

Introduction to

SELF-DRIVING

WITH CARLA

Start





Agent de conduite autonome basé sur l'apprentissage par renforcement

- Déplacer en autonomie
- Adapter sa vitesse à la densité du trafic
- Gérer l'environnement Carla





- L'utilisation des caméras RGB va être trop gourmande
- Environnement Carla est trop lourd pour effectuer beaucoup d'épisode d'entraînement
- Focaliser sur la qualité de la reward et la sélection du modèle pour apprendre avec le moins d'épisodes possible





Environnement de conduite urbain virtuel

Utilisation au cours du projet de plusieurs outils nous donnant des observations

- capteur position
- capteurs vitesse
- capteurs collision
- lidar
- waypoint
- caméra rgb





LES OBSERVATIONS

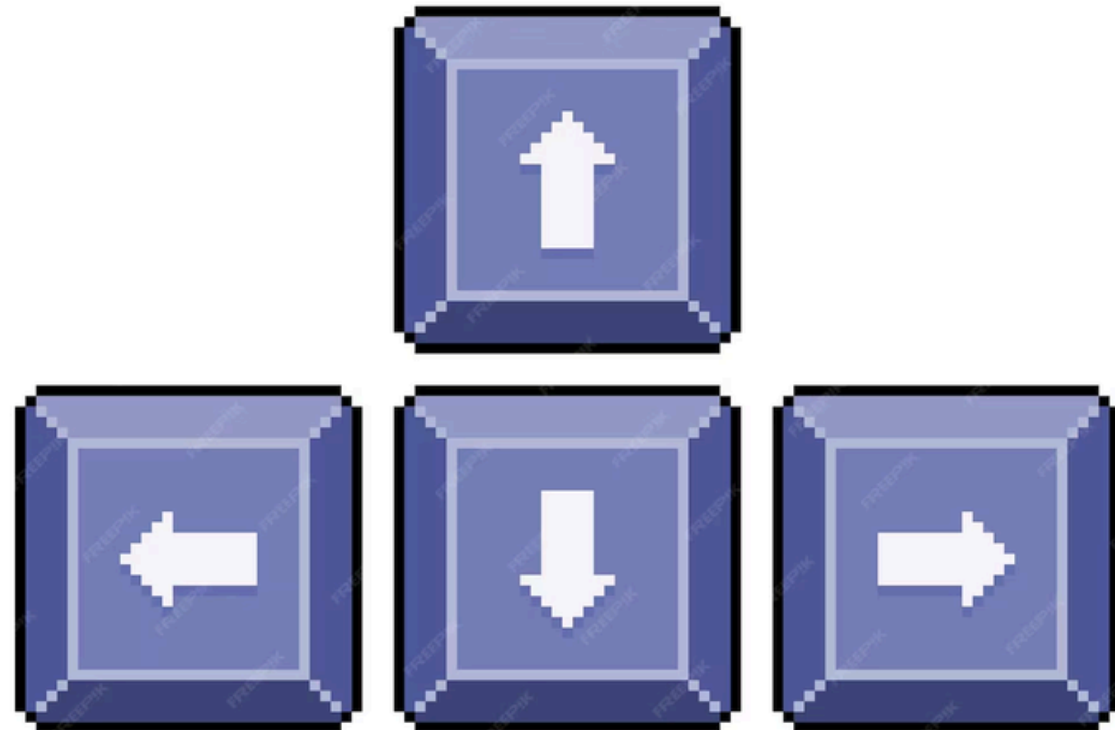
37 observations finales (inputs finales)

- 16 distance mini (lidar)
- 16 distance moyenne (lidar)
- 1 collision
- 1 position
- 1 vitesse
- 1 lane offset
- 1 lane angle





