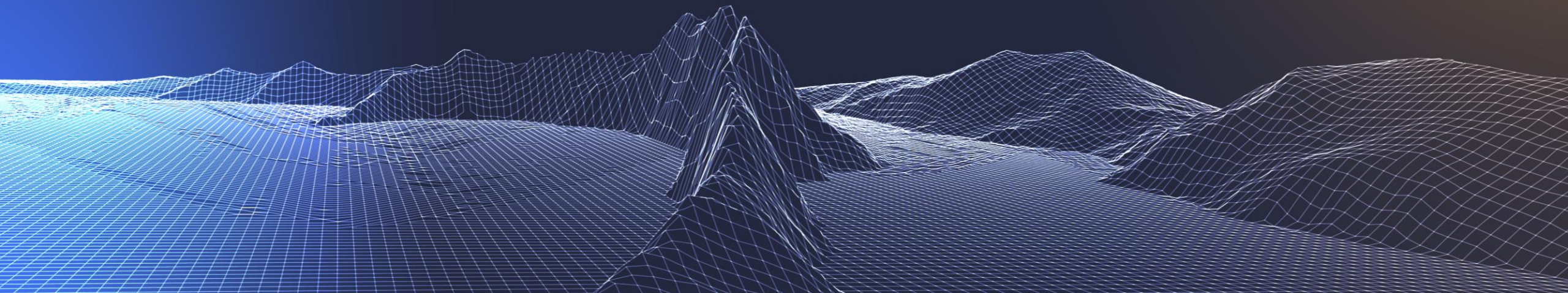




Présentation Drlfox
Delfox – Dassault

13/03/2026



1. **Présentation de Delfox**
2. **Historique & Contexte**
3. **Construction de DRLfox**
4. **Fonctionnalités principales**
5. **Modes de groupe**
6. **Interfacage simulateurs**
7. **Démo live**

➤ **DELFOX**

- Abderrazzak Sabri
- Maxime Rey
- Kévin Jolly
- Guillaume Blaclard

➤ **Dassault**

- À compléter



Pionnière européenne de l'IA par renforcement profond, Delfox développe des solutions embarquées **permettant à vos systèmes d'agir de manière autonome, adaptative et coordonnée** dans des environnements opérationnels complexes.

Expert en intelligence artificielle, Delfox propose **des solutions à fort impact** pour aider ses clients à **relever des défis complexes** et intégrer l'IA efficacement au sein de leur organisation.

Soutenu par deux grands acteurs de la défense.



100 french best inventors



100 startups to invest



Challenge Open Innovation



Year AI Start-up



MMT Batch I & II



- Premier framework interne : **RLFW**, construit sur **RLlib**.
 - → **Problème : trop abstrait, difficile d'y développer de nouvelles techniques RL avancées**
- Besoin identifié :
 - **Modularité** pour créer et tester facilement de nouveaux algorithmes
 - **Vectorisation GPU** pour accélérer l'entraînement
 - **Flexibilité multi-agent** pour gérer plusieurs entités en parallèle
- Analyse de l'état de l'art:
 - **SB3** : simple mais trop limité (pas de multi-agent)
 - **RLlib** : Algorithmes de bases implementés, mais lourd et rigide
 - **Tianshou** : vectorisé, mais incomplet
 - **TorchRL** : intéressant, car propose des briques utiles (buffers, rollouts, fonctions de loss, politiques d'exploration...) mais ne réponds pas au besoin directement

- Approche Delfox : capitaliser sur **les acquis de RLFW** et exploiter **les briques TorchRL**
- Migration et amélioration des techniques déjà présentes dans RLFW :
 - **Curriculum learning, apprentissage hiérarchique**, multi-agent
 - Optimisation et **vectorisation GPU** du code existant
- Ajouts spécifiques Delfox :
 - Nouvelles méthodes: **self-play, CTDE, orchestration par groupes**
 - **API Gym généralisée** pour connecter facilement différents simulateurs
- Résultat : une **suite modulaire et performante**, combinant l'expérience RLFW et la puissance de TorchRL, prête à évoluer vers des cas industriels

- **Algorithmes disponibles** : PPO, DDPG, SAC, Behaviour Cloning (imitation learning), Self-play, Curriculum Learning, CTDE
- **Types de politiques** : MLP, politiques scriptées ou statiques, politiques mixtes, aléatoires ; extensibles à CNN et Transformers
- **Gestion multi-agent par groupes** : orchestration centralisée via TensorDict, mélange de politiques possibles par sous-environnement
- **Support complet des espaces d'actions** : discrets, multi-discrets, continus ou hybrides
- **Code vectorisé et optimisé GPU** : parallélisation native pour accélérer l'entraînement et gérer de grands volumes de simulations/données
- **Monitoring et suivi** : Tensorboard, callbacks configurables, enregistrement des métriques et modèles

