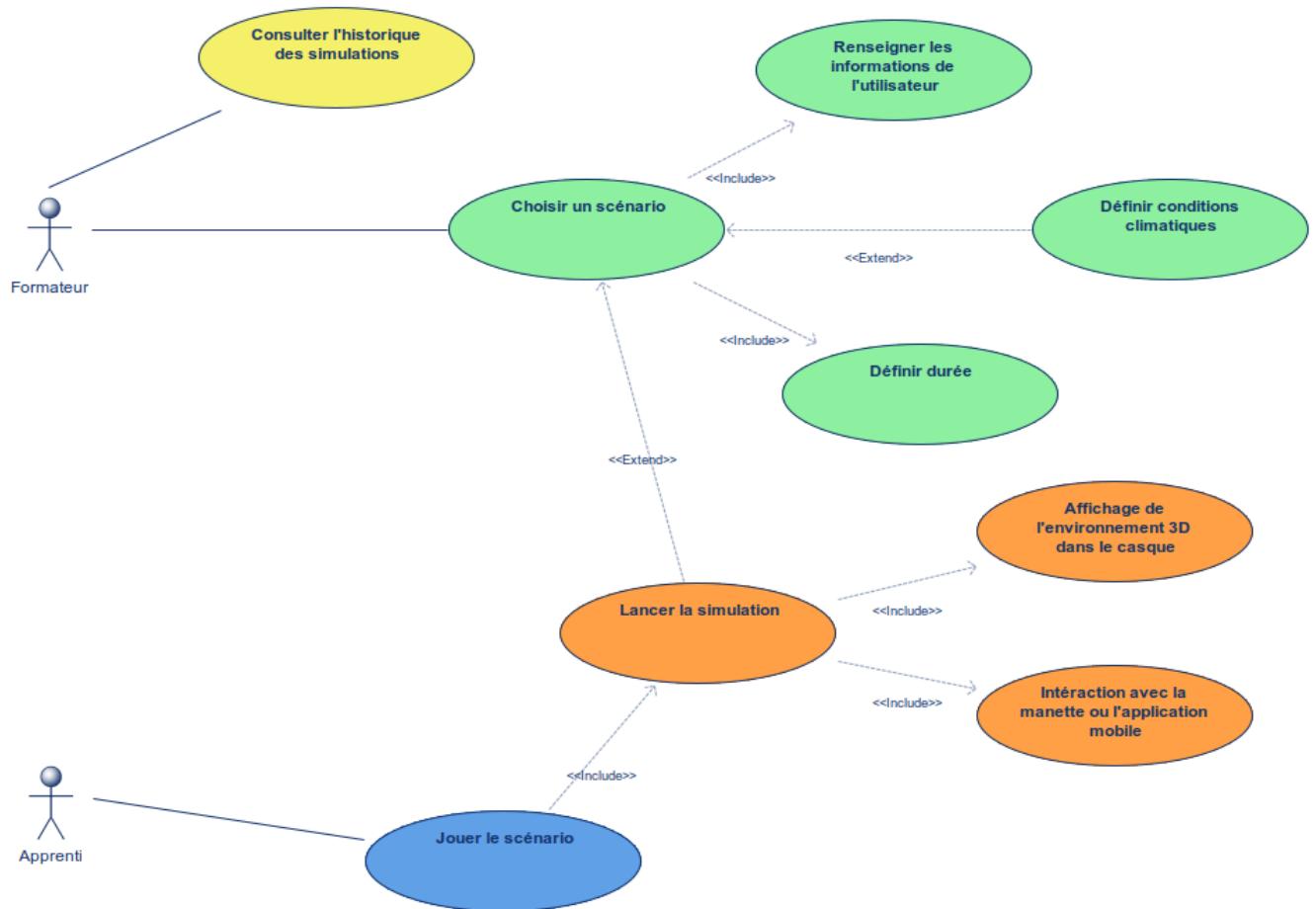


Figure 1 : Diagramme des cas d'utilisations

Voici la description écrite du cas d'utilisation “Lancer la simulation” :

Pré-conditions :	<ul style="list-style-type: none"> - Un scénario doit être sélectionné par l'utilisateur. - Les informations de l'apprenti doivent être renseignées. - Les paramètres tels que la durée de la simulation et les conditions climatiques doivent être définis.
Post-conditions :	<ul style="list-style-type: none"> - Le formateur à un retour du casque, il voit ce que l'apprenti fait. - Les objectifs accomplis sont envoyés à l'interface utilisateur pour savoir ce qui à été réussi ou non. - Le timer descend jusqu'à arriver à 0. - Toutes les données (objectifs réussi, durée...) sont écrites dans la base de données pour garder une trace de la simulation.
Scénario principal :	<ul style="list-style-type: none"> - Placement du pilote au point de départ - Tous les paramètres sont pris en compte (pluie, vent, scénario...) - L'environnement se lance dans le casque - Les objectifs sont consultés pour être pris en compte - Déclenchement d'un timer - L'apprenti peut faire bouger le drone grâce aux commandes qui sont récupérés - Les objectifs peuvent se valider automatiquement - Le formateur peut valider les objectifs manuellement
Alternatives :	<ul style="list-style-type: none"> - <u>Paramètres incomplets</u> : Si des paramètres manquent (ex. durée non renseignée), le système affiche un message d'erreur demandant à l'utilisateur de compléter les champs requis. - <u>Erreur système</u> : Si un problème technique survient (ex. VR non connectée), le système affiche un message d'erreur technique. - <u>Panne de matériel</u> : Si le matériel (casque VR, capteurs, etc.) est déconnecté ou dysfonctionnel, le système arrête le processus et envoie une alerte pour demander une intervention technique.

