Smart Dash

Rapport final

El Bouroumi Anas
Grossman Jeremy
Perez Ioann

Tuteur : Guénaël Cabanes

2023/2024

Tables des matières

Introduction

Le présent rapport offre une analyse complète du projet "Smart Dash", retraçant son évolution de la phase de conception jusqu'aux résultats escomptés.

Ce projet ambitieux a pour but le développement d'une intelligence artificielle (IA) capable de maîtriser Geometry Dash, un jeu de plateforme réputé pour sa difficulté. Dans ce jeu, un cube avance automatiquement et le joueur doit le faire sauter avec précision pour éviter divers obstacles.

Notre objectif principal est de permettre à l'IA de naviguer avec succès à travers des niveaux inédits, prouvant ainsi sa capacité à s'adapter et à réagir face à de nouveaux défis.

Afin d'atteindre cet objectif, nous avons développé une version simplifiée de Geometry Dash. Cette version, dépourvue d'éléments graphiques, nous permet de calculer les positions du joueur et des obstacles de manière efficace, facilitant ainsi un entraînement accéléré de l'IA. Nous l'avons baptisée le "moteur de jeu".

Les principales fonctionnalités de notre projet incluent un calcul dynamique des positions, un système d'entraînement efficace favorisant une adaptation rapide de l'IA, et la capacité pour celle-ci de s'ajuster à des environnements jamais rencontrés auparavant. Notre méthodologie englobe le développement d'un moteur de jeu épuré, l'implémentation d'algorithmes d'apprentissage de pointe, la conception de scénarios d'entraînement variés, et une optimisation continue des performances pour un apprentissage efficace.

Les résultats attendus incluent le développement d'un moteur de jeu fonctionnel, une IA capable de jouer sur des niveaux connus et inconnus, ainsi qu'un moteur graphique épuré.

Analyse

Ce projet complexe peut être décomposé en plusieurs sections distinctes, chacune jouant un rôle crucial dans le succès global de l'initiative.

Moteur de Jeu

Le développement du moteur de jeu représente la première étape cruciale de notre projet. Cette phase englobe la conception et l'implémentation d'un moteur de jeu simplifié pour Geometry Dash. L'objectif est de créer une plateforme fonctionnelle avec laquelle l'IA pourra interagir et évoluer.
Les principaux éléments à prendre en compte dans cette section comprennent le calcul dynamique des positions et la gestion des collisions par rapport au bloc et au pique des terrains..

Nous avons donc créé une version très simplifiée du jeu, où le joueur se déplace bloc par bloc à chaque intervalle de temps.

Diagramme de classe du début :

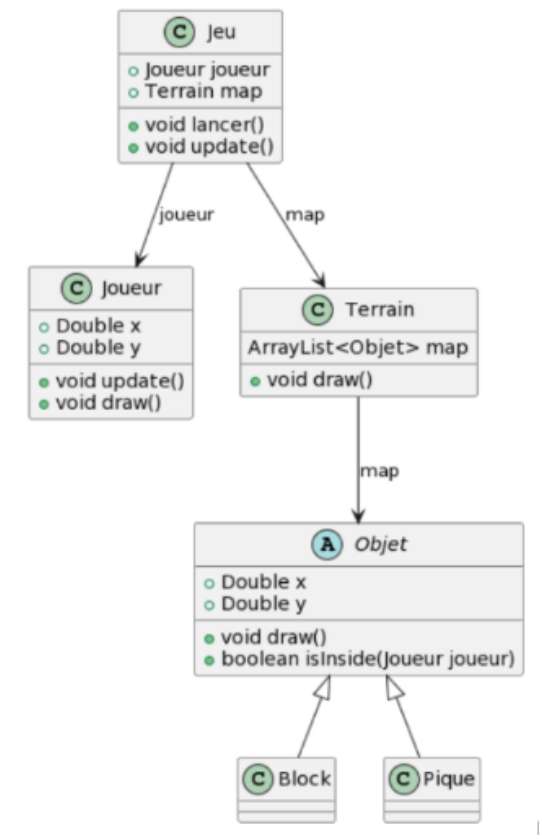


Diagramme de classe S5 :

