Fabien Cocheteux – A4 GDP – IIM

TP – Intelligence Artificielle

I – Règles du jeu :

Le jeu se joue en un contre un et en temps limité fixe. Chaque joueur pilote un vaisseau spatial dans un champ d'astéroïde et doit :

- Capturer le plus de zones de contrôle possible.
- Toucher le plus de fois possible son adversaire (tirs et mines)

Un vaisseau peut effectuer les actions suivantes :

- Se propulser.
- Tourner
- Tirer
- Poser des mines
- Émettre une onde de choc

Propulsion : La propulsion du réacteur pousse le vaisseau devant lui. La puissance de propulsion peut être modulée de 0 à 100%.

Orientation : Le vaisseau se tourne progressivement vers une orientation donnée (angle en degré). La vitesse maximale de rotation est fixe.

Énergie : Chaque tirs, mines et ondes de choc coûte au vaisseau de l'énergie. Le vaisseau regagne de l'énergie avec le temps. La propulsion diminue jusqu'à 50% du regain d'énergie.

Tir : Le tir part en ligne droite jusqu'à toucher un obstacle solide : vaisseau, astéroïde ou mine. Il est possible de tirer sur une mine pour la détruire.

Mine : La mine s'active au bout de une seconde une fois posée. Tout vaisseau passant dans son rayon d'activation déclenche la mine.

Pénalité : Un vaisseau touché par une mine ou un tir est pénalisé pendant plusieurs secondes. La vitesse maximale du vaisseau pénalisé est fortement réduite.

Onde de choc : L'onde de choc repousse les adversaires à proximité. Les contrôles d'un vaisseau repoussé (propulsion, direction, tir, mine, onde de choc) sont désactivés pendant une seconde. L'onde de choc ne rapporte pas de point. Il est possible de détruire une mine avec une onde de choc.

En fin de partie :

- Chaque zone contrôlée rapporte 1 point.
- Chaque tir ou mine touchant un joueur non-pénalisé rapporte 1 point à son adversaire.

II – Stratégies :

Les game-designers doivent analyser dans un document différentes stratégies possibles permettant de maximiser les chances de victoire de jeu.

En expliquant leurs choix, les game-designers doivent par la suite décrire les stratégies adaptées à leur IA.

III – Système d'IA :

L'intelligence artificielle doit être réalisée avec au moins l'un des trois systèmes suivants (voir pluiseurs !) :

- FSM,
- Behavior Tree
- Utility AI

Un document de spécification de ce système doit être produit par les game-designers. Ce document décrit :

- Les états ou actions de l'IA,
- Les paramètres du blackboard,
- Les transitions ou embranchements,
- Les calculs heuristiques servant à la décision (si existants).

Ce système doit être implémentée au projet par les programmeurs. L'utilisation de plugins d'IA tiers est autorisée.

Toute la configuration éditeur doit être réalisée par les game-designers (câblage des FSM, construction de Behaviour Tree, composition des actions avec scorers de l'Utility AI).

L'intelligence artificielle doit s'appuyer sur un Blackboard, soit fourni par Animator ou Behavior Tree, soit implémenté par les programmeurs. Ce blackboard doit être tenu à jour à chaque Update.

IV – Configuration d'IA

L'IA doit être aussi modulaire et configurable que possible. Les différents comportements de l'IA doivent être proprement séparés en modules.

Le système d'IA doit exposer autant de paramètres que nécessaire pour permettre aux game-designers de composer et régler librement ce système.

Les game-designers produiront une IA « facile » et une IA « difficile » en modifiant le comportement de l'IA et en ajustant les paramètres.

V – Niveaux de tests

Les game-designers produiront différentes scènes de test pour valider le comportement de l'IA dans des environnements simples.

L'IA doit savoir au moins :

- Se diriger vers une zone de contrôle
- Trouver les zones de contrôle non-conquises les plus proches (chemin),
- Éviter les astéroïdes,
- Éviter les mines,
- Tirer sur une cible immobile (vaisseau ou mine).

VI – Programmation

Les programmeurs doivent réaliser l'IA dans un contrôleur basé sur le code du fichier « ExampleController ».

Le contrôleur développé dérive du script « BaseSpaceShipController » et surcharge les méthodes :

- Initialize : En début de jeu, cette fonction permet de récupérer une référence vers le vaisseau contrôlé ainsi que les informations sur l'état initial de l'environnement de jeu.
- UpdateInput : A chaque boucle de jeu, cette fonction fournie des informations sur l'environnement de jeu et renvoie les inputs du vaisseau.

La structure « GameData » contient toutes les données nécessaires à l'analyse du contexte de jeu :

- Vaisseaux
- Points de contrôle
- Astéroïdes
- Mines
- Tirs
- Temps restant

Des propriétés sont disponibles sur les différentes classes pour vous faciliter l'accès aux données.

La méthode « UpdateInput » renvoie une structure de type « InputData » contenant :

- « thruste » : la puissance de propulsion en pourcentage entre 0 et 1.
- « targetOrient » : l'orientation vers laquelle le vaisseau doit se tourne, en degrés. Une orientation de 0 degré dirige le vaisseau vers la droite du plateau de jeu.
- « shoot » : indique si le vaisseau doit tirer.
- « dropMine » : indique si le vaisseau doit poser une mine.
- « FireShockwave »: indique si le vaisseau doit émettre une onde de choc.

Il est interdit d'altérer le fonctionnement normal du jeu :

- Ne pas modifier le code existant (demander en cas de nécessité)
- Ne pas modifier la position ou la physique des objets du jeu
- Ne pas créer ou détruire d'éléments de jeu sur la scène
- Ne rien rajouter sur la scène autre que les scripts permettant à votre IA de fonctionner (sur le contrôleur)

Afin de limiter les conflits lors du rassemblement des IA dans un même projet :

- **Le code produit doit être placé dans un namespace au nom de votre équipe !**
- **Les assets (scripts, prefabs, ...) doivent être placés dans un dossier au nom de votre équipe à la racine du dossier Teams (Teams/MaTeam) !**