# FICHE : PoC & Choix Technologiques

**=> Après avoir rempli cette fiche :convertir ce document en PDF et le téléverser dans votre dépôt Github dans un répertoire nommé 'doc'.**

## Compte rendu de la veille technologique PoC

## Questions de RECHERCHE sur la PoC

Je pense que la difficulté majeure de la preuve de concept va être de faire communiquer Python (qui gère les modèles d'apprentissage) et Unity (le moteur de jeu 3D). Ce n'est pas un problème en soi car je compte utiliser des outils qui sont spécifiquement conçus pour ça. Cependant, ils ont l'aire plutôt complexe a mettre en place et à prendre en main.

## PoC = Preuve de concept

**Quel genre de preuve de concept minimale pourrait valider que le problème n'existe pas ou qu'une solution a été trouvée ? Décrivez chaque élément du code requis.**

Pour commencer, je pense utiliser python et Pytorch pour contrôler une fonctionnalité simple de mon projet dans Unity. Par exemple, je pourrais implémenter le déplacement d'un GameObject dans un environnement 3D qui serait réactif à la présence d'obstacles.

## Les marque-pages identifiés lors de vos recherches

Lien vers une page publique contenant vos marque-pages collaboratifs ou lister les marque-pages directement ici. Pour chaque lien : URL, nom de la page et description sommaire.

Page Notion :

https://www.notion.so/bd150619fc814bf9ae140ab82ad7ea0e?v=34fb4ad10743428db0501a4d9a7bae6b

# Les preuves de concept

Pour chaque preuve de concept réalisée : identifier le but de la preuve de concept (ce qu'elle vérifie), le lien vers le sous-répertoire de votre dépôt GitHub qui contient le code de la preuve de concept ainsi que les résultats de votre expérimentation, puis, finalement, vos conclusions.

Au moins une preuve de concept doit être documentée et réalisée.

# Première POC réalisée

| | |
|---|---|
| **Preuve :** | Unity peux interagir avec un modèle d'apprentissage en python |
| **URL Github :** | https://github.com/cegepmatane/projet-specialise-2022-Matheo-Galuba/tree/PoC1/PoC |

# Expliquez votre Poc

**Décrivez la Poc en détail.**

L'objectif est de mettre en place l'environnement nécessaire pour que Unity puisse interagir avec Python avec la librairie "Unity ML agent". On souhaite aussi que le modèle soit fonctionnel avec un certain nombre d'entrées (capture de l'environnement 3D dans Unity) et de sorties (actions concrètes dans Unity).

Dans le détail, un gameobject unity sera conditionné à avancer toujours tout droit et pourra sonder les obstacles devant lui au moyen d'un raycast. La partie intelligence artificielle en Python devra lui faire éviter les obstacles.

**Que PROUVE la Poc ?**

Cette PoC prouve que Unity peut échanger des données avec le modèle d'apprentissage de manière bilatérale. C'est-à-dire que le modèle peut connaître des informations de la part de l'environnement 3D dans Unity (ex : distance d'un obstacle devant, coordonnées, etc…) et agir en conséquence avec des actions prédéterminées (ex : avancer, reculer, etc…)

**Que reste-t-il à prouver ?**

La partie évolutionniste de mon projet : un algorithme de sélection naturelle qui réitère l'expérience sur des générations uniquement constitué du meilleur modèle de la génération précédente avec des altérations (mutations) aléatoires.

**Quels sont vos résultats de la Poc ?**

# Première technologie sélectionnée (la nouvelle)

**Technologie :** Librairie d'intelligence artificiel PyTorch pour Python

**URL :** https://pytorch.org/

## Justifier votre choix technologique pour cette technologie

**Expliquer à l'aide d'une argumentation rationnelle votre choix technologique. Établir votre justification à l'aide de liens avec les fonctionnalités, contraintes et risques de votre projet. Un tableau comparatif permettant de synthétiser votre réflexion pourrait être un apport judicieux à vos explications.**

Il existe principalement deux librairies python qui ont pour but de créer des modèles d'apprentissage profond dans Unity : PyTorch et TensorFlow.

Je n'ai utilisé aucune de ces deux librairies par le passé mais au cours de mes recherches sur l'intégration d'IA dans Unity, j'ai trouvé plus de résultats pertinents avec PyTorch qu'avec Tensor FLow. De plus, d'après des articles comparatifs, PyTorch serai un peu plus facile a prendre en main que TensorFlow ce qui a fini de déterminer ma décision.

J'ai vraiment réfléchi sur ces deux libraires qui sont vraiment très similaires donc j'ai décidé de faire ma fiche de comparaison dessus. Cf : Ma fiche de comparaison : https://docs.google.com/presentation/d/1I9EQ0xb6P17-xXolSpcOsdqIueDFJLrxob3Lt8X8flo/edit?usp=sharing

# Deuxième technologie sélectionnée (la connue)

**Technologie :** Unity

**URL :** https://unity.com/fr

## Justifier votre choix technologique pour cette technologie

**Expliquer à l'aide d'une argumentation rationnelle votre choix technologique. Établir votre justification à l'aide de liens avec les fonctionnalités, contraintes et risques de votre projet. Un tableau comparatif permettant de synthétiser votre réflexion pourrait être un apport judicieux à vos explications.**

Pour le choix du moteur graphique de mon projet, il y avait plusieurs solutions : Unity, Unreal Engin, Godot, et créer un système de rendu moi même.

J'ai choisi Unity car il intègre des librairies spécifiques pour l'intelligence artificielle et peut interagir avec un modèle d'apprentissage Python. Ces deux technologies sont massivement utilisées donc je dispose de beaucoup d'aide de la part de la communauté. Le scripting en C# est puissant et permet de la flexibilité contrairement au système de programmation nodale de Unreal Engin que je trouve personnellement très bien pour ces jeux "classiques" mais limitant dans mon cas.

Enfin, j’ai éliminé la dernière option par économie de temps.