Recommandations tout document :

- format PDF
- texte taille 12
  - justifié gauche et droite
- figures lisibles et de bonne taille
  - Légendes sur les figures
- Se relire
- privilégier la clarté du propos
- Eviter le Jargon
  - Expliquer les mots techniques
  - éviter les abréviations
    - les expliciter la première fois si il faut vraiment les utiliser en note de bas de page ou parenthèses
  - Si beaucoup de pages, regrouper abréviations et définitions à la fin du document
- mettre en oeuvre les cours de MHR

Contraintes :

- mettre TOUS nos documents en markdown (.md) en plus du pdf, sur le git.
- écrire un README.md expliquant comment installer le logiciel
- écrire une documentation technique pour utilisateur (!= développeur)

Recommandations cahier des charges :

- 1 cahier des charges FONCTIONNELLES
  - compréhensible par un non informaticien
- 1 cahier des charges TECHNIQUES et METHODOLOGIQUES
  - compris par les membres de l'équipe et le tuteur
- la première page comporte le nom du projet, date, responsable référent du projet, auteurs ayant contribué, et adresse du git

Cahier des charges Fonctionnelles:

- présentation contexte (préciser client, besoins, contraintes, existant)
- Etude des objectifs (analyse des besoins)
  - Explication précise des fonctionnalités attendues
  - utilisateurs/acteurs
  - diagramme de cas d'usage (DCU)
  - userstories
  - wireflow
- Calendrier et priorisation des objectifs
  - En accord avec le client, fixer des jalons avec des objectifs
  - Fixer les priorités sur les objectifs. MoSCoW
  - Détailler les objectifs en sous objectifs

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 fonctionnelles

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.

Objectifs :

Nous avons très rapidement opté pour faire différentes IA qui pourraient jouer toutes seules à un jeu simple. Nous sommes très intéressés par la notion de temps réel et de l'apprentissage.
Notre objectif serait de comparer différentes méthodes d'analyse du jeu sur des bots différents pour les comparer entre elles sur plusieurs facteurs comme le temps d'apprentissage ou l'efficacité de la méthode.

Le livrable sera une application exécutable avec un .exe. L'interface doit permettre de lancer le jeu en choisissant les paramètres désirés. L'utilisateur pourra choisir de contrôler l'avatar pour jouer lui-même ou bien déléguer la partie à l'un des bots développés, puis éventuellement apporter des changements de règle au jeu.

En fonction des paramètres choisis, l'utilisateur peut jouer lui-même ou assister à la partie d'un ou plusieurs bots en même temps. Chaque partie de chaque bot (ou du joueur) enregistre des données stockées au format json ou xml.

L'utilisateur peut choisir un paramètre "Rush" privilégiant la vitesse de calculs à l'affichage graphique, accélérant le jeu au maximum et réalisant des parties en boucle pour obtenir des données exploitables et comparables le plus vite possible.

Le menu principal du jeu permet d'accéder à une interface de visualisation des données collectées en jeu, affichables en graphiques et tableaux, permettant de comparer efficacement les performances des différents bots et du joueur humain en fonction des paramètres de partie choisis.

Le Jeu :

Le jeu de base :

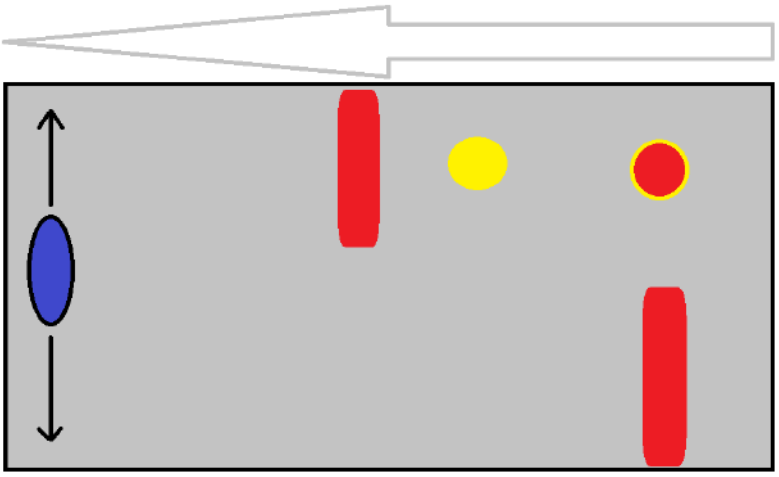
Jetpack Joyride est un jeu dit “Runner” car il consiste simplement à aller le plus loin possible tout en survivant aux dangers que le jeu lui oppose. Les règles du jeu sont simples, laissant le joueur décider de se propulser son avatar vers le haut ou de le laisser tomber pour faire bouger son personnage verticalement tandis que le terrain et donc ses obstacles avancent horizontalement vers lui de plus en plus vite.

(capture d’écran d’une partie type)

Notre jeu :

Pour faire jouer nos IA nous allons donc reproduire le jeu sur Unity en adaptant certains aspects pour faciliter la compréhension et l’apprentissage de l’IA :

- Les objets qui ajoutent des règles très situationnelles sont retirés (comme les véhicules ou bonus modifiant l'expérience interactive du jeu).
- Un système de score sera ajouté. La maximisation du score sera l’objectif de l’IA au-delà du temps de survie. Le score aura une augmentation linéaire en fonction du temps de survie et pourra être augmenté en ramassant des pièces. Mais de mauvaises pièces peuvent apparaître et faire baisser le score. Les IA devront ainsi adapter leur chemin pour tenter de maximiser leur score quitte à prendre des risques.
- Le jeu original peut faire apparaître des missiles pouvant tuer le joueur. Ces missiles ne seront initialement pas implémentés, des tests nous permettront de savoir si ils sont un ajout viable au jeu après le développement du premier bot.



Les Bots :

nb. Pour comprendre la suite, il faut savoir que les IA fonctionnant par apprentissage nécessitent l'apport d'un certain nombre de données de sources définies pour fonctionner, données qu'elles apprennent à corréler par tâtonnement afin de trouver la meilleure approche et apporter les meilleures réponses.

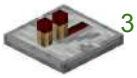
Notation MoSCoW:
Must Should Could Would

S) Le sachant (Apprentissage):  [1]

Cette IA avancera à l'aveugle en ayant en tête les positions de chaque objet sur l'écran. On s'attends à ce que l'IA comprenne que l'objet représentant son personnage ne doit pas s'approcher des objets néfastes entraînant la mort ou une baisse de score.

M) L'observateur (Apprentissage)  [2]

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.

M) L'Itérateur (Procédural)  [3]

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.

C) L'Itérateur intelligent(Procédural et neuronal)  [4]

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.

D'autres IA viendront si les idées et les deadlines nous le permettent.

---

[1] Représentation du Sachant par l'icône d'un 'entonnoir' du jeu Minecraft. L'entonnoir symbolise l'entrée de nombreuses données résultant en une réponse simple.
[2] Représentation de l'Observateur par l'icône d'un 'observateur' du jeu Minecraft. L'icône de l'observateur symbolise l'observation de changements dans l'environnement du bot entraînant une prise de décision.
[3] Représentation de l'itérateur par l'icône d'un 'répéteur' du jeu Minecraft. Le répéteur symbolise la répétition des itérations de calcul de chemins différents pour ensuite sélectionner la meilleure option.
[4] Représentation de l'itérateur intelligent par l'icône d'un 'comparateur' du jeu Minecraft. Le comparateur symbolise la comparaison des différents chemins calculés par l'algorithme par l'IA pour élaborer un raisonnement économisant des itérations de calcul.

L’Application :

Menu principal :