LES ACTIONS



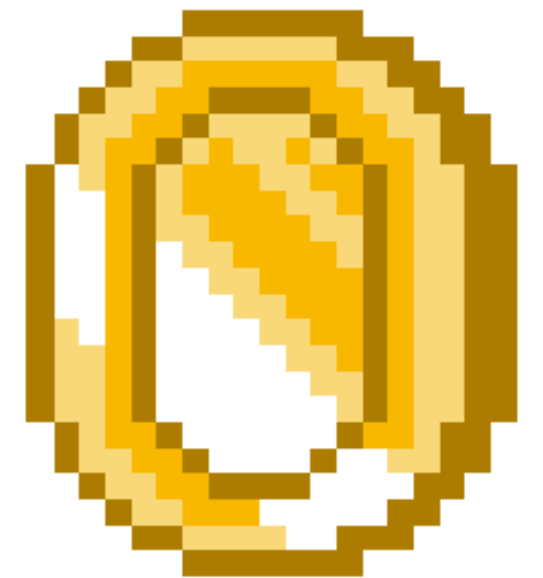
- direction (-1 à 1)
- throttle (0 à 1)
- brake (0 à 1)



COMPOSITION RÉCOMPENSE

Récompense à chaque step :

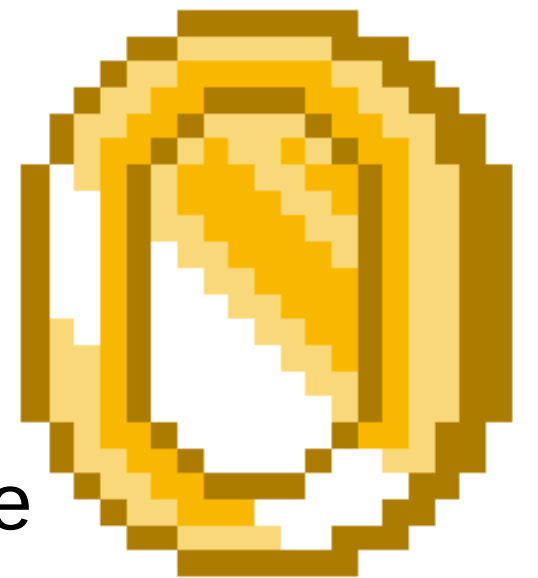
- **Base** : favorise la survie
- **Lane_keeping** : favorise min lane offset et lane_angle
- **Consistency_bonus** : favorise lane_keeping soutenue
- **Exploration** : favorise la distance par rapport au spawn