Affichage de l'environnement 3D dans le casque VR :

- Pré-conditions	<ul style="list-style-type: none"> - La simulation doit être lancée - La connexion entre la radiocommande et le casque doit être établie - La connexion entre l'ordinateur et le casque est établie
- Scénario principal	<ul style="list-style-type: none"> - L'apprenti voit l'environnement dans le casque - Le formateur voit l'environnement sur l'ordinateur
- Post-conditions	<ul style="list-style-type: none"> - L'apprenti peut voir l'environnement 3D dans le casque
- Post-condition échec	<ul style="list-style-type: none"> - L'environnement n'est pas chargé dans le casque

Interaction avec la manette ou l'application mobile :

- Pré-conditions	<ul style="list-style-type: none"> - La simulation doit être lancée - La connexion entre la radiocommande et le casque doit être établie - La connexion entre l'ordinateur et le casque est établie
- Scénario principale	<ul style="list-style-type: none"> - L'apprenti peut manipuler la manette et la radiocommande ce qui fait bouger le drone dans l'environnement 3D
- Post-conditions	<ul style="list-style-type: none"> - L'apprenti peut faire bouger le drone avec la manette ou l'application mobile
- Post-condition échec	<ul style="list-style-type: none"> - L'apprenti bouge la manette mais le drone ne bouge pas

Tables utilisées pour le cas d'utilisation :

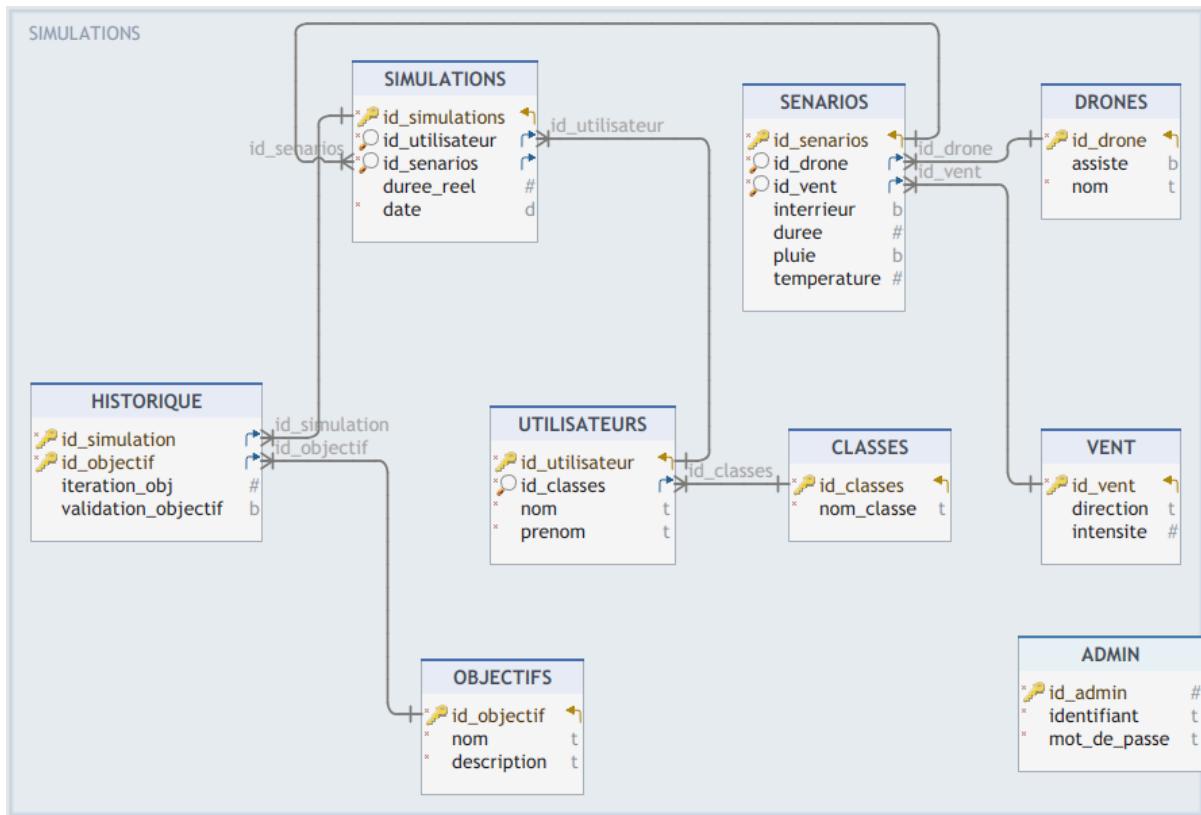


Figure 2 : Base de données du projet

Pour réaliser ce cas d'utilisation nous allons avoir besoin des tables OBJECTIFS et HISTORIQUE. Ces deux tables vont nous permettre d'enregistrer les objectifs qui ont été réussi pendant la simulation et ceux qui ont été échoués. La table OBJECTIFS permet de définir les objectifs, et la table HISTORIQUE va permettre de définir un nombre de fois auquel l'objectif doit être fait pour qu'il soit réussi pendant la simulation (ex: passer dans un cerceau, grâce à la table HISTORIQUE cet objectif devient passer dans 3 cerceaux.)

Choix du protocole de communication :

Nous avons fait le choix de coder le serveur en C++, sur le framework QT Creator. Nous avons choisi ce langage de programmation puisque c'est un langage que nous avons appris pendant les 2 années de BTS. De plus, nous avons choisi de le faire sur le framework QT Creator, car nous avons eu l'habitude de l'utiliser durant le BTS également.



Figure 3 : Logo Framework QT



Figure 4 : Logo langage C++

Nous avons fait le choix de communiquer en requête HTTP, et en TCPsocket, entre la manette et le casque VR.