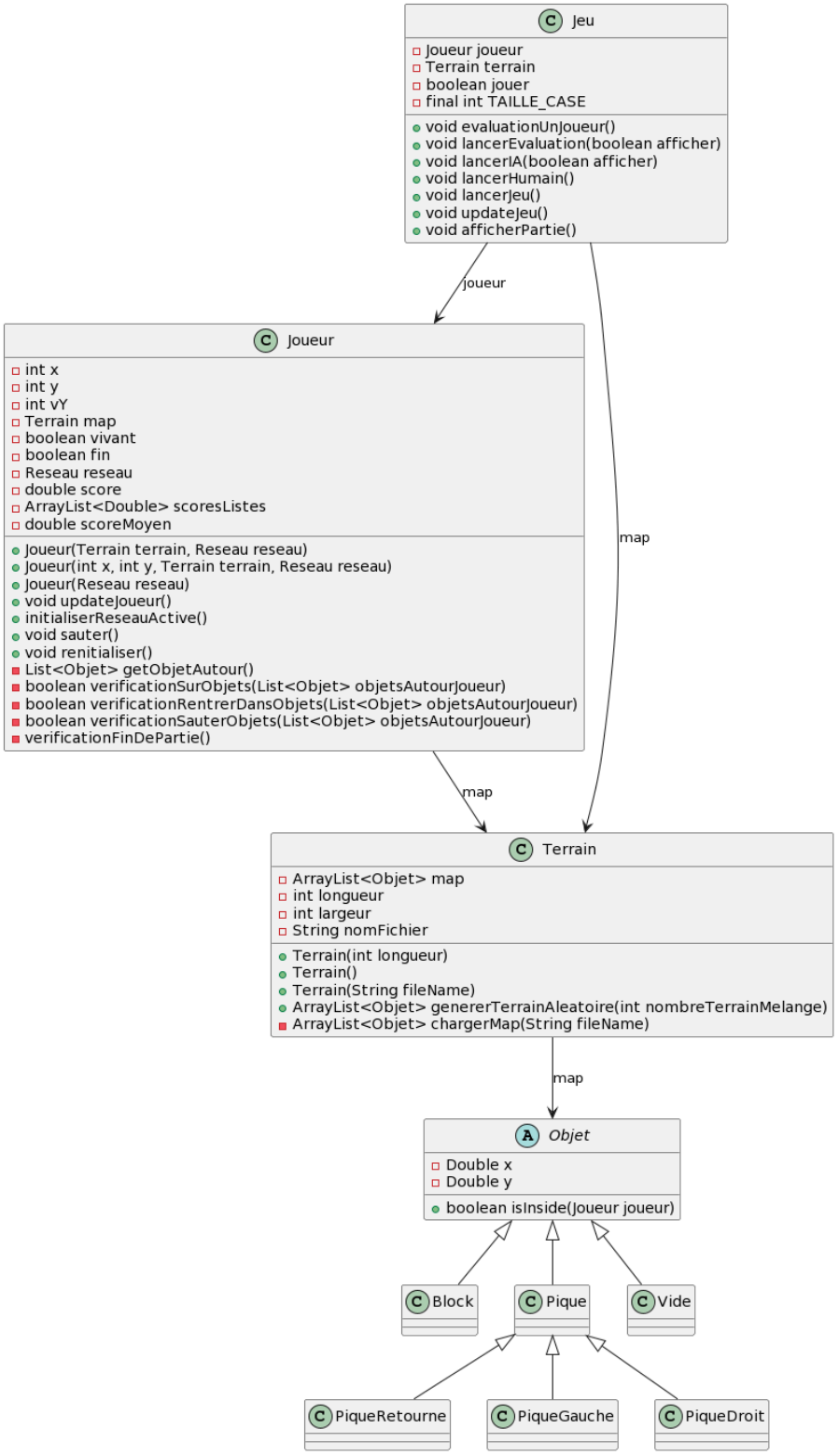
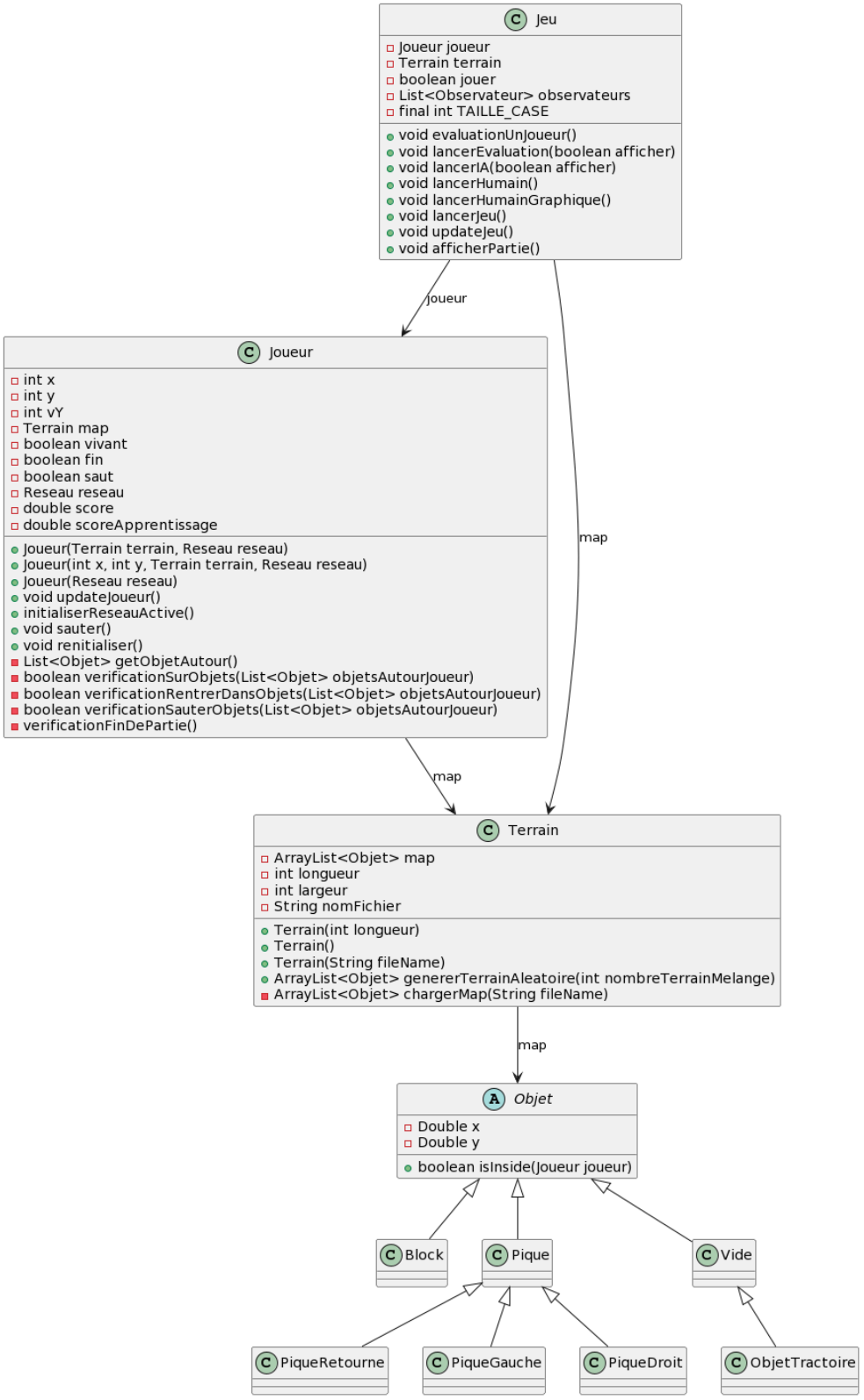