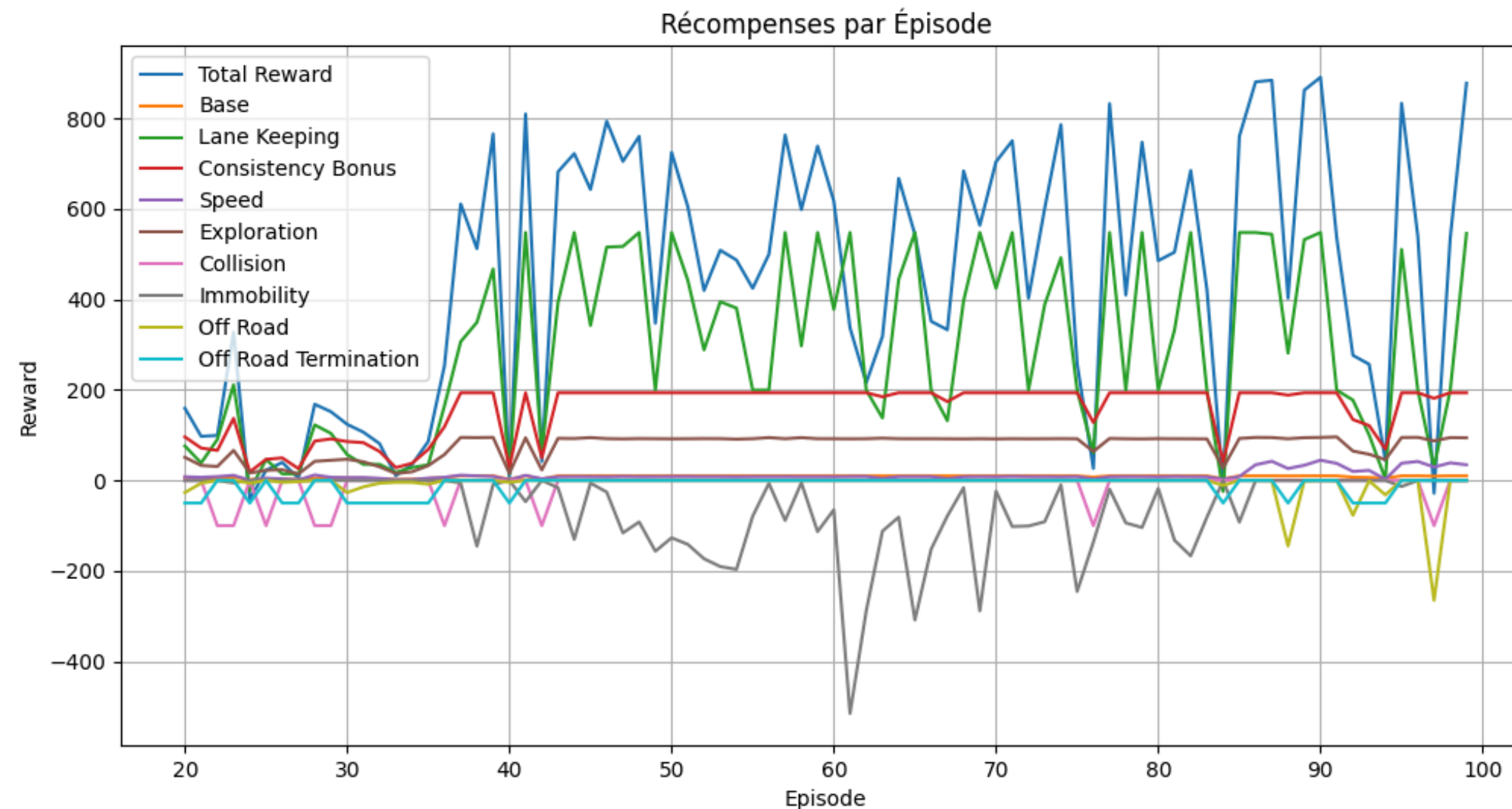
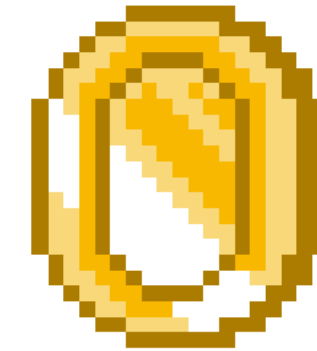
COMPOSITION RÉCOMPENSE

Pénalités critiques :

- **Collision** : pénalise les collisions
- **Immobility** : pénalise l'immobilité excessive
- **Off_road** : pénalise nb step trop loin ou desaxé de la voie
- **Off_road_terminaison** : pénalise fortement sortie de route



EVOLUTION RÉCOMPENSE



Mise en place d'un clipping pour stabiliser la récompense à chaque step et éviter les valeurs extrême, non représenté dans le graphique

Le graph montre entrainement d'un modèle selectionné pour sa convergence rapide après les 10 premiers épisodes

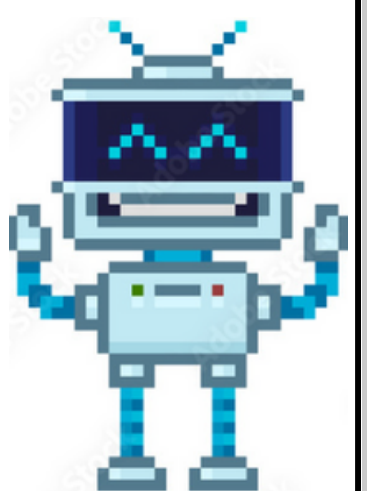
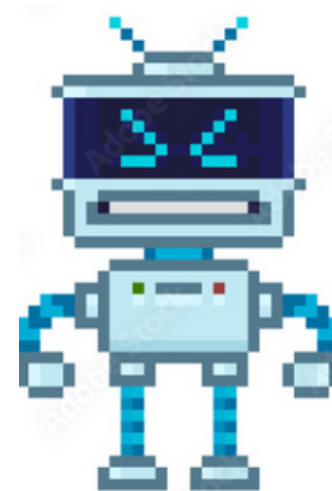
DÉCOUVERTE DE CARLA