Figure 5 : Logo protocole HTTP



Figure 6 : Logo protocole TCP/IP

Nous avons fait ces choix puisque au début nous avions réalisé un serveur HTTP pour communiquer grâce aux requêtes GET, sauf que Unreal Engine n'a pas de plugin gratuit pour recevoir ce type de message et une contrainte de notre projet est que les logiciels, les services et les outils doivent être gratuit. Donc pour ne pas impacter le travail fait pour la communication de l'application mobile, nous avons réalisé un serveur qui reçoit des requêtes HTTP (GET) et qui les renvoie en requêtes TCP sockets au programme du casque. Puisque les TCP sockets sont le seul type de sockets qui ont un plugin gratuit sous Unreal Engine.

Voici l'explication des axes utilisés par la radiocommande et l'application mobile, la radiocommande et l'application mobile fonctionnent dans le mode 2 de la figure 7. Ensuite il y a les valeurs qui permettent au drone de bouger dans l'environnement virtuel :

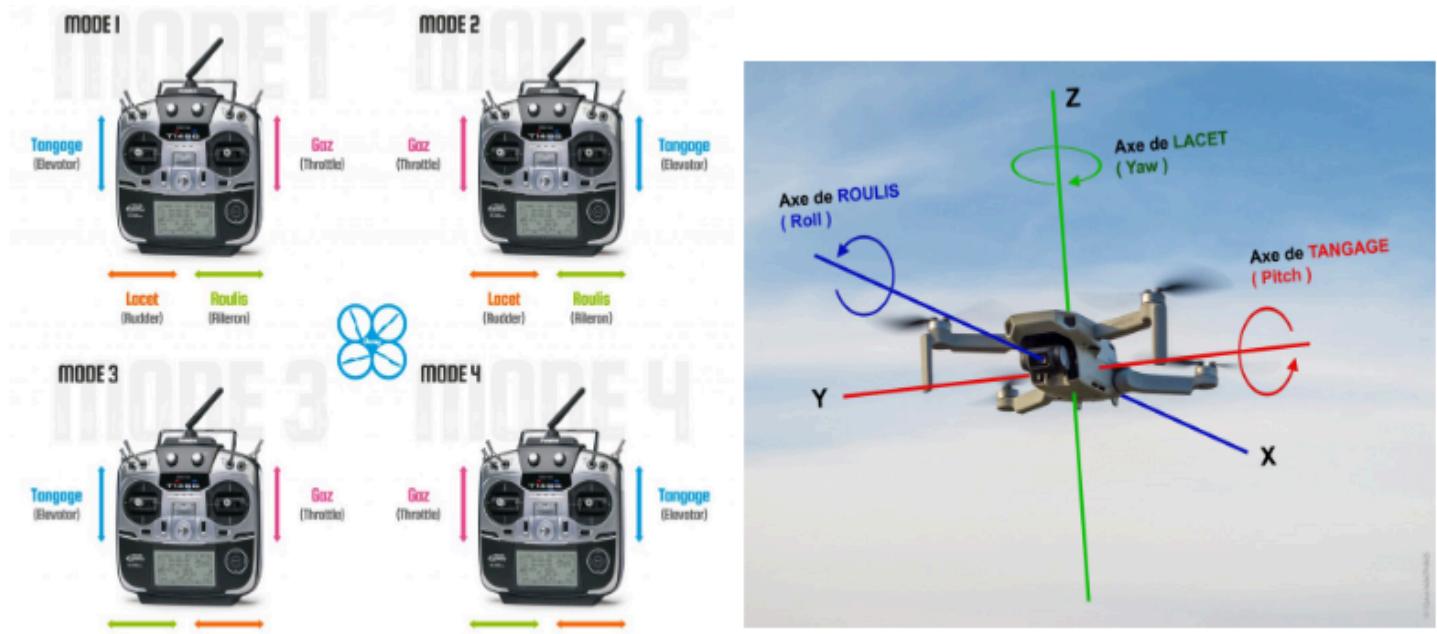


Figure 7 : Configuration radiocommande et description des axes

Axes	Valeurs application mobile	Valeurs radiocommande
Gaz	gaz:"0 à 100"	gaz:"-54 à 82"
Lacet	lacet:"-100 à 100"	lacet:"-56 à 67 "
Tangage	tangage:"-100 à 105"	tangage:"-97 à 64"
Roulis	roulis:"-100 à 100"	roulis:"-60 à 63"

En revanche ces valeurs étant trop grandes pour Unreal Engine, nous devons diviser les valeurs pour rendre la simulation du vol de drone plus réaliste.

Pour la radiocommande et l'application mobile, nous divisons les valeurs de gaz par 50, et les valeurs de lacet, tangage, roulis par 300.

Voici le synoptique du protocole de communication entre la radiocommande ou l'application mobile et le casque VR :