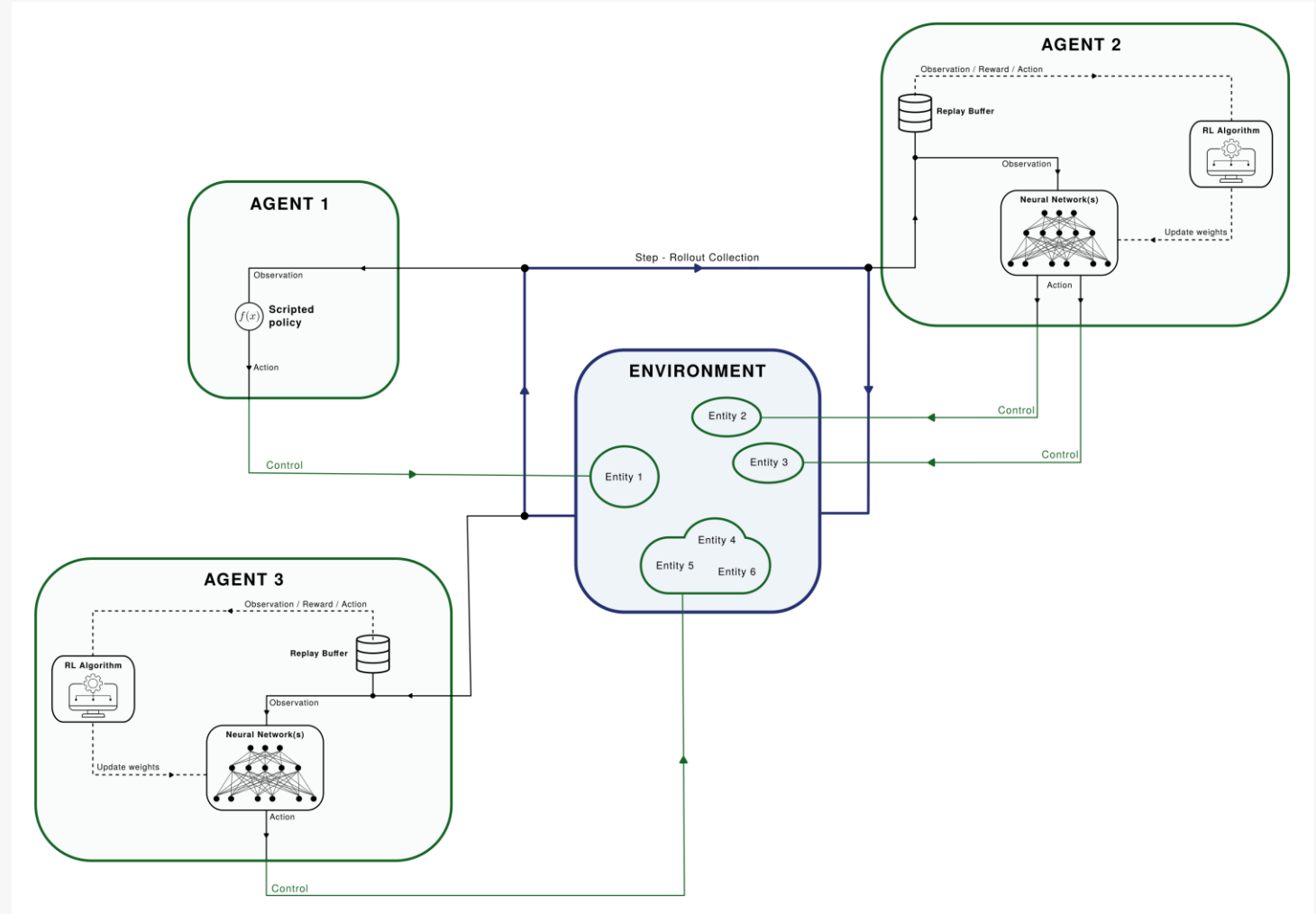
Discussion

Modes de groupe

Dans drlfox, à un Agent RL est associé :

