Projet Jetpack DinoAI

Comparaison de Différentes Intelligences Artificielles dans un Jeu Simple

Cahier des charges techniques

Par :
Victor Descamps
Simon Catanese
Ethan Richard

Tuteur :
Florent Madelaine

fait le 18/11/2023

Sommaire :

1. Contexte

Sujet :

Florent Madelaine.
Expérimentation Bots pour un ou des jeux.

Programmer un jeu original avec idéalement des éléments qui en font un challenge pour la réalisation d'IA (hasard, phase d'enchère changeant le payoff, multijoueur, combinatoire importante, ...). Il conviendra de choisir un jeu pour lequel il est possible de réduire la combinatoire du jeu de manière naturelle et de désactiver/activer des règles pour en étudier des variantes naturelles. Il faudra ensuite créer, améliorer et expérimenter avec divers algorithmes pour converger vers des bots offrant un challenge raisonnable pour un humain.

2.Technologies envisagées :

2.1. Plateforme de développement

Le projet sera développé sur Unity, version 22.3.11f1.

Unity est un moteur de jeu gratuit et simple d'accès offrant une grande liberté de création et une grande collection d'outils divers.

De nombreux cours sont disponibles sur internet pour apprendre son utilisation.
Unity est aussi compatible avec de nombreuses plateformes dont Windows, MacOs et Linux.

2.2. Langage de programmation

Le langage de programmation utilisé sera C# pour le développement dans l'environnement Unity. Il s'agit du langage natif de la plateforme.

C# est un langage de programmation objet très ressemblant au Java.

2.3. Utilisation de ML-Agents

ML-Agents sera intégré pour la conception et l'entraînement des intelligences artificielles. La version 2.0.1 de ML-Agents sera utilisée.

ML agent est un asset (outil additionnel) interne à Unity qui permet d'encadrer un apprentissage neuronal. Les scripts d'IA seront développés en utilisant ML-Agents pour permettre aux bots d'apprendre à jouer au jeu de manière autonome.

Les modèles neuronaux seront entraînés en utilisant des algorithmes d'apprentissage par renforcement.

2.4. Modélisation des personnages et des environnements

Des sprites 2D (images) seront créés pour représenter l'avatar, les obstacles et l'environnement du jeu.

L'outil de graphisme Photoshop sera majoritairement utilisé pour la conception des sprites.

2.5 Contrôles du jeu

Les contrôles du jeu seront mis en œuvre pour un joueur humain ainsi que pour les IA.

La seule commande permettra de modifier la position verticale du joueur tandis que le niveau défilera tout seul de droite à gauche.

2.6 Gestion du jeu

Un système de gestion de jeu sera mis en place, incluant le chargement du niveau, la gestion des collisions, la génération procédurale d'obstacles et la sauvegarde des scores et de données diverses.

Les données enregistrées durant l’exécution de l’application seront stockées dans des fichiers json ou xml dans le répertoire de l’application.

2.7 Tests et débogage

Des scénarios de test exhaustifs seront élaborés pour chaque composant du jeu.

Les fonctionnalités seront soumises à des tests unitaires. Des outils de débogage de Unity seront utilisés pour résoudre les problèmes potentiels.

3. Équipe de développement

- **Ethan, Simon et Victor portent tous les 3 le rôle de chef de projet et de développeur.**
- **Simon sera également responsable des tests de l'application**
- **Ethan sera aussi impliqué dans la documentation du projet**
- **Victor sera plus spécifiquement impliqué dans l'implémentation des systèmes neuronaux dans le jeu.**

Nous allons tenter de réaliser le développement en pair programming, avec une approche test-driven.

L'outil en ligne Trello nous servira à planifier les tâches.

4.Les IA :

Les concepts d'IA suivant sont expérimentaux et seront sujets à des changements suivant l'avancée du développement du projet et de notre documentation sur la programmation de systèmes neuronaux.