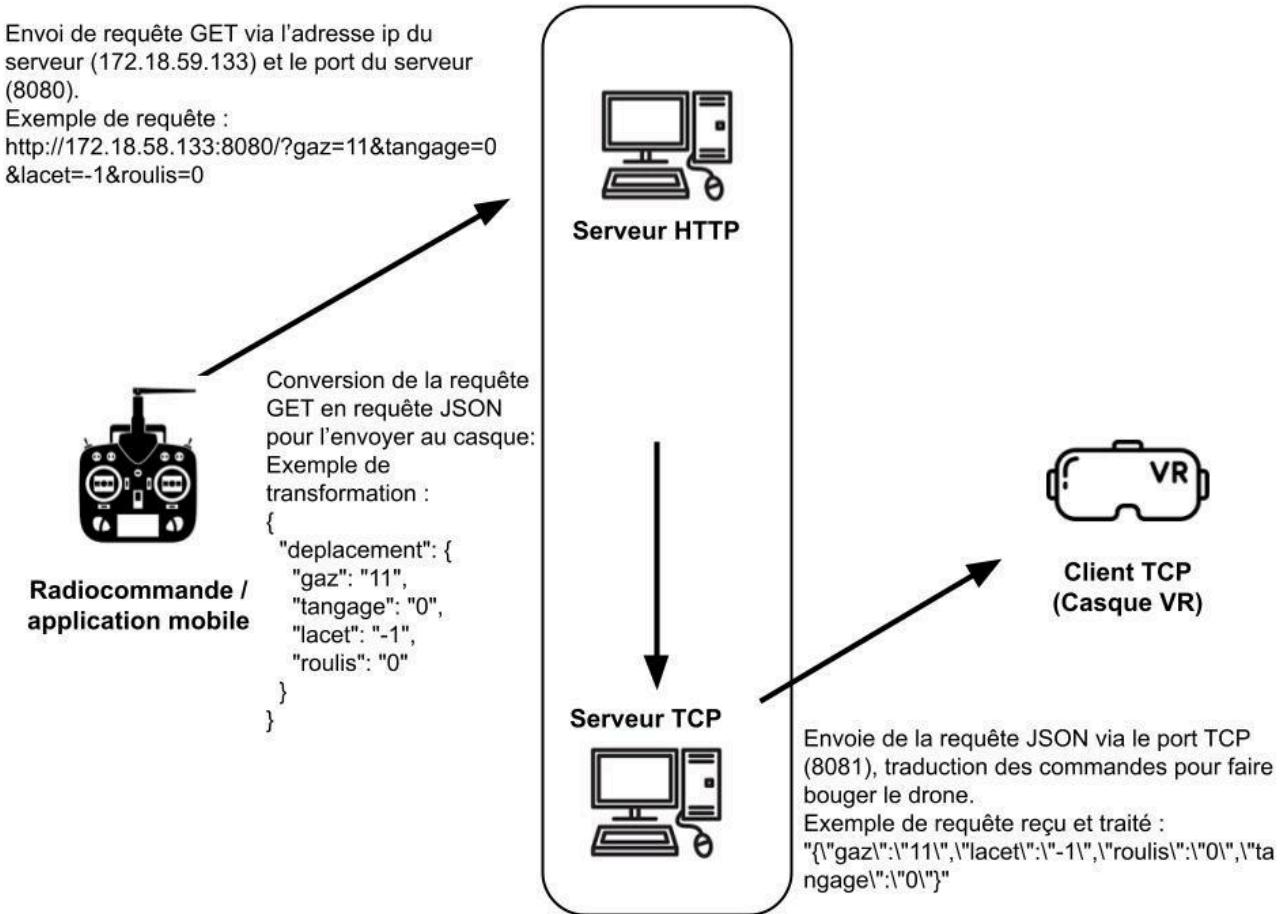


Figure 8 : Synoptique protocole de communication radiocommande, casque VR

Dans ce synoptique nous voyons la radiocommande ou l'application mobile, ensuite nous voyons le serveur HTTP, et le serveur TCP qui sont sur la même machine et donc sur le même programme. J'ai fait le choix de mettre deux machines pour le serveur dans le synoptique afin d'améliorer la lisibilité et la compréhension de celui-ci. Enfin nous voyons le client TCP, donc dans notre cas, le casque VR.

Nous voyons donc comment transitent les requêtes, tout d'abord le client web donc la radiocommande ou l'application mobile envoie une requête HTTP, au serveur HTTP via l'adresse ip et le port du serveur, donc dans notre cas sur l'adresse ip 172.18.59.133 et sur le port 8080. Ensuite le serveur traduit la requête en objet JSON et la transmet au serveur TCP. Enfin quand le client TCP est connecté au port ainsi qu'à l'adresse ip du serveur TCP mais également qu'il est prêt à recevoir alors le serveur TCP, envoie sur le port 8081. Dans notre cas, la requête JSON comme nous pouvons la voir dans l'exemple du synoptique.