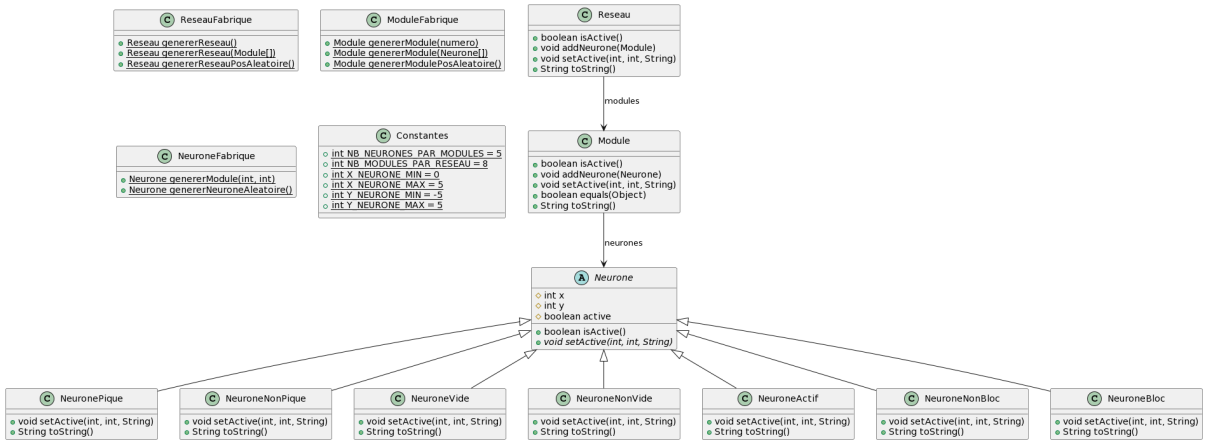
Diagramme de classe final :



IA et Apprentissage

La partie dédiée à l'intelligence artificielle et à l'apprentissage constitue l'essence de notre projet. Notre objectif est de concevoir un système d'intelligence artificielle apte à apprendre à jouer à Geometry Dash. Cela nécessite le développement de réseaux neuronaux modulaires, l'implémentation d'algorithmes d'apprentissage évolutif, et la conception de scénarios d'entraînement diversifiés. Nous visons à habiliter l'IA à prendre des décisions éclairées en fonction de l'environnement du jeu. Pour faciliter la création d'instances de différents types de réseaux, nous avons également instauré des fabriques.

Diagramme de classe final de l'IA:



Moteur Graphique

Bien que notre objectif principal soit de développer une IA sans besoin d'une interface graphique, une section consacrée au moteur graphique est essentielle. Cela peut inclure la création d'un moteur graphique simplifié pour permettre un affichage visuel, même s'il n'est pas utilisé par l'IA. Cette partie peut également aborder des aspects liés à la convivialité et à la visualisation des résultats de l'IA.

Diagramme de classe final du moteur graphique :

