Composant : Projet Warcraft

Samuel Bell-Bell , Eric Lim

12 mai 2015

1 Introduction

1.1 Présentation

Nous avons, pour ce projet, fait une conception par contrat.
Nous n'avons fait aucun supplément indiqué dans l'énoncé du projet. Une phase importante du développement a été l'ensemble des étapes à faire pour le moteur du Jeu. En effet, nous avons centralisé toutes les fonctionnalités dans le moteur du Jeu. Les autres services (Mines, Route...) sont donc assez minimales.

Warcraft étant un jeu de stratégie en temps réel sur un grand terrain, les principaux problèmes sont la gestion du terrain et des commandes qui sont dans cette version simplifiée au nombre de 3.

2 SPECIFICATION

Le dossier specs contient les spécifications que nous avons écrites sous forme .txt.
Dans cette partie, comme les services autres que moteurJeu sont minimales, nous n'allons pas en parler, à la place, nous allons évoquer le moteur de jeu et les principaux problèmes que ca nous a causé.

2.1 MoteurJeu et modélisation

Comme dit précédemment, le moteur de jeu est la pièce centrale du Jeu. Nous avons utilisé le pattern Require/provide pour ce composant. Il requiert

des villageois, mines, hôtels de ville... pour fonctionner.
Nous avons modélisé les positions de tout les élements du jeu dans le moteur de Jeu sous forme de 2 Hashmap<Object, Point>, l'un regroupant les positions des villlageois, l'autre celles des bâtiments.
Cela est du au fait que la position des bâtiments est statique tout au long de la partie (ce qui fait qu'il n'y a besoin de tester qu'une seule fois leur position et c'est après l'initialisation) alors que la position des villageois évolue constamment au fil du jeu ce qui implique de tester leur position après chaque appel de la fonction pasJeu.

Pour les déplacements, nous avons découpé l'angle de déplacement en 8 pour modéliser les 8 directions possibles de déplacement (est, sud est, etc.).

Pour la corvée, nous avons fait 2 listes d'Integer.
Le premier villageoisAttente a n cases (n correspondant au nombre de villageois). Si le villageois d'indice i dans la liste des villageois est en corvée, on stockera l'indice i la valeur pasCourant + 16. Cette valeur correspond au numéro du pas jeu dans lequel on devra débloquer le villageois.

Le deuxième MineMinee aussi de n cases, chacunes modélisant un villageois, permet de savoir si un villageois mine (dans ce cas la valeur dans le tableau sera différent de -1) et dans quel mine il se trouve(on retire les pièces d'or lorsque le villageois a fini de miner).

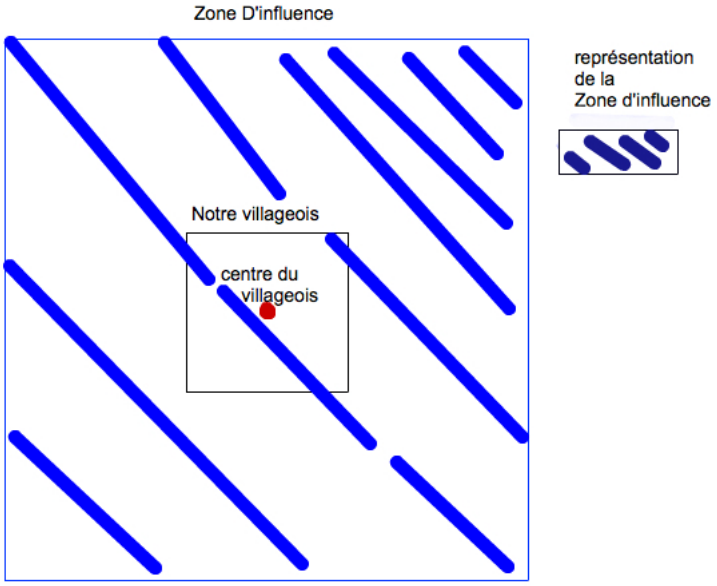
Chaque case du modèle est un carré de 1x1 pixel. Les villageois et bâtiments sont représentés par des carrés de pixels.(10x10 pour un villageois, 50x50 pour les bâtiments autres que muraille et route).
Cette modélisation a posé des soucis pour savoir si un bâtiment/villageois se trouvait hors de la carte ou pas. Dans ce cas il fallait tester si les 4 sommets du carré se trouvait ou non sur la carte. A noter que nous n'avons pas gérer le problème des collisions. Plusieurs villageois peuvent donc se trouver sur la même case que ce soit de la même race ou d'une race différente.