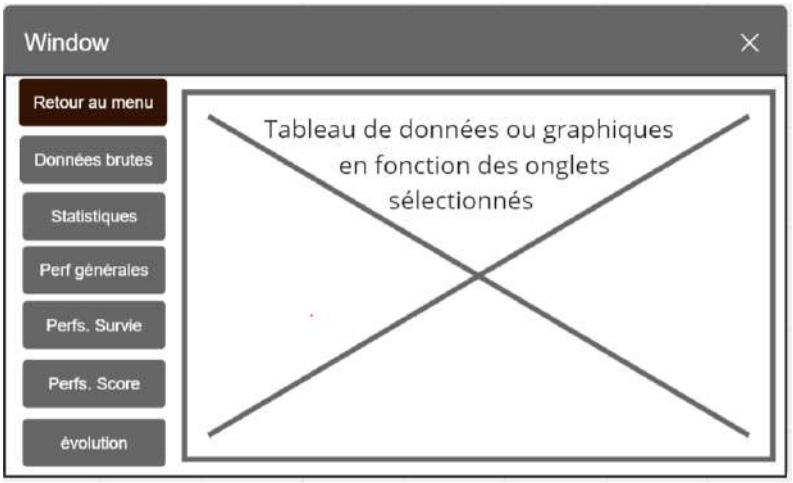
Permet de naviguer entre les deux parties majeures de l’application.

Le bouton “Lancer le jeu” redirige vers un menu de configuration de la partie.

Le bouton “Consulter les données” redirige vers une interface de visualisation des données récoltées au long des parties enregistrées.

Affichage des données :

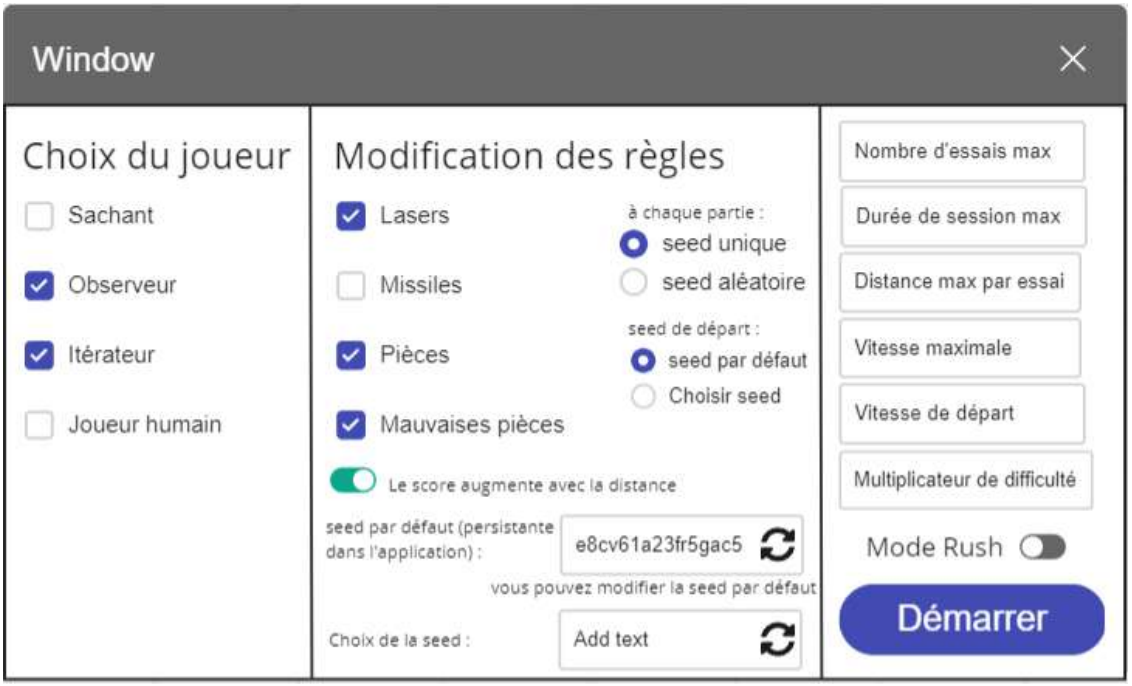
Cette interface traite les données enregistrées automatiquement pendant toutes les parties jouées. Les bots utilisés et les règles de chaque partie permettent de différencier chaque ensemble de données.
Les données traitées seront affichées sous forme de tableaux ou graphiques selon ce qui semble le plus adapté.
Différents onglets permettent de filtrer l’affichage et obtenir des informations plus précises sur certains aspects.



L’objectif est de permettre à l’utilisateur de pouvoir comparer facilement de multiples aspects de chaque bots de manière concrète et rigoureuse après avoir recueilli suffisamment de données.