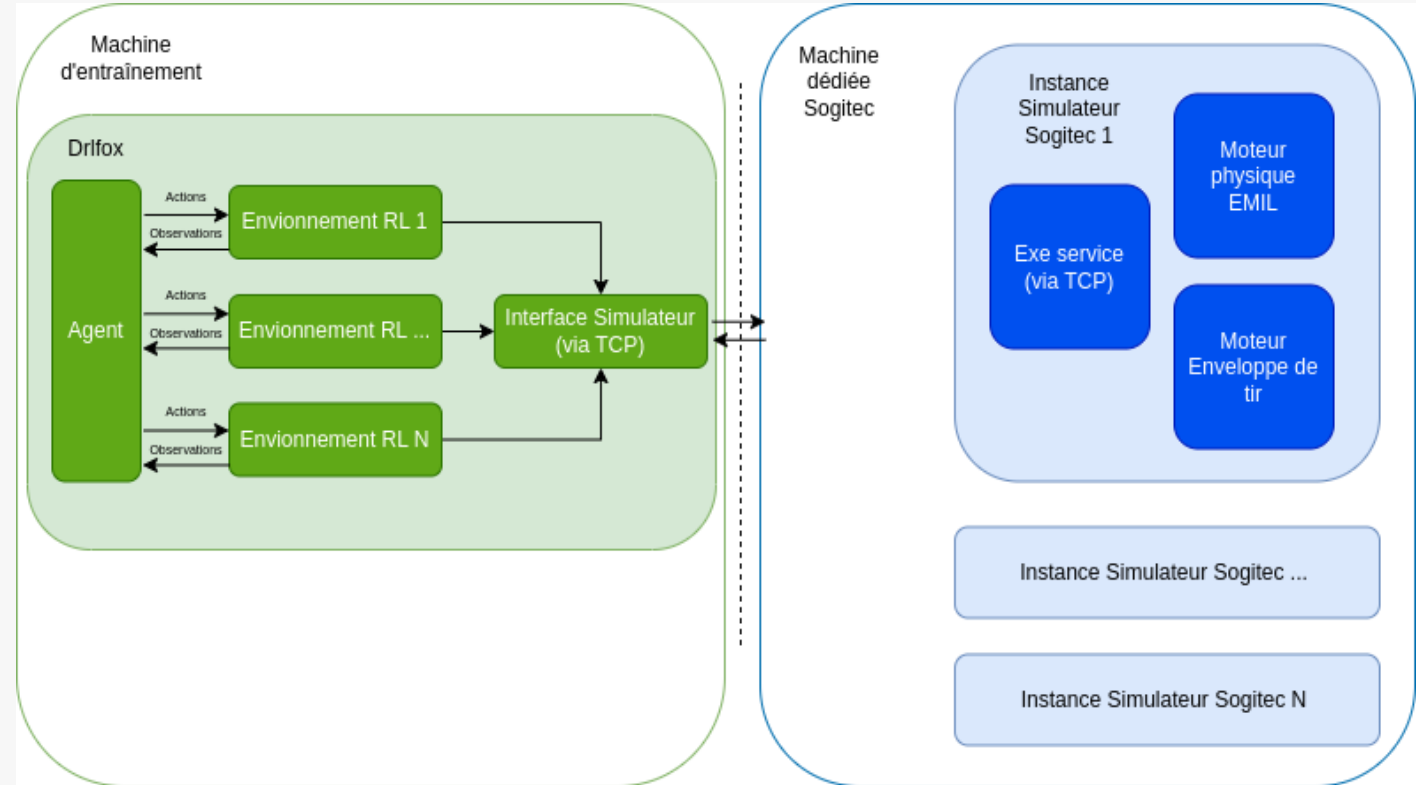
- Une **politique** (scriptée, réseau...)
- Un **replay buffer**
- Un **optimizer** (Adam, ...)
- Un **algorithme** d'entraînement (PPO, DDPG...)
- Des **callbacks** spécifiques (Tensorboard...)
- Un **ID** de groupe utilisé par les TensorDict

