# Examen réparti Mars 2015 – Warcraft ™ !

Binh-Minh Bui-Xuan

2 heures. Tout document personnel non-électronique autorisé.

## Introduction

Ce sujet fait partie d'un devoir global (examen réparti 1 + projet) consistant à formaliser, à l'aide du langage de spécification de service vu en cours, un modèle simplifié du mode solo du jeu `Warcraft: Orcs and Humans` ™, développé par Blizzard en 1994.

Dans cet épisode de la série, comme dans tout épisode de cette série à l'innovation discutable, les joueurs dispensent des ordres à gogo à leur chers sujets – orcs ou humains – en appuyant et déplaçant furieusement leur main sur un objet rond à boutons cliquables...


## Enoncé de l'examen

Le barême est indicatif. L'objectif est d'exploiter au mieux le langage de spécification étudié en cours. On souhaite obtenir une spécification *cohérente*, *complète* et bien modulaire. Décrire les services suivants :

**Villageois (5 points)**
Le service Villageois représente les sujets les moins favorables dans la société Azeroth. Ces parfaits exemples de la servitude sont distingués non seulement par leur `race` et leurs dimensions – plus précisément leur `largeur` et `hauteur` – mais aussi par leur `force`, leur `vitesse` et les précieux `points_de_vie` leur restant. On dispose également d'un observateur donnant leur état de service, c.à.d. si le villageois `est_mort` ou si l'on peut continuer à l'utiliser sans trop de problème. Finalement, on dispose d'un observateur indiquant la `quantité_d_or` que le villageois, à défaut de posséder, en est chargé du transport. Il est à noter que la race d'un villageois est obligatoirement RACE.ORC ou RACE.HUMAN. Lorsque la situation permet, ce service offre l'opération de `retrait` de points de vie, qui est probablement la seule chose notable que l'on peut s'attendre de tels figures.

**Mine et Hôtel de Ville (3 points)**
La Mine d'or représente les ressources les plus convoitées dans le monde d'Azeroth, où les villageois travaillent dans la joie du Prophète-joueur et, généralement, aussi au profit de celui-ci. Ces objets de choix sont reconnaissables par leur dimensions (`largeur` et `hauteur`), leur quantité d'`or_restant`, et leur `etat_d_appartenance`, indiquant si oui ou non cette ressource est attribuée au joueur. Ce service offre l'opération de `retrait` de quantité d'or, et un observateur indiquant si la mine `est_laminée` : chose qui arrive dès que sa quantité d'`or_restant` n'a plus de valeur profitable. Pendant le partiel, le service HôtelVille ne diffère au service `Mine` que par son nom. Il n'est pas à recopier sur la feuille d'examen.

**Moteur du jeu (12 points)**
Le service MoteurJeu modélisera le jeu lui-même. Son squelette est donné ci-dessous et on n'aura pas à définir d'autres observateurs que ceux définis dans le squelette. Ce service aura comme principale opération le calcul d'un pas de jeu. Lors de ce calcul, on passera en paramètre un ordre éventuel à dispenser à au plus un villageois par pas de jeu : se déplacer selon une direction, prendre possession d'une mine abandonnée, voire vider ses poches au profit du joueur.

Le MoteurJeu supervise un terrain de jeu vide sur lequel se trouvent exactement un HôtelVille, trois Villageois, et un nombre indéfini de Mine. Le terrain de jeu est représenté par deux entités, largeur et hauteur, constamment fixées à certaines valeurs raisonnables. Les villageois sont initialement positionnés sur le terrain à moins de 51 pixels de l'hôtel de ville, leur autres caractéristiques sont initialisées par des valeurs libres à l'imagination du lecteur. L'hôtel de ville est par défaut attribué au joueur, et contient 16 en quantité d'or. Une mine apartient au joueur lorsqu'un villageois en prend la possession et la visite régulièrement. Si une mine n'est pas visitée pendant 51 pas de jeu, elle devient abadonnée et son

`etat_d_appartenance` change de manière correspondante. Un villageois peut visiter une mine ou un hôtel de ville s'il se trouve à moins de 51 pixels par rapport à cet objet.

Le nombre maximal de pas de jeu sera fixé à l'avance et on prendra soin de détecter la fin du jeu avec un observateur adéquat : partie gagnée ou perdue, selon le point de vue de chacun. On ne pourra entreprendre un pas de jeu supplémentaire si la partie est terminée. Les possibilités de terminaison de partie sont les suivantes : 1. le joueur a réussi à amasser 1664 en quantité d'or dans son hôtel de ville avant que le nombre maximum de pas de jeu ait été atteint : il part (enfin) rejoindre le dîner familial dans la joie de l'accomplissement ; 2. le joueur, malgré lui, ne réussit pas à s'enrichir après que le nombre maximum de pas de jeu a été atteint : la partie est finie, tristement, le joueur n'a plus qu'à se venger sur les chips.