2.1.1 zone d'influence

Les commandes ENTRERHOTELVILLE et ENTRERMINE pour la fonction pasJeu requiert que le villageois se trouve à moins de 51 pixels de la mine/hôtel de ville pour y entrer.
Le problème ici est que, comme nous avons modélisé les villageois et les bâtiments sous formes de carré, la condition pour entrer va donc être changer.

Cette condition devient alors que la distance entre au moins un point du carré du villageois et un point du carré du bâtiment dans lequel il veut entrer soit inférieur à 51 pixels.

Pour résoudre ce problème, nous avons décider de créer un carré modélisant la “zone d’influence du villageois”.
Ce carré, de côté 51 + longueur du villageois/2 pixels a pour centre le centre du carré représentant le villageois permet de palier à ce problème. En effet, il suffit juste alors juste de tester si le carré de la zone d’influence et le carré du bâtiment dans lequle le villageois veut entrer s’intersecte ce qui est très simple en JAVA avec la méthode intersects de la classe Rectangle.



3 TEST

Nous avons fourni un ensemble de test montrant que l’implémentation,l’implementationError et le contrat marche bien , à l’aide de ceux-ci nous avons pu découvrir de nombreux bugs de notre implémentation de moteur de jeu.

Pour écrire les tests, nous avons utilisé Junit 4 :
-Nous avons fait 36 tests au total dont certain avec plusieurs assert.

3.1 Objectif de tests

L'ensemble des tests ont été fait avec la même architecture que les tests vu en cours et en TD. Le dossier contenant ces descriptions se trouve dans le dossier obj_tests qui possède les fichiers :
hotelVille-obj-tests.txt
Mine-obj-tests.txt
MoteurJeu-obj-tests.txt
Muraille-obj-tests.txt
Route-obj-tests.txt
Villageois-obj-tests.txt

Dans le build.xml, la commande ctest lance les tests du contrat, btest les test sur l'implémentation bugué et enfin test les tests sur l'implémentations.

4 Message d'erreur version bugué

Messages d'erreur des 36 tests que nous avons fait sur l'implémentationError :
testSetAbandonCompteur, SetAbandonCompteur de la hdv ne s'est pas fait correctement
testAbandonedPositif, abandoned de hdv ne s'est pas fait pas correctement
appartenance : HUMAIN
testAccueilPositif, L'accueil de la hdv n'a pas fonctionnee
testSetOrRestantPositif, orRestant de la hdv ne s'est pas fait correctement
testSetAppartenance, SetAppartenance de la hdv ne s'est pas fait correctement
testDepotPositif, Le depot de l'hdv n'a pas fonctionnee
testInitPositif, L'or restant de la hdv ne s'est pas initialisee correctement
init, La hauteur du villageois ne s'est faite pas correctement
estMort, Le retrait de vie qui doit detruire la muraille n'a pas fonctionne correctement
retrait, Le retrait de vie de la muraille n'a pas fonctionne correctement
init, La hauteur de la muraille ne s'est pas initialisee correctement
test10_assert1 :HOTEL DE VILLE n'est pas DEVENUE HUMAINE alors qu'elle le devrait
test7 : le villageois s'est deplace en dehors du terrain puis doit etre remis a sa position d'origine normalement mais ce n'est pas le cas
test9 : le villageois 1 HUMAIN et LE VILLAGEOIS 3 ORC , sont censes

etre mis sur la meme case et pourtant.
test8 : le villageois 1 prend possession de son hotel de ville
test5 : Le villageois 1 ne s'est pas bien deplace
test6 : le villageois s'est deplace dans la muraille , c'est impossible
test11 : l'hotel de ville n'est pas abandonnee alors qu'elle le devrