Projet Logiciel Transversal "Heroes of might and magic"

Mehdi NACER - Maxime HEURTEAU

Table des matières

Objectif

Présentation générale

L’objectif du projet est de réaliser un jeu de stratégie tour par tour, s’inspirant du jeu «Heroes of might and magic» dont les règles sont développées dans la section suivante.

Règles du jeu

L’univers du jeu est un monde médiéval-fantastique, qui sera représenté sur une carte en 2D avec une vision vue dessus.

Au début du jeu, chaque joueur possède un territoire avec un peuple, un héros et une armée. Le peuple permet de générer des ressources. Pour se développer, le joueur utilise les ressources.

Le jeu peut passer en mode "guerre", lorsque le joueur déclenche une attaque contre une autre armée. L’armée est menée par le héros. Elle suit le héros dans ses déplacements et ses attaques. Si l’armée gagne la guerre, elle gagne des ressources et agrandit le territoire, dans le cas contraire elle perd des ressources et des territoires, son armée est aussi vidée de ses soldats. Le principe de la guerre est de vider les points de vie de l’adversaire grâce à des attaques qui auront un niveau donné. Les attaques sont en tour par tour.

ON REMPLACE LE GAIN DE TERRAIN PAR UN GAIN DE NIVEAU QUI PERMET D’AVOIR DES CAPACITÉS EN FONCTION DU NIVEAU.

La taille maximum de l’armée est fonction de la taille des territoires.

Les différents personnages sont:

- les villageois: leurs nombres dépend de la taille du territoire du joueur. Ils permettent de créer des ressources.
- les soldats: ils existent différents types de soldats: cavaliers, archers... Ils coûtent des ressources pour être entraînés. Ils sont contrôlés par le héros et sont définis par des points de vie et une attaque.
-le héros: le héros est choisi au début du jeu parmi une liste de héros. Il a des pouvoirs spécifiques. Il mène l’armée de soldats pendant les attaques.

Les différents bâtiments sont:

-les mines: elles contiennent les ressources. Les villageois vont chercher dans ces mines pour avoir des ressources.
-la caserne: elle permet en échange de ressources et d’un temps donné d’avoir des soldats.
-un château: sert au héros pour se reposer et donnera accès au menu (sauvegarde, langue, son...). Le château aura aussi des points de vies qui seront ajoutés à ceux des soldats lors d’une attaque adverse.

Conception Logiciel

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

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.

Description des états

L'états du jeu est modélisé par deux états: un états pour le jeu en général (nommé village) qui modélise le village et un état qui modélise la phase de guerre. Chacun des états est formés par un ensemble d'éléments fixes et d'élément mobiles. Chaque élément est défini par:

- ses coordonnées dans la grilles (x,y)
- sa taille
- son identifiant d'élément

État de jeu village

Cet état modélise le village du joueur. Ces éléments fixes sont les bâtiments (château, mine, caserne) et les éléments de décors, et ces éléments mobiles sont le héros et les villageois.

État des éléments fixes

La grille est défini en deux types de cases: les cases «libres» et les cases «non-libres».

Cases libres: Les cases libres sont modélisés par de l'herbe et pourront être chevauché par les élément mobiles.

Cases «non-libres»: Les cases non-libres ne peuvent pas être chevauché par les éléments mobiles et sont soit un bâtiments soit un éléments de décors.

État des éléments mobiles

Les éléments mobiles possède une direction (aucune, droite, gauche, haut, bas), une vitesse, une position.

