# Cahier des charges Techniques:

- plus court et peut être complété en plusieurs fois
- détail des technologies envisagées en justifiant les choix
- bibliographie (article, livres, sites, vidéos)
- Esquisse propositions de design (diagramme de classe ou BD, complet ou non)
- METHODOLOGIE
    - Méthodes de travail envisagées (agile, pairprogramming, test driven…)
    - Outils à mettre en place (trello, convention de codage, orga repo, outils, scripts de déploiement, etc)
    - UTILISER UN REPO GIT à renseigner
        - politique de tests ambicieuse (tests unitaires reproductibles)
        - documenter convenablement le code
        - réfléchir au déploiement du logiciel et à la documentation utilisateur ( != développeur)
    - expliquer le fonctionnement de l'équipe
        - rôles des membres
    - Diagramme de Gantt prévisionnel reprennant les jalons
        - à mettre à jour pour tenir compte du travail réel

      comparer à therme les diagrammes prévisionnels et réels

# 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 :

# 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.

# Technologies envisagées :

## Unity :

Unity est un moteur de jeu simple d'accès offrant une grande liberté de création et une grande collection d'outils divers. Unity utilise des objets scriptables en C#.
**ML agent** est un asset (outil additionnel) interne à Unity qui permet d'encadrer un apprentissage neuronal. Nous ne nous sommes pas encore formés à son utilisation mais une présentation détaillée sera faite en temps et en heure.

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

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.

# Fonctionnement de l’équipe

Nos rôles ne sont pas encore clairement définis mais certaines pistes d’attibution de tâches existent déjà :

- Victor est le plus documenté sur l’utilisation de l’IA dans Unity
- Simon a une affinité avec le graphisme et pourra s’occuper des visuels du jeu à côté du travail fonctionnel

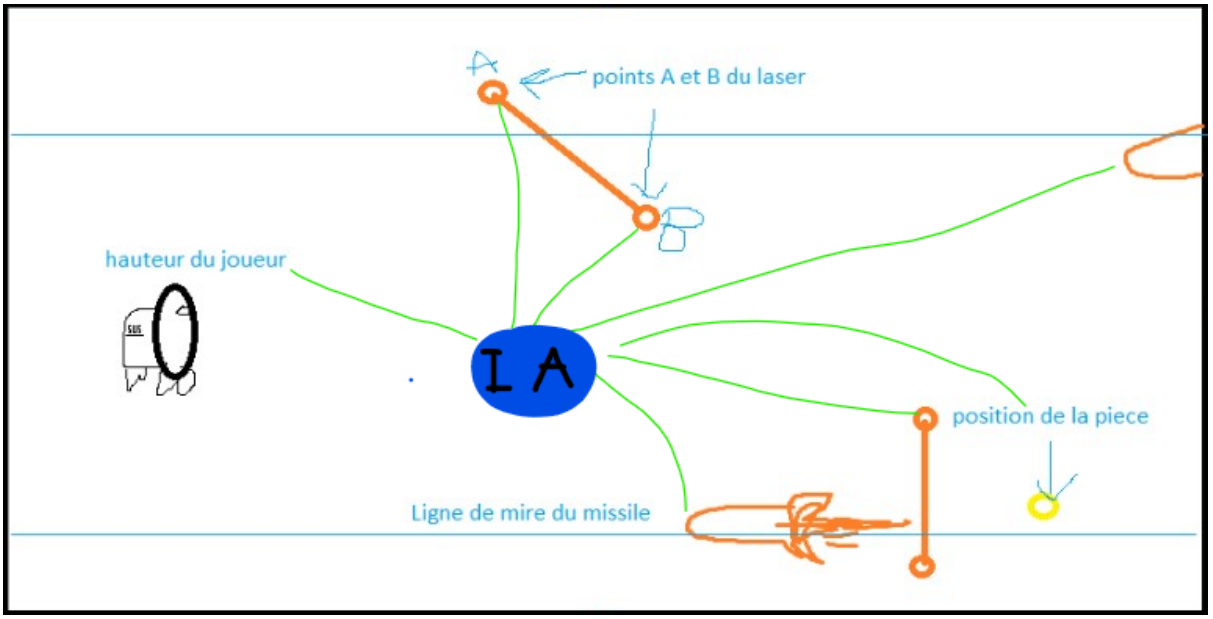
Notre workflow sera probablement en pair-programming et test driven.
Un Trello sera mis en place dans les jours qui viennent.

# Les IA :

À noter que les listes d’input aux IA sont encore expérimentales, il n’est pas exclu qu’il y en ai qui changent, s'ajoutent ou même disparaissent de la liste d’ici la fin du projet.

## Le sachant (Apprentissage): 

Cette IA avancera à l’aveugle en recevant les coordonnées de chaque objet sur l'écran ainsi que d’autres informations telles que la vélocité verticale du joueur et la vitesse de défilement. On s'attend à ce que l’IA comprenne que l’objet représentant son personnage ne doit rester à distance des objets néfastes.