EVOLUTION DU PROJET & MODÈLE

- Etude de l'environnement
- Contrôle de Carla et Blueprint
- Spawn voiture, caméra et contrôle

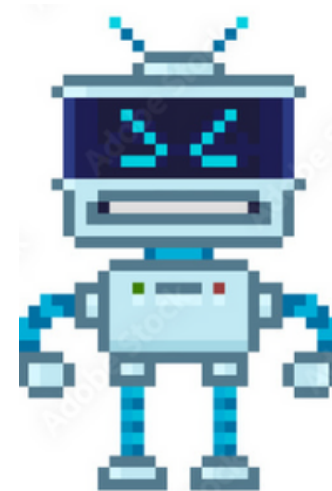
Difficulté à faire tourner
l'environnement sur nos machines

Entraînement pas facile mais optimisable





EVOLUTION DU PROJET & MODÈLE

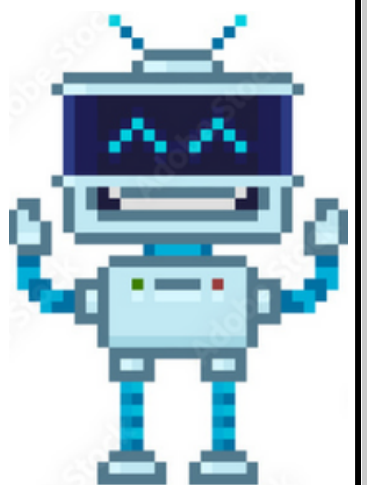


PREMIERS MODÈLES

- Structure l'environnement (code)
- Ajout de capteur sur la voiture (lidar, caméra rgb, ect)
- Utilisation d'un DQN + Replay buffer et encodeur pour lidar
- Mise en place de boucle d'entrainnement et graph pour suivi évolution reward

Difficultée de trouver de la performance avec DQN
Modèle vue en cours insuffisant dans notre situation pour la V0

Nécessaire de chercher un modèle plus efficient car entrainnement DQN insuportable par nos machines



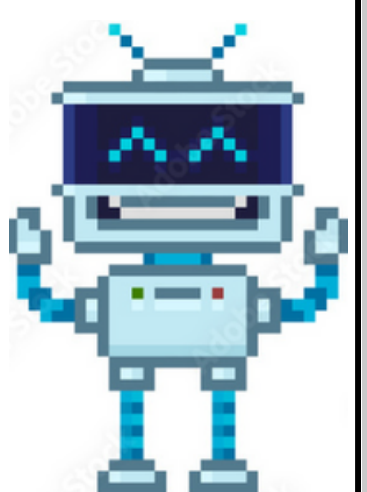
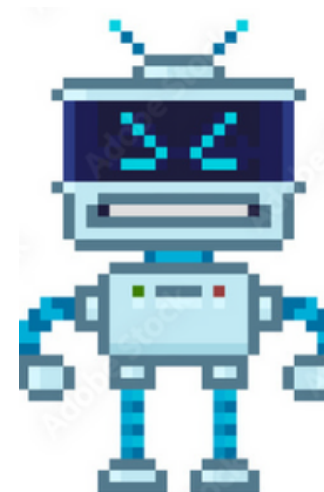
MODÈLE PLUS EFFICIENT

- Recherche d'un nouveau modèle et approche
- Selection d'un modèle policy gradient (TD3 / SAC / PPO)
- Suppression de l'encodeur du lidar et caméra rgb et simplification lidar
- Intégration de lane_offset et lane_angle
- Optimisation charge GPU et CPU

Iteration d'entraînement pour rechercher l'efficience mais toujours pas mal de difficulté. Perte de motivation car loin du livrable finale et la convergence de nos modèles résulte plus de la chance que de l'optimisation de la reward