Il est envisagé d’implémenter une fonctionnalité d’export des données en tableau Exel.

Configuration d’une partie :



Cette interface permet à l’utilisateur de configurer une partie ou une suite de parties auxquelles il fera participer un ou plusieurs bots ou bien lui-même.

Les règles du jeu peuvent être modifiées pour soumettre les bots à des conditions de jeu diverses.

Des éléments comme les missiles ou les pièces peuvent être activés ou désactivés. D’autres paramètres peuvent être modifiés comme la vitesse maximale pouvant être atteinte par le jeu ou le nombre d’essais maximums et la durée de session maximum (la session s’arrête lorsqu’une des deux limites est atteinte).

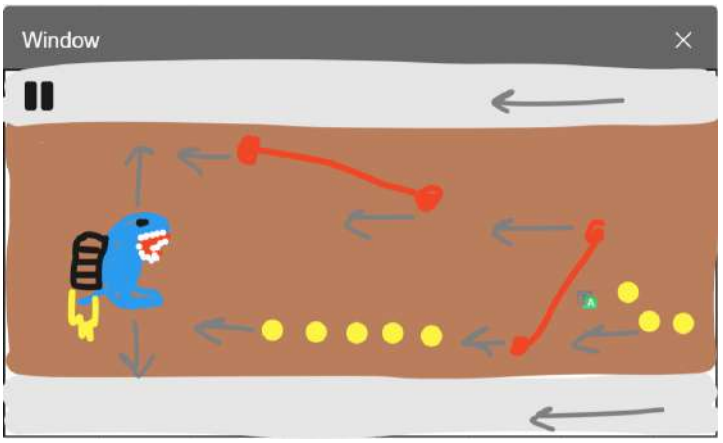
Chaque partie peut comporter une seed aléatoire ou garder une seed constante. La seed constante peut être la seed par défaut, conservée dans les fichiers du jeu et non modifiée depuis le début du développement, ou bien une autre seed choisie par l’utilisateur. La seed par défaut peut aussi être modifiée.

Enfin, le mode Rush permet de maximiser la vitesse du jeu pour accélérer la collecte de données et l’apprentissage des bots. Ce mode de jeu remplace le rendu graphique par une console affichant en temps réel différentes informations. La vitesse du mode rush dépend entièrement de la puissance de l’ordinateur combinée à la complexité de calcul générée par les bots.

Il est envisagé de permettre de choisir un niveau d'entraînement pour chaque bot.

Le jeu :

L’écran de jeu exécutera en temps réel la reproduction du jeu Jetpack Joyride en rendu graphique (ou console si le mode Rush est activé).
L’avatar, représentant la célèbre mascotte officieusement officielle du BUT INFO, Dino, peut se déplacer de haut en bas. Le décor et les autres éléments du jeu se déplacent de droite à gauche.
Dino doit se déplacer verticalement de manière à éviter les lasers et autres malus et attraper le plus de pièces possibles.



Si plusieurs bots jouent ensemble, leurs avatars apparaîtront en transparence. L’avatar du joueur humain sera toujours opaque.

Si plusieurs parties sont configurées, chaque partie suivante commencera dès la fin de la partie en cours.

A la fin de la dernière partie, l’application changera d’interface pour afficher le menu de consultation de données ne comprenant que les données obtenues pendant la session jouée.

Prévision des Étapes de création

Nous avons décidé d'organiser le processus de création du projet en travaillant simultanément sur plusieurs axes afin de rester constamment informés de son évolution et d'éviter d'être dépassés, tout en permettant à chacun de commencer à travailler sur des aspects additionnels. Ainsi, nous avons structuré la création en plusieurs étapes :

Semaine 1-3 : Apprentissage des outils (Unity, C#, Python, ML Agent)
Semaine 3-4 : Conception de l'interface
Semaine 5-6 : Conception du jeu
Semaine 7-8 : Création de l'IA "Sachant"
Semaine 9-10 : Création de l'IA "Observateur"
Semaine 11-12 : Création de l'IA "Calculateur"
Semaine 13-14 : Création de l'IA "Calculateur Intelligent"
Semaine 15-16 : Collecte de données des IA

diagramme de Gantt disponible dans cahier des charges technique.