

# **Développement d'une Intelligence artificielle en utilisant le module ML-Agents sur Unity3D**

Rapport de stage 2<sup>ème</sup> Année

présenté par

Chaves D'Carvalho Matheus



Entreprise :

Sigma Clermont

Tuteur d'entreprise :

Christophe BASCOUL

Tuteur SIGMA Clermont :

Christophe BASCOUL

Date : Aout 2020

Référence du fichier : ChavesDCarvalho\_ Matheus\_RapportStage2A.pdf

**SIGMA Clermont**

Campus de Clermont-Ferrand – CS 20265 – 63178 Aubière– France  
Tél. +33 (0)4 73 28 80 00 – Fax +33 (0)4 73 28 81 00 –<http://www.sigma-clermont.fr/>



## FICHE D'IDENTIFICATION DU DOCUMENT

### Rapport de Stage 2<sup>ème</sup> Année

**TITRE DU DOCUMENT :** Développement d'une Intelligence artificielle en utilisant le module ML-Agents sur Unity3D  
Development of Artificial Intelligence using the ML-Agents module on Unity3D

**AUTEUR(S) :** Chaves D'Carvalho Matheus – Mécanique Avancée - Pôle MMS

Date du document	Nbre de pages	Référence du document
12/11/2020	62	ChavesDCarvalho_ Matheus_RapportStage2A.pdf

**ABSTRACT:**

Mainly aiming to guide future professors and students that have interest in this area, this document presents in detail the development of an Artificial Intelligence capable of piloting a drone to a position marked with 3 coordinates (x, y, z). Two different control approaches are developed in this project, in the first one there is an independent stability control system and in the second one the stability control must be done by the artificial intelligence. After extensive testing, it is clear that the first approach is more satisfying for the objective of reaching the destination without a collision.

**Keywords:** artificial intelligence, unity3d, ml-agents, drone

**RESUME :**

En portant un objectif plutôt académique de orientation aux futures professeurs et étudiants qui portent un intérêt pour ce thème, ce document présente en détails le développement d'une Intelligence Artificielle capable de piloter un drone jusqu'à une position marqué avec 3 coordonnées (x,y,z). Deux abordages de contrôles différents sont développés dans ce projet, dans le premier il y a un système de contrôle de stabilité indépendant et dans la deuxième le contrôle de stabilité doit être fait par l'intelligence artificielle. Après nombreux tests, il est clair que le premier abordage est plus satisfaisant à l'objectif d'arriver au destin sans une collision.

**Mots clés :** intelligence artificielle, unity3d, ml-agents, drone

Visa de l'entreprise :

Ce rapport de stage a été visé par (Nom, Prénom, Qualité) : .....

Date :

Signature :

Je demande que ce rapport soit confidentiel (non diffusé sur l'intranet de l'école) : ☐ OUI ☐ NON

# SOMMAIRE

SOMMAIRE .....	5
<b>CHAPITRE 1. INTRODUCTION .....</b>	<b>5</b>
1.1. Livraison avec Drones.....	5
1.2. ML-Agents .....	5
1.3. Unity3D.....	6
1.4. Objectif.....	6
<b>CHAPITRE 2. SOFTWARES .....</b>	<b>7</b>
2.1. Unity3D.....	7
2.2. Python 3.8.5 .....	9
2.3. ML-Agents .....	12
2.4. Nvidia CUDA Toolkit.....	16
2.5. Nvidia cuDNN .....	17
<b>CHAPITRE 3. ENVIRONNEMENT .....</b>	<b>19</b>
3.1. Des structures.....	19
3.2. La destination.....	22
3.3. Barrières .....	25
<b>CHAPITRE 4. DRONE .....</b>	<b>27</b>
4.1. Modèle 3D .....	27
4.2. Capteurs .....	28
4.3. Agent.....	30
4.3.1 Méthodes importantes .....	33
<b>CHAPITRE 5. HYPERPARAMÈTRES .....</b>	<b>35</b>
5.1. Trainer.....	36
5.2. Batch size .....	36
5.3. Beta .....	37

5.4. Buffer Size .....	37
5.5. Epsilon .....	38
5.6. Lambd .....	38
5.7. Max Steps.....	39
5.9. Checkpoint Interval.....	39
5.10. Keep Checkpoints .....	39
5.11. Num Epoch .....	40
5.12. Normalize.....	40
5.13. Summary Freq.....	40
5.14 Time Horizon .....	40
<b>CHAPITRE 6. CONTRÔLE.....</b>	<b>42</b>
6.1. Contrôle avec stabilité créée en avance .....	42
6.1.1. Récompenses.....	42
6.1.2. Les tests.....	43
6.2. Contrôle sans stabilité créée en avance.....	47
6.2.1. Récompenses.....	52
6.2.2. Les tests.....	53
<b>CHAPITRE 7. CONCLUSION .....</b>	<b>58</b>
<b>BIBLIOGRAPHIE.....</b>	<b>60</b>

# CHAPITRE 1. INTRODUCTION

## 1.1. Livraison avec Drones

Les drones sont l'une des technologies logistiques les plus étudiées ces dernières années. Ils combinent des caractéristiques technologiques correspondant aux tendances actuelles de l'industrie des transports et de la société comme l'autonomie, la flexibilité et l'agilité. Parmi les différents concepts d'utilisation des drones en logistique, la livraison de colis est l'un des scénarios d'application les plus populaires. Des entreprises comme Amazon testent les drones en particulier pour la livraison sur le dernier kilomètre, dans le but de réduire à la fois le coût total et d'augmenter la satisfaction des clients grâce à des livraisons rapides. Les drones étant des véhicules électriques, ils sont également souvent considérés comme un moyen de transport respectueux de l'environnement.

Un problème grave rencontré actuellement est l'autonomie. Comme un des records d'autonomie d'un drone, il est possible de citer le Drone US-1 fabriqué par un ancien ingénieur de Tesla, il est capable de voler pendant plus de 70 minutes avec des charges utiles jusqu'à 2,8 kg. Dans ce scénario, une vraie préoccupation est la réduction maximum de caméras et capteurs que demandent une grande quantité d'énergie.

Dans ce projet, il est présenté une méthode de faire l'interaction du drone avec l'extérieur sans utiliser une caméra ou des capteurs qui utilisent beaucoup de batterie.

## 1.2. ML-Agents

La boîte à outils Unity Machine Learning Agents (ML-Agents) est un projet open source qui permet aux jeux et aux simulations de servir d'environnements pour la formation d'agents intelligents. Les agents peuvent être formés à l'aide de l'apprentissage par renforcement, de l'apprentissage par imitation ou d'autres méthodes d'apprentissage automatique via une API Python simple à utiliser.

Avec une communauté croissante et vraiment active, cette boîte à outils est un moyen puissant d'enseigner un agent intelligent. Mais surtout, est un moyen beaucoup agréable de familiariser nouveaux étudiants avec l'intelligence artificielle.

En tirant parti de la facilité d'utilisation et de la puissance de cet outil, dans ce projet un agent intelligent a été enseigné par la méthode d'apprentissage par renforcement dans un environnement créé dans l'Unity3D.

### **1.3. Unity3D**

Unity3D est un puissant moteur 3D multiplateforme et un environnement de développement convivial. Assez facile pour le débutant et assez puissant pour l'expert, Unity devrait intéresser tous ceux qui souhaitent créer facilement des jeux et des applications 3D pour mobile, ordinateur de bureau, Web et consoles.

Unity offre aux utilisateurs la possibilité de créer des jeux en 2D et 3D, il prend en charge les API suivantes : Direct3D sur Windows et Xbox 360 ; OpenGL sur MacOS et Linux ; OpenGL ES sur Android et iOS ; WebGL sur Internet. Les programmeurs utilisent majoritairement C# comme langage de code dans la plateforme.

Dans les jeux 2D, Unity permet l'importation de « sprites » et d'un moteur de rendu de monde 2D avancé. Pour les jeux 3D, Unity permet de spécifier la compression de texture et les paramètres de résolution pour chaque plateforme prise en charge par le moteur de jeu, et prend en charge le mappage de relief, le mappage de réflexion, le mappage de parallaxe, l'occlusion de l'environnement spatial de ombres d'écran (SSAO) en utilisant des shadow maps, des effets de post-traitement pour la texture et le rendu plein écran.

En tirant parti de la facilité d'utilisation et de la puissance de cet outil, dans ce projet un environnement 3D a été créé pour simuler une structure urbaine.

### **1.4. Objectif**

Ce travail a comme objectif enseigner une intelligence artificielle à contrôler un drone dans un environnement 3D qui simule une structure urbaine. Une coordonnée avec trois axes (x,y,z) est donnée au programme et il doit être capable de contrôler le drone jusqu'à la position désirée sans quelques collisions.

Un autre objectif souhaité est de possibilitiser l'utilisation future de la démarche dans ce travail comme forme d'implémentation d'une classe ou travail pratique dans le thème d'Intelligence Artificielle. Bien comme familiariser l'auteur avec les différents défis pour l'application de la technologie d'intelligence artificielle.



## CHAPITRE 2. SOFTWARES

Comme un des objectifs souhaités est de permettre une utilisation future de ce travail comme une orientation dans la création d'un support de cours, il est important d'avoir l'existence d'un chapitre dont l'objectif est de montrer chaque logiciel utilisé et comment faire l'installation correctement. La mauvaise installation de ces programmes peut créer un problème de compatibilité dans le moment d'exécution.

### 2.1. Unity3D

Dans le site officiel, il y a une option « get started », où vous pouvez choisir l'option « individual » et après « student » et après s'inscrire.

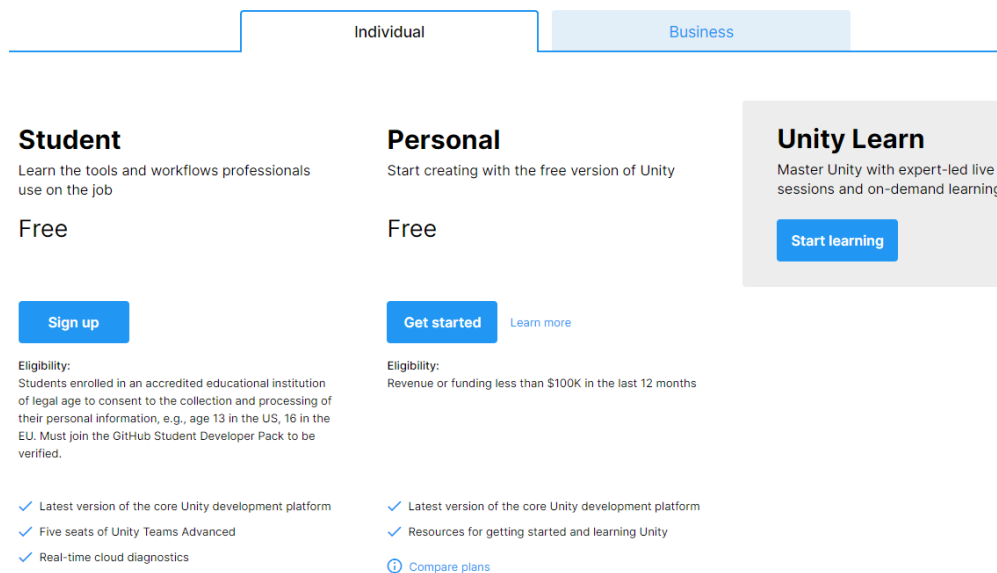


Figure 1 : Options pour télécharger l'Unity Hub.

Cela étant, vous pouvez télécharger le Unity Hub. Il s'agit d'une interface où vous pouvez accéder tous les versions d'unity installés et télécharger des versions nouvelles ou anciennes.

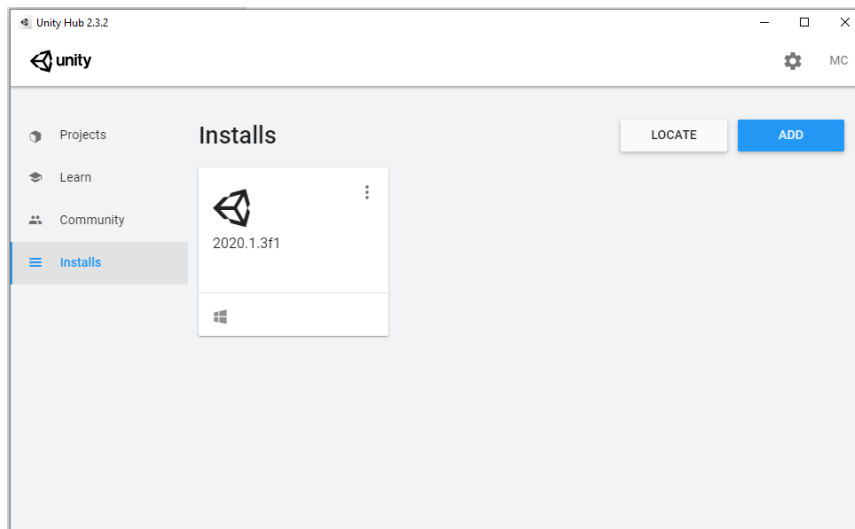


Figure 2 : L'onglet ou vous pouvez choisir une version d'Unity pour télécharger.

Dans l'onglet « Projects », vous pouvez créer un nouveau projet dans le bouton « NEW » ou ouvrir un projet qui déjà existe dans la liste ou dans votre ordinateur avec le bouton « ADD ».

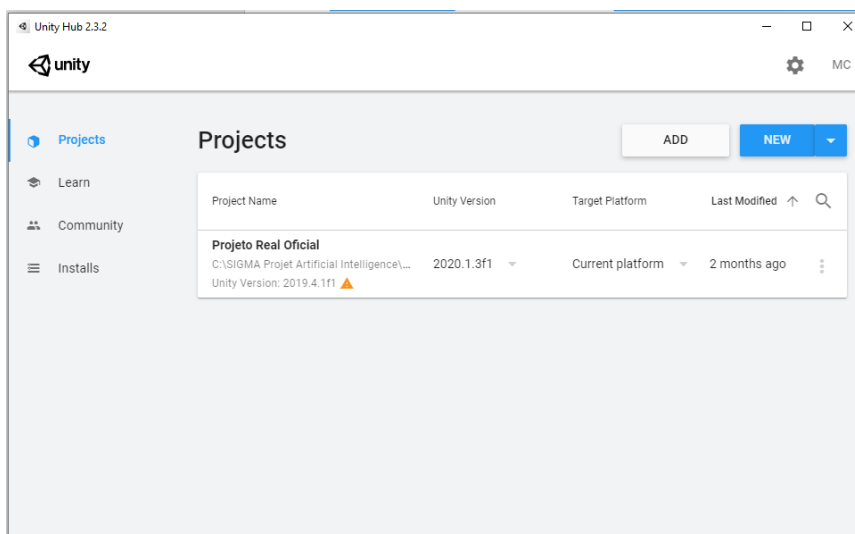


Figure 3 : L'onglet avec liste de projets.

## 2.2. Python 3.8.5

Après l'installation d'unity, il est important d'installer Python pour pouvoir utiliser l'API responsable pour enseigner notre agent intelligent.

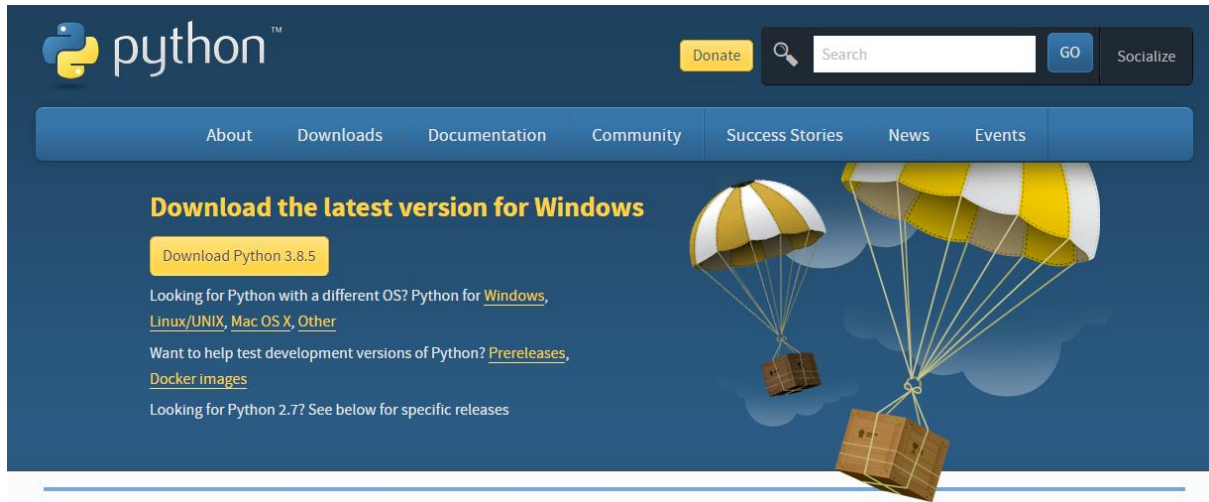


Figure 4 : Exemple de bouton ou vous ne devez pas télécharger python.

Dans la première page du site il y a une option de téléchargement. Ne cliquez pas dessus. Dans cette option vous allez télécharger une version 32 bit et normalement vous avez besoin d'une version 64 bit. Vous pouvez aller dans la partie « downloads » et télécharger la version 64 bit.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
<a href="#">Gzipped source tarball</a>	Source release		e2f52bcf531c8cc94732c0b6ff933ff0	24149103	<a href="#">SIG</a>
<a href="#">XZ compressed source tarball</a>	Source release		35b5a3d0254c1c59be9736373d429db7	18019640	<a href="#">SIG</a>
<a href="#">macOS 64-bit installer</a>	Mac OS X	for OS X 10.9 and later	2f8a736eeb307a27f1998cfd07f22440	30238024	<a href="#">SIG</a>
<a href="#">Windows help file</a>	Windows		3079d9cf19ac09d7b3e5eb3fb05581c4	8528031	<a href="#">SIG</a>
<a href="#">Windows x86-64 embeddable zip file</a>	Windows	for AMD64/EM64T/x64	73bd7aab047b81f83e473efb5d5652a0	8168581	<a href="#">SIG</a>
<a href="#">Windows x86-64 executable installer</a>	Windows	for AMD64/EM64T/x64	0ba2e9ca29b719da6e0b81f7f33f08f6	27864320	<a href="#">SIG</a>
<a href="#">Windows x86-64 web-based installer</a>	Windows	for AMD64/EM64T/x64	eeab52a08398a009c90189248ff43dac	1364128	<a href="#">SIG</a>
<a href="#">Windows x86 embeddable zip file</a>	Windows		bc354669bffd81a4ca14f06817222e50	7305731	<a href="#">SIG</a>
<a href="#">Windows x86 executable installer</a>	Windows		959873b37b74c1508428596b7f9df151	26777232	<a href="#">SIG</a>
<a href="#">Windows x86 web-based installer</a>	Windows		c813e6671f33a269e669d913b1f9b0d	1328184	<a href="#">SIG</a>

Figure 5 : Version 64 bit du Python.