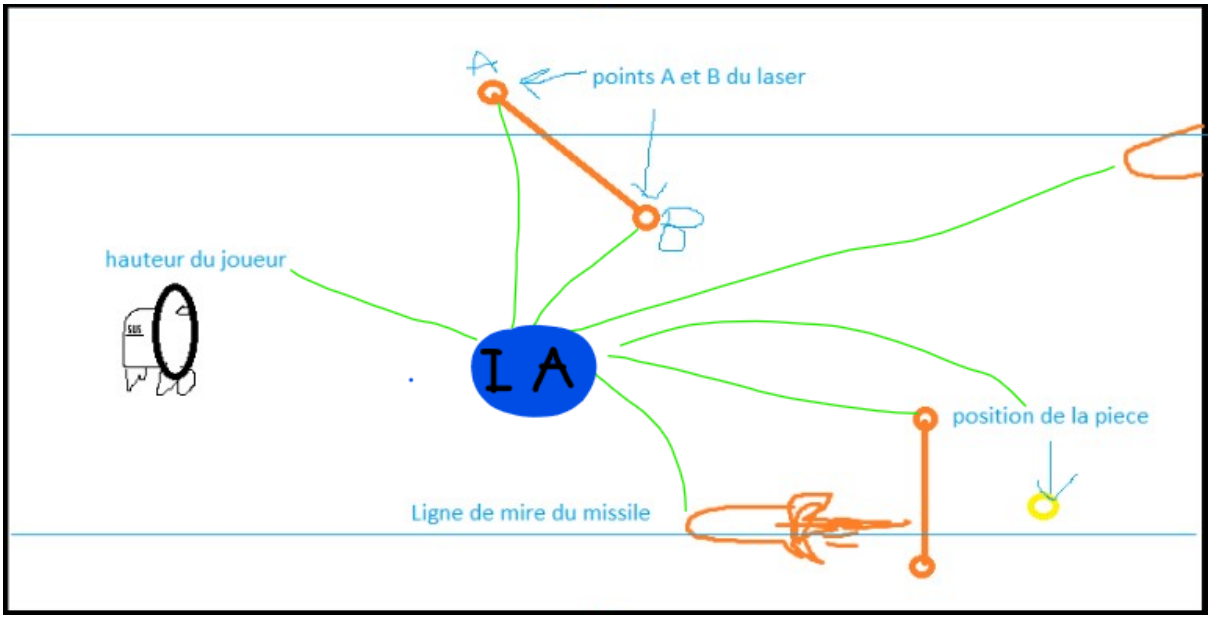
Le rôle des bots est de décider à chaque instant *T* de faire monter ou de laisser descendre l'avatar. Tous auront une source de données différentes leur permettant de prendre une décision.

L'omniscient (Neuronal) 

Ce bot connaît un grand nombre de données brutes. Entre autres :

- La vitesse de défilement du jeu,
- La position verticale du joueur,
- Les coordonnées des deux points des deux lasers les plus proches,
- Les coordonnées de tout autre danger proche,
- les coordonnées d'une zone comportant des pièces

L'IA devra réussir à faire les bonnes corrélations dans ce grand nombre de données pour prendre les bonnes décisions au cours du temps. Elle n'obtiendra pas de données enrichies pouvant faciliter la prise de décision, ces données brutes seront son seul guide.



L’observateur (Neuronal) 

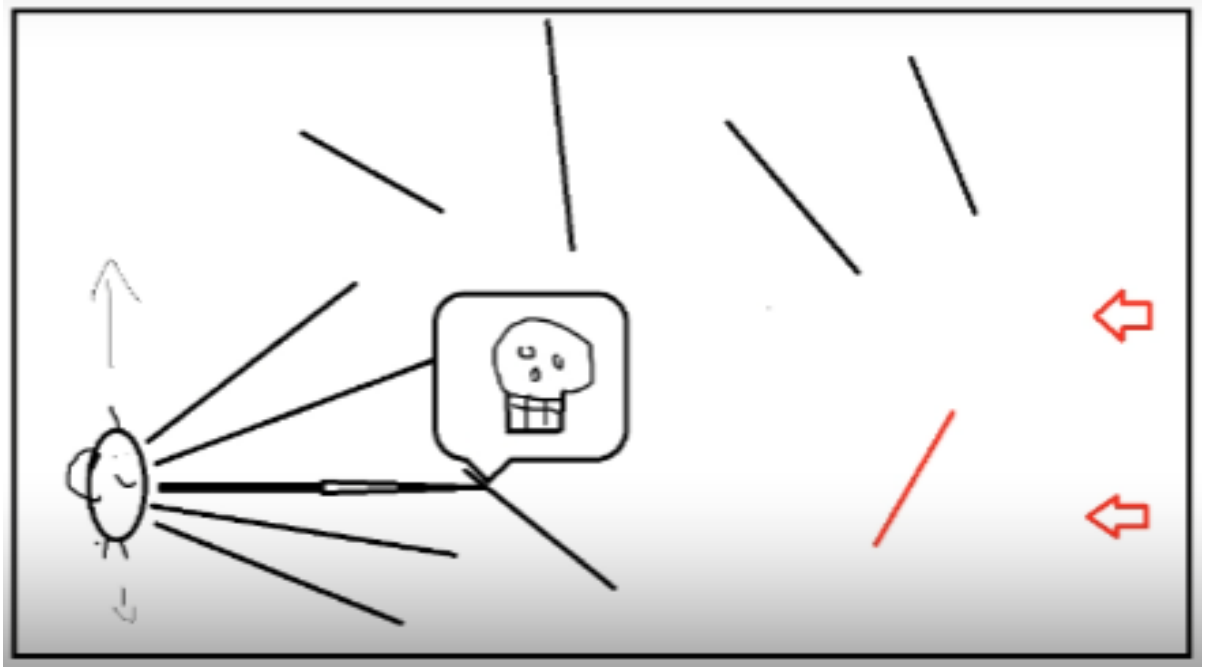
Ce bot reçoit en temps réel des données de proximité de l’avatar avec les éléments de l’environnement et des données sur la nature de ces éléments.