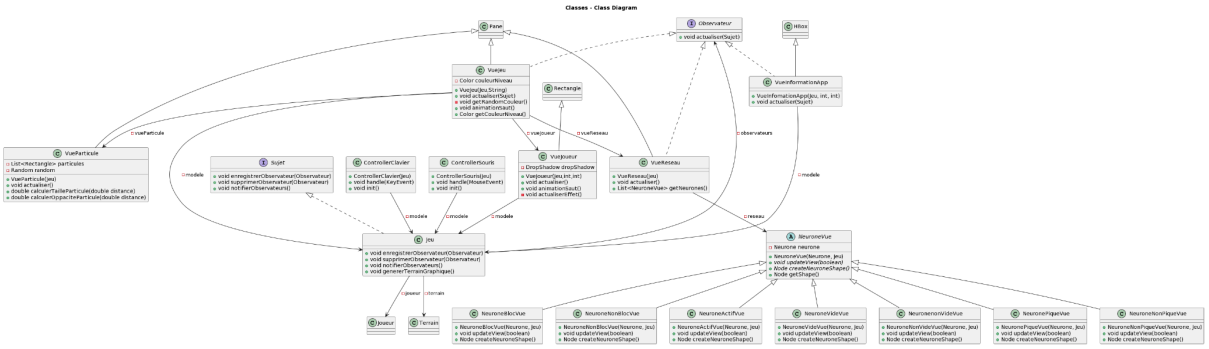
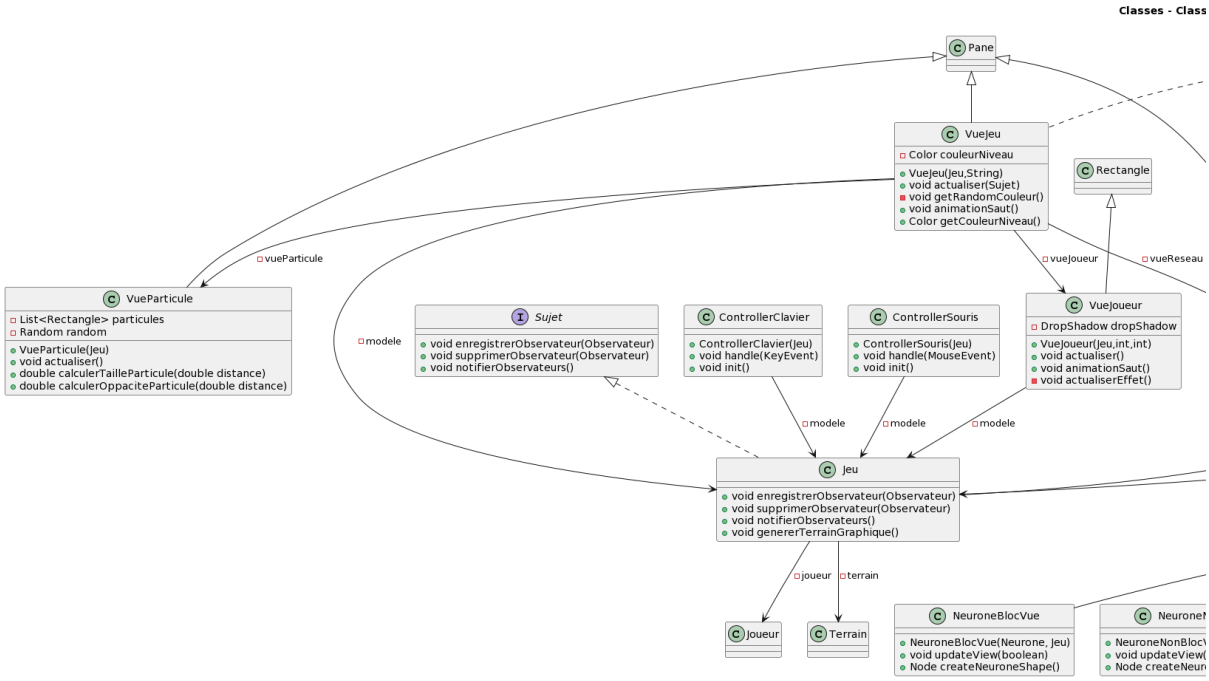
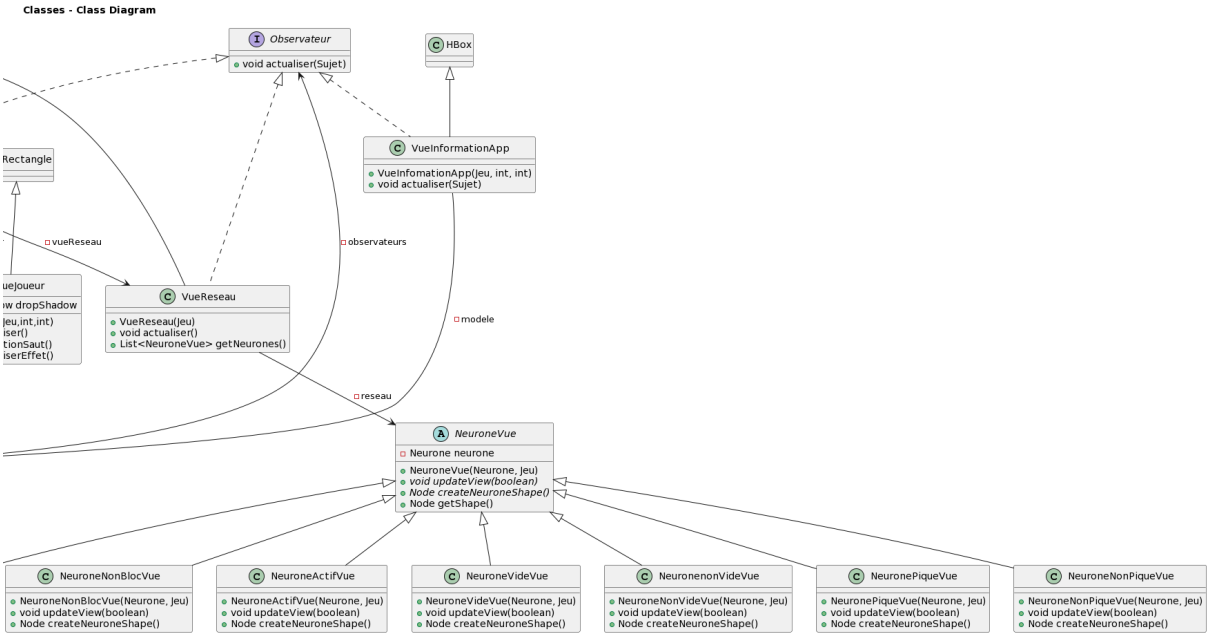


Diagramme de classe final du moteur graphique en 2 parties :





Interface Utilisateur

Au fil des itérations, nous avons décidé d'introduire une interface utilisateur dans notre projet, dans le but de faciliter son utilisation aussi bien pour les utilisateurs que pour nous-mêmes. Cette interface permet aux utilisateurs de sélectionner l'intelligence artificielle en fonction de ses performances, ainsi que de choisir le terrain sur lequel tester cette IA. Concernant le choix du terrain, une sélection de terrains préétablis est disponible, avec également la possibilité de générer un terrain aléatoire que l'IA n'a jamais rencontré auparavant.

