ArenIA Project

Alexandra Blanc, Alp Yucesoy,
Anne Pourcelot, Clément Espeute,
Quentin "Johnny" Guye, Samory Ka, Simon Belletier

Sommaire

Part I
Introduction

Proposition de scénar 1

Année 2142, les progrès technologiques ont stagné, et les ressources commencent sérieusement à s'épuiser. L'humanité a entièrement automatisé toutes ses actions, plus personne n'a besoin de travailler ni même de sortir de chez soi. En revanche, les robots asservis à la collecte de ressources ont fleuris de partout et une nouvelle forme de contrebande est née. Ce n'est plus au contrebandier de récupérer les ressources directement, mais à lui de commander à ces robots de le faire pour lui, le mieux possible ... et surtout mieux que les autres !

Bien entendu, les robots sont totalement inaptes sans aide humaine pour les configurer et sont limités en tâches. Les vaillantes loquent que nous sommes devenus en 2142 ont pour but de concevoir les Intelligences Artificielles (IA) qui permettront aux robots d'accomplir leurs tâches.

Le monde ni vraiment dévasté, ni réellement entretenu, est un mélange de rues laissées à l'abandon, et de bâtiments emplis de lueurs de vies avachies dans leur sièges de bureau. On trouve dans ces rues des plantes totalement aléatoire qui ne laissent place à la civilisation que sur les axes principaux de passage des être mécanisés.

Part II
Fonctionnement du jeu

Vous l'avez compris, en tant que joueur, vous êtes une de ces loques physique mais brillante intellectuellement et devez créer les IAs du ou des robots que vous contrôlerez dans le but de réaliser des objectifs.

1 Objectifs

1.1 King of the hill

C'est simple, vous etes deux adversaires sur le terrain, mais il n'y a qu'une ressource collectable. Votre but est de récupérer avant votre adversaire toutes les ressources dont vous avez besoin en restant le plus longtemps possible dans la zone de collecte.

1.2 DeathMatch

Un face à face, les poussières qui volent, la musique western, Clint Eastwood une cigarette au bec. Vous vous regardez dans le blanc des diodes, le premier qui réduit l'autre à l'état de boite de conserve inanimée gagne.

1.3 Capture the flag

Simple : chacun convoite une ressource que l'autre a pour compléter sa construction. Le soucis, c'est qu'il ne peut rien céder. Le but étant de récupérer la ressource dans la base de l'autre et de revenir chez soi...sans se faire déconnecter par l'autre !

2 PvE et PvP

2.1 PvE : Player vs Environnement

Comme au début vous etes un novice, il est impossible pour vous de développer tout le potentiel de votre robot. Mais ! Fort heureusement, il existe un manuel du jeune développeur, votre robot est envoyé sur le terrain avec progressivement au fil de missions un panel de fonctionnalités qui s'ouvre.

Le programme lui-même peut se faire en ligne de code, mais le guide du débutant permet de se faire la main sur de la programmation en organigramme qui permet d'avoir des jolis blocs fonctionnels.

2.2 PvP : Player vs Player

C'est là que tout devient intéressant : après la partie PvE qui permet de connaître les outils pour lutter dans ce monde, vient la partie PvP. C'est l'heure de réellement récupérer les ressources. Le principe, vous choisissez un robot, son IA et vous le lachez dans la nature avec une mission (le mode de jeu). Manque de bol, vous allez tomber sur un adversaire qui avait la même mission que vous : il faudrait être meilleur ou lui passer sur les boulons !
Le système devrait trouver des adversaires sensiblement de même niveau s'il en existe et dans le même mode de jeu. Chaque partie rapporte de l'expérience et un de l'elo qui correspond à

votre niveau de joueur, pour éviter que vous ne tombiez contre des contrebandiers complètement ingérables ou au contraire, que vous ne soyez un newbie bully. Ca n'a rien de glorieux de "pawn du noob".
Quand vous voulez lancer une mission, vous soumettez un billet à un serveur qui va se charger de vous trouver un adversaire. Une fois l'adversaire trouver, il exécutera la mission et vous dira qui a gagné. Pour voir le match, il faudra alors relancer la dite mission pour que l'action se déroule sous vos yeux.