Pendant l'installation, il est important de faire attention à l'installation du pip3. Normalement, l'installation va comprendre le pip3, mais si vous avez choisi l'option personnalisable, vous devez être sûr d'installer le pip3.

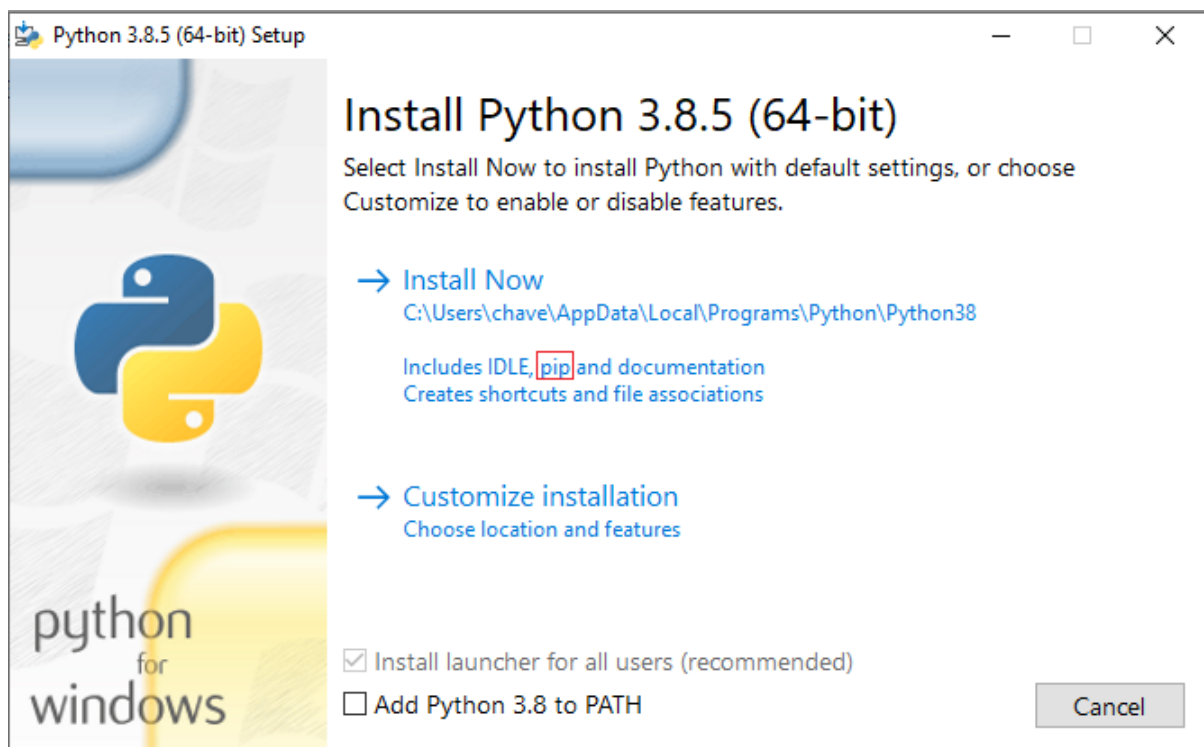


Figure 6 : PIP est normalement inclus dans l'installation.

Il faut faire attention à la dernière option aussi. Elle vient décochée, mais il est important de la cocher. Avec cette option, il est possible d'utiliser les commandes de notre API dans une fenêtre de commande ouverte n'importe le fichier.



Figure 7 : Option que vous permettez d'utiliser les commandes python dans un fichier différent.

À la fin, normalement il y a une option pour augmenter la limite de caractères. Il est fortement recommandé de l'accepter.

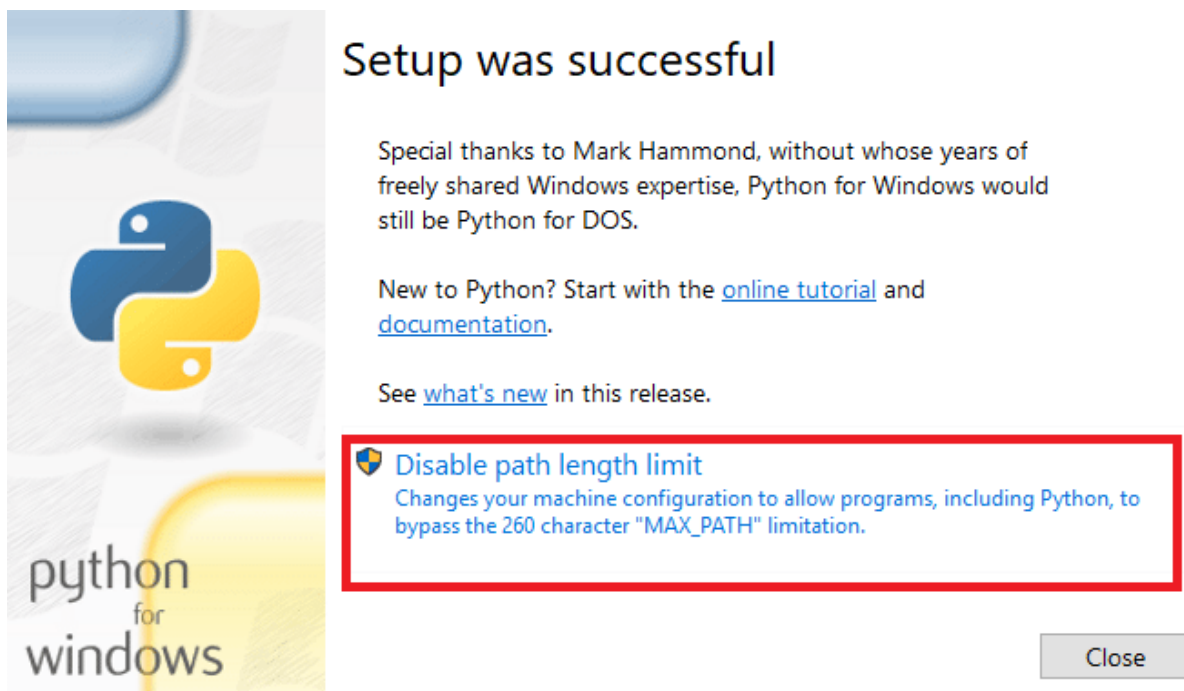


Figure 8 : Option pour désactiver la limite de caractères dans le nom d'un fichier.

Une observation, il n'est pas nécessaire d'installer Anaconda pur utiliser cette API.

## 2.3. ML-Agents

Depuis que l'installation d'unity et Python sont faites, il est temps d'installer la boîte aux outils ML-Agents.

Vous devez, en première moment, télécharger ou cloner le github de ML-Agents. Il faut juste accéder au site, aller dans l'option vert « Code » et cliquer dans l'option désiré. Il est recommandé de juste télécharger un ZIP.

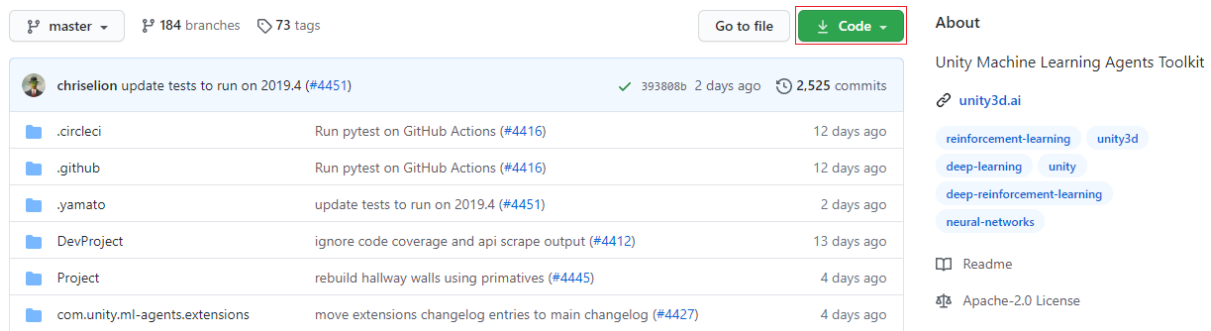


Figure 9 : Bouton pour télécharger le fichier ML-Agents.

Cela étant, vous devez ouvrir votre projet dans l'unity en cliquant dans le projet désiré (voir figure 3). Pour installer la boîte aux outils, il faut aller à l'option « Window » et « Package Manager ».

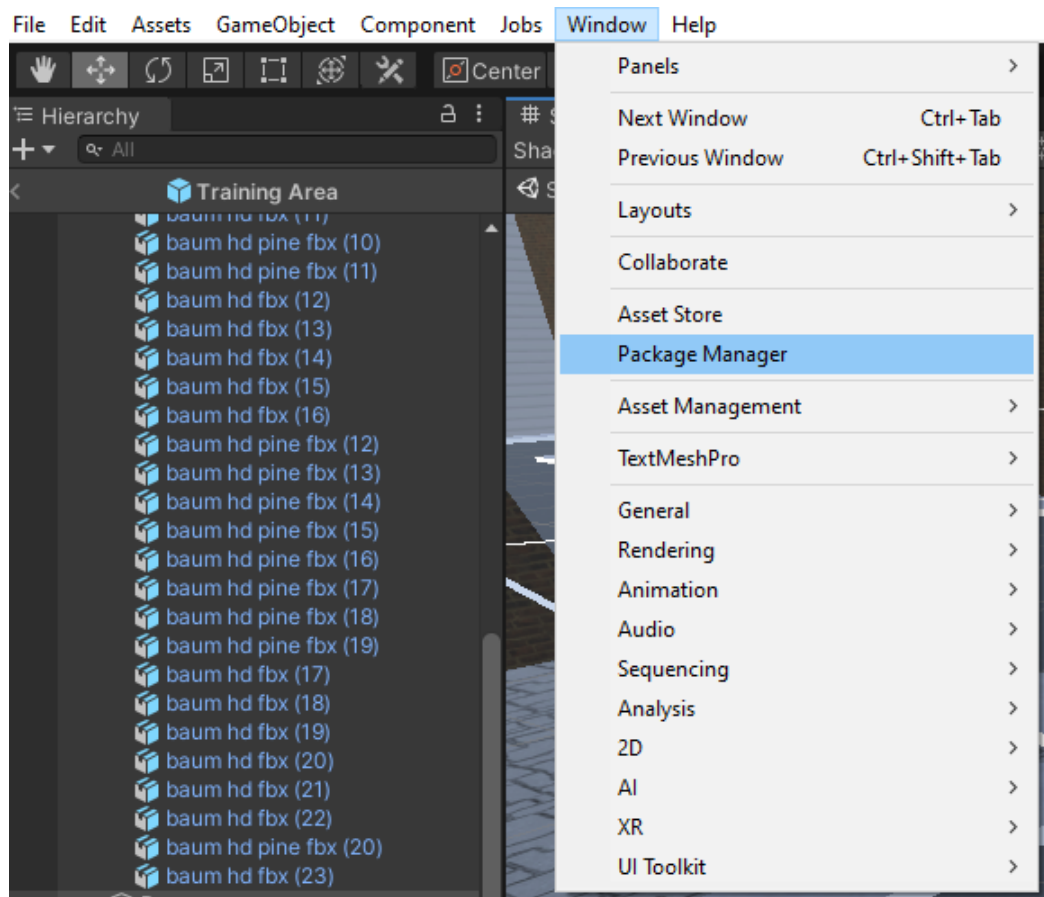


Figure 10 : Window --> Package Manager.

Dans le bouton « + », vous pouvez ajouter d'un fichier trouvé dans l'ordinateur.

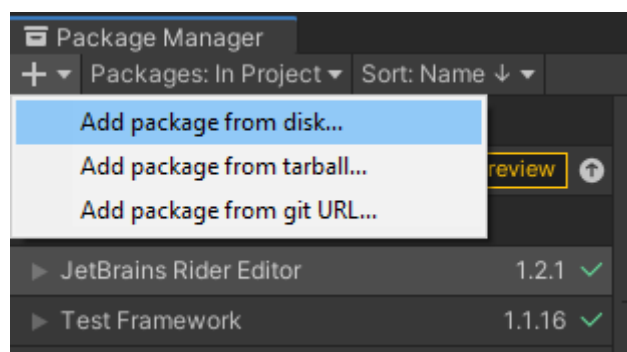


Figure 11 : Dans le bouton + vous pouvez ajouter un fichier trouvé dans l'ordinateur.

Trouvé dans le fichier « ml-agents-master/com.unity.ml-agents », il y a l'archive « package.json ». Sélectionnez-lui.

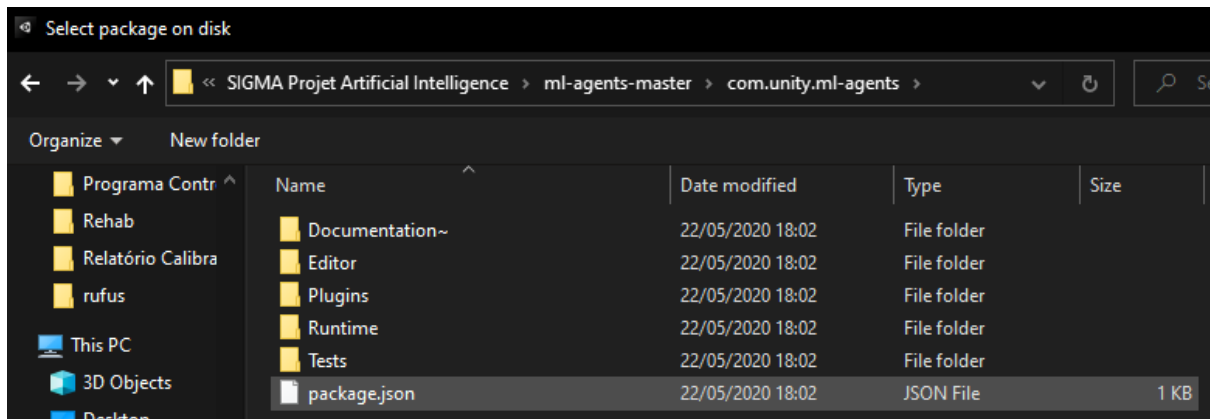


Figure 12 : Il faut ajouter l'archive .json pour importer ML-Agents.

Pour la partie Unity, il est prêt. Maintenant il reste installer ml-agents au python. Si vous utilisez Windows, vous devez maintenir le bouton SHIFT enfoncé tout en faisant un clic droit dans le « Desktop ». Sélectionnez l'option pour ouvrir le PowerShell.

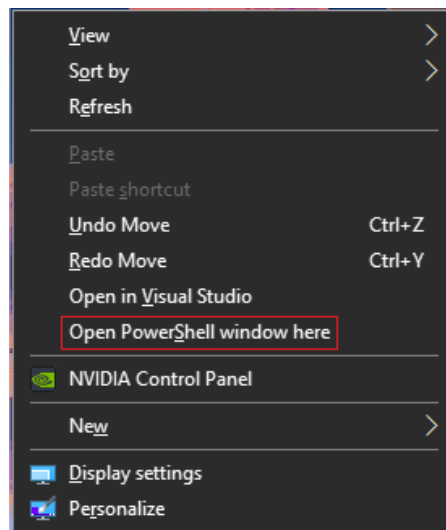


Figure 13 : Instruction pour ouvrir la fenêtre de commandes.

Cela étant, vous devez écrire la commande « `pip3 install mlagents` » dans la fenêtre ouverte.



```
Windows PowerShell
PS C:\Users\chave\Desktop> pip3 install mlagents
Collecting mlagents
  Downloading mlagents-0.19.0-py3-none-any.whl (141 kB)
    | 141 kB 2.2 MB/s
Collecting h5py>=2.9.0
  Downloading h5py-2.10.0-cp38-cp38-win_amd64.whl (2.5 MB)
    | 2.5 MB 939 kB/s
Collecting mlagents-envs==0.19.0
  Downloading mlagents_envs-0.19.0-py3-none-any.whl (68 kB)
    | 68 kB 1.5 MB/s
Collecting pyyaml>=3.1.0
  Downloading PyYAML-5.3.1-cp38-cp38-win_amd64.whl (219 kB)
    | 219 kB 2.2 MB/s
Collecting protobuf>=3.6
  Downloading protobuf-3.13.0-py2.py3-none-any.whl (438 kB)
    | 438 kB 2.2 MB/s
Collecting Pillow>=4.2.1
  Downloading Pillow-7.2.0-cp38-cp38-win_amd64.whl (2.1 MB)
    | 2.1 MB 726 kB/s
```

Figure 14 : Instruction pour installer le ML-Agents.

Probablement, vous avez reçu un message d'erreur en disant que la version du « numpy » est incompatible.

```
Collecting pyasn1<0.5.0,>=0.4.6
  Downloading pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
    | 77 kB 5.5 MB/s
Using legacy setup.py install for wrapt, since package 'wheel' is not installed.
Using legacy setup.py install for termcolor, since package 'wheel' is not installed.
ERROR: mlagents-envs 0.19.0 has requirement numpy<1.19.0,>=1.14.1, but you'll have numpy 1.19.1 which is incompatible.
ERROR: tensorflow 2.3.0 has requirement numpy<1.19.0,>=1.16.0, but you'll have numpy 1.19.1 which is incompatible.
```

Figure 15 : Message d'erreur en disant que la version du "numpy" est incompatible.

Vous pouvez installer une version plus ancienne pour éviter le problème de compatibilité. Ecrivez le commande « `pip install numpy==1.18.5` » dans la même fenêtre.

```
PS C:\Users\chave\Desktop> pip install numpy==1.18.5
Collecting numpy==1.18.5
  Downloading numpy-1.18.5-cp38-cp38-win_amd64.whl (12.8 MB)
    | 12.8 MB 6.8 MB/s
Installing collected packages: numpy
Attempting uninstall: numpy
```

Figure 16 : Instruction pour installer la version correcte.