Maquettes de l'interface :

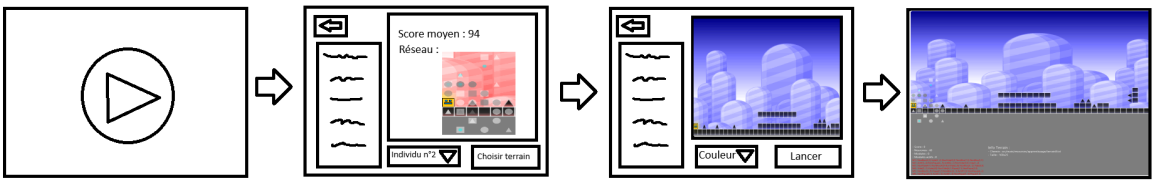
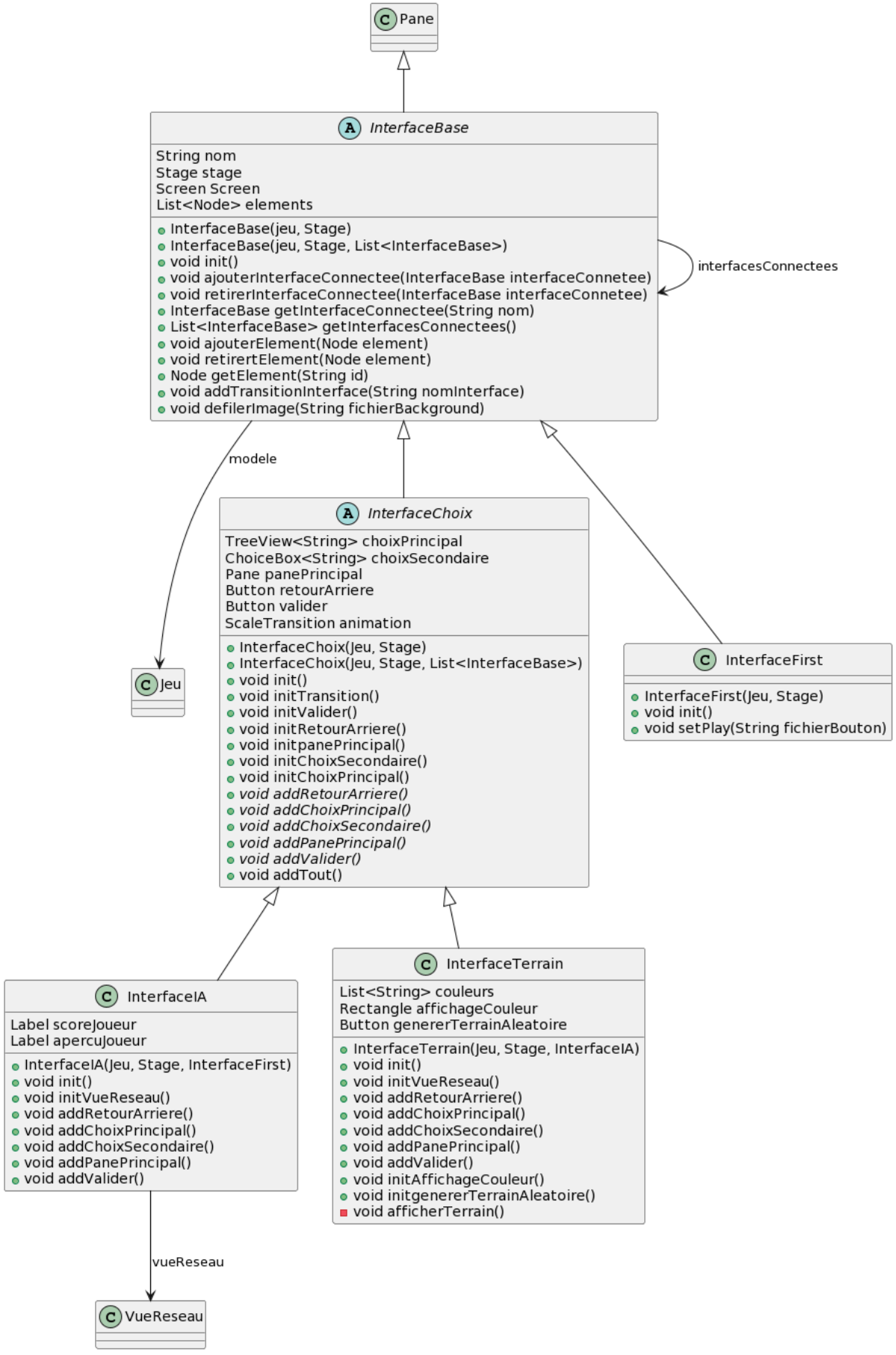


Diagramme de classe final de l'interface utilisateur :

Diagramme de classe - Interface



Evolution par rapport à l’étude préalable :

Pour résumer l'évolution par rapport à l'étude préalable, nous avons décidé de synthétiser tout cela dans un tableau qui récapitulerait notre projet sur nos trois axes différents.

CRÉATION DE L'IA	CRÉATION DU MOTEUR JEU	CRÉATION DU MOTEUR GRAPHIQUE
Création de neurones simples ✓	Création de niveau ✓	Représentation des éléments 2D ✓
Réseau de neurones simple ✓	Saut et déplacements ✓	Animation du personnage ✓
Algorithme d'apprentissage ✓	Collisions ✓	Interface utilisateur ✓
Enregistrement de l'apprentissage ✓		
Simulation en parallèle ✓		
Générateur de terrain aléatoire ✓		

Nous pouvons conclure que le projet est indéniablement une réussite, puisque le cahier des charges a été rempli. Seules quelques tâches optionnelles n'ont pas été finalisées, mais elles ne sont pas essentielles pour le projet.

Cependant, la partie ajoutée en cours de développement, à savoir rendre l'IA flexible, n'a pas été achevée et les résultats ne sont pas très convaincants. Cela ouvre néanmoins la voie à une poursuite du projet, car il reste du travail à accomplir. Par exemple, il serait possible d'améliorer le moteur de jeu pour le rendre plus fluide et se rapprocher davantage d'une expérience de jeu authentique. De plus, nous pourrions enrichir le jeu en ajoutant plus de pièges et de fonctionnalités pour le rendre encore plus captivant, à l'image de la version originale.