2.3 Coder son IA

Le jeu est pensé pour permettre aux débutants en programmation de jouer tout de même sans en avoir jamais fait au préalable. Le tutoriel est construit pour que de manière empirique, tout ce qui fait un bon code soit ajouté petit à petit. A cet effet, il est donc possible de coder de deux manières différents : en organigramme ou en ligne de code.

Organigramme

C'est la version accessible à tout un chacun. Très visuelle, elle fonctionne par blocs. Chaque blocs représentés par un gros rectangle ou autre forme parfaitement géométrique représente une instruction. Les instructions ont un impact sur l'intelligence du robot. Il devient alors capable de tirer, de rouler, ou de courir. Plus tard, on rajoutera les blocs "conditionnels" et les boucles qui lui permettront de recommencer ses actions plusieurs fois et de trouver des conditions pour les arrêter ou les entamer. Et avec les blocs "variable" le robot pourra retenir des informations qu'il connaissait auparavant. Ce qui fait que le robot devient à même de connaitre la position des ennemis qu'il voit, retenir la position d'un ennemi qu'il ne voit plus, compter l'énergie et la vie qui lui reste ou même anticiper une réaction ennemie.
Les blocs sont reliés par des flèches permettant facilement de suivre le chemin dans l'IA et comprendre ce que fait le robot en un seul coup d'œil.

Ligne de code

On rentre dans l'atmosphère du code, des lignes plus sévères et plus strictes en syntaxe et donc plus hermétiques. En échange, il devient plus rapide et plus efficace de faire son IA. On utilisera la syntaxe du langage de programmation "LUA". Les lignes de codes permettent autant que l'organigramme mais avec une souplesse accrue.

Part III
Robots

Qu'est-ce qu'on a sous la main pour ces arènes ?
Et bien une petite liste de robots avec chacun leur points forts et points faibles. Un robot tireur, un robot sprinter, un robot réparateur, tout les choix existent et sont combinables par modules.

1 Caractéristiques

Voyons un peu ses caractéristiques

Points de Vie : Une jauge de points de vie qui correspond à son degrés de casse. A zéro, le robot est H-S et ne pourra être réparé que par un humain après l'arène.

Energie : Une jauge qui représente l'énergie dont dispose le robot. Elle se régénère progressivement dans l'arène et est consommée par des sprints ou autres compétences. A zéro, le robot est inactif et doit attendre d'en récupérer suffisamment pour exécuter ses actions. Les mouvements de base et la vision n'ont aucuns coûts en énergie.

Degats : Capacité d'un robot à faire des dégâts à un autre.

Bouclier : Capacité d'un robot à encaisser les dégâts. (Algorithme non défini encore, mais sera connu du joueur)

Réparation : Capacité d'un robot à se réparer ou à en réparer un autre.

Vitesse : Vitesse de déplacement du robot (affectée par le terrain)

Vision : La distance maximale à laquelle voit le robot (affectée par le terrain)

2 Classe

Les robots ne sont pas naturellement doués partout, d'autant plus qu'un robot n'est de toute manière pas naturel. Ils ont donc chacun leurs spécificités qui jouent sur leurs caractéristiques et compétences. Quelques exemple d'archétypes :

2.1 L'équilibré

Bon en tout excellent en rien. C'est un peu le couteau suisse, il peut tout faire, mais il existera toujours des autres robots qui feront certaines taches mieux que lui. Idéal sur une mission nécessitant de la versatilité.

PV : Normal.

Energie : Normale.

Degats : Normaux

Bouclier : Normal.

Réparation : Normale.

Réparation max : Normale.

Vitesse : Normale.

Vision : Normale.

2.2 Le sprinteur

Son truc à lui, c'est de rush l'objectif. La collecte rapide c'est son truc, faire tourner les autres robots en bourrique aussi.

PV : Pas des masses ...

Energie : Survolté !