Normalement, vous devez être prêt pour commencer à utiliser le ML-Agents avec la CPU.

Si vous souhaitez utiliser la GPU pour faire les calculs, il faut installer le Nvidia CUDA toolkit et cuDNN.

## 2.4. Nvidia CUDA Toolkit

Vous devez accéder au site Nvidia Developer et chercher l'archive de versions du CUDA. Il faut télécharger la version 10.1, les nouvelles versions ne sont pas compatibles avec le ML-Agents.

### CUDA Toolkit Archive

Previous releases of the CUDA Toolkit, GPU Computing SDK, documentation below, and be sure to check [www.nvidia.com/drivers](http://www.nvidia.com/drivers) for more recent products.

[Download Latest CUDA Toolkit](#)

#### Latest Release

[CUDA Toolkit 11.0 Update1](#) (Aug 2020), [Versioned Online Documentation](#)

#### Archived Releases

[CUDA Toolkit 11.0](#) (May 2020), [Versioned Online Documentation](#)

[CUDA Toolkit 10.2](#) (Nov 2019), [Versioned Online Documentation](#)

[CUDA Toolkit 10.1 update2](#) (Aug 2019), [Versioned Online Documentation](#)

[CUDA Toolkit 10.1 update1](#) (May 2019), [Versioned Online Documentation](#)

[CUDA Toolkit 10.1](#) (Feb 2019), [Online Documentation](#)

[CUDA Toolkit 10.0](#) (Sept 2018), [Online Documentation](#)

[CUDA Toolkit 9.2](#) (May 2018), [Online Documentation](#)

[CUDA Toolkit 9.1](#) (Dec 2017), [Online Documentation](#)

*Figure 17 : Version compatible du CUDA Toolkit.*

Il faut installer tous les composants.

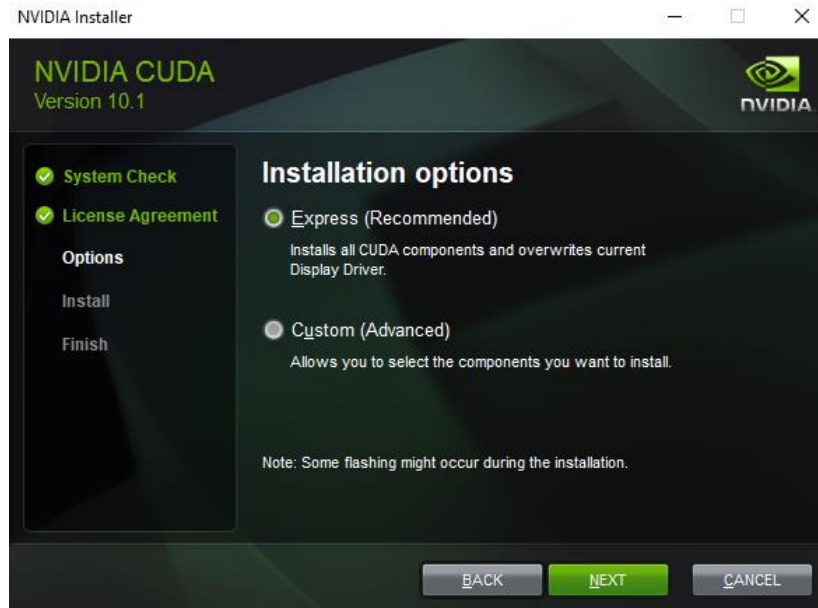


Figure 18 : Installation du NVIDIA CUDA Toolkit.

## 2.5. Nvidia cuDNN

Il faut aussi installer le cuDNN. Le problème est que pour le télécharger il faut avoir un compte développeur NVIDIA. C'est simple à créer et quelqu'un peut le faire, mais ils peuvent vous demander plus d'un jour pour avoir la vérification de données.

Attention, il faut télécharger la version compatible avec le CUDA 10.1 et avec Windows, c'est-à-dire pour l'instant la version 7.6.5.

Il s'agit d'un package avec plusieurs DLLs, il faut l'ouvrir et coller dans le dossier où CUDA 10.1 est installé. Normalement il est dans C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1

## cuDNN Archive

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

[Download cuDNN v8.0.2 \(July 24th, 2020\), for CUDA 11.0](#)

[Download cuDNN v8.0.2 \(July 24th, 2020\), for CUDA 10.2](#)

[Download cuDNN v8.0.2 \(July 24th, 2020\), for CUDA 10.1](#)

[Download cuDNN v8.0.1 RC2 \(June 26th, 2020\), for CUDA 11.0](#)

[Download cuDNN v8.0.1 RC2 \(June 26th, 2020\), for CUDA 10.2](#)

[Download cuDNN v7.6.5 \(November 18th, 2019\), for CUDA 10.2](#)

[Download cuDNN v7.6.5 \(November 5th, 2019\), for CUDA 10.1](#)

### Library for Windows, Mac, Linux, Ubuntu and RedHat/Centos(x86\_64architecture)

[cuDNN Library for Windows 7](#)

[cuDNN Library for Windows 10](#)

[cuDNN Library for Linux](#)

[cuDNN Library for OSX](#)

[cuDNN Runtime Library for Ubuntu18.04 \(Deb\)](#)

[cuDNN Developer Library for Ubuntu18.04 \(Deb\)](#)

[cuDNN Code Samples and User Guide for Ubuntu18.04 \(Deb\)](#)

*Figure 19 : Liste de versions disponibles du NVIDIA cuDNN.*

Si vous n'avez pas eu un message d'erreur, votre ordinateur doit être prêt pour exécuter le ML-Agents et faire les calculs avec la GPU.

## CHAPITRE 3. ENVIRONNEMENT

Afin de développer un environnement similaire à ce qu'un drone réel pour rencontrer, il faut penser à quel endroit un drone sera inséré. Normalement, un drone capable de faire des livraisons vas se rencontrer dans un environnement urbain avec des édifices, rues, parcs et arbres.

Il faut penser aussi à la nécessité de considérer un troisième axe dans l'adresse finale, parce qu'il y a toujours la possibilité d'envoyer un colis jusqu'à une fenêtre ou terrasse. Ou quand même la possibilité d'avoir futurément une destination standard au-dessus des bâtiments pour drones capables de faire des livraisons.

### 3.1. Des structures

L'idée est d'enseigner un drone à faire des livraisons dans un environnement virtuel, créer un modèle de réseaux neural capable de piloter ce drone à travers la ville virtuel vers une destination en 3 coordonnées (3 axes). La structure de la ville virtuelle doit simuler l'environnement où le drone sera inséré, dans le cas d'un drone projeté pour faire des livraisons, cet endroit doit être proche à une ville urbaine avec des édifices, rues, parcs, arbres et différentes altitudes.

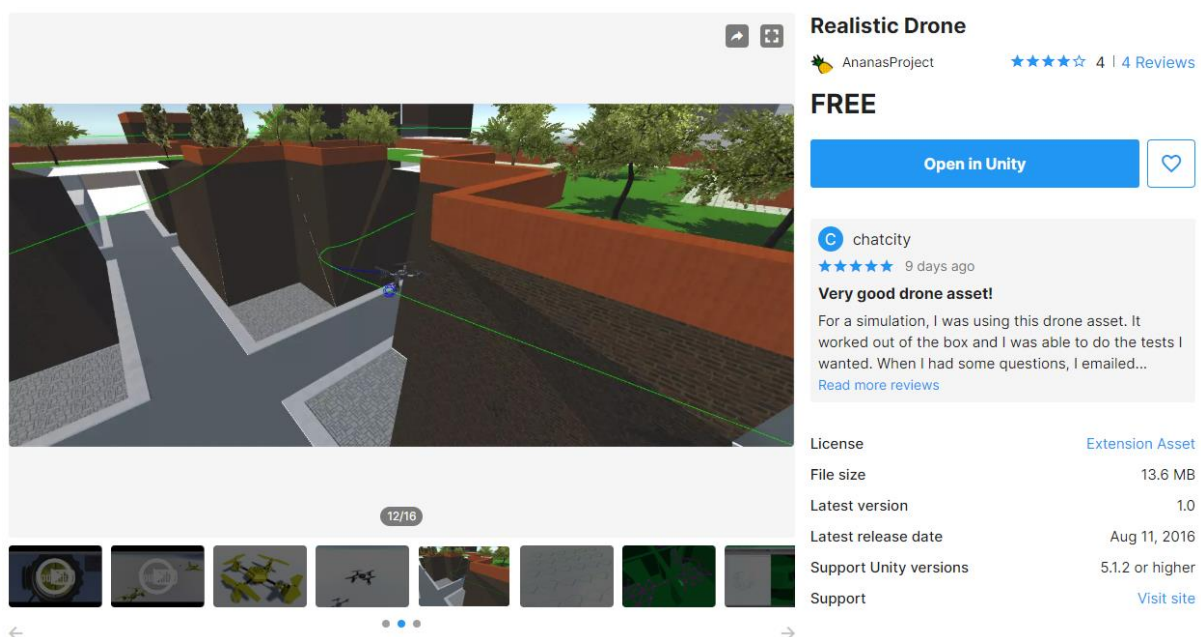
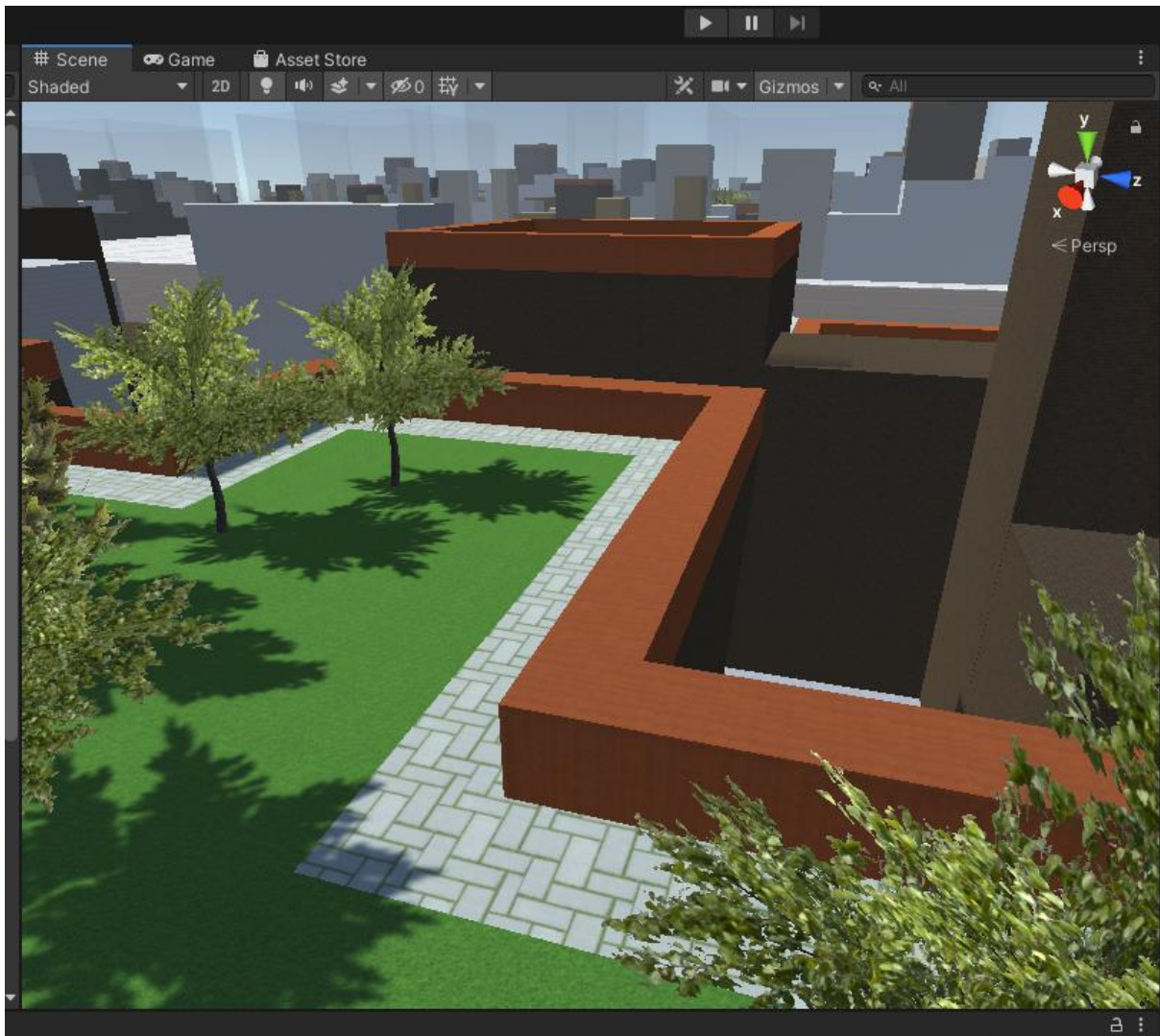


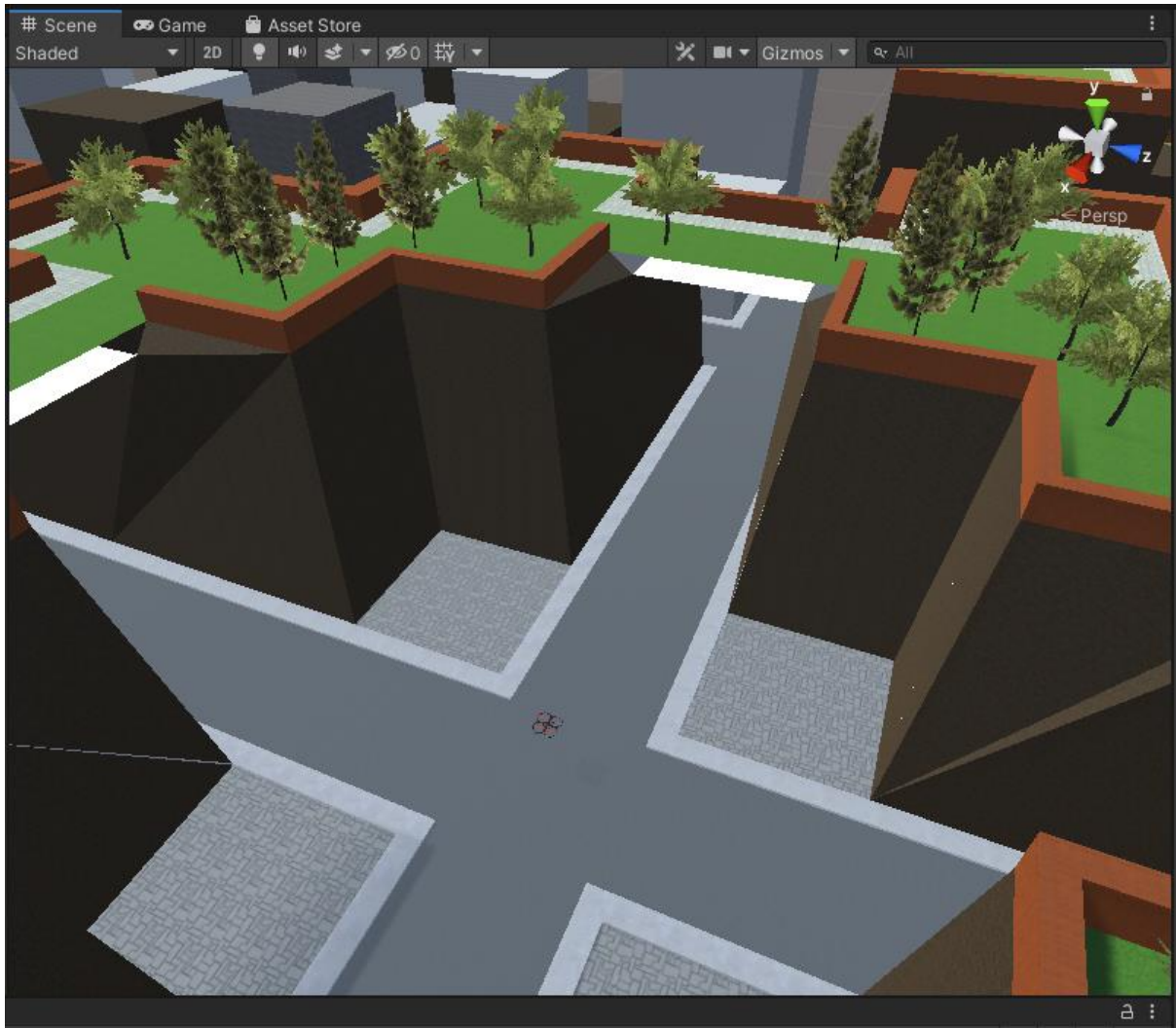
Figure 20 : Page de téléchargement de assets Realistic Drone.

Dans l'unity store, il est possible de télécharger des différents « assets » en contenant des éléments 3D, ainsi que « scenes » prêtes. Il y a un asset que s'appelle « Realistic Drone », lequel a un environnement urbain qui s'intègre parfaitement dans la description souhaitée. Il a été créé par le groupe AnanasProject.



*Figure 21 : Environnement créé pour être utilisé dans la simulation.*





*Figure 22 : Environnement créé pour être utilisé dans la simulation et le drone au centre.*

Il a été utilisé dans ce projet pour simuler la structure d'une ville urbaine. Il représente bien les édifices, rues, parcs et il y a même des viaducs. Un autre point important ce qu'il représente bien les différentes altitudes des édifices.



Figure 23 : Environnement créé pour être utilisé dans la simulation, mais en montrant les édifices.

Chaque élément est doté d'un « Box Collider » capable de savoir quand il entre en contact avec le drone. Toujours quand le drone se choque avec une partie de la ville, il retourne à position initiale est la destination est changée.

### 3.2. La destination

Une partie très importante dans l'environnement est la structure responsable pour marquer la destination finale du drone. Il s'agit d'un espace parallélépipède un peu plus grand que le drone e beaucoup plus haut.



