LOG8235 – Agents intelligents pour jeux vidéo

Hiver 2024

TP No. 2

Numéro d'équipe : 04

1951042 - Abderrahim Laribi

2057242 - Ayoub Chihab

2077446 - Félix Lamarche

2307835 - Léo Valette

Soumis à :

Daryl Barampaze

17 Mars 2024

Ce qui suit est la liste des fichiers et des fonctions modifiées pour section du travail à accomplir:

1. Génération du navmesh

La génération du navmesh se fait à partir de l'éditeur d'Unreal. Pour prendre en compte les dalles de type "Death", il faut modifier le blueprint et ajouter un component "NavModifier".

Il faut aussi s'assurer que les paramètres du navmesh correspondent aux dimensions des agents.

2. Calcul et affichage d'un chemin

Dans le fichier STDAIController.cpp :

- GetBestCollectible() : Retourne le collectible le plus près de l'agent.
- ShowNavigationPath(): Affiche le chemin suivi par l'agent.

3. Parcours d'un chemin

Dans le fichier STDAIController.cpp :

- UpdatePlayerInteraction() : Décide de l'état de l'agent et de se diriger vers un collectible.
- GetBestCollectible() : Retourne le collectible le plus près de l'agent.

Dans le fichier SDTPathFollowingComponent.cpp :

- FollowPathSegment() : Dans le cas d'un parcours sans saut, on utilise la fonction FollowPathSegment venant de la classe parente.
- SetMoveSegment() : Détermine si l'agent est devant un saut ou non.

4. Rajout de navlink

Il faut ajouter les Navlinks depuis l'éditeur d'Unreal avec le "Place Actors Panel". Ensuite, nous les avons modifiés en définissant l'Area Class comme étant "SDTNavArea_Jump".

5. Parcours d'un chemin comportant des navlinks

Dans le fichier SDTPathFollowingComponent.cpp:

- FollowPathSegment() : Suit le chemin courant ou suivre une jump curve dépendamment de la distance de l'agent du navlink cible. Définie une variable jumpCurveTime entre 0 et 1 pour faire du blending d'animation.
- SetMoveSegment() : Définie AtJumpSegment et le mode de mouvement du contrôleur IA.

6. Mis à jour du comportement de poursuite

Dans le fichier STDAIController.cpp et STDAIController.h :

- GoToBestTarget(): Bouge l'agent vers la meilleure destination, et si sa destination est le joueur utilise la fonction MoveToActor() pour être plus précis et MoveToLocation() pour des emplacements spécifiques.
- UpdatePlayerInteraction(): Décide, à partir de l'état de l'agent et du joueur de l'état présent de l'agent et de sa destination. Le mouvement présent peut être interrompu et redirigé. La poursuite se produit lorsque le joueur est visible et la position du joueur devient la destination. Lorsque le joueur n'est plus visible, sa dernière position reste la destination pour maintenir en mémoire son dernier emplacement.
- CheckPlayerVisibility() : Met à jour si l'agent voit le joueur via des Raycasts et met à jour la dernière position du joueur.

7. Mis à jour du comportement de fuite

Dans le fichier STDAIController.cpp et STDAIController.h :

- GoToBestTarget(): Bouge l'agent à la destination choisie de fuite.
- UpdatePlayerInteraction(): L'agent devient en fuite lorsqu'il voit le joueur et le joueur est powered-up. L'agent continue de fuire tant que le joueur n'est plus powered-up. Lorsque l'agent fuit en direction du joueur, il change de destination de fuite pour mieux éviter le joueur.
- GetBestFleeLocation() : Retourne une FleeLocation qui est dans la direction opposée du joueur en utilisant un produit scalaire entre la direction de l'agent vers l'emplacement de fuite et la direction de l'agent vers le joueur.

8. Ajout d’animation de déplacement

9. Ajout d’animation de saut

- Avec le jumpCurveTime, on pose le jumpProgress pour blend le saut avec la loop et l'atterrissage allant de 0 à 1.
- Avec GetVelocity() et VectorLength() on get la vitesse de l’agent pour blend les animations Idle, walk et run, allant de 0 à 600.
- Un blend poses by bool pour blend les animations de course avec de celles de saut.