Degats : Raisonnablement.

Bouclier : Basique..

Réparation : Normale.

Réparation max: Faible.

Vitesse : Exceptionnelle !

Vision : Normale.

2.3 Le fracasseur

Un robot dont le but est de détruire son adversaire plutôt que d'utiliser la ruse (la ruse, c'est comme fuir, c'est lâche).

PV : Beaucoup.

Energie : Normale.

Degats : Élevés !

Bouclier : Robuste !

Réparation : Faible.

Réparation max: Faible. A chaque coup il arrache des éléments à l'autre robot qui lui permet de se constituer un petit stock de pièces de rechange.

Vitesse : Normale.

Vision : Réduite.

2.4 Le réparateur

Celui-ci préfère le calme et la tranquillité plutôt que de semer la mort et la dévastation. Il a plus un rôle de soutien et prend tout son sens dans des combats à plusieurs robots par équipe.

PV : Normal.

Energie : Normale.

Degats : Faibles.

Bouclier : Convenable

Réparation : Medic' !

Réparation max: Incroyable !

Vitesse : Normale. Améliorée si la cible de son déplacement est un allié mal en point.

Vision : Normale.

2.5 L'assassin

Un robot très peu résistant mais ayant une capacité à détruire rapidement sa cible. Pour compenser son manque de résistance, il gagne en potentialité de surprise.

PV : Peu...

Energie : Elevée.

Degats : Insane !!

Bouclier : Minime...

Réparation : Minime...

Réparation max: Faible.

Vitesse : Rapide.

Vision : Accrue !

3 Personnalisation

Les robots ont bien entendu des archétypes bien marqués pour qu'on sache contre quoi l'on se bat, mais ils ont aussi des atouts supplémentaires qui sont dépendants de la classe mais personnalisables. De plus, chaque robot à trois compétences précises dépendante des personnalisations. Il existe quatre sortes de personnalisation, les voici.

3.1 La tete

C'est le module de vision et la vitesse de rotation de de la tête autour de son support.

3.2 L'arme

Son angle est indépendant de l'angle de vision et a une vitesse de rotation propre aussi.
Ce module définit la portée de l'arme et sa cadence de tir et définit le premier slot de compétence : l'arme.

3.3 Roues

Ici on a la vitesse de rotation du robot par rapport au sol et sa vitesse en déplacement linéaire. Le deuxième slot est dépendant de celui-ci et est une compétence intimement liée à la classe.

3.4 Chipset

Ajoute directement la troisième compétence. C'est juste une puce placée dans la tete qui ne permet pas d'améliorer le matériel mais d'en permettre un usage plus efficace. Elle permet donc l'accès à des fonctions particulières supplémentaires ou des améliorations des autres compétences justifiées par une "optimisation" de ces dernières. Ce slot une compétence passive. Il ne faut pas l'activer soit même mais provoquer ses conditions d'activation.

4 Liste des compétences pour chaque archétypes

4.1 Equilibré

Armes

- Arme standard, portée moyenne, dégats moyens.

Chassis

- Trousse de première réparation

Chipset

- A voir

4.2 Sprinteur

Armes

- Portée réduite et dégâts légers.

Châssis

- Pack turbo : propulse le robot et brule les robots proches.

Chipset

- A voir

4.3 Fracasseur

Armes

- Portée réduite, dégats lourds.

Châssis

- Berserk : gagne un stéroïde de vitesse et de cadence.

Chipset

- Destruction partielle : récupère une partie des PVs du robot qu'il détruit en stocks de réparation.

4.4 Assassin

Armes

- Portée extremement faible, dégats monstrueux.

Chassis

- Stealthed : se rend invisible pour les autres robots pour une courte durée.

Chipset

-

4.5 Réparateur

Armes

- Portée corps à corps, dégâts pour la forme

Chassis

- Medic' : se répare lui-même ou répare un robot allié.

Chipset

- Ambulance : gagne en vitesse s'il vise une cible alliée avec peu de points de vie.
- Garage : gagne en vitesse de réparation

5 Chipsets communs

Taunty : donne la possibilité de taunt son adversaire, lui signifiant clairement l'absence de chipset.