Diagramme de séquence du serveur

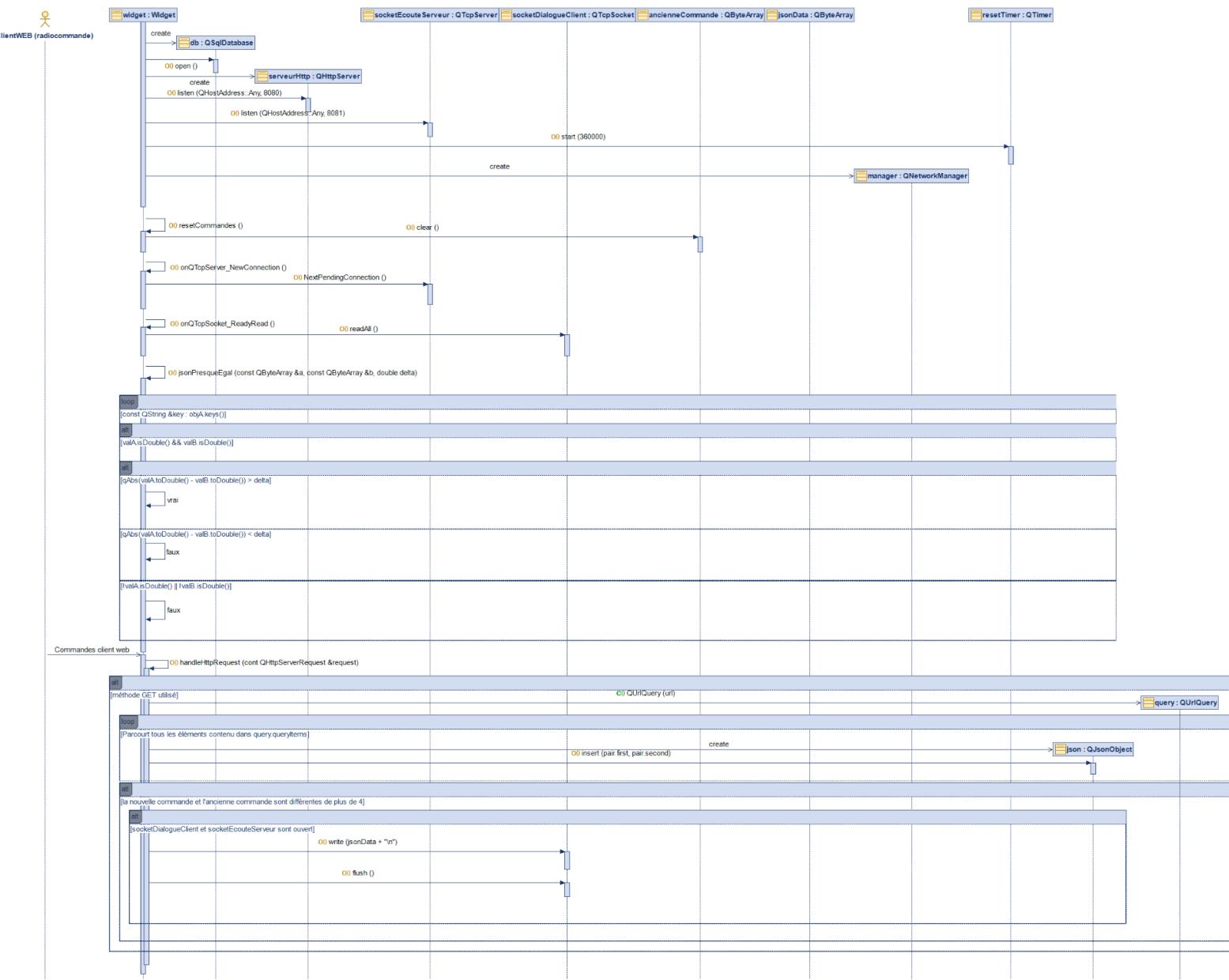


Figure 9 : Diagramme de séquence serveur HTTP/TCP

Voici une partie du diagramme de séquence du serveur, dans ce diagramme nous pouvons voir en partie les connexions des serveurs HTTP, et TCP. Ensuite nous pouvons voir la fonction qui va permettre de ne pas envoyer trop de fois les mêmes commandes, si les commandes sont différentes de moins de 4, alors elles ne sont pas envoyées. Et enfin nous pouvons voir la gestion de la réception des commandes venant d'un client WEB, et l'envoi au client TCP.

Diagramme de classe du serveur :

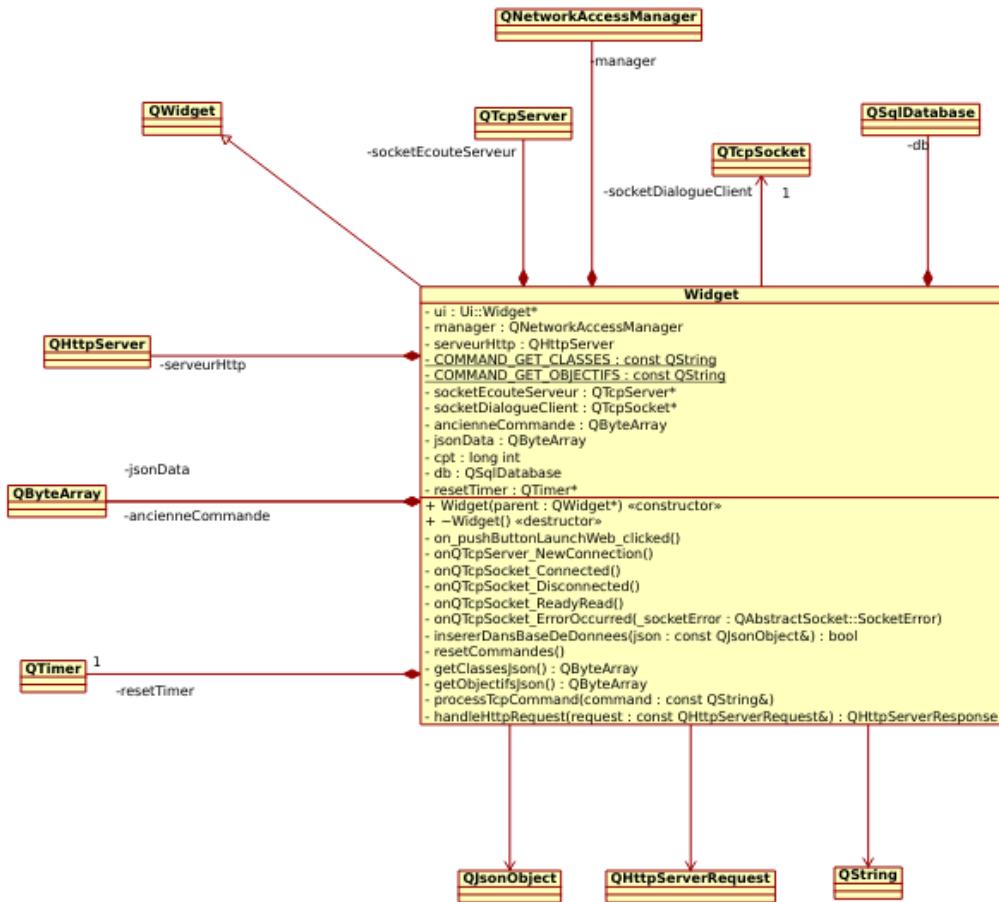


Figure 10 : Diagramme de classe serveur HTTP/TCP

Voilà le diagramme de classe du serveur, dans ce diagramme nous pouvons voir chaque méthodes utiles au serveur. Mais surtout toutes les relations avec les classes du framework QT (QTimer, QTcpserver, QTcpsocket...).