Un ensemble de RayCast[1] partant de l’avatar dans des directions précises seront utilisés pour recueillir les données nécessaires.
A la manière d’un radar de voiture, l’IA devra analyser sa proximité avec des obstacles puis prendra une décision pour ajuster sa trajectoire.

L’observateur devra apprendre à éviter les objets néfastes s’approchant trop de l’avatar et essayer de se rapprocher des pièces pour maximiser son score.

Cinq RayCasts couvrent probablement un champ de vision suffisamment large et dense pour une analyse efficace de l’environnement, mais des nombres différents seront essayés pour essayer de maximiser les performances du bot.

La longueur des RayCasts correspondra à la taille de l’écran, et les premiers objets rencontrés dans leur trajectoire seront traités comme des données utilisables par le système neuronal de l’observateur. Leur distance par rapport à l’avatar et leur nature (bordure, obstacle, bonus, malus) seront transmises au bot.



[1]Raycast = vecteur partant d’un point dans une direction donnée sur une certaine distance. Peut rapporter des données, par exemple en cas d’objet rencontré sur le parcours du vecteur (nature, distance, etc). Comparable à un pointeur laser de chantier mesurant des distances précises.

L’Itérateur (Procédural) 

Le bot itérateur ne fonctionne pas par apprentissage mais par itérations de calculs. Son fonctionnement consiste à calculer de nombreux chemins différents par itérations puis à sélectionner le meilleur chemin trouvé pour poursuivre la partie. À chaque intervalle , le bot pourra effectuer et maintenir une action (monter ou descendre) le temps de cet intervalle. L’ensemble des actions forme un chemin donné.

Le moteur de jeu va tester virtuellement les chemins à tester et le bot va itérer des calculs de suite d’actions un certain nombre de fois. Les chemins proposés amenant à la collision avec un obstacle seront écartés. Le chemin sans collision et apportant le plus de points au score sera sélectionné et le bot recommencera une suite d’itérations de calculs vers la moitié de ce chemin pour améliorer sa trajectoire.

Si tous les chemins mènent à une collision avec un obstacle, le chemin retardant le plus la collision sera sélectionné. L’ensemble des actions possibles dans une suite d’intervalles représente un arbre binaire où chaque chemin est un parcours de la base de l’arbre vers une feuille. L’itérateur procède à un parcours en profondeur de l’arbre pour trouver le meilleur chemin.

Nous envisageons d’utiliser l’approche de l'algorithme glouton afin d’obtenir un chemin satisfaisant sans dépenser trop de temps de calcul en parcourant l’arbre entier.Pendant le parcours de l’arbre, si un chemin testé mène à une collision avec un obstacle ou avec un bonus, l’itérateur reviendra avec une nouvelle suite d’actions

L’ itérateur intelligent (Procédural et neuronal) 

Bot optionnel que l’on implémentera si le temps nous le permet.

Nous n’avons pas encore de concept clair pour ce bot, nous chercherons une possible implémentation de réseau neuronal fonctionnant de pair avec le calcul itératif de chemins pour améliorer les performances du bot précédent.

5. Livrables

Le dépôt git du projet contiendra le build de la dernière version fonctionnelle de l’application en plus des documentations annexes. Le projet Unity en lui-même est partagé entre les membres du PT sur la plateforme de collaboration officielle du Unity. Nous veillerons à inclure le code source du projet final dans le dépôt git en plus du build complet de l’application.

6. Bibliographie :

Apprentissage du C# :

Série de vidéos sur les aspects du C# dans la programmation sous Unity
https://www.youtube.com/watch?v=IFayQioG71A&list=PLzDRvYVwl53t2GGC4rV_AmH7vSvSqjVmz&pp=iAQB

Documentation unity :

Documentation officiel de la plateforme, utile pour tout besoin technique.
https://docs.unity.com/

Documentation ML-Agents :

Documentation officielle de ML-Agents qui nous servira à implémenter des systèmes neuronaux.
https://unity-technologies.github.io/ml-agents/ML-Agents-Toolkit-Documentation/