Rapport TP - FSM + Algorithme de pathfinding IA

Maurice Fotso Njiojip
Philippe Montplaisir-Morissette
Étienne Primeau
Ahmed Yassine

Modifications apportées au programme existant

La base de code que nous avons utilisé afin d'intégrer notre *FSM* ainsi que notre algorithme de *pathfinding* est le projet open source "Multiplayer FPS". C'est un jeu de tir à la première personne multijoueurs basé sur le moteur de jeu Unity 3D. Le jeu comportait déjà une bonne base qui permettait à plusieurs joueurs de se connecter en réseau et de jouer ensemble.

Nous avons décidé d'inclure le code de la *FSM* dans de nouveaux scripts bien répertoriés dans de nouveaux dossiers à même le projet pour respecter les bonnes pratiques de lisibilité et maintenabilité du code. Tout le code de la *FSM* se retrouve dans le dossier "StateMachine", du dossier "Scripts". Nous avons également adapté le code du fusil du joueur (classe "FpsGun") et plus particulièrement la méthode "Shoot()" pour détecter les collisions des balles et les intégrer aux conditions de notre FSM.

Pour le *pathfinding*, nous avons choisi d'utiliser l'outil "AI Navigation" disponible dans les *packages* de l'éditeur de Unity pour implanter un système de *NavMesh* et de *pathfinding* pour notre PNJ. Pour ce faire, nous avons été en mesure de créer une surface de type *NavMesh* partout sur les objets de la scène en créant un objet de type "NavMesh Surface". Cet objet permet de *bake* rapidement une surface navigable dans le niveau et différents paramètres de reconnaissance du terrain peuvent être configurés.

Enfin, en intégrant l'usage de l'outil "AI Navigation" à notre code nous avons pu accéder à son intelligence artificielle pour gérer les déplacements entre notre PNJ et le *NavMesh* dans la scène. En cherchant ensuite dans la documentation, nous avons pu déterminer que l'algorithme de *pathfinding* utilisé par l'outil était un algorithme de type A*.

Déroulement du projet

Avant l'implémentation de la FSM nous avons évalué nos options issues du travail pratique fait en classe. Nous avions une machine à états construite en un seul script qui aurait pu faire l'affaire (pour s'y référer nous l'avons conservée dans le dossier "TP_AI" sous le dossier "Scripts"), mais dans l'optique d'être en mesure d'avoir un code facilement extensible nous avons opté pour l'option d'une FSM divisée en

plusieurs parties. À partir d'une interface pour les états et d'une FSM de base, nous avons produit une FSM spécifique à notre PNJ qui lui permet de changer d'état selon certaines conditions. Au départ, le PNJ patrouille entre deux points sur la carte, lorsque le joueur s'approche à une certaine distance le PNJ commence à le poursuivre. Si le joueur s'éloigne trop, le PNJ va retourner patrouiller. Lorsque le joueur se fait poursuivre, il peut tirer sur le PNJ ce qui le fera fuir et évaluer sporadiquement sa distance au joueur et reprendre la poursuite ou retourner patrouiller.

Un autre enjeu que nous avons rencontré pour l'intégration de la FSM et de l'algorithme de pathfinding est que nous avons choisi de travailler dans un projet existant. Plusieurs personnes ont contribué à ce projet *open source* avant nos interventions. Parfois les bonnes pratiques n'ont pas été respectées et les nouveaux éléments n'ont pas été ajoutés de manière unifiée rendant le code difficile à lire et le rendant moins flexible pour nos intégrations. Nous avons donc passé beaucoup de temps à comprendre le fonctionnement du projet. Nous avons tout de même réussi à développer une bonne compréhension du matériel et à intégrer le code du fusil du joueur (FpsGun) dans les conditions de la FSM. De plus, nous nous sommes assurés de suivre les bonnes pratiques de programmation en jeu vidéo en découpant notre code en plusieurs scripts et en suivant une nomenclature précise pour ne pas ajouter davantage de lourdeur à la base du projet.

En utilisant l'outil de *NavMesh* de Unity, nous avons fait le choix d'utiliser l'algorithme A* pour le *pathfinding*, ce qui convient à un jeu en 3D pour trouver les chemins les plus courts entre deux points. Il aura fallu adapter nos paramètres de *baking* du *NavMesh* pour l'adapter à un monde en 3D complexe déjà existant et être certain que le PNJ ne reste pas coincé à certains endroits.

Puisque que nous avions une bonne idée de la manière d'implémenter la FSM en plus de la pratique précédente, nous avons initialement peu utilisé les outils d'intelligence artificielle générative. Cependant, une fois le projet fonctionnel nous avons décidé de demander à ChatGpt de réviser le code que nous avons intégré et s'assurer du respect des bonnes pratiques de programmation.

Enfin, la gestion du travail d'équipe et de l'utilisation de la plateforme GitHub s'est bien déroulée. Avant de commencer le travail, nous nous sommes réparti le travail selon nos disponibilités et les différents éléments à intégrer au projet. Nous avons utilisé du *pair programming* pour nous aider dans l'intégration de la FSM. Avant, chaque *commit* nous avons aussi demander à un de nos collègues de faire du *code review* pour s'assurer que le nouveau code était bien écrit et ne contenait pas d'erreurs. À chaque étape, nous nous sommes également assurés de bien tester les nouvelles *features* avant de l'intégrer dans le projet source. Nous avons utilisé la plateforme Discord pour communiquer et s'organiser autour des tâches à faire et des disponibilités de chacun des membres de l'équipe.

Voici le lien du code review de chatGpt:
https://chat.openai.com/share/81a99b92-50d2-4918-bb6c-f4927acb0696
Il était très intéressant de constater l'évaluation de ChatGpt. Il nous a amené différentes améliorations potentielles qui aideraient la maintenabilité et la modularité de notre FSM. Il ne semble toutefois pas avoir repéré d'erreurs majeures qui méritaient une intervention. De plus, il a même souligné plusieurs de nos bonnes pratiques