Choix logiciel Environnement 3D :

Caractéristiques	Unity	Unreal Engine
Langage de programmation	C#	C++
Graphismes	Moins réaliste	Graphismes AAA, rendu photoréaliste
Plateformes supportés	Très large (PC, mobile, console..)	Principalement PC et consoles haut de gamme
Performances	Pas très optimisé pour les gros projets	Excellent gestion de projets complexes
Support des casques VR	Supporte le Oculus Quest 2	Supporte le Oculus Quest 2
Blueprint	Pas de blueprint	Blueprint intégré
Idéal pour	Jeux mobiles, 2D, projets multiplateformes	Jeux 3D AAA, expériences réalistes, cinématiques et simulations

Dans notre cas, le choix se portera sur Unreal Engine. Premièrement, du fait que Unreal Engine se code avec du C++, un langage qui a été appris pendant le BTS alors que Unity se code avec du C# qui n'a pas été appris durant le BTS. Deuxièmement, malgré beaucoup de qualités communes, Unity n'a pas de blueprint intégrés, il faut les installer grâce à des plugins. Et troisièmement, Unreal Engine est idéal pour faire des projets de simulations alors que Unity n'est pas fait en premier lieu pour cela et notre projet consiste à faire une simulation donc Unreal Engine est plus approprié.

Brève explication des blueprints :

Qu'est-ce qu'un Blueprint ?

Le système de script visuel Blueprint dans Unreal Engine est un langage de programmation visuel qui utilise une interface basée sur les nœuds pour créer des éléments de jeu.

Comment ça fonctionne ?

Un Blueprint est comme un script visuel. Dans un blueprint il y a :

- Des nœuds (Nodes) : Ce sont des blocs d'actions (ex : déplacer un objet, jouer un son...).
- Des fils/lignes : Ils relient les nœuds entre eux. Ils montrent l'ordre d'exécution ou les valeurs transmises.

Il y a deux types de fils :

- Fils blancs (ligne de vie) → le flux d'exécution (dans quel ordre les choses se passent).
- Fils colorés (bleu, vert, rouge...) → le flux de données (valeurs comme un nombre, une position, un texte...).

Types de Blueprint :

Il existe plusieurs types, mais les plus fréquents sont :

- Blueprint d'Acteur (Actor Blueprint) : pour créer des objets interactifs (ennemis, portes, objets...).
- Blueprint de Niveau (Level Blueprint) : pour gérer des événements liés aux niveaux (exemple : un joueur entre dans une zone).
- Blueprint de Classe de Personnage (Character / Pawn) : pour créer et gérer un personnage.

Voici un exemple de Blueprint :

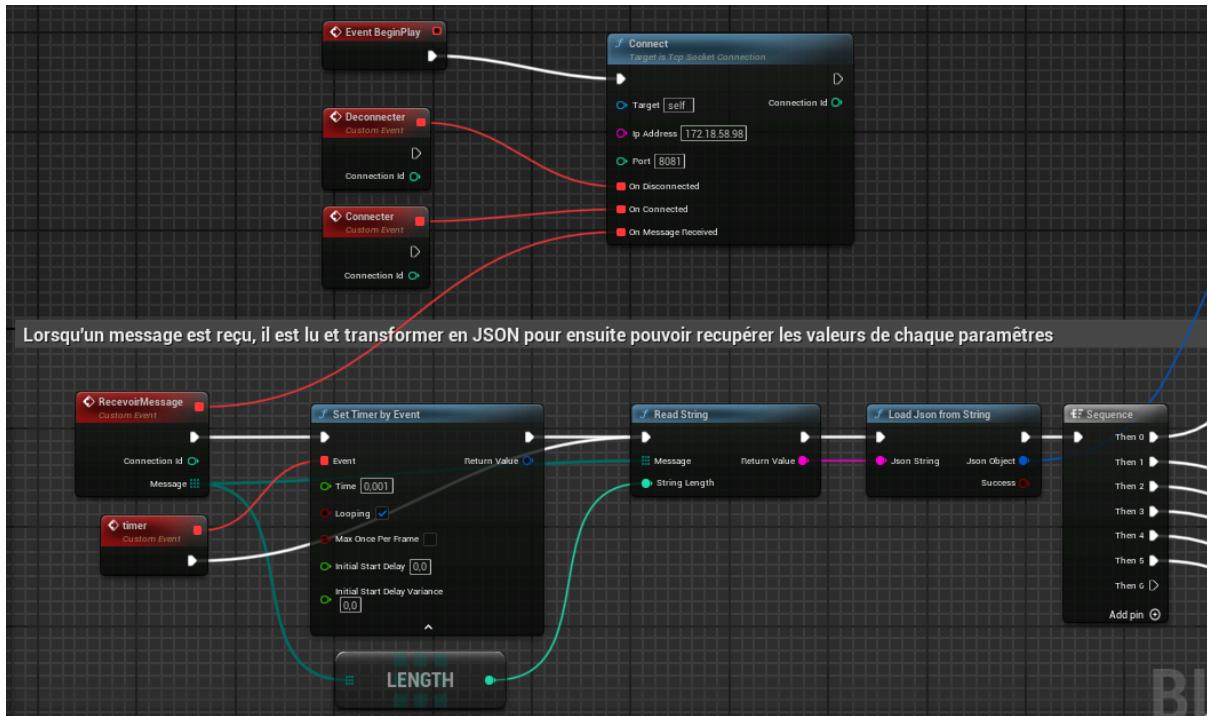


Figure 11 : Exemple de Blueprint Unreal Engine

Dans ce blueprint on peut voir la ligne de vie donc le fil blanc (qui donne l'ordre du déroulement du code), il y a les nodes (blocs d'actions qui font les actions exemple : connect), et enfin les fils de couleurs (ces fils servent à faire circuler les valeurs exemple : json object).