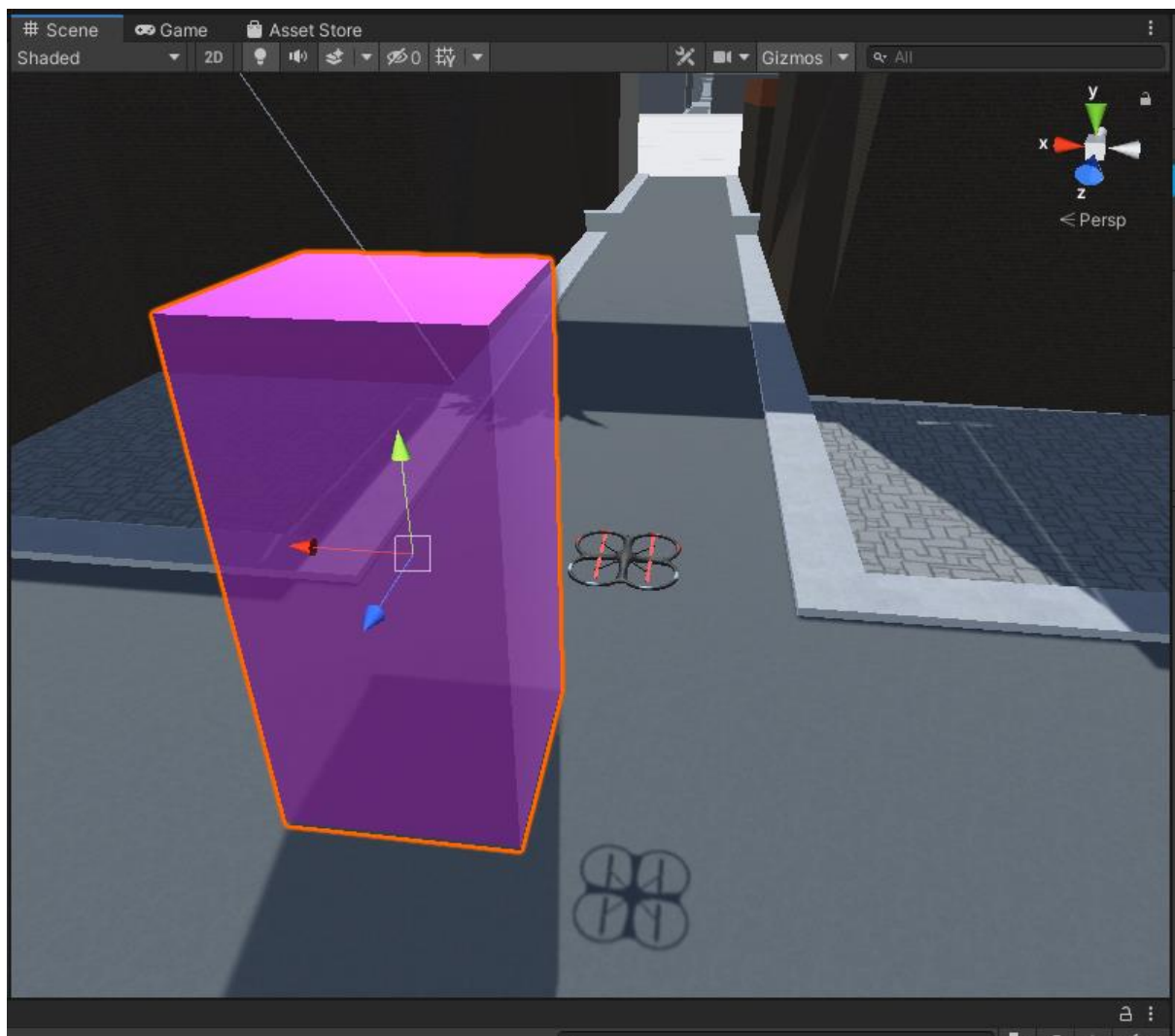


Figure 24 : Comparaison entre le drone et le "Target".

Dans le code, il s'appelle « Target » et possède un « Box Collider » capable de savoir quand le drone entre en contact avec lui. Sa position en trois axes est très importante, elle est le destin final du drone.

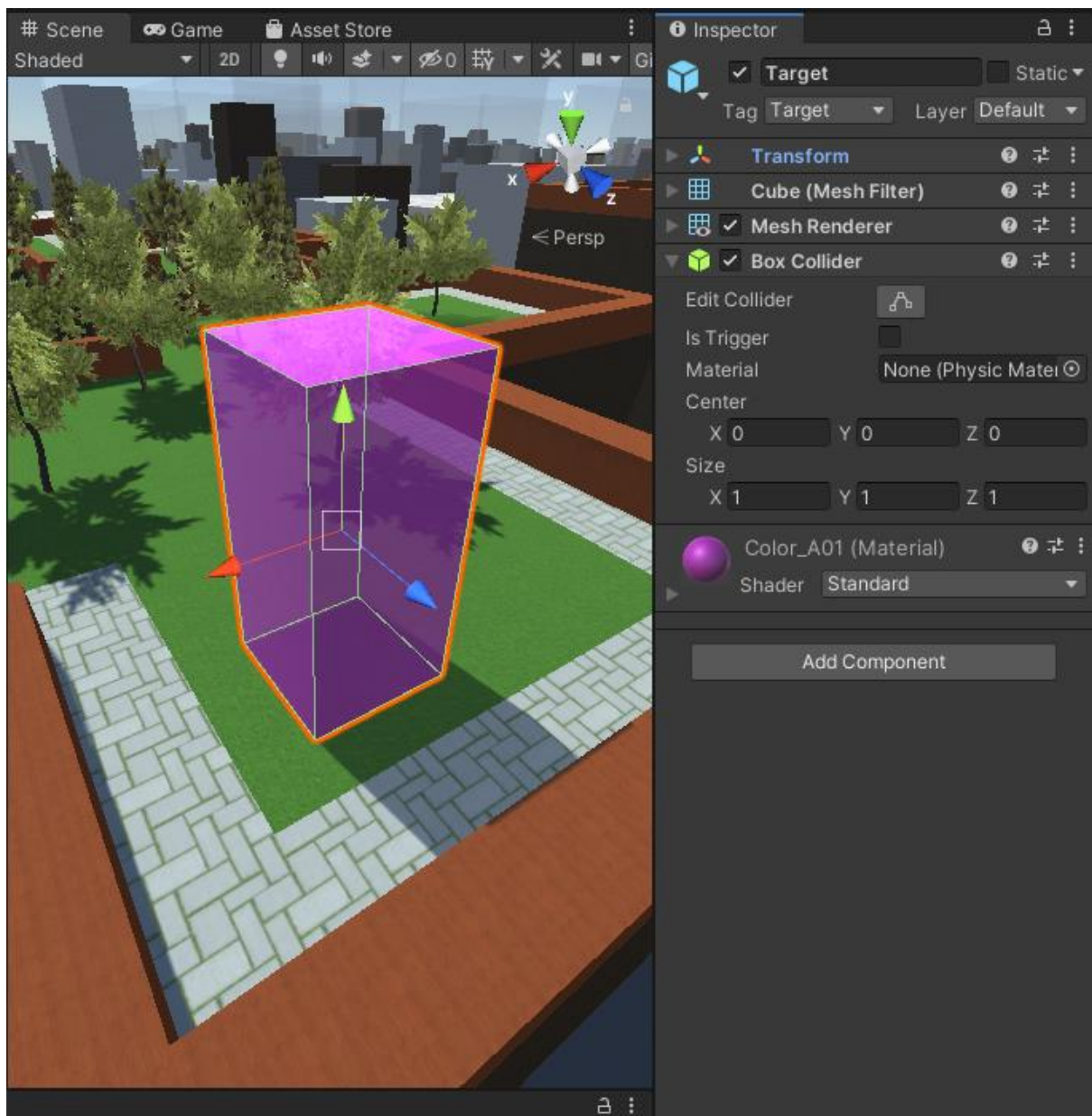


Figure 25 : Propriétés du "Target".

Avec les commandes programmées, sa position change aléatoirement à chaque réinitialisation vers une des 16 positions pensés avant.

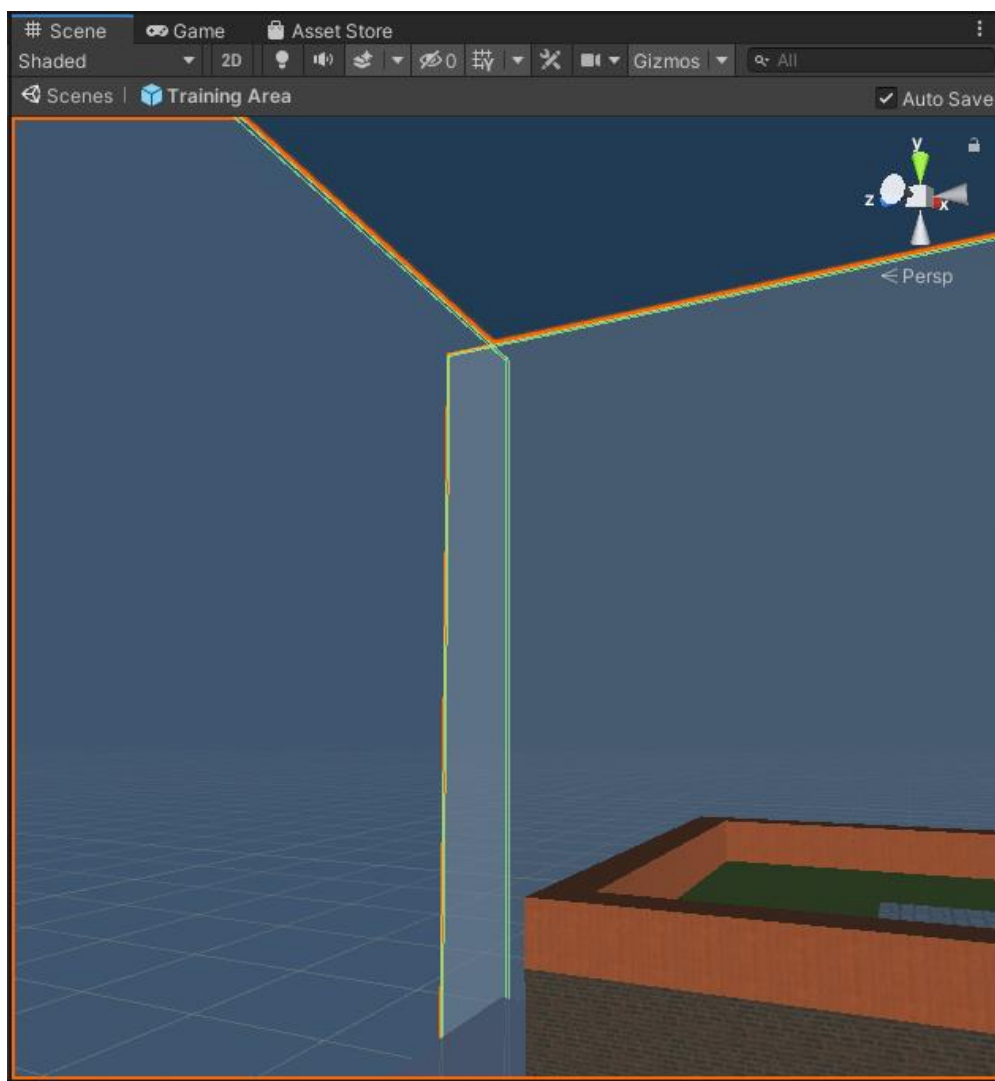
La réinitialisation va être faite toujours quand le drone arrive au Target, quand le drone se choque a quelque partie d'endroit ou quand le temps limite arrive.

### 3.3. Barrières

Après beaucoup tester l'environnement et affronter plusieurs problèmes, une conclusion a été réfléchi. Il faut avoir une barrière à la frontière de la ville pour éviter la sortie du drone pendant l'enseignement.

Comme l'enseignement est fait par essais et erreurs, il y avait plusieurs fois où le drone sort de la ville jusqu'au temps limite.

Afin d'éviter cette situation indésirable, des murs visibles au drone avec un « Box Collider » ont été créés. Les barrières fonctionnent comme une partie de la ville, si le drone choque contre elles, une réinitialisation va être forcé.



*Figure 26 : Barrières qui définissent la frontière d'espace d'apprentissage.*

Ils se trouveront toujours à la frontière de la ville.

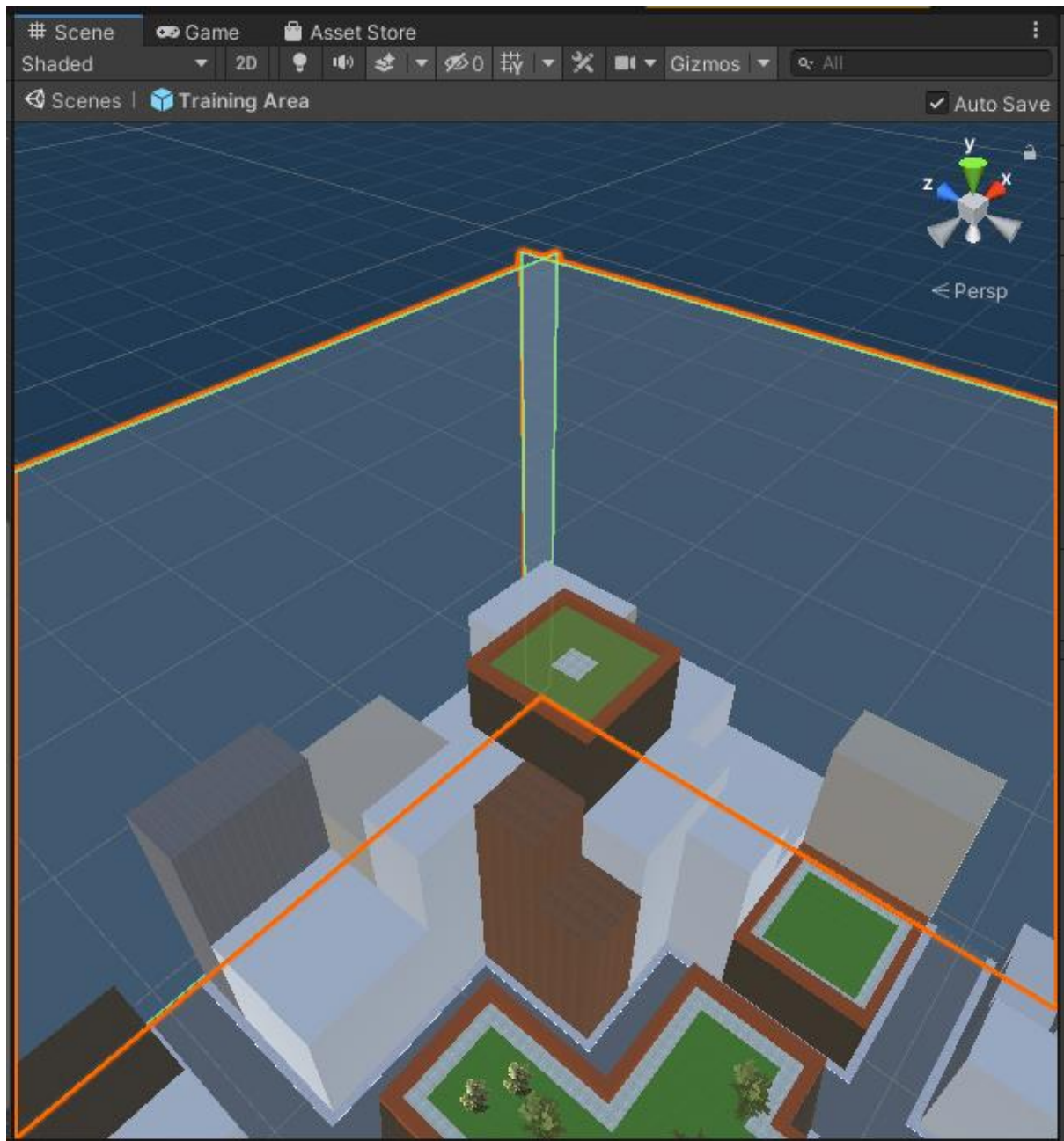


Figure 27 : Barrières qui définissent la frontière d'espace d'apprentissage.

Mais une situation observée est que le drone ne vole pas dans une altitude très haut. Pour respecter la condition de temps limite et s'orienter vers la destination, il finit toujours par adopter une stratégie consistant à continuer à voler à une hauteur inférieure. En conséquence, il n'y a pas une raison de mettre une barrière au-dessus.



## CHAPITRE 4. DRONE

### 4.1. Modèle 3D

Le modèle 3D du drone utilisé dans ce projet est une partie de l'Asset qui s'appelle « Quadcopter controller ». Il est développé pour l'utilisateur GBAndrewGB.

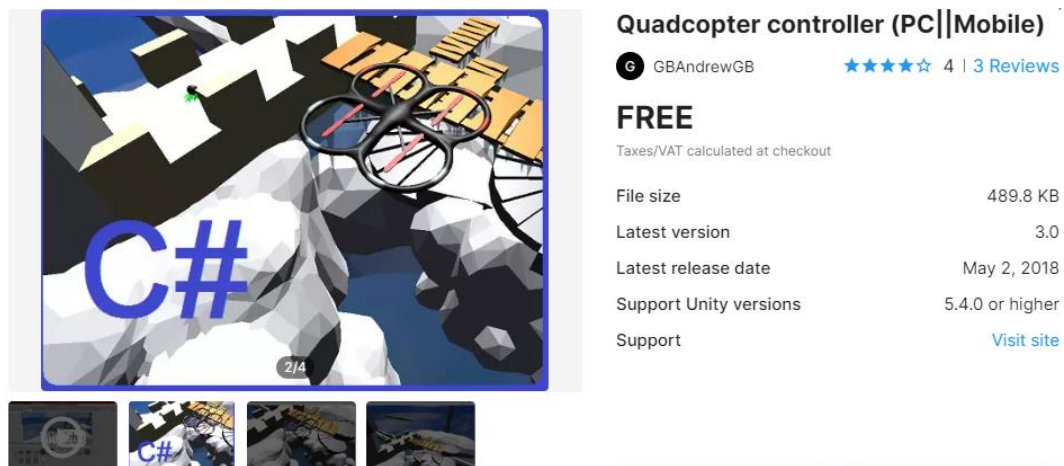


Figure 28 : Page de téléchargement d'Assets Quadcopter controller.

