Projet Logiciel Transversal

Matthieu DEBARGE – Romain LOURY

Table des matières

1 Objectif

1.1 Présentation générale

L'objectif du projet « Guerre des gangs » est la réalisation d'un jeu de stratégie pouvant se jouer seul contre une intelligence artificielle ou à plusieurs. Le jeu se déroule sur une carte Il s'agit d'un jeu au tour par tour, l'état du jeu n'est modifié qu'à la fin d'un tour après que chaque joueur ait renseigné son coup. Le principe est simple, conquérir les territoires des gangs adverses en déplaçant ses recrues sur une carte.

1.2 Règles du jeu

La carte est composée d'un ensemble de territoires (31) et de 6 quartiers généraux. Chacun des joueurs (physique ou intelligence artificielle) possède un quartier général, le but du jeu est de conquérir les QG de ses adversaires. Chaque joueur commence avec un certain nombre de recrues dans son QG. Pour conquérir d'autres territoires, le joueur peut envoyer une partie de ses recrues sur des territoires adverses, il s'agit d'une attaque. Lorsqu'un joueur envoie un nombre de recrues supérieur au nombre de recrues présent sur un territoire alors il obtient ce territoire. En revanche si un joueur attaque un territoire adverse ayant plus de recrues, le joueur attaquant se voit perdre toutes les recrues qu'il a engagées dans l'attaque et n'obtient pas le territoire qu'il souhaitait, le territoire attaqué conserve toutes ses recrues, ou une partie en fonction du nombre de recrues l'ayant attaqué. En cas d'égalité le territoire gagnant est attribué aléatoirement. A chaque tour tous les joueurs définissent au maximum 3 déplacements d'un territoire vers un autre. Tout l'intérêt du jeu et de la stratégie est d'essayer d'anticiper les coups des adversaires, pour savoir où placer ses recrues. Après chaque tour, chaque joueur se voit attribuer une certaine valeur d'argent lui permettant d'acheter de nouvelles recrues. Plus le joueur possède de territoires, et de QG, plus il obtient d'argent. Pour ajouter de la stratégie au jeu, il existe des cartes, qui peuvent aussi être achetées par le joueur. Les cartes sont plus ou moins puissante en fonction de leur prix, il existe trois catégories de cartes de puissance différentes, le joueur peut en acheter une seule à chaque tour et ne connait pas ses effets lorsqu'il l'achète, il connait uniquement la puissance de la carte. Les cartes ont des effets sur le jeu, voici une liste non exhaustive d'effets introduits par application d'une carte : réduction de l'effectif de recrues adverse d'un certain coefficient, ajout d'un certain nombre de recrues à un secteur, annulation des attaques des ennemies sur un ou plusieurs territoires, et d'autres. Le jeu s'arrête lorsque tous les QG sont conquis par un joueur. Lorsqu'un joueur n'a plus de QG ni de territoires, la partie est perdue pour lui.

1.3 Conception Logiciel

Présenter ici les packages de votre solution, ainsi que leurs dépendances.

2 Description et conception des états

L'objectif de cette section est une description très fine des états dans le projet. Plusieurs niveaux de descriptions sont attendus. Le premier doit être général, afin que le lecteur puisse comprendre les éléments et principes en jeux. Le niveau suivant est celui de la conception logiciel. Pour ce faire, on présente à la fois un diagramme des classes, ainsi qu'un commentaire détaillé de ce diagramme. Indiquer l'utilisation de patron de conception sera très appriécé. Notez bien que les règles de changement d'état ne sont pas attendues dans cette section, même s'il n'est pas interdit d'illustrer de temps à autre des états par leur possibles changements.

2.1 Description des états

Un état du jeu est défini par l'état de l'ensemble de ses éléments. Tous les éléments sont fixes, la carte ne se déplace pas au cours du jeu, ni les autres éléments. Au cours du jeu, certains attributs évoluent et d'autres sont fixes. Voici la liste des éléments du jeu.

-Elément **« Gang »** : Les gangs sont au nombre de 5 et sont définis par 5 attributs, l'argent disponible, le nombre de recrues, les cartes possédées, le nom du gang fixe lui aussi, un booléen définissant si le Gang est encore en vie.

-Elément **« Map »** : La carte est définie par une texture de fond fixe de couleur blanche, elle est composée de secteurs qui ont la couleur blanche du fond par défaut, puis leur couleur évolue avec le jeu.

-Elément **« Secteur »** : Les secteurs sont définis par un ensemble de tuiles dont la position est fixe sur la carte. Le secteur possède une texture attribuée aux tuiles correspondante à la couleur du gang auquel il appartient, ainsi qu'un nombre de recrues présent sur le secteur, le nombre de recrues sera affiché avec un Sprite, positionné à un endroit fixe de la carte. De plus à chaque secteur correspond des secteurs adjacents, les secteurs étant fixes au cours du jeu, les adjacences sont fixes elles aussi.

-Elément **« Carte »**: elles sont au nombre de 9, et de 3 types différents, A, B, ou C.
Type A: -$nombre\ de\ recrues\ des\ secteurs\ adjacents \times 0.25$
-$nombre\ de\ recrues\ des\ secteurs\ adjacents + 10$
-$Annulation\ des\ attaques\ provenant\ des\ secteurs\ adjacents$

Type B :-$nombre\ de\ recrues\ des\ secteurs\ adjacents \times 0.50$
-$nombre\ d'alliés\ de\ l'attaque\ \times 2$
-$Annulation\ des\ cartes\ ennemies\ joués\ dans\ les\ secteurs\ adjacents$

Type C :-$nombre\ de\ recrues\ du\ secteur\ attaqué\ \times 0.50$
-$nombre\ d'alliés\ d'une\ attaque\ +\ 10$
-$Annulation\ des\ cartes\ jouées\ dans\ le\ secteur\ attaqué$

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

Illustration 1: Diagramme des classes d'état

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémente pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

Illustration 2: Diagramme de classes pour le rendu

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects lié au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

Illustration 3: Diagrammes des classes pour le moteur de jeu

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est un parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

Illustration 4: Diagramme de classes pour la modularisation