Plugins Unreal Engine nécessaire :

Il y a 4 plugins qui sont nécessaires pour ce projet sur Unreal Engine, tout d'abord il y a les plugins nécessaires au fonctionnement du casque. C'est-à-dire les plugins qui vont permettre de lancer notre projet dans le casque VR, ces 2 plugins sont :



Figure 12 : Image plugin MetaXR



Figure 13 : Image plugin Meta XR Platform

Ensuite pour la connexion au serveur 2 autres plugins sont nécessaires ces plugins sont :



Figure 14 : Image plugin VaRest

Le plugin VaRest, ce plugin va servir à faire des requêtes GET ou POST, il va également permettre d'envoyer ou recevoir des données en JSON, donc de les encoder ou les décoder...

Et enfin le dernier plugin nécessaire est le plugin TCPsockets, ce plugin va permettre la connexion au serveur TCP, via le port et l'adresse ip du serveur. Celui-ci va également permettre de gérer les événements venant du serveur, donc par exemple les messages reçus, la connexion ou la déconnexion...



Figure 15 : Image plugin TCPsockets

Importation des modèles de drones sous Unreal Engine :



Figure 16 : Image Drone DJI Tello

Dans Unreal Engine il n'y a pas de modèle de drone directement, il faut en importer.
Pour importer le modèle de drone, donc dans notre cas le drone DJI Tello, il faut avoir un fichier avec une de ses extensions :

- .FBX
- .OBJ
- .GLTF
- .GLB
- .DAE
- .ABC

Il n'y a que ces extensions qui sont supportées sur Unreal Engine.

Fiche de tests unitaire

Voici la fiche de tests unitaire qui teste la réception des commandes pour faire bouger le drone dans l'environnement virtuel.

Dans le cas de notre fiche de test unitaire nous allons utiliser Unreal Engine 5.5.4 pour le programme du casque donc le programme qui va traiter les commandes reçus, et QT 6.8.2 et QT Creator 4.8.1 pour le serveur HTTP / TCPSockets qui va renvoyer les commande reçus via la radiocommande ou l'application mobile au programme du casque.

Fiche de tests unitaire			
Nature	Fonctionnel	Référence	TW1 - ET3 - 001
Module	Interaction avec la manette ou l'application mobile	Auteur	GERARD Lenny
Date de création / mise à jour		4 mars 2025	
Objectifs		Tester que la réception des commandes fonctionne correctement	
Condition du test			
État initial du module		Environnement du test	
Ordinateur	PC	Android (Casque VR) et linux	
Programme	BP_TCP	Unreal Engine 5.5.4, QT 6.8.2 et QT Creator 4.8.1	
Base de données (pas utile pour cette méthode)			
Procédure de test			
Lancement de l'application		Le formateur lance la simulation via Unreal Engine	
Repère	Opérations	Résultats attendus	
1	L'environnement se charge pour être affiché dans l'environnement virtuel	L'apprenti voit l'environnement virtuel dans le casque avec les objectifs et les paramètres définis par le formateur choisi par le formateur	
2	Dans le terminal on lance la commande curl "172.18.59.133:8080/?gaz=0&lacet=0&roulis=0&tangage=0"	Le drone ne bouge pas car le gaz est à 0, le champ lacet est à 0 aussi, le champ roulis est à 0, et enfin le champ tangage est à 0 également.	

3	Dans le terminal on lance la commande curl "172.18.59.133:8080/?gaz=100&lacet=0&roulis=0&tangage=0"	Le drone monte de 100% car le champ gaz est à 100, le champ lacet gauche est à 0, le champ roulis est à 0, et enfin le champ tangage est à 0.
4	Dans le terminal on lance la commande curl "172.18.58.133:8080/?gaz=0&lacet=100&roulis=0&tangage=0"	Le drone tourne à fond sur lui-même vers la droite car le champ gaz est à 0, le champ lacet est à 100, le champ roulis est à 0, et enfin le champ tangage est à 0.
5	Dans le terminal on lance la commande curl "172.18.59.133:8080/?gaz=0&lacet=0&roulis=100&tangage=0"	Le drone vers la droite à fond car le champ gaz est à 0, le champ lacet est à 0, le champ roulis est à 100, et enfin le champ tangage est à 0.
6	Dans le terminal on lance la commande curl "172.18.59.133:8080/?gaz=0&lacet=0&roulis=0&tangage=100"	Le drone tourne sur lui-même vers l'avant à fond car le champ gaz est à 0, le champ lacet est à 0, le champ roulis est à 0, et enfin le champ tangage est à 100.
7	Dans le terminal on lance la commande curl "172.18.59.133:8080/?gaz=100&lacet=0&roulis=100&tangage=0"	Le drone monte et tourne vers la droite car le champ gaz est à 100, le champ lacet gauche est à 0, le champ roulis est à 100, et enfin le champ tangage est à 0.
8	Dans le terminal on lance la commande curl "172.18.59.133:8080/?gaz=0&roulis=0&tangage=100"	Si la commande est incomplète alors tous les paramètres avec une valeur sont pris en compte et les autres gardent leurs anciennes valeurs

Bilan sur la planification personnelle

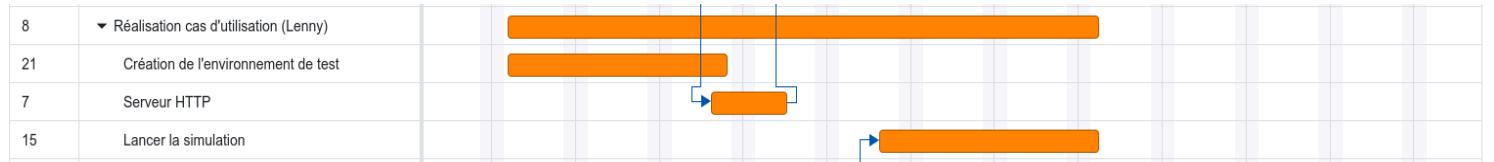


Figure 17 : Extrait du diagramme de Gantt

Durant le projet nous avions réalisé une première planification mais je n'ai pas pu suivre celle-ci correctement.