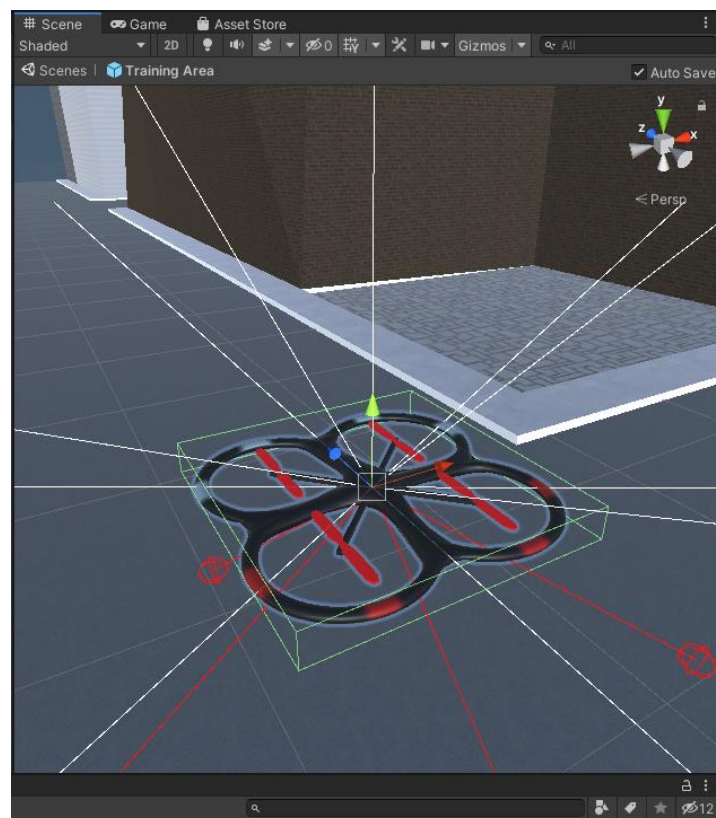


Figure 29 : Image du drone avec son « box collider ».

Il est un modèle simple d'un drone avec quatre rotors, parfait pour simuler la taille réelle d'un drone dans l'espace virtuel créé, il est également équipé d'un « BoxCollider ». Mais juste le modèle 3D et l'animation ont été utilisés dans ce projet, les autres composants présents dans l'inspecteur d'unity sont créés après.

## 4.2. Capteurs

L'interaction du drone avec l'environnement se fait par 17 rayons avec une longueur d'environ 5 mètres distancés de 60 ou 30 degrés entre chaque un. Le component responsable pour créer des rayons s'appelle « Ray Perception Sensor 3D » et vous pouvez contrôler la longueur de chaque rayon, les objets détectables, la quantité de rayons, les degrés englobés pour tous les rayons, la taille de la sphère de détection, bien comme la position initiale et finale dans l'axe vertical et les couleurs.

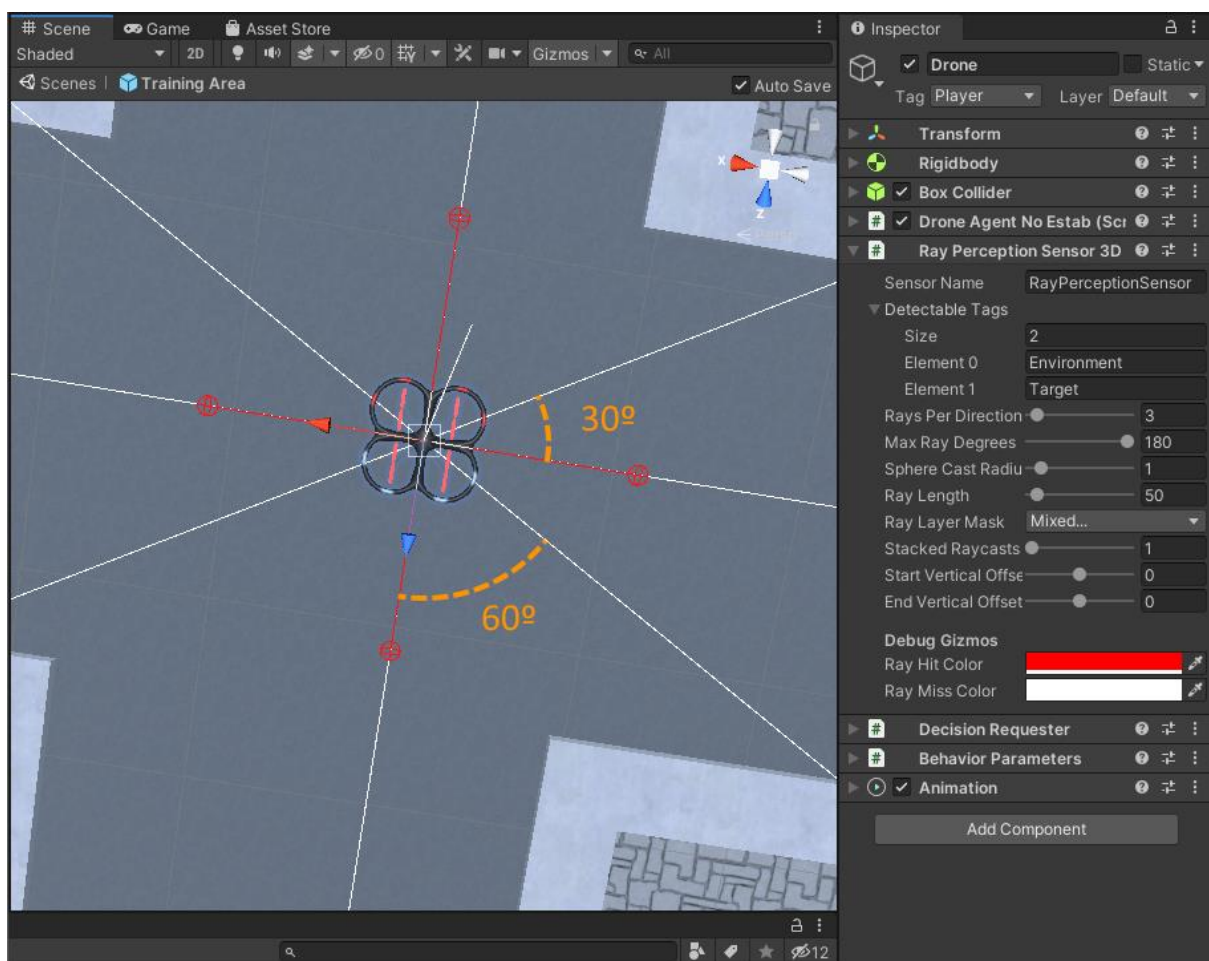


Figure 30 : Propriétés des rayons utilisés comme capteurs.

Il y a encore quelques problèmes avec ce component. Un des problèmes est la raison pour avoir 30 et 60 degrés quand l'idéal est d'avoir juste 45 degrés. Ce problème est causé pour l'impossibilité de choisir exactement les angles entre les rayons et de n'avoir pas une division exacte entre chaque un quand l'option « max ray degrees » est marqué 180.

Le deuxième problème est l'impossibilité de tourner l'axe de référence pour les 180 degrés. Mais il y a une solution pour celui-ci. Il n'est pas possible de tourner l'axe de référence dans le component directement, mais il est encore possible de créer un « child GameObject » avec la référence désirée. En créant plusieurs « GameObjects », il est possible de simuler bien l'orientation des capteurs ultrasoniques dans un vrai drone.

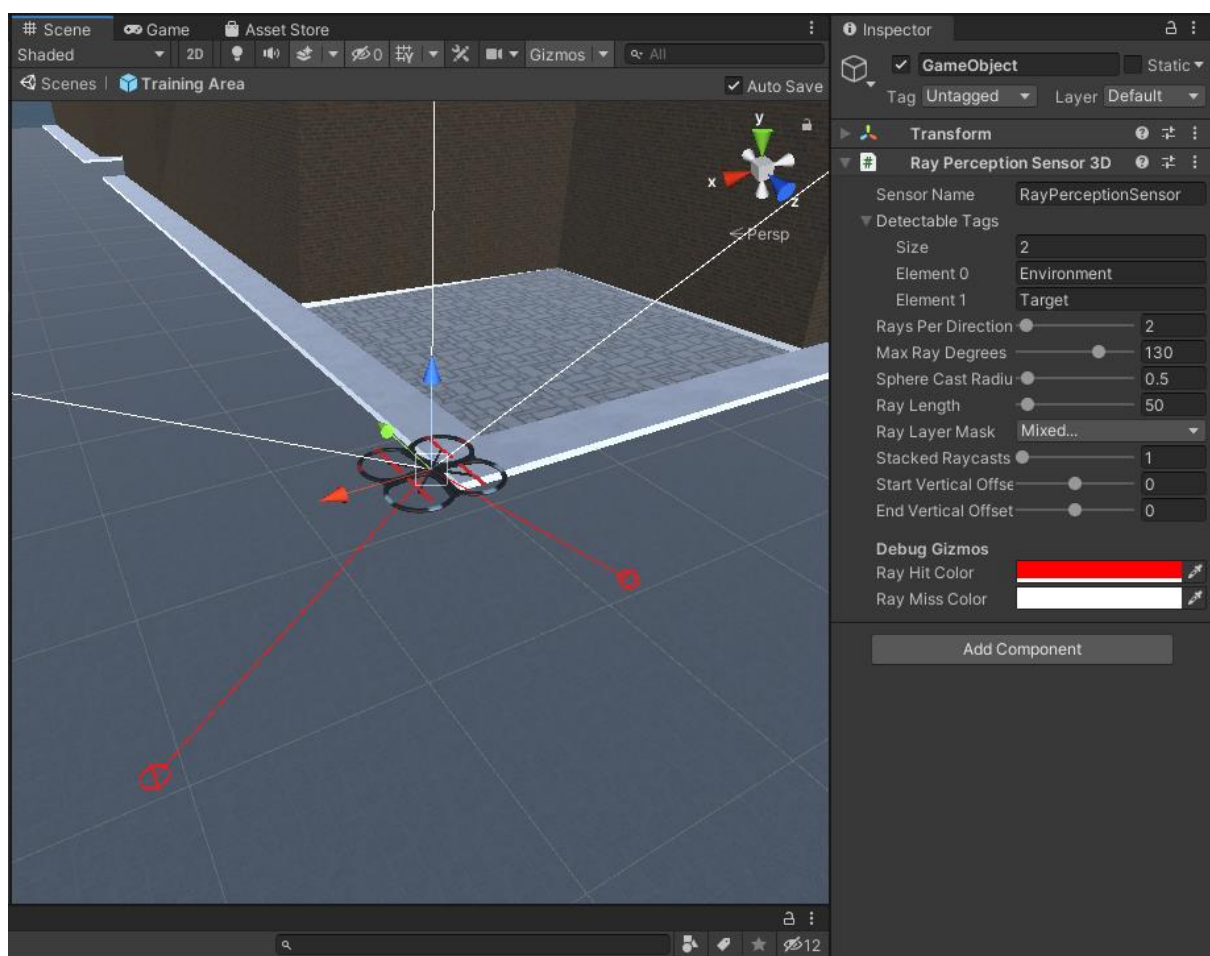


Figure 31 : Capteurs dans le plan XZ (référence global).

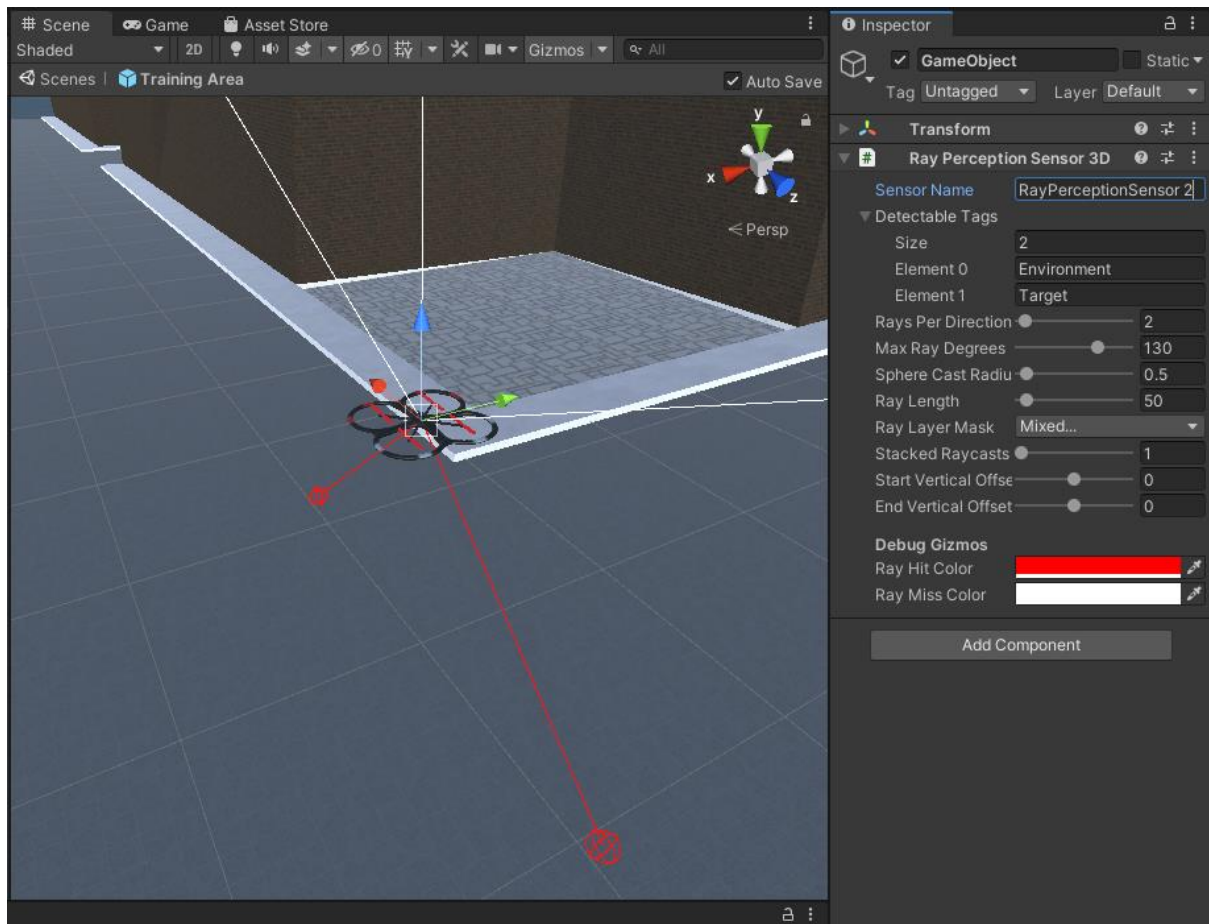


Figure 32 : Capteurs dans le plan YZ (référence global).

### 4.3. Agent

L'agent intelligent qui va être enseigné et va être responsable pour commander le drone est un ensemble de composants.

Le premier composant et le plus important est le « script » principal, ici appelé de « Drone Agent No Estab », il est responsable pour :

- Commencer, terminer et réinitialiser chaque épisode d'enseignement ;
- Envoyer à l'API python les données reçus dans les capteurs rayons ;
- Exécuter l'action reçu de l'API python et pour calculer les récompenses et les l'envoyer.

Dans ce projet, plusieurs fonctions ont été ajoutés au script original de ML-Agents, lequel est maintenant aussi responsable pour repositionner le « Target » et vérifier les collisions. Attention au champ « max step », il faut définir une quantité maximum de pas



possibles avant forcer une réinitialisation, il est ce paramètre le responsable pour contrôler le temp maximum de chaque épisode d'enseignement.

Le deuxième est appelé « Decision Requester », il est responsable pour contrôler si l'agent va faire des pas entre décisions envoyés. Il y a le paramètre appelé « decision period » responsable pour la quantité de pas effectués entre chaque décision.

Le troisième est appelé « Behavior parameters », il est responsable pour un ensemble de fonctions. Elles sont :

- Définir le nom du model ;
- Définir la taille du vecteur d'observation qui sera envoyé à l'API python. Il est normalement une source d'erreur récurrente qui empêche l'apprentissage de commencer, donc il faut faire attention à ce paramètre quand il y a un message d'erreur en parlant du numéro d'observations ;
- Définir e type d'action (discrète ou continu) et sa taille ;
- Définir le modèle à être utilisé. S'il n'y a pas un modèle sélectionné, un nouveau modèle va être formé avec le nom stipulé dans le premier paramètre.
- Définir si le calcules seront faites pour la CPU ou GPU ;
- Définir le type de comportement, s'il sera le default, inférence ou heuristique. Ce dernier vous permet de tester les contrôles et vérifier, avant l'enseignement, s'ils sont bien faits.
- Si seront utilisés des capteurs faits dans un « child GameObject ».