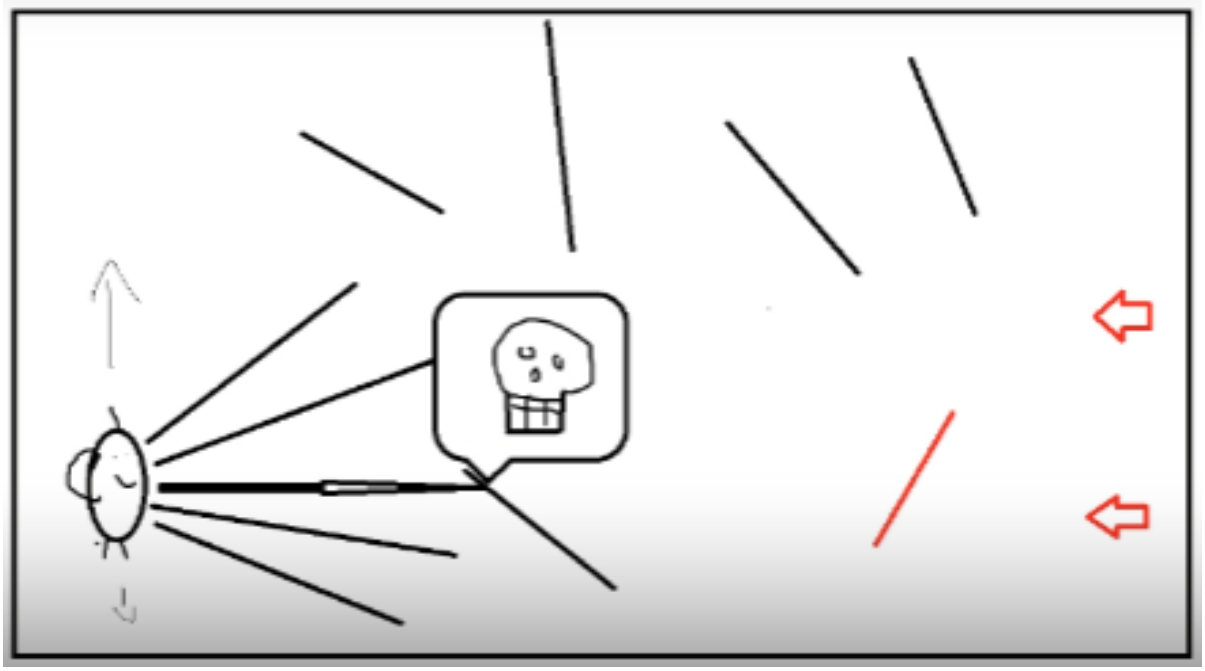

Pour se faire on donnera à l’ia les données suivante:

-vitesse du jeu
-position et vitesse y du joueur
-position des 2 prochains lasers et 2 prochains missiles

La position d'un laser sera probablement communiquée par un couple de coordonnées représentant les 2 points de celui-ci.

## L'observateur (Apprentissage) 

Cette IA ne connaît pas les coordonnées des objets à l'écran. L'observateur possède en revanche un champ de vision lui permettant de repérer sur certains angles la distance d'objets par rapport à l'avatar. On s'attend à ce que l'IA apprenne à éviter les objets néfastes s'approchant trop de l'avatar.
Le champ de vision sera composé d'un nombre fixe de raycast[1] couvrant la zone en face du joueur.



Pour se faire on donnera à l'ia les données suivante:

- vitesse du jeu
- position x du joueur
- pour chaque raycast :
    - nature de l'objet touché (bordure, danger, pièce)
    - distance de l'objet touché par rapport à l'avatar

Il est difficile de savoir à l'avance combien de raycast nous allons utiliser.

[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.

## Le calculateur (Algorithme) 

Le bot Calculateur ne fonctionne pas par apprentissage mais par un algorithme. Au fil de la partie, le calculateur doit calculer par itérations différents chemins en essayant de s'approcher du chemin le plus sûr possible. Par intervalle de temps, le calculateur essaiera de prolonger le meilleur chemin qu'il a trouvé et testera des déviations possibles apportant un éventuel avantage sur l'espérance de survie et d'obtention de points.

Pour réaliser cette IA nous allons diviser chaque seconde en plusieurs "séquences" pendant lesquelles le jetpack sera ou non activé ce qui permet de prévisualiser l'ensemble des chemin possible dans un arbre. par exemple si on divise une seconde en 4 et qu'on regarde 4 secondes en avant cela nous donne un arbre de 2^16 chemins, trop grand pour être calculable en temps réel mais à cela on pourra soustraire les enfants des chemins ayant touché un danger, les chemin trop similaire à un chemin déjà trouvé ou déjà éliminé et autre optimisations.

## Le calculateur apprenant (Algorithme & IA) 

Bot optionnel fonctionnant comme le calculateur à la différence que l’on essaiera d’y implémenter une IA qui devra faire de meilleurs choix de prolongement et de déviation de chemin afin d’augmenter l’efficacité et les performances du bot Calculateur. On s'attend à ce que l’IA apprenne à trouver des chemins plus sûrs en prenant en compte les chemins déjà testés.

fonctionnement pas encore mis au clair (mais utilisant l’IA du Calculateur en priorisant grâce à l’apprentissage l’esquive de schéma d’obstacles qu’il a deja pu observer)