La seule opération du service MoteurJeu concerne les commandes joueur : on mettra à jour les positions des villageois suivant ces commandes, ainsi que les profits éventuels résultant de leurs actions béates :

- visiter une mine abandonnée affecte celle-ci au joueur ;
- visiter une mine implique le début de la corvée dans cette mine, en ce qui concerne le villageois ;
- une corvée dure 16 pas de jeu, pendant lesquels le villageois reste à la mine, travaillant en chantant ;
- à la fin d'une corvée de mine, le joyeux villageois réapparaît à côté de la mine en question, en soutirant 1 quantité d'or de celle-ci ;
- visiter un hôtel de ville vide les poches du villageois au profit de l'hôtel de ville, le villageois est prêt à repartir travailler dès que ses poches sont allégées.

On donne ci-dessous une version incomplète de MoteurJeu (à ne pas recopier sur la feuille d'examen). Ajouter toutes les observations nécessaires afin de rendre la spécification de MoteurJeu complète.

**service** : MoteurJeu
**types** : enum RESULTAT{GAGNE, PERDU},
enum COMMANDE{RIEN, DEPLACER, ENTRERMINE, ENTRERHOTELVILLE}
**observators** :
**const** largeurTerrain : [MoteurJeu] → int
**const** hauteurTerrain : [MoteurJeu] → int
**const** maxPasJeu : [MoteurJeu] → int
pasJeuCourant : [MoteurJeu] → int
estFini : [MoteurJeu] → boolean
resultatFinal : [MoteurJeu] → RESULTAT
**pre** resultatFinal(M) **require** estFini(M)
**const** numeroesVillageois : [MoteurJeu] → Set<int>
getVillageois : [MoteurJeu] × int → Villageois
**pre** getVillageois(M,num) **require** num ∈ numeroesVillageois(M)
positionVillageoisX : [MoteurJeu] × int → int
**pre** positionVillageoisX(M,num) **require** num ∈ numeroesVillageois(M)
positionVillageoisY : [MoteurJeu] × int → int
**pre** positionVillageoisY(M,num) **require** num ∈ numeroesVillageois(M)
**const** numeroesMine : [MoteurJeu] → Set<int>
getMine : [MoteurJeu] × int → Mine
**pre** getMine(M,num) **require** num ∈ numeroesMine(M)
**const** positionMineX : [MoteurJeu] × int → int
**pre** positionMineX(M,num) **require** num ∈ numeroesMine(M)
**const** positionMineY : [MoteurJeu] × int → int
**pre** positionMineY(M,num) **require** num ∈ numeroesMine(M)
hotelDeVille : [MoteurJeu] → HôtelVille
**const** positionHotelVilleX : [MoteurJeu] → int
**const** positionHotelVilleY : [MoteurJeu] → int
peutEntrerMine : [MoteurJeu] × int × int → boolean
**pre** peutEntrerMine(M,numVillageois,numMine) **require** numVillageois ∈ numeroesVillageois(M)
∧ numMine ∈ numeroesMine(M)
peutEntrerHotelVille : [MoteurJeu] × int → boolean
**pre** peutEntrerHotelVille(M,numVillageois) **require** numVillageois ∈ numeroesVillageois(M)
**Constructors** :
init : int × int × int → [MoteurJeu]
**pre** init(largeur,hauteur,maxPas) **require** largeur≥ 600 ∧ hauteur≥ 400 ∧ maxPas≥ 0
**Operators** :
pasJeu : [MoteurJeu] × COMMANDE × int × int → [MoteurJeu]

$$\textbf{pre}\ \text{pasJeu(M,commmand,numVillageois,argument)}\ \textbf{require}\ \begin{cases} \neg\text{estFini(M)} \\ \text{c=DEPLACER} \Rightarrow 0 \leq \text{argument} \leq 360 \\ \text{c=ENTRERMINE} \Rightarrow \begin{cases} \text{argument} \in \text{numeroesMines(M)} \\ \text{peutEntrerMine(M,numVillageois,argument)} \end{cases} \\ \text{c=ENTRERHOTELVILLE} \Rightarrow \text{peutEntrerHotelVille(M,numVillageois)} \end{cases}$$

**Observations** :
[invariants]
$0 \leq$ pasJeuCourant(M) $\leq$ maxPasJeu(M)

### Question bonus : le mode multi-joueur, la sélection multiple

En mode multi-joueur deux joueurs s'affrontent pour un maximum de `quantité_d_or` dans leur hôtel de ville, et généralement pas de chips. La sélection multiple se traduit par la possibilité de dispenser une commande (commune) à un groupe de villageois pendant un seul pas de jeu. Expliquer les changements et/ou modifications qu'il faudrait apporter aux spécifications précédentes pour étendre le mode solo en mode multi-joueur, et inclure dans le *gameplay* la sélection multiple.