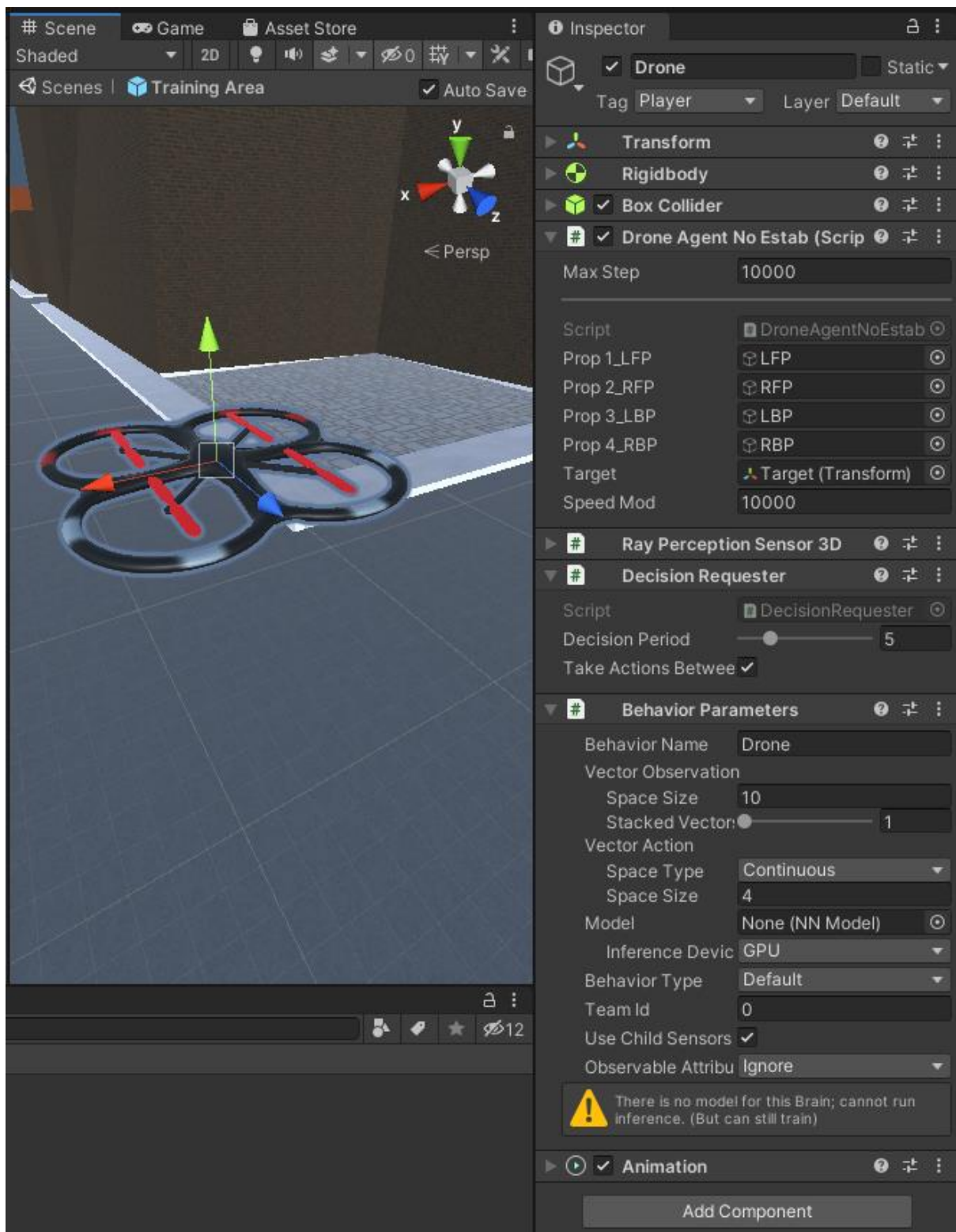


Figure 33 : Propriétés générales du Drone.

### 4.3.1 Méthodes importantes

Le processus d'enseignement dans la boîte à outils ML-Agents implique l'exécution d'épisodes au cours desquels l'agent tente de résoudre la tâche. Chaque épisode dure jusqu'à ce que les agents résolvent la tâche, échouent (collision), ou expirent (prend trop de temps à résoudre ou échouent à la tâche). Au début de chaque épisode, la méthode « `Agent.OnEpisodeBegin()` » est appelée pour configurer l'environnement d'un nouvel épisode. Généralement, la scène est initialisée de manière aléatoire pour permettre à l'agent d'apprendre à résoudre la tâche dans diverses conditions. La scène peut être terminée avec la méthode « `Agent.EndEpisode()` ».

L'agent envoie les informations que nous collectons à l'API python, qui les utilise pour prendre une décision. Lorsque vous entraînez l'agent (ou utilisez un modèle entraîné), les données sont introduites dans un réseau neuronal en tant que vecteur de caractéristiques. La méthode utilisée dans ce cas est le « `Agent.CollectObservations(VectorSensor sensor)` ».

La dernière partie du code de l'agent normalement est la méthode « `Agent.OnActionReceived (float[] vectorAction)` », qui reçoit les actions et attribue la récompense. Dans cette méthode, le vecteur « `vectorAction` » est le vecteur où chaque composant est responsable pour une action. Par exemple, dans ce projet dans le deuxième contrôle développé, le « `vectorAction` » est composé pour 4 valeurs de vitesse de rotation, une pour chaque hélice du drone.

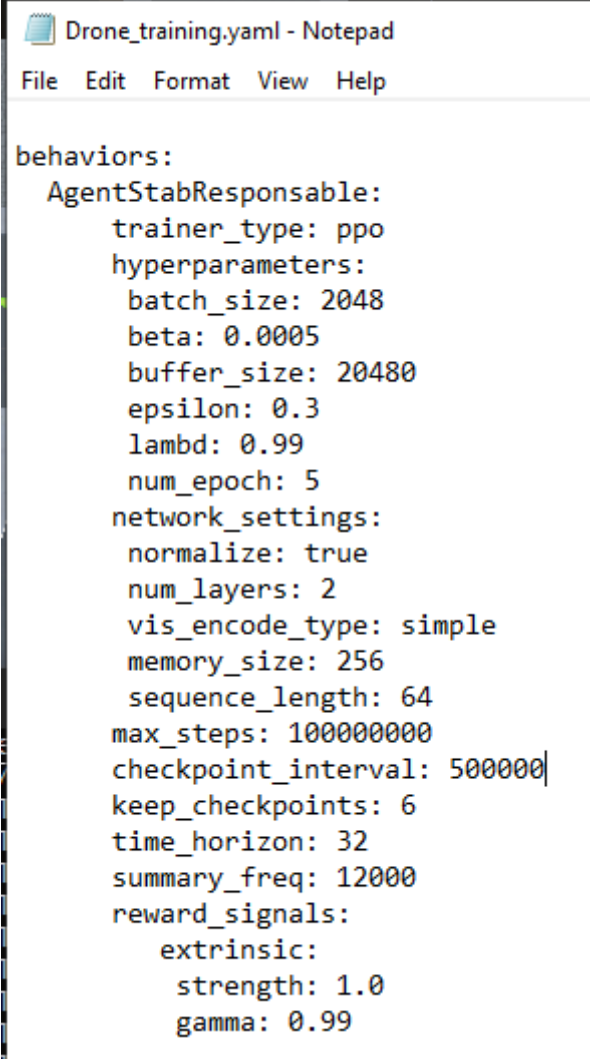
- `OnEpisodeBegin()`
- `CollectObservations(VectorSensor sensor)`
- `OnActionReceived(float[] vectorAction)`



## CHAPITRE 5. HYPERPARAMÈTRES

L'enseignement d'une intelligence artificielle est contrôlé par hyperparamètres. C'est-à-dire qu'il y a beaucoup d'options modifiables qui ont une interférence dans le processus de traitement de données réalisé par l'API python. Il faut les connaître si vous avez l'envie d'optimiser le processus d'enseignement et accélérer le temps de convergence.

Les hyperparamètres qui sont changés dans ce projet par rapport au défaut sont expliqués dans ce chapitre.



```
behaviors:
  AgentStabResponsable:
    trainer_type: ppo
    hyperparameters:
      batch_size: 2048
      beta: 0.0005
      buffer_size: 20480
      epsilon: 0.3
      lambda: 0.99
      num_epoch: 5
    network_settings:
      normalize: true
      num_layers: 2
      vis_encode_type: simple
      memory_size: 256
      sequence_length: 64
      max_steps: 100000000
      checkpoint_interval: 500000
      keep_checkpoints: 6
      time_horizon: 32
      summary_freq: 12000
    reward_signals:
      extrinsic:
        strength: 1.0
        gamma: 0.99
```

Figure 34 : Hyperparamètres utilisés dans le départ des tests. Ils ne sont pas le standard.

## 5.1. Trainer

Le « Trainer » est l'hyperparamètre plus important, il va définir l'algorithme utilisé par l'API python. Pour ML-Agents, il y a deux options disponibles : PPO ou SAC. Il faut connaître les cas d'utilisation de chaque algorithme, de cette façon vous pouvez choisir le plus efficace pour vous.

L'algorithme SAC est plus efficace dans le traitement de beaucoup de données, donc il est plus efficace dans un environnement très complexe lequel n'est pas possible de simuler plus d'une instance au même temps. Le principal problème de l'algorithme SAC est son instabilité, sa convergence n'est pas autant garantie que l'autre option, l'algorithme PPO.

L'algorithme PPO est beaucoup plus utilisé que l'algorithme SAC, la principale raison est parce qu'il est beaucoup plus stable que le SAC et sa convergence est pratiquement une garantie. Il est très efficace pour des enseignements dans un environnement plus simple où il n'y a pas beaucoup de données collectées, qui sont lesquelles dans vous pouvez faire beaucoup d'instances.

Dans ce projet, l'algorithme choisi est le PPO. L'environnement créé est simple, il y a plusieurs instances et il n'y a pas beaucoup des données envoyées à l'API python.

## 5.2. Batch size

Le Batch Size est un hyperparamètre de descente de gradient qui contrôle le nombre d'échantillons d'apprentissage à traiter avant la mise à jour des paramètres internes du modèle. Cela doit toujours être plusieurs fois plus petit que `buffer_size`. Si vous utilisez un espace d'action continu, cette valeur doit être élevée (de l'ordre de 1000). Si vous utilisez un espace d'action discret, cette valeur doit être plus petite (de l'ordre de 10 s).

La différence entre espace d'action continu et discret est si vous voulez changer les actions possibles dans un moment spécifique ou pas. Par exemple, si vous voulez un agent capable de se déplacer dans un avion et de sauter, nous pourrions définir deux branches (une pour le mouvement et une pour le saut) parce que nous voulons que notre agent puisse se déplacer et sauter simultanément. Nous définissons la première branche pour avoir 5 actions possibles (ne pas bouger, aller à gauche, aller à droite, reculer, avancer) et la seconde pour avoir 2 actions possibles (ne pas sauter, sauter). Cet exemple démontre un espace d'action

discret. Si vous n'avez pas l'envie de changer les actions, vous êtes en train de créer un espace d'action continu.

Gamme typique :

- (Continu - PPO) : 512 – 5120. Celui-ci est le cas de ce projet ;
- (Continu - SAC) : 128 - 1024 ;
- (Discret, PPO & SAC) : 32-512.

Attention : « Batch\_size », « Buffer-size » et « Num\_epochs » sont corrélés. Une explication plus simple est que le modèle est mis à jour à chaque fois qu'il est obtenu le nombre de pas de « buffer\_size ». Au cours de cette mise à jour, le « buffer » est divisé en lots de taille « batch\_size » et est effectué une mise à jour du modèle sur chacun de ces lots un par un. Ce processus est répété « num\_epochs » plusieurs fois.

### 5.3. Beta

Beta est la force de la régularisation d'entropie, ce qui rend la politique plus aléatoire. Cela garantit que les agents explorent correctement l'espace d'action pendant la formation. Augmenter ce nombre garantira que des actions plus aléatoires seront prises. Cela doit être ajusté de telle sorte que l'entropie (mesurable à partir de TensorBoard) diminue lentement parallèlement à l'augmentation de la récompense. Si l'entropie baisse trop rapidement, augmentez la valeur beta. Si l'entropie baisse trop lentement, diminuez la beta.

- Gamme typique : 1e-4 - 1e-2
- Par défaut = 5.0e-3

### 5.4. Buffer Size

« Buffer Size » est le nombre d'expériences à collecter avant de mettre à jour le modèle. Correspond au nombre d'expériences à collecter avant de procéder à tout apprentissage ou mise à jour du modèle.

PPO : Cela doit être plusieurs fois plus grand que « batch\_size ». En règle générale, un « buffer\_size » plus grand correspond à des mises à jour d'entraînement plus stables.

SAC : La taille maximale de la mémoire tampon d'expérience - de l'ordre de milliers de fois plus longue que vos épisodes, afin que SAC puisse apprendre des expériences anciennes et nouvelles.

- Gamme typique : PPO : 2048 – 409600 ; SAC : 50000 – 1000000 ;
- Par défaut = 10240 pour PPO et 50000 pour SAC

## 5.5. Epsilon

L'Epsilon influence la rapidité avec laquelle le modèle peut évoluer pendant la formation. Correspond au seuil acceptable de divergence entre l'ancienne et la nouvelle politique lors de la mise à jour de la descente de gradient. La définition de cette valeur faible entraînera des mises à jour plus stables, mais ralentira également le processus de formation.

- Gamme typique : 0,1 - 0,3 ;
- Par défaut = 0,2.

## 5.6. Lambd

Cela peut être considéré comme la mesure dans laquelle l'agent s'appuie sur son estimation de valeur actuelle lors du calcul d'une estimation de valeur mise à jour. Les valeurs faibles correspondent à se fier davantage à l'estimation de la valeur actuelle (qui peut être un biais élevé), et les valeurs élevées correspondent à se fier davantage aux récompenses réelles reçues dans l'environnement (ce qui peut être une variance élevée). Le paramètre fournit un compromis entre les deux, et la bonne valeur peut conduire à un processus d'entraînement plus stable.

- Gamme typique : 0,9 - 0,99 ;
- Par défaut = 0,9.



## 5.7. Max Steps

Nombre total d'étapes (c'est-à-dire observation collectée et action prise) qui doivent être prises dans l'environnement (ou dans tous les environnements si vous utilisez plusieurs en parallèle) avant de terminer le processus d'enseignement. Si vous avez plusieurs agents avec le même nom de comportement dans votre environnement, toutes les mesures prises par ces agents contribueront au même nombre « max\_steps ».

- Gamme typique :  $5e5$  -  $1e7$  ;
- Par défaut = 500000.

## 5.8. Checkpoint Interval

Le nombre d'expériences collectées entre chaque point de contrôle par l'API python. Un maximum de points de contrôle « keep\_checkpoints » est enregistré avant que les anciens ne soient supprimés. Chaque point de contrôle enregistre les fichiers .nn dans le dossier « results ».

- Par défaut = 500000.

## 5.9. Keep Checkpoints

Le nombre maximum de points de contrôle du modèle à conserver. Les points de contrôle sont enregistrés après le nombre d'étapes spécifié par l'option checkpoint\_interval. Une fois le nombre maximum de points de contrôle atteint, le point de contrôle le plus ancien est supprimé lors de l'enregistrement d'un nouveau point de contrôle.

- Par défaut = 5 ;

## 5.10. Num Epoch

Nombre de passages à effectuer dans le tampon d'expérience lors de l'optimisation de la descente de gradient. Plus le batch\_size est grand, plus il est acceptable de le faire. Diminuer cela garantira des mises à jour plus stables, au prix d'un apprentissage plus lent.

- Gamme typique: 3 – 10 ;
- Par défaut = 3.

## 5.11. Normalize

Indique si la normalisation est appliquée aux entrées d'observation vectorielle. Cette normalisation est basée sur la moyenne courante et la variance de l'observation vectorielle. La normalisation peut être utile dans les cas de problèmes de contrôle continu complexes, mais peut être nuisible avec des problèmes de contrôle discrets plus simples.

- Par défaut = false.

## 5.12. Summary Freq

Nombre d'expériences à collecter avant de générer et d'afficher des statistiques d'entraînement. Cela détermine la granularité des graphiques dans « Tensorboard ».

- Par défaut = 50000.

## 5.13. Time Horizon

Combien d'étapes d'expérience à collecter par agent avant de l'ajouter au « buffer » d'expérience. Lorsque cette limite est atteinte avant la fin d'un épisode, une estimation de valeur est utilisée pour prédire la récompense globale attendue de l'état actuel de l'agent. En tant que tel, ce paramètre fait un compromis entre une estimation de variance moins biaisée, mais plus élevée (« time horizon » long) et une estimation plus biaisée, mais moins variée (« time

horizon » court). Dans les cas où il y a des récompenses fréquentes dans un épisode, ou les épisodes sont prohibitifs, un nombre plus petit peut être plus idéal. Ce nombre doit être suffisamment grand pour capturer tous les comportements importants dans une séquence d'actions d'un agent.

- Gamme typique : 32 – 2048 ;
- Par défaut = 64.

## CHAPITRE 6. CONTRÔLE

Dans un premier moment, l'idée été d'enseigner l'agent à contrôler un drone déjà programmé. C'est-à-dire enseigner l'agent à contrôler le drone avec les mêmes commandes qui peuvent être faits par une humaine avec un télécommande. Dans cette situation, un contrôle de stabilité doit exister en avance.

Au but de quelques semaines, l'inefficacité du processus précédent a été remarquée et un contrôle où l'intelligence artificielle est aussi responsable pour la stabilité a été pensé.

### 6.1. Contrôle avec stabilité créée en avance

Le but de ce contrôle est de contrôler un drone déjà programmé, c'est-à-dire que l'agent n'est pas responsable pour le contrôle de stabilité et qu'il peut faire juste les commandes qui peuvent être faits par une humaine avec une manette.

S'il n'est pas responsable pour le contrôle de stabilité, il faut en exister un pour maintenir le drone stable pendant la simulation et il doit avoir accès au « GameObject » du drone. En relation au contrôle de stabilité pendant la simulation, il faut juste que le drone soit stable, donc, en tant que le drone soit stable, la façon de comme le contrôle marche a une interférence petite dans l'enseignement, par suite un contrôle très simple a été créé en utilisant des outils existent dans l'Unity.

Le contrôle de stabilité consiste en une série de « if » responsable pour vérifier la position du drone en chaque axe et appliquer un torque contraire s'il commence à s'incliner. Si l'inclinaison est plus que 10 degrés, le contrôle va appliquer un torque grand, si elle est inférieure à 10 degrés, le contrôle va appliquer un torque petit jusqu'à stabilité.

Pour vérifier la position angulaire, il a été utilisé les angles de Euler avec la méthode « `.transform.localEulerAngles` » et pour appliquer un torque, il a été utilisé la méthode « `.AddRelativeTorque` ».

#### 6.1.1. Récompenses

Le système d'essai et d'erreur seulement sera efficace si le système de récompense est bien fait. Ce dernier est responsable pour dire à l'API python si elle est en train de faire les actions désirées.

A chaque itération, à chaque décision, l'API va recevoir un feedback. Ce feedback est la ponctuation accumulée dans cette itération et est l'unique moyen de contrôler ce que l'API est en train de faire. Donc est un système vraiment important dans ce projet, il doit être bien pensé.

Dans ce premier contrôle, il y a l'intérêt de contrôler avec l'API juste si le drone se déplace dans la direction désirée et s'il entre en collision avec quelque partie de la ville. Il y a aussi l'intérêt de contrôler le temps nécessaire pour arriver à la destination.

Donc, l'agent va recevoir une récompense :

- S'il se déplace vers le « Target » ;
- S'il arrive à la position du « Target ».

Et une punition :

- S'il entre en collision avec une structure ;
- S'il s'éloigne du « Target » ;
- Aussi une ponctuation négative pour chaque étape qu'il effectue afin de diminuer le nombre d'étapes nécessaires pour atteindre la destination.