MAIS un Agent RL n'est pas **pas présent** dans l'env directement



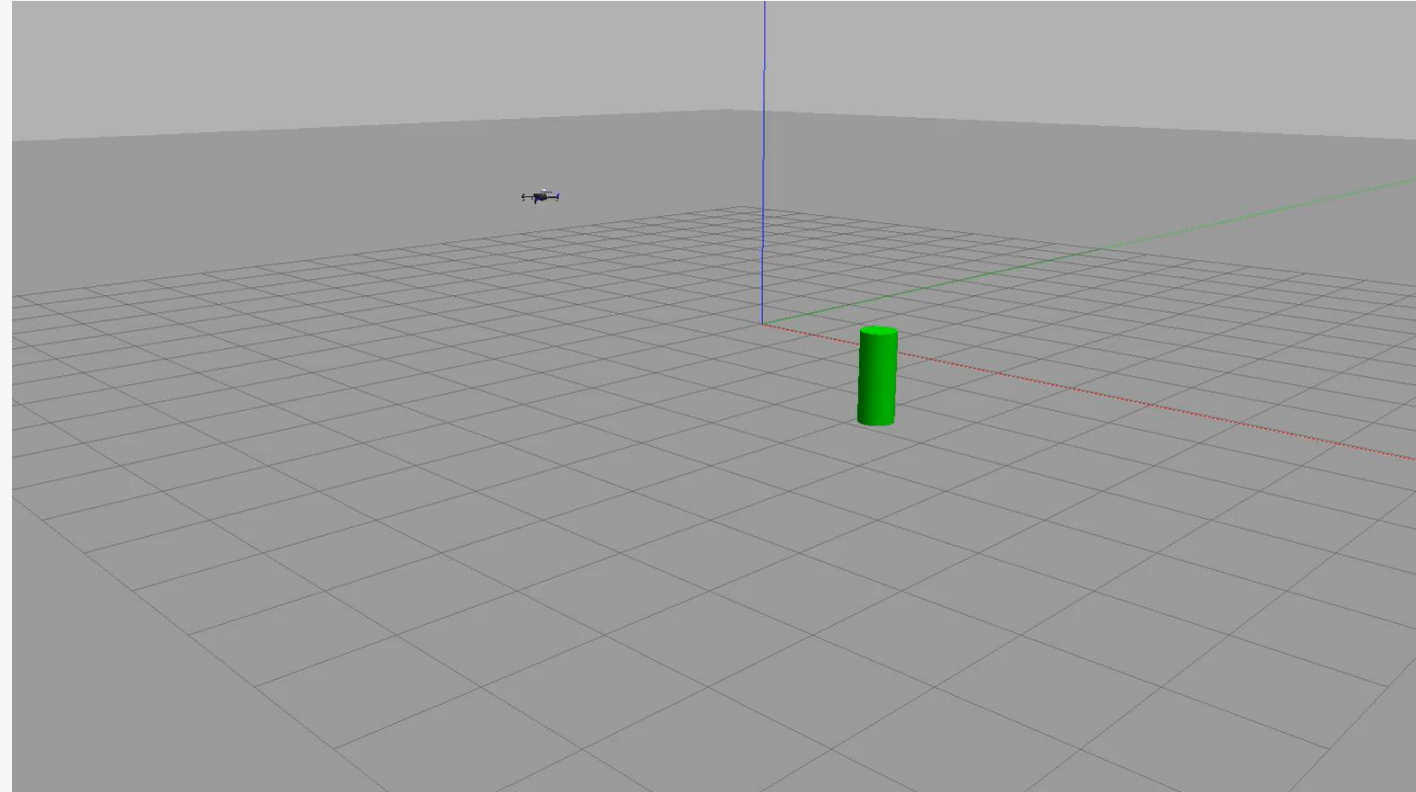
Entraînement avec DRLFox et le simulateur Sogitec:

- L'agent est entraîné à partir d'un **simulateur tiers** fourni par **Sogitec**.
- **Plusieurs environnements RL** sont exécutés **en parallèle** sur la machine d'entraînement (DRLFox).
- **Chaque environnement RL** intègre une **interface TCP** qui communique avec **sa propre instance** du simulateur Sogitec, hébergée sur **une machine distante dédiée**.
- L'**environnement RL** joue le rôle de **wrapper** : il **contient la logique d'orchestration** pour démarrer/arrêter les instances côté Sogitec et **gérer les échanges réseau**.



Entraînement avec DRLFox et le simulateur Gazébo:

- **Exemple** : agent entraîné **1h30** avec DRLfox sur Gazebo+PX4 (vidéo).
- L'agent est entraîné dans un environnement réaliste avec **Gazebo** (physique 3D) et **PX4** (autopilote).
- L'interface passe par **ROS + MAVROS**, reliant actions envoyées par DRLfox et observations issues des capteurs simulés.
- Plusieurs instances Gazebo+PX4 peuvent tourner en parallèle, chacune reliée à un agent DRLfox.
- DRLfox orchestre :
 - envoi des commandes vers PX4
 - lecture des capteurs via ROS
 - calcul des récompenses et apprentissage RL
- **Résultat** : entraînement fluide et stable, même dans des environnements complexes.



- **Exécution complète sur GPU** : DRLfox vectorise les calculs et exploite la parallélisation native de PyTorch
- **Simulation + entraînement** : si le simulateur est compatible GPU, DRLfox s'aligne et permet une chaîne 100% accélérée
- **Scalabilité** : support d'environnements parallèles (ex. 5, 50, 500) avec remplissage rapide des buffers d'expérience
- **Optimisation des rollouts** : collecte d'échantillons efficace, même en grande dimension
- **Résultat** : réduction drastique des temps d'apprentissage et meilleure reproductibilité des expériences

Discussion

Démo pratique

Live coding