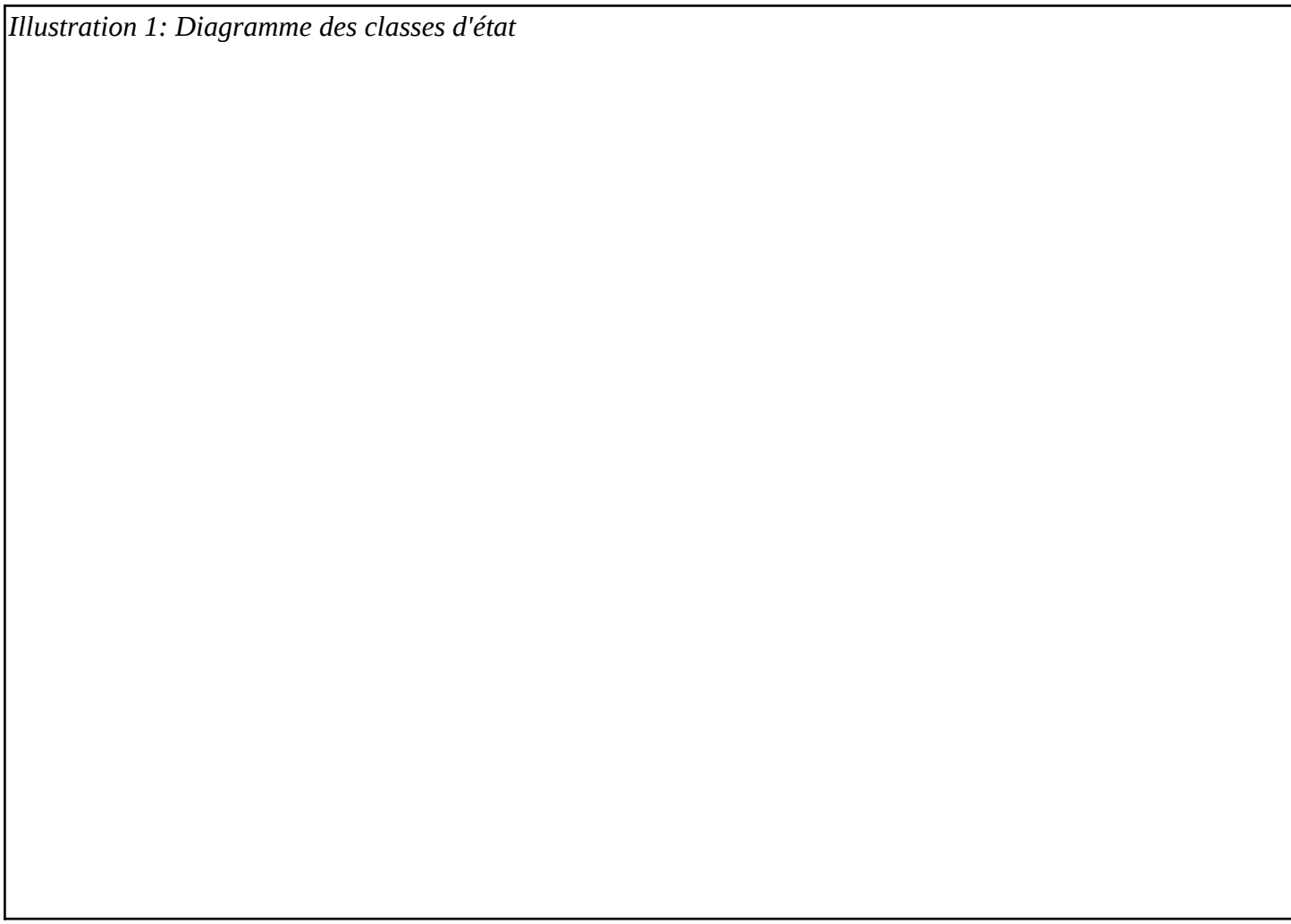
Conception logiciel

Conception logiciel : extension pour le rendu

Conception logiciel : extension pour le moteur de jeu

Ressources

Illustration 1: Diagramme des classes d'état



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.

Stratégie de rendu d'un état

Conception logiciel

Conception logiciel : extension pour les animations

Ressources

Exemple de rendu

Illustration 2: Diagramme de classes pour le rendu

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.

Horloge globale

Changements extérieurs

Changements autonomes

Conception logiciel

Conception logiciel : extension pour l'IA

Conception logiciel : extension pour la parallélisation

Illustration 3: Diagrammes des classes pour le moteur de jeu

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.

Stratégies

Intelligence minimale

Intelligence basée sur des heuristiques

Intelligence basée sur les arbres de recherche

Conception logiciel

Conception logiciel : extension pour l'IA composée

Conception logiciel : extension pour IA avancée

Conception logiciel : extension pour la parallélisation

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.

Organisation des modules

Répartition sur différents threads

Répartition sur différentes machines

Conception logiciel

Conception logiciel : extension réseau

Conception logiciel : client Android

Illustration 4: Diagramme de classes pour la modularisation