### **6.1.2. Les tests**

Avant commencer l'entraînement, c'est toujours une bonne idée de tester votre environnement en contrôlant l'agent à l'aide du clavier. Pour ce faire, vous devrez utiliser la méthode Heuristic (). Et sélectionner l'option « Heuristic only » dans le composant « Behavior Parameters » de votre Agent.

Un point important dans l'entraînement est faire plusieurs instances qui vont apprendre au même temps. Pour le faire, il faut créer un « prefab » de votre espace d'enseignement et le répliquer un certain numéro de fois. Ici est important avoir connaissance de la puissance de votre ordinateur et trouver une quantité qu'il est capable de gérer.

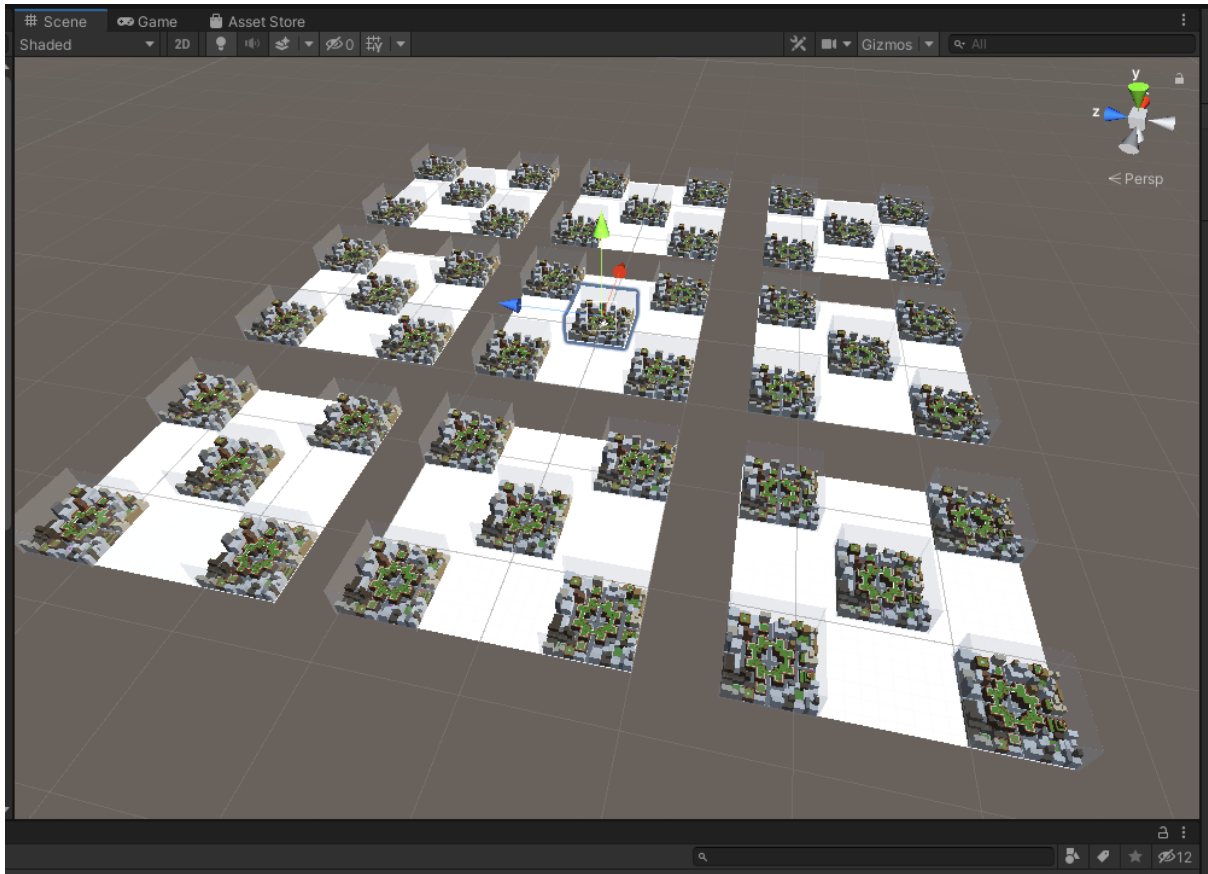


Figure 35 : Image qui montre au centre le « prefab » utilisé pour l'apprentissage. Autour il y a des copies pour augmenter la quantité d'agents qui bougent au même temp.

L'ordinateur utilisé dans ce projet avait les configurations suivantes :

- Système Opérationnel : Windows 10 Home 64-bit ;
- Processeur : Intel Core i7-10875H ~2.3GHz (16 CPUs) ;
- Mémoire : 32 Go
- Vidéo : NVIDIA GeForce RTX 2060

Pour commencer l'apprentissage, il faut utiliser la commande suivante dans la fenêtre de commande du Windows (PowerShell, par exemple) :

```
mlagents-learn <trainer-config-file> --env=<env_name> --run-id=<run-identifiant>
```

Où:

- « <trainer-config-file> » est le chemin du fichier du « .yaml » de configuration d'agent. Celui-ci contient toutes les valeurs d'hyperparamètres.

- « <env\_name> » (facultatif) est le nom (y compris le chemin) de votre exécutable Unity contenant les agents à entraîner. Si « <env\_name> » n'est pas passé, l'apprentissage aura lieu dans l'éditeur. Appuyez sur le bouton « PLAY » dans Unity lorsque le message «Start training by pressing the Play button in the Unity Editor» s'affiche à l'écran.
- « <run-identifiant> » est un nom unique que vous pouvez utiliser pour identifier les résultats de vos courses d'entraînement.

```
mllagents-learn config/ppo/Drone_training.yaml --run-id=DroneTest16
```

Figure 36 : Commande utilisé pour commencer l'apprentissage.

Après l'apprentissage, les résultats peuvent être accédés avec la commande :

« *tensorboard --logdir results --port 6006* »

```
tensorboard --logdir results --port 6006
```

Figure 37 : Commande utilisé pour accéder aux résultats.

Et vous pouvez les accéder dans un navigateur avec la URL :

<http://localhost:6006/>

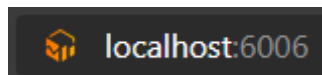


Figure 38 : URL pour visualiser les graphiques des résultats dans le navigateur.

Dans le premier contrôle, le principal aspect considéré été la compétence de faire des livraisons sans avoir une collision et la vitesse dont il été capable de le faire.

Depuis le premier test, l'agent été capable de faire des livraisons sans avoir un impact. Donc le facteur principal de comparaison entre les tests été considéré la vitesse de livraison, c'est-à-dire la durée moyenne des épisodes.

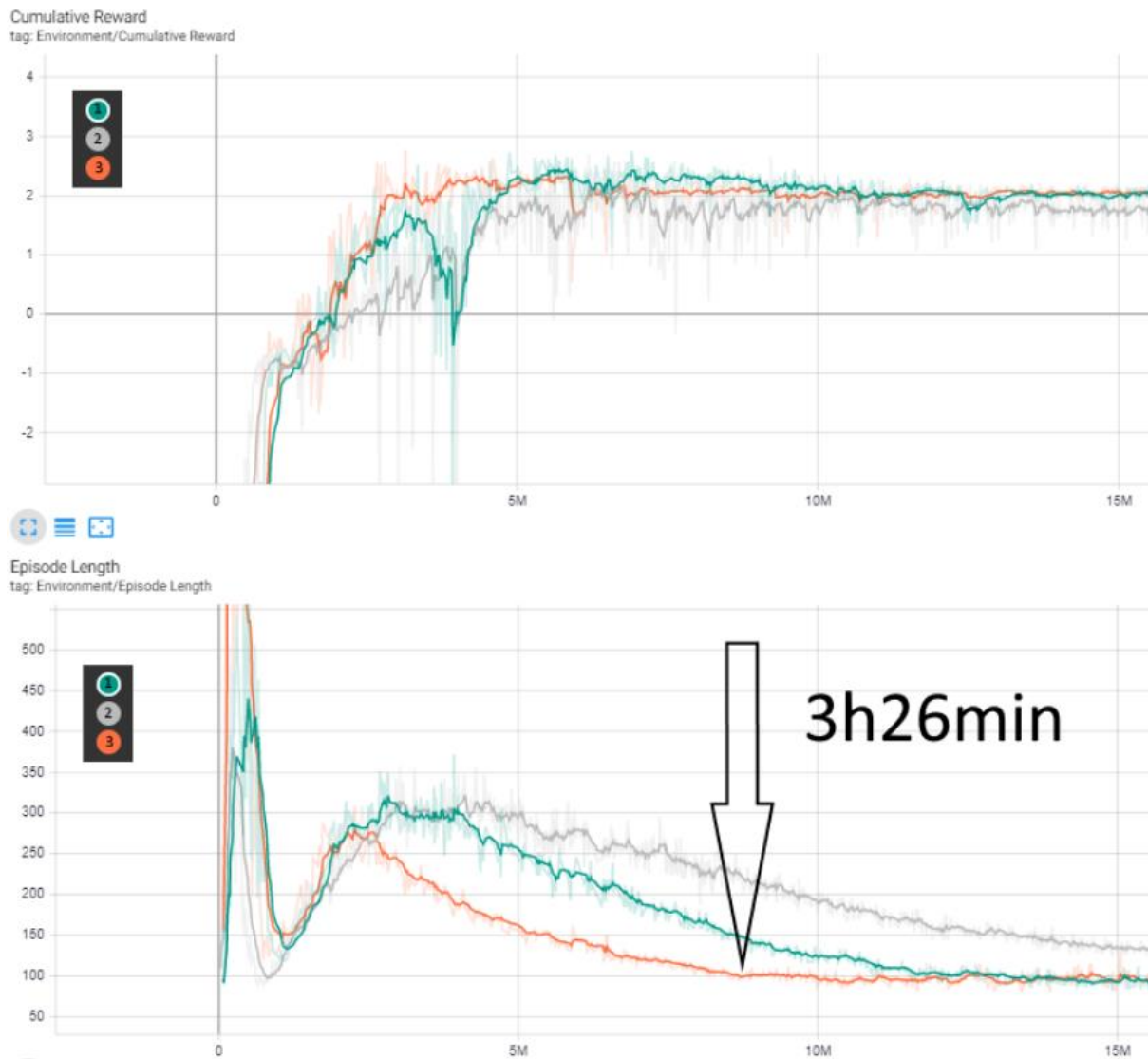


Figure 39 : Graph A : Récompenses cumulés par Quantités de pas. Graph B : Duration d'épisode (en pas) par Quantité de pas. 3h 26 min est le temps nécessaire pour avoir une simulation de 8,76 millions de pas avec l'ordinateur utilisé dans ce projet.

- 1) Le premier test a suivi les paramètres établis dans le chapitre 5 Hyperparamètres ;
- 2)  $\text{Beta} = 0,0005 \rightarrow 0,005$  et  $\text{time\_horizon} = 32 \rightarrow 64$  ;
- 3)  $\text{Lambd} = 0.99 \rightarrow 0,98$  et  $\text{num\_epoch} = 5 \rightarrow 6$ .

Le troisième test a été satisfaisant en considérant l'objectif.



## 6.2. Contrôle sans stabilité créée en avance

La raison d'existence de ce contrôle sans stabilité est d'appliquer les commandes directement sur le drone sans passer pour un système de contrôle préexistant, de cette façon il est possible d'avoir une réponse plus rapide avec moins de traitement en temps réel.

Ce contrôle consiste en contrôler directement les quatre hélices du drone. Mais dans un drone réel, normalement le paramètre contrôlé peut être la vitesse de rotation de chaque moteur avec une fonction de transfert en passant cette donnée contrôlée à un signal PWM. Alors que dans l'unity, il n'est pas suffi de tourner les hélices, il faut créer des vecteurs de force directement sur chaque une pour bien simuler la réalité.

Pour cela, il est nécessaire de créer une relation mathématique en faisant la relation entre la rotation d'une hélice et la force produite, aussi appelée poussée.

L'approche mathématique utilisé dans ce projet est basée sur une approche développée par Gabriel Staples, un ingénieur à station de l'armée de l'air du Cap Canaveral aux États-Unis. Le développement commence avec la deuxième loi de Newton en considèrent la vitesse comme constant. De cette façon les molécules de l'air ont un débit massique à travers l'hélice.

$$F = \frac{d(mv)}{dt} = \left(\frac{dm}{dt}\right)v = \dot{m}v \quad (1)$$

Il s'agit d'un point de départ très courant pour les ingénieurs en propulsion à réaction et les spécialistes des fusées, car cette équation est très courante dans ces deux domaines. Pour un aéronef statique, il est su que toute la vitesse des molécules d'air traversant une hélice contribue à la poussée puisque les molécules d'air ont commencé à être stationnaires, et ont été accélérées à cette vitesse. Par conséquent, la poussée de l'hélice,  $F$ , pour un aéronef stationnaire (statique) est :

$$F = \dot{m}V_f \quad (2)$$

Pour un aéronef en mouvement, seule la vitesse de l'air qui est due au fait que l'air a été accéléré par l'hélice est ce qui contribue à la poussée. C'est-à-dire, le changement de vitesse est ce qui compte :

$$F = \dot{m}\Delta V$$

$$F = \dot{m}(V_f - V_{ac}) \quad (3)$$

Notez que sur la base de cette équation, lorsque la vitesse du drone,  $V_{ac}$ , augmente, la poussée diminue. Cela est dû au fait que la vitesse de sortie de l'hélice (ou vitesse induite) est approximativement constante, et donc le résultat de  $(V_f - V_{ac})$  s'approche de zéro lorsque la vitesse maximale est atteinte.

Puisque  $\dot{m}$  est égal à la densité de l'air multipliée par la section transversale à travers laquelle l'air circule, multipliée par la vitesse de l'air, il est obtenu :

$$\dot{m} = \rho A V_f$$

$$F = \rho A V_f (V_f - V_{ac})$$

$$F = \rho A V_f^2 - \rho A V_f V_{ac} \quad (4)$$

Où  $A$  est l'aire de la section transversale ou l'aire du disque de rotor recouverte par une hélice en rotation.  $A$  est donc l'aire d'un cercle :

$$A = \pi r^2$$

$$A = \frac{\pi d^2}{4} \quad (5)$$

Où  $r$  est le rayon de l'hélice, et  $d$  est le diamètre de l'hélice, en unités de mètres.

En remplaçant en  $A$  ci-dessus, il est obtenu ce qui suit pour l'équation de poussée dynamique de l'hélice théorique :

$$F = \rho \frac{\pi d^2}{4} V_f^2 - \rho \frac{\pi d^2}{4} V_f V_{ac} \quad (6)$$

En simplifiant, en factorisant le terme d'area, il est obtenu :

$$F = \rho \frac{\pi d^2}{4} (V_f^2 - V_f V_{ac}) \quad (7)$$

Rappelez-vous,  $V_f$  est la vitesse de sortie de l'air, ou la vitesse induite de l'air par une hélice, à travers une hélice, et  $V_{ac}$  est la vitesse de l'aéronef (ou, plus précisément, la vitesse du courant libre).  $F$  est la poussée,  $\rho$  la densité de l'air et  $d$  est le diamètre de l'hélice.

En supposant que  $V_f$  est approximativement égal à la vitesse de pas de l'hélice. Le pas d'une hélice d'avion radiocommandé est normalement mesuré en pouces (in), et représente la distance théorique vers l'avant qu'une hélice se déplacerait, en fonction de son angle de pas, si elle tournait exactement d'un tour. La vitesse de pas dépend uniquement de la vitesse de rotation et du pas de l'hélice et se présente comme suit :

$$V_{pas}(mph) = V_{rot}(rpm) \cdot Pas(in) \frac{1 ft}{12 in} \frac{1 mile}{5280 ft} \frac{60 min}{1 hr} \quad (8)$$

Maintenant, vous pouvez brancher la vitesse de pas (Eq8) dans (Eq7), à la place de  $V_f$ , pour obtenir une estimation de la poussée de l'hélice. En réglant  $V_{ac}$  à zéro, il y a ce qui suit pour le calcul de la poussée statique. Le diamètre de l'hélice,  $d$ , et le pas de l'hélice,  $Pas$ , étant en unités de pouces, et la poussée,  $F$ , sortant en unités de newtons (N). Le 0,0254 est un facteur de conversion pour convertir les pouces en mètres, car il y a 0,0254 m/in. Et la densité de l'air  $\rho = 1,225 kg/m^3$ .

$$F(N) = 1,225 \frac{\pi(0,0254 \cdot d(in))^2}{4} (V_{rot}(rpm) \cdot 0,0254 \cdot Pas(in) \cdot \frac{1 \text{ min}}{60 \text{ sec}})^2 \quad (9)$$

Afin de terminer le développement mathématique, il est nécessaire de choisir un modèle d'hélice à être utilisé dans le calcul. Le drone quadrirotor choisi comme référence est le « Sky Hero Spyder » créé par Sky-Hero, il est un drone capable de supporter plus de 7kg de charge et batteries. Il utilise des hélices de fibre de carbone avec la configuration 10x5'', c'est-à-dire 10 pouces de diamètre et 5 pouces de pas.



Figure 40 : Concept du Sky Hero Spyder.

Finalement, la formule utilisée dans ce projet dans la partie avec le contrôle sans stabilité créée en avance est la suivante :

$$F(N) = 1,225 \frac{\pi(0,0254 \cdot 10(in))^2}{4} (V_{rot}(rpm) \cdot 0,0254 \cdot 5(in) \cdot \frac{1 \text{ min}}{60 \text{ sec}})^2$$

$$F(N) = 0,30625 \cdot \pi(0,254)^2 \cdot (V_{rot}(rpm) \cdot \frac{0,127}{60})^2 \quad (10)$$

Avec cette expression, le contrôle va consister en recevoir de l'API python une valeur désiré de vitesse de rotation et le code dans l'unity va faire la conversion vers la poussée réalisée par chaque hélice du drone. La force gérée sera appliquée sur chaque hélice, en utilisant un vecteur vers haut avec la magnitude obtenue avec l'équation 10 (Eq10), le vecteur sera appliqué sur le point central d'hélice correspondant, avec un système de coordonnées local (en utilisant le « GameObject » du drone comme référence). Il est important d'utiliser le système local du drone, parce que le vecteur de poussée d'une hélice réel va être toujours orienté vers la partie en dessus d'hélice. Se la référence locale n'est pas utilisée, le vecteur va toujours s'orienter vers haut et peut être une source grande d'erreur.