Réalisation

La phase de réalisation de notre projet a nécessité une approche méthodique et collaborative, impliquant des efforts significatifs dans divers domaines. Cette section détaille l'architecture logicielle adoptée, les tests de validation effectués, ainsi que les défis rencontrés tout au long du processus.

Architecture Logicielle

Moteur de Jeu

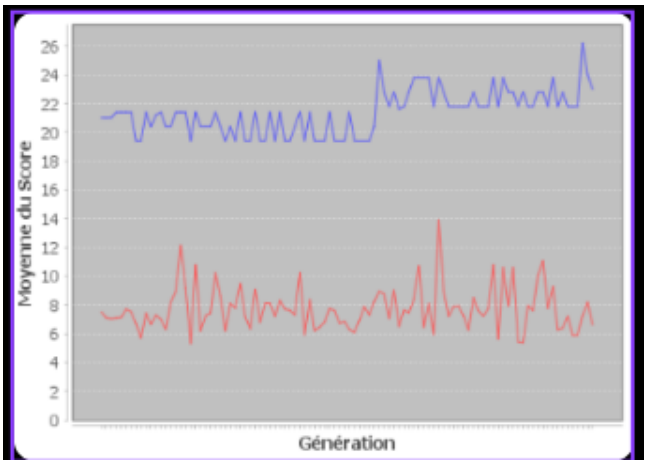
Pour la conception du moteur de jeu, nous avons divisé les responsabilités entre différents composants, facilitant ainsi la gestion du mouvement du joueur, le calcul des collisions et la gestion de l'environnement et des terrains.

IA et Apprentissage

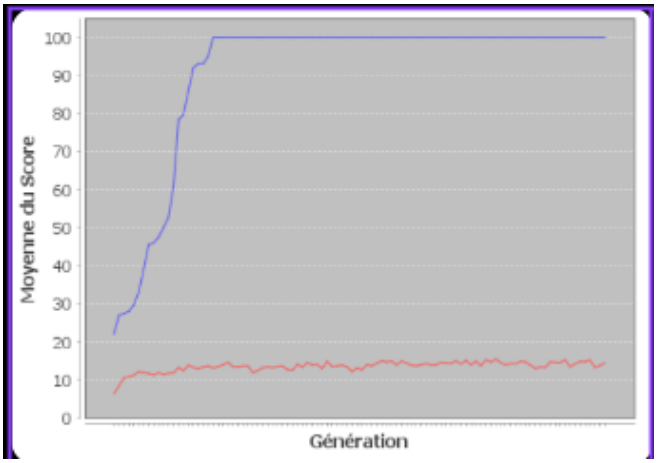
L'architecture logicielle de la partie IA est basée sur des réseaux neuronaux modulaires.
Chaque module correspond à une fonction spécifique, comme la prise de décision sur le moment du saut. Cette approche modulaire facilite l'extension du système pour gérer des situations plus complexes. Nous avons de plus implémenté des algorithmes d'apprentissage par évolution (plus précisément NEAT), permettant à l'IA d'ajuster ses comportements en fonction des résultats obtenus lors de scénarios d'entraînement.

Pendant cette phase, nous avons mis en place différentes versions de notre algorithme NEAT. Nous avons alors obtenu différents résultats.

Pour commencer, examinons la première version :



Nous avons ensuite notre version finale qui nous permet d'avoir une IA qui apprend sur nos terrains de base. Les résultats obtenus sont les suivants :



Cet algorithme, en plus de l'autre, fait varier les positions des joueurs et les fait évoluer sur plusieurs terrains

Moteur Graphique

Pour nous, le moteur graphique n'était pas une partie essentielle du projet, mais nous l'avons tout de même développé. Nous avons créé un moteur graphique permettant d'afficher en temps réel une partie de Geometry Dash. Au fil des itérations, nous l'avons amélioré en ajoutant des effets, des couleurs, des fonds, etc., afin de nous rapprocher le plus possible de la version originale.

Ensuite, nous avons décidé de concevoir une interface utilisateur pour simplifier l'utilisation de SmartDash. Au lieu de devoir lancer l'application dans notre IDE et configurer manuellement les terrains et les IA, cette interface permettrait de le faire de manière intuitive.

Nous avons alors créé des maquettes et nous avons essayé de les respecter autant que possible dans la conception de l'interface utilisateur.