Passage de la V0

EVOLUTION DU PROJET & MODÈLE



DERNIÈRE OPTIMISATION



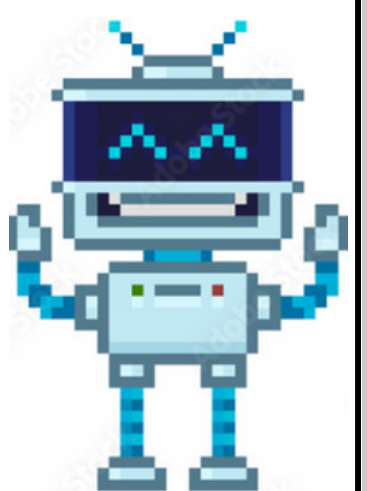
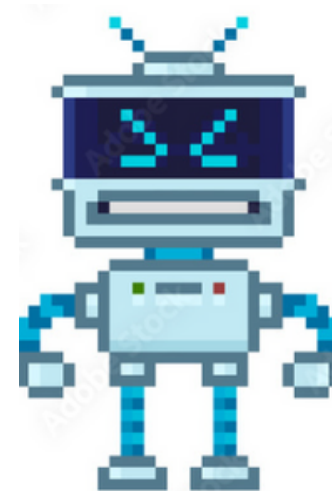
EVOLUTION DU PROJET & MODÈLE



- Modification légère des poids de la reward
- Lancement d'entraînement en boucle pour espérer avoir le modèle le plus performant possible
- Essaie de la V1

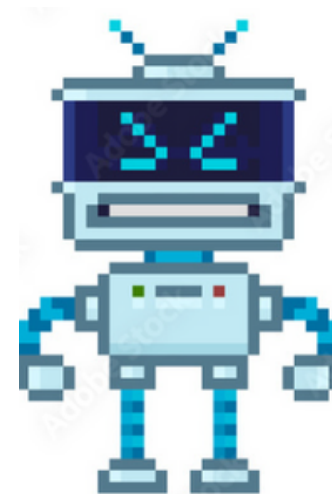
Plus le modèle est performant plus sa vitesse et les collisions qu'il prend sont intenses en s'entraînant