De cette façon, en utilisant cette technique, il est possible de contrôler individuellement chaque hélice comme dans un drone réel. Le méthode pour le faire s'appelle « .AddForceAtPosition » et vous pouvez choisir comme paramètres du méthode le vecteur force et la position d'application. Dans ce cas, le vecteur est un vecteur vers l'orientation Y du drone avec la magnitude obtenue dans l'équation précédent et la position d'application est le centre de chaque hélice.

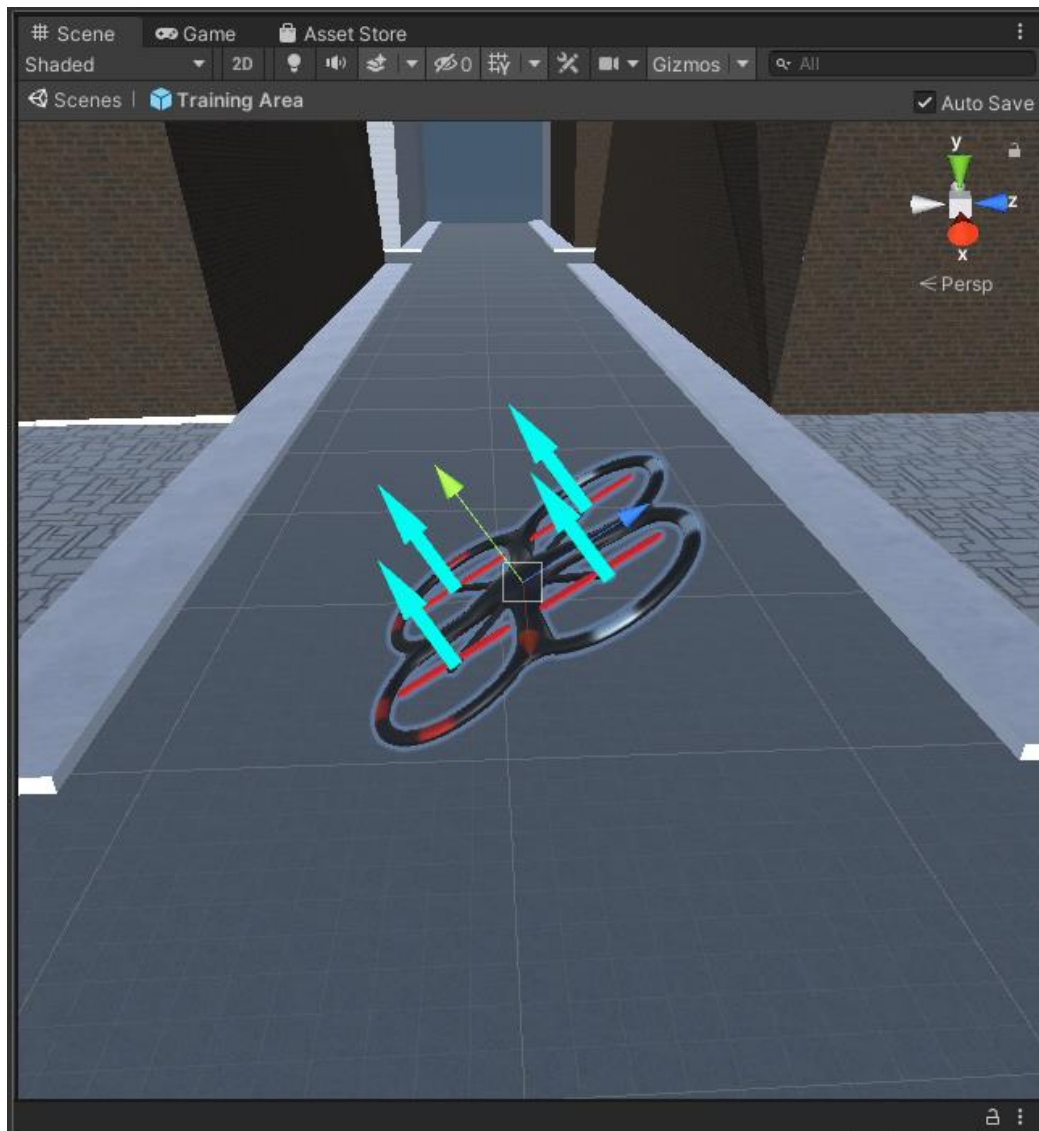


Figure 41 : Image qui montre les vecteurs de force en agissant sur chaque hélice.

### 6.2.1. Récompenses

Maintenant, plus qu'avant, le système de récompenses a une fonction très importante. Il va interférer directement au contrôle de stabilité parce que maintenant l'API python va être responsable pour le développer pendant l'enseignement.

Pour être sûr que l'agent va développer un contrôle de stabilité, il faut créer un système de punition et récompense efficace et effectif. Si le drone perd de stabilité, il faut d'avoir une punition un peu plus haut que le normal et une récompense constante si la stabilité est maintenue.

Donc, les points de récompense sont :

- S'il se déplace vers le « Target » ;

- S'il arrive à la position du « Target » ;
- S'il maintient l'inclinaison des axes horizontaux entre -30 et 30 degrés ;
- Si l'accélération angulaire actuelle sur les axes horizontaux est inférieure à l'accélération angulaire mesuré dans une itération avant.

Et les points de punitions sont :

- S'il entre en collision avec une structure ;
- S'il s'éloigne du « Target » ;
- Une ponctuation négative pour chaque étape qu'il effectue afin de diminuer le nombre d'étapes nécessaires pour atteindre la destination ;
- S'il ne maintient pas l'inclinaison des axes horizontaux entre -30 et 30 degrés ;
- Si l'accélération angulaire actuelle sur les axes horizontaux est supérieure à la précédente.

### **6.2.2. Les tests**

Dans les tests dans ce deuxième contrôle, le paramètre plus considéré été la stabilité du drone et la compétence d'être capable d'arriver à la destination sans rien toucher.

Il n'avait pas vraiment une manière de mesurer la capacité d'arriver à la destination sans rien toucher en graphique et les montrer ici, mais il n'y a pas une raison pour le faire une fois que aucun entraînement a réussi à le faire.

Chaque entraînement a pris en environ 6h avant que l'auteur l'abandonne et décide de commencer un autre épisode. Mais, en regardant les graphiques de récompenses, il est possible de constater qu'après en environ 1h ou 2,7 millions de pas il commence à converger.

Cumulative Reward  
tag: Environment/Cumulative Reward

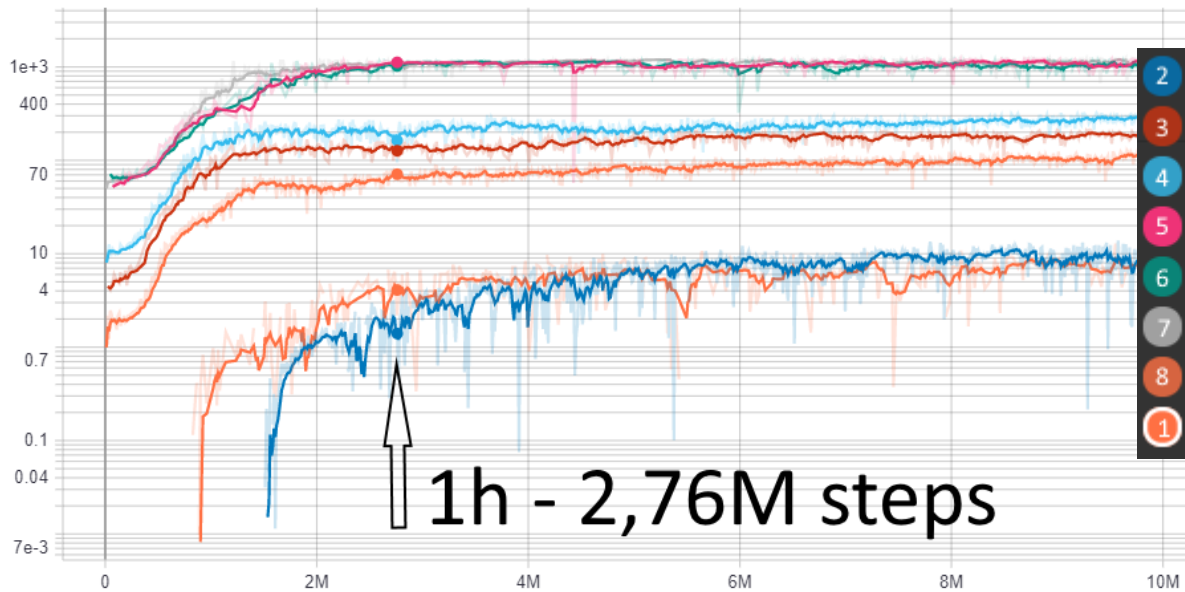


Figure 42 : Graph de récompenses cumulés par quantité de pas. L'axe Y a été mis en échelle logarithmique pour permettre la visualisation simultanée de tous les graphs et les comparer. Les couleurs entre 8 et 1 sont très proche, donc je vous explique que le graph 8 est le graph le plus bas.

Il est compliqué de se baser sur les graphiques de récompenses parce que quand il y a un changement dans les valeurs de récompenses, le graphique va changer complètement. Vous pouvez comparer des exemples :

Cumulative Reward  
tag: Environment/Cumulative Reward

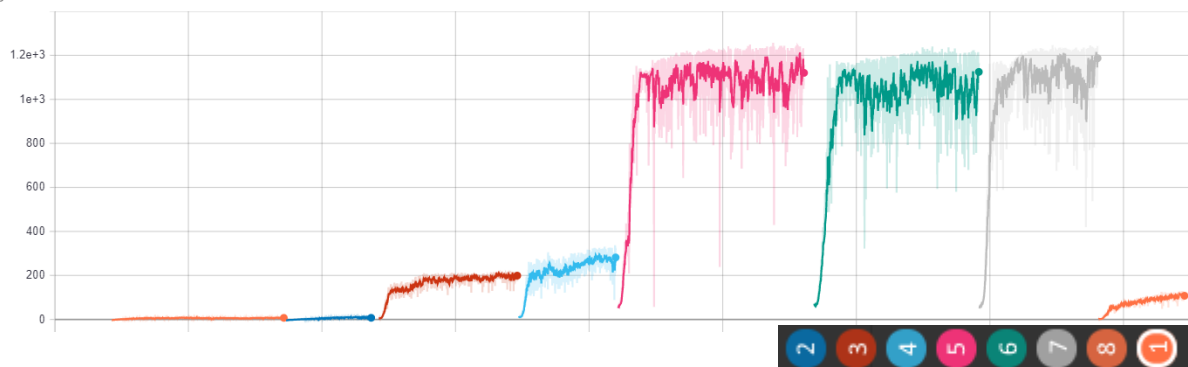


Figure 43 : Graph de récompenses cumulés par quantité de pas. L'axe de quantité de pas est déformé afin de comparer mieux la valeur de récompense dans chaque test. Le graph 1 est le dernier.

Les paramètres changés étés plutôt les hyperparamètres et les récompenses, mais même après plusieurs tentatives, le drone n'arrive pas à une bonne stabilité et, en conséquence, il va toujours faire une collision. Vous pouvez trouver à suivre les coordonnées relatives aux graphiques montrés.





Figure 44 : Légende des figures précédents.

- 1)En suivant le modèle montré dans le chapitre 5, Hyperparamètres.
- 2)buffer\_size = 102400. Après le changement de buffer\_size, une baisse considérable de performance a été observé dans l'ordinateur. Donc dans le test prochain il faut diminuer la valeur.
- 3)buffer\_size = 61440. Augmentation de la punition pour collision et de la récompense pour « douceur ».
- 4)Augmentation de la récompense pour arriver au « Target » et de la punition pour une grande quantité de pas.
- 5)Lambda 0.99 -> 0.98. Augmentation de la récompense pour « douceur » et pour maintenir la stabilité.
- 6)Diminution de la punition pour collision et augmentation de la récompense pour arriver à la destination.
- 7)Epsilon 0.3->0.2
- 8)Diminution de toutes les récompenses et punitions. Mais les hyperparamètres étaient gardés.

Vous pouvez accéder à plusieurs graphiques qui montrent la progression et efficacité de l'apprentissage. Mais en les regardant, vous pouvez constater qu'il n'avait pas beaucoup de changement sauf les graphiques de récompenses. Il avait aussi quelque changement dans le déclin de l'entropie, mais c'était aussi petit qu'il n'y a pas vraiment une considération à être fait.

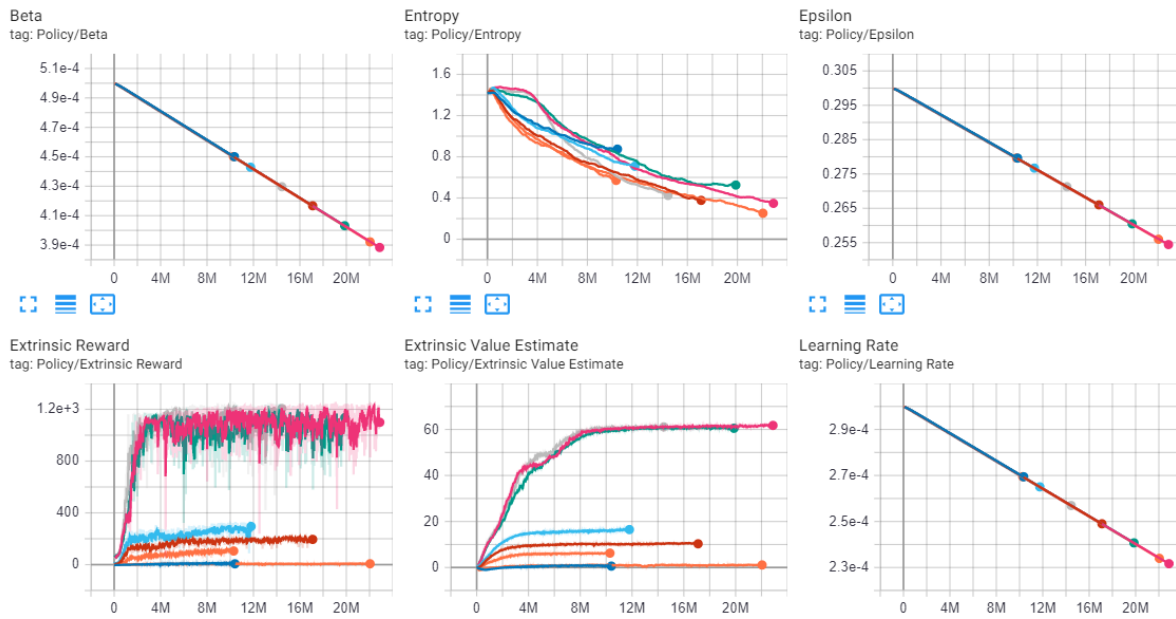


Figure 45 : Plusieurs graphes en comparant les tests. Vous pouvez regarder une différence considérable juste dans les graphes de récompenses.

Il avait plusieurs modifications et tentatives, mais aucune n'a réussi à une bonne stabilité et à éviter des collisions.



## CHAPITRE 7. CONCLUSION

Après vérifier les deux contrôles, il est possible de constater que le premier contrôle été plus satisfaisant dans l'objectif de faire une délivrance. Avec un contrôle de stabilité indépendant, la stabilité est garantie et le drone n'ira pas arriver à faire une collision.

Il probablement coûtera plus cher que le deuxième et utilisera plus pouvoir de computation du microcontrôleur du drone, mais la simulation montre qu'il est plus satisfaisant.

Une autre situation observée est la difficulté de valider un épisode d'apprentissage, parce que les validations sont toujours basées sur des graphiques de récompenses cumulés. Si vous changez le système de récompenses, il est presque impossible avoir une comparaison graphique. Donc l'unique comparaison vraiment effective est la comparaison visuelle dans l'environnement Unity.

Un point important à considérer est que ce projet avait un objectif plutôt académique et l'objectif d'orienter prochains étudiants ou professeurs dont l'envie est d'explorer ce thème. Donc malgré ce projet peut être le départ pour un projet beaucoup plus ambitieux, un projet de créer un vrai drone dont le software de contrôle est entraîné dans une réalité virtuelle, le projet est encore beaucoup loin de cette réalité. Il y a beaucoup des points qu'il faut considérer avant affirmer que la simulation représente bien la réalité, points comme personnes, oiseaux, avions, réseaux électriques, véhicules et services désirés par les clients, comme par exemple s'il est désiré que le drone reste arrêté quelques minutes avant une fenêtre jusqu'à l'arrivée du client ou quoi il doit faire si le package est perdu pendant le chemin. Il y a beaucoup de détails à penser que ne sont pas abordés ici mais qui peuvent être pensés à la future.



## BIBLIOGRAPHIE

**Juliani, A., Berges, V., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D.** *Unity: A General Platform for Intelligent Agents*. 2020 <https://github.com/Unity-Technologies/ml-agents>.

**G. Staples.** *Propeller Static & Dynamic Thrust Calculation*. <https://www.electricrcaircraftguy.com/2013/09/propeller-static-dynamic-thrust-equation.html>

**Sky-Hero.** *Sky Hero Spyder X8 ARF*. 2020 <https://www.onedrone.com/store/sky-hero-spyder-x8-arf.html>

**NullPointerException.** *3D Printed Quadcopter*. 2015. <https://www.instructables.com/id/DIY-Quadcopter-Suggestions-OR-What-can-go-wrong/#discuss>

**Dayarian, I., Savelsbergh, M., Clarke, J.-P.** *Same-day delivery with drone resupply*. 2017. [http://www.optimization-online.org/DB\\_FILE/2017/09/6206.pdf](http://www.optimization-online.org/DB_FILE/2017/09/6206.pdf).

**T. Kirschstein** *Transportation Research Part D: Transport and Environment, Volume 78*, January 2020. <https://doi.org/10.1016/j.trd.2019.102209>.

**C. Cheng, Y. Adulyasak, L. Rousseau.** *Drone routing with energy function: Formulation and exact algorithm*. *Transportation Research Part B: Methodological* Volume 139, September 2020, Pages 364-387. Polytechnique Montréal and CIRRELT, Montréal, H3C 3A7, Canada. Septembre, 2020. <https://doi.org/10.1016/j.trb.2020.06.011>