Batterie de rechange : donne la possibilité de recharger totalement sa batterie une fois par partie.

Icare : permet de franchir les obstacles de petites tailles.

Not alone : inflige des dégâts aux robots de proches du robot porteur lors de la mort de ce dernier.

Part IV
Rendu minimum

Bien sur tout ça c'est bien beau, mais pour l'instant on y est pas encore. Alors pour premiers objectifs pour que le jeu fonctionne sur de bonnes bases, voici une liste exhaustive.

1 Conception d'une Arène

Il faut concevoir un environnement de jeu de préférence modulable qui sera le terrain de jeu.
Dans le minimum, il suffit d'un terrain avec des cases repérées par des indices uniques.
Il faut prévoir de quoi modifier les conditions de victoires, repérer les évènements inhérents à l'arène.
L'arène doit contenir des obstacles et des zones différentes qui pourraient impacter le déplacement, la portée et ou la visibilité d'un robot.
Si on opte pour des un système avec des effets aléatoires, alors il faut pouvoir les recréer lors d'un récapitulatif de bataille. Donc générer pour chaque partie une seed aléatoire pour que seule la clé ait besoin d'être stockée pour que les valeurs de l'aléatoire soit régénérées.
Anticiper une possible modification de l'arène par des évènements soit liés au robot soit liés à des évènements de l'arène.

2 Robots

Il faut des robots avec IA dynamique qui s'exécutera à partir du script codé par le joueur.
Le repérer par un flag pour connaitre son propriétaire et donc s'il est allié ou ennemi.
Créer son système de déplacement et d'orientation qui soit récupérable par et pour n'importe quel robot visible d'un autre.
Choisir un système de vision. Dans l'idée, ce serait un cercle autour du robot qui peut être réduit à un cône (paramétrable) et qui serait masqué par les obstacles de la carte.
Le robot doit pouvoir utiliser une compétence de dégâts (tirs ou coups au corps à corps).
Posséder des jauges de points de vie et d'énergie, les autres statistiques sont là pour l'évolution et l'interaction statistiques avec le décors.
Gérer à minima les collisions avec l'arène et les autres entités.
Préparer les différents slots d'attaques disponibles. Seul le slot d'arme est nécessaire pour un premier jet. Le deuxième est à envisager rapidement, le troisième est bonus.

Autres

Surveiller la réparation.

3 Interface de code

Cette partie est à anticiper car elle est au cœur du projet, mais elle ne viendra qu'après avoir mis en place les mécaniques de jeu.

Lignes

Une interface pour écrire du code (un traitement de texte avec peut-être une coloration syntaxique).
Un ensemble d'instructions listées et documentées (va falloir faire du code propre les amis !).
Des instructions de debug comme une instruction qui garde le chemin du robot.

Organigramme

De même, une liste des blocs qui correspondent aux commandes organisées par utilités.
Bien faire ressortir via des codes couleurs les usages des blocs : chaque bloc a un type, chaque type à une couleur, voici ceux qu'il faut implémenter à minima :

variable : bloc de définition d'une variable.

boucle : bloc initiant une boucle et décrivant une boucle. Dans l'absolu, il est tout à fait possible en organigramme de ne se servir que des conditions. Mais cela revient à un usage de goto et il est préférable d'en utiliser un minimum. Cela permettra en plus d'alléger le code.

test : bloc de test de type if. De même un bloc else if ou else n'existe pas vraiment, il n'existe donc qu'un seul bloc test qui est if.

opération : bloc effectuant des opérations sur des variables

appel de fonction : bloc appelant une fonction, qui peut parfois nécessiter des variables en paramètre

début/fin de script : bloc définissant le début et la fin des scripts d'IAs.

Anticipations nécessaires diverses

Penser à faire les strings affichées dans un fichier séparés pour permettre un changement de langue facile.
Préparer les endroits où lancer du son et les mettre dans deux filles, une BGM, une SFX.