Rendre l’IA flexible

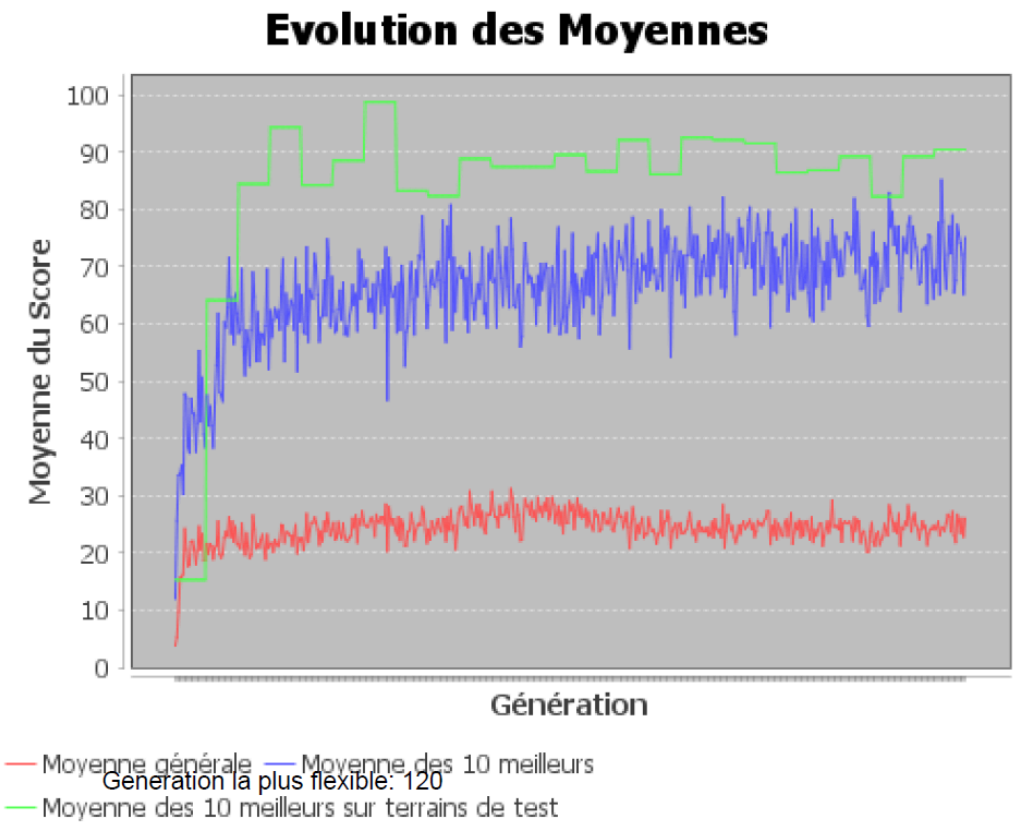
Pour rendre l'IA aussi flexible que possible, nous avons dû recourir à de nouvelles méthodes d'apprentissage qui nous permettent de modifier facilement sa configuration. De plus, nous avons mis en place un générateur de terrain aléatoire

afin d'obtenir des données de test variées. Ce générateur crée une trajectoire aléatoire à partir de laquelle le terrain est généré.

Nous avons ensuite utilisé ces données de test pour démontrer qu'en utilisant notre nouveau système d'apprentissage, qui ajuste dynamiquement le nombre de neurones et de modules en fonction du score, l'IA deviendrait plus flexible.

Le principe est simple : plus notre modèle contient de neurones, plus il risque de sur-apprendre, tandis qu'un modèle avec peu de neurones risque de sous-apprendre. Il est donc essentiel de trouver un juste équilibre, et les données de test permettraient de déterminer si cette approche a réussi ou échoué.

Nous avons alors obtenus les résultats suivant pour la partie rendre l'IA flexible :



Génération la plus flexible: 120

Tests de Validation

La phase de tests a été cruciale pour garantir le bon fonctionnement de notre solution et ainsi éviter de se retrouver avec un jeu qui ne fonctionne pas correctement où des neurones défectueux lors de l'apprentissage.

Nous avons effectué des tests de validation à plusieurs niveaux :

Unitaires : Chaque module du moteur de jeu et de l'IA a été soumis à des tests unitaires pour vérifier son bon fonctionnement individuel.

Intégration : Nous avons évalué l'interaction entre les différents composants du système, garantissant une communication fonctionnelle.

Scénarios d'Entraînement : Des tests ont été réalisés avec des scénarios d'entraînement variés pour évaluer la capacité d'adaptation de l'IA à des situations diverses.

Moteur Graphique : Les fonctionnalités du moteur graphique ont été testées pour s'assurer de leur conformité aux attentes.

Difficultés Rencontrées

Durant ce projet nous avons pu faire face à quelques difficultés. Parmi les difficultés les plus notables, nous avons rencontré :

- Complexité de l'Apprentissage : La conception d'un système d'apprentissage capable de s'adapter efficacement à des environnements complexes a posé des défis, nécessitant des itérations fréquentes. En effet il a fallu choisir judicieusement des méthodes de mutation et de croisement optimal afin d'avoir un taux d'apprentissage si trop faible ni trop important et ainsi donc obtenir la convergence attendue.

- Complexité du réseau de neurones initial : Il a également fallu choisir une structure permettant à l'IA de pouvoir apprendre correctement, sans quoi l'IA convergent beaucoup trop tôt. Nous devons également faire attention à ce que la structure ne soit pas trop importante non plus afin d'éviter le surapprentissage de l'IA sur les niveaux d'apprentissage. Cela en vient à modifier le nombre de modules, le nombre de neurones par modules, les positions initiales..

- Nous avons également eu quelques difficultés plus générales en ce qui concerne les effets de bords des objets java étant données qu'on définissait les réseaux de neurones des joueurs à chaque génération.

- Nous avons également rencontré des difficultés pour rendre l'IA flexible afin qu'elle puisse s'adapter à différents types de terrains. Nous avions établi des théories sur la manière de parvenir à cette flexibilité, mais dans la pratique, nous avons rencontré des complications. Par exemple, dans notre nouvelle méthode d'apprentissage, nous avions mis en place un système où nous faisions varier le nombre de modules et de neurones grâce à un mécanisme de punition dans le score. Théoriquement, un score légèrement moins élevé mais avec moins de neurones devait être considéré comme meilleur qu'un score plus élevé avec beaucoup de neurones. Cette approche visait à éviter le sur-apprentissage en permettant à l'IA d'être plus adaptable. Cependant, notre système de notation était défectueux, et il pénalisait fortement l'IA, ce qui conduisait à un modèle très faible.