Au début nous voulions faire un serveur HTTP, mais nous nous sommes rendus compte qu'il n'y avait pas de plugin gratuit sur Unreal Engine pour communiquer correctement en HTTP sur chaque plugin il manquait des fonctionnalités. Ainsi, nous avons dû modifier le serveur ce qui nous a fait prendre du retard sur la mise en place du serveur de communication.

En revanche, à part ce problème, aucunes autres tâches n'a pris du retard puisque dans la prévision nous avions essayé de laisser plus de temps pour les tâches avec de la compréhension d'Unreal Engine.

Versionning :

Toute ma partie de projet à été versionné grâce à l'outil Github, voici quelques exemples de ce versionnage :

Ciel2_CPP / servweb /		
		Add file ...
 Igerard1	serveur fonctionnel	fb146ee · 2 months ago
Name	Last commit message	Last commit date
..		
main.cpp	Add files via upload	3 months ago
servweb.pro	Add files via upload	3 months ago
servweb.pro.user	serveur fonctionnel	2 months ago
servweb.pro.user.1abee8c	Add files via upload	3 months ago
widget.cpp	serveur fonctionnel	2 months ago
widget.h	serveur fonctionnel	2 months ago
widget.ui	serveur fonctionnel	2 months ago

Figure 18 : Extrait github

Ciel2_CPP / servwebFin /		
		Add file ...
 Igerard1	Add files via upload	3940d0d · 2 weeks ago
Name	Last commit message	Last commit date
..		
main.cpp	Add files via upload	3 weeks ago
servweb.pro	Add files via upload	3 weeks ago
servweb.pro.user	Add files via upload	2 weeks ago
servweb.pro.user.b44ea29	Add files via upload	3 weeks ago
widget.cpp	Add files via upload	2 weeks ago
widget.h	Add files via upload	2 weeks ago
widget.ui	Add files via upload	3 weeks ago

Figure 19 : Extrait github

Apport personnel du projet

Ce projet m'a permis d'apprendre beaucoup de notions.

Dans un premier temps le projet m'a permis de m'instruire sur la formation BIMD que je ne connaissais pas du tout avant. Et que j'ai pu suivre pour apprendre toutes les caractéristiques d'un drone, y compris les spécificités d'un vol de drone.



Figure 20 : Logo formation BIMD

Ensuite durant ce projet, il a fallu que j'apprenne comment fonctionne Unreal Engine et plus particulièrement les Blueprints via Unreal Engine. En effet, il a fallu que je cherche quels protocoles de communications pouvaient être utilisés avec Unreal Engine mais surtout quels protocoles peuvent être utilisés grâce à des plugins gratuits puisque c'était l'une des contraintes du projet.

De plus ce projet m'a permis de faire un serveur qui reçoit des messages via un protocole mais qui traduit ces mêmes messages dans un autre protocole (c'est à dire le serveur qui reçoit des messages dans le protocole HTTP mais qui les renvoie au programme du casque en TCPsocket)

Enfin ce projet m'a permis de découvrir le casque de réalité virtuelle, auparavant je n'avais jamais essayé.



Figure 21 : Casque Oculus Quest 2

GERARD Lenny ETU-3	Simulation de pilotage de drone	22
--------------------	---------------------------------	----

Difficultés rencontrées et solutions :

Durant ce projet plusieurs problèmes ont été rencontrés :

- la prise en main d'Unreal Engine et de ses Blueprints

Pour ce problème j'ai suivi des formations en ligne pour apprendre à prendre en main ce logiciel, ces fonctionnalités et du coup forcément les blueprints, car c'est une possibilité de programmer que je ne connaissais pas du tout avant de débuter le projet.



Figure 22 : Logo Unreal Engine

- Protocole de communication entre le casque, l'application mobile et la radiocommande

Il fallait trouver un protocole de communication qui convient à la radiocommande, l'application mobile et le casque, tout en trouvant des plugins Unreal Engine gratuits car c'est une contrainte du projet.

La solution que nous avons trouvé pour ce problème a été de faire un serveur qui reçoit des messages venant de la radiocommande et de l'application mobile en requêtes GET donc via le protocole HTTP, et qui renvoie en protocole TCP au casque.

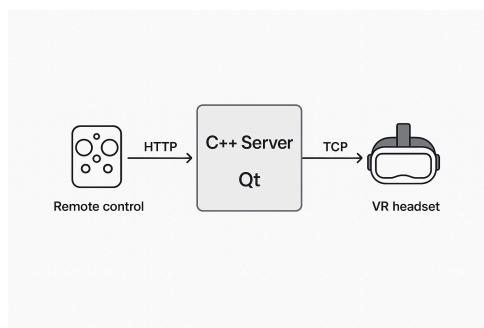


Figure 23 : Schéma communication

Axes d'améliorations :

Pour finir, dans ce projet il y a plusieurs axes d'améliorations qui pourraient voir le jour. Tout d'abord, la première amélioration pourrait être de modéliser les hélices du drone, pour que la simulation soit encore plus réaliste.



Figure 23 : Schéma modélisation hélices

Ensuite, il pourrait y avoir un retour vidéo du drone sur l'application mobile, pour que quand le drone soit trop éloigné l'apprenti puisse toujours avoir un œil sur le drone.



Figure 24 : Schéma retour vidéo sur application mobile

Il y a de nombreux autres points d'améliorations sur ce projet. Toutefois, ces deux points sont les plus importants pour améliorer le réalisme de chaque simulation.