=> Arrête l'entraînement car le pc bloque à cause des calculs





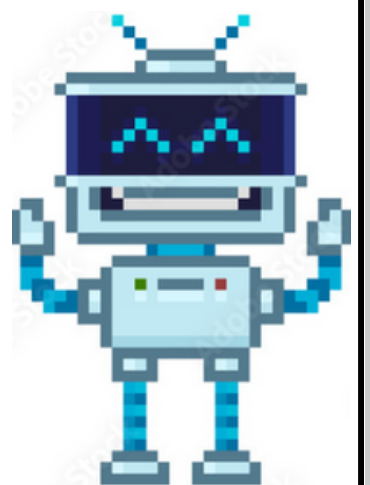
PPO

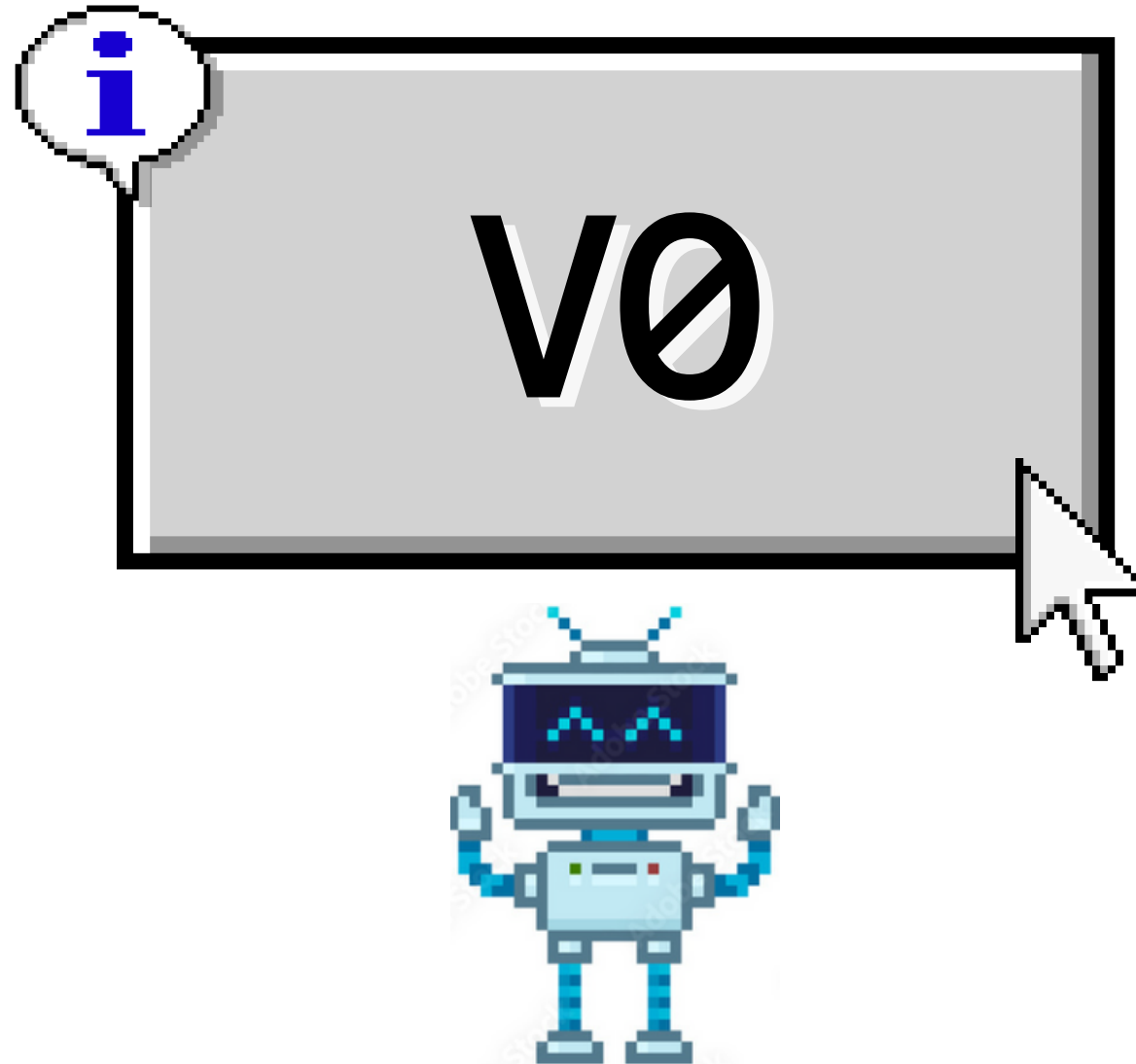
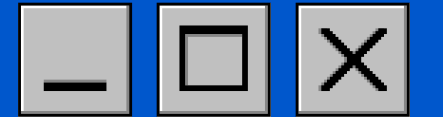


CHOIX FINAL

- Modèle de **policy gradient** plus pertinent = output en valeur continu (pas discrétisation = +précis et +exploration)
- Modèle de type **on-policy** (réduit la divergence)
- Très stable avec du **clipping**
- Input only (non compatible avec image)
- Nécessite bien moins d'input et d'épisode pour converger
- Plus facile à implémenter (SAC / TD3)

= Moins couteux en ressource





RÉSULTATS

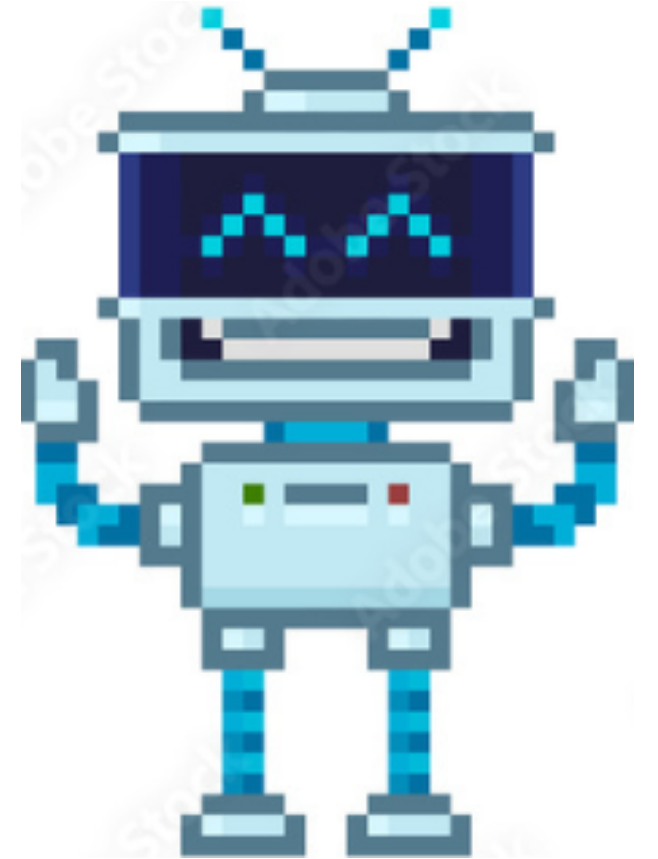


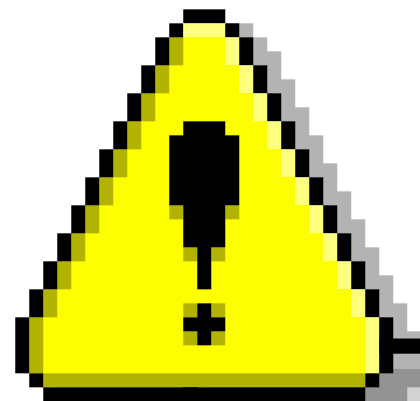
Notre agent fait:

- Suivre la route
- Ne change pas de voie
- des vecteurs d'observations, un ensemble d'actions, une fonction de récompense

Ne fait pas :

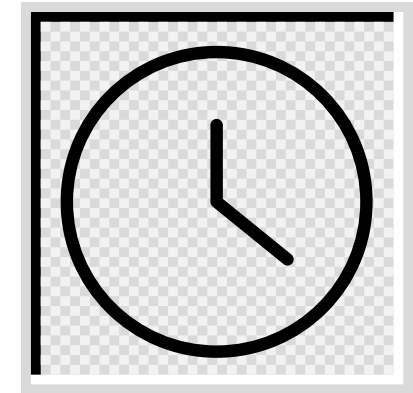
- Suivre les virages
- Correctement adapter sa vitesse face aux obstacles
- changer de voie



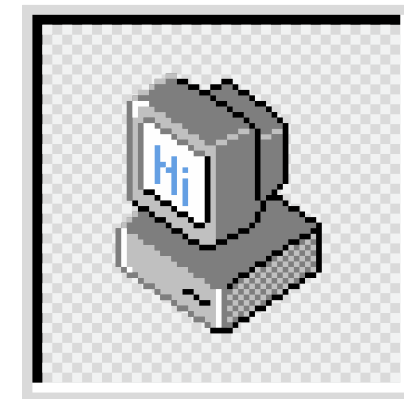


**MANQUE DE
RESSOURCE**

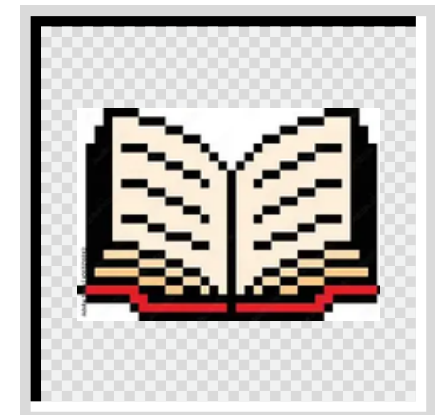
Manque de temps

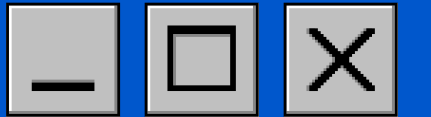


Manque de CPU & GPU



Manque de compétence





CONCLUSION

Projet intéressant mais trop prétentieux pour les ressources disponible

Les résultats sont objectivement pas suffisant

Travail honorable au vue des ressources allouée

Louis LAMBERT

Théo CONDAMIN