En dépit de ces défis, notre approche collaborative et notre engagement envers des tests rigoureux ont permis de surmonter ces obstacles et d'atteindre des résultats significatifs dans la réalisation de notre projet

Au cours du projet, nous avons également rencontré le départ d'un membre de l'équipe, ce qui a représenté une difficulté initiale puisque nous nous retrouvions avec une charge de travail conséquente à gérer à seulement trois personnes. Cependant, au fil des itérations, nous avons réussi à surmonter cette contrainte et le manque de personnel ne s'est plus fait ressentir.

Conclusion

"Smart Dash" a été choisi comme sujet de notre projet tutoré dans le but d'intégrer l'intelligence artificielle dans l'univers des jeux vidéo, plus précisément dans le jeu Geometry Dash. Notre objectif ambitieux était de développer, en plus d'une version bloc par bloc du jeu d'origine, une IA capable de maîtriser ce jeu complexe, illustrant ainsi les progrès et les capacités de l'apprentissage automatique.

Ayant atteint cet objectif avant la fin de notre projet, nous avons ensuite cherché à rendre notre IA flexible sur des terrains inconnus pour elle. Bien que des progrès notables aient été réalisés dans cette direction, nous sommes encore loin de parvenir à une flexibilité totale à 100%. Pour y parvenir, il faudrait que l'IA apprenne à généraliser au maximum toutes les règles du jeu.

Nous avons alors remarqué que lorsque l'IA possède une structure bien équilibrée (avec un nombre optimal de neurones pour éviter le surapprentissage ou le sous-apprentissage), et qu'elle apprend sur des terrains générés aléatoirement à chaque génération, elle se rapproche de ce que l'on pourrait appeler une "IA flexible". Il serait alors nécessaire de développer un générateur de terrain aléatoire capable de créer des niveaux comportant tous les pièges possibles, afin d'obtenir la meilleure IA possible. Ceci permettrait d'apporter une continuité et une évolution au projet.

En conclusion, le projet "Smart Dash" a été une expérience enrichissante, mettant en lumière les complexités et les possibilités de l'intégration de l'intelligence artificielle dans les jeux vidéo. Malgré les défis rencontrés, nous avons finalement obtenu des résultats très satisfaisants. Cela nous a permis d'acquérir une connaissance approfondie d'une nouvelle structure d'IA ainsi que de nouveaux algorithmes dans ce passionnant domaine de l'intelligence artificielle, ce qui ne pourra être que bénéfique pour notre future carrière en informatique.

Mode d’emploi

Pour exécuter l'application finale, vous pouvez vous rendre sur un IDE tel queIntelliJ et lancer la classe "UserInterface" qui se situe dans le package application.

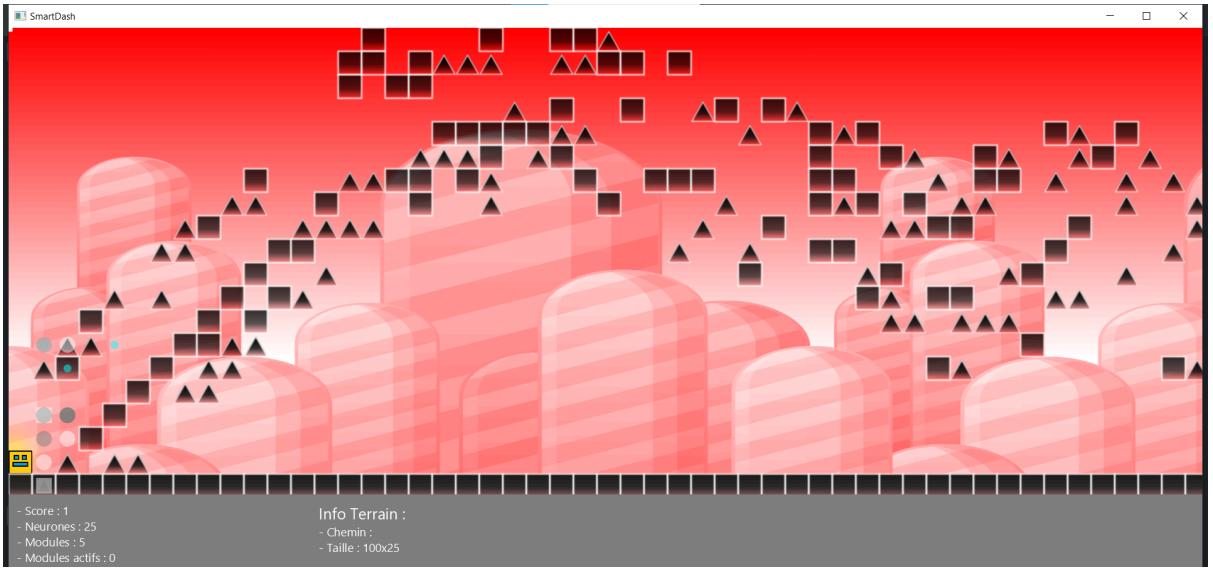
Alternativement, vous pouvez naviguer vers le répertoire et exécuter le fichier "smartdash.jar", ce qui lancera l'application mais sans l'interface utilisateur. Pour ce faire, il faudra vérifier que la bonne version de Java soit installée, la version 21.

Pour lancer un apprentissage, il faudra alors exécuter la classe "Apprentissage", qui se situe dans le projet.

La classe est assez bien documentée grâce aux commentaires, elle est assez intuitive.

Comment jouer :

Après avoir lancé l'application, une page comme celle-ci sera alors affichée :



Pour jouer, vous devrez alors appuyer sur le bouton "A", ce qui lancera l'IA et lui permettra de jouer.

Pour revenir au début de la partie, vous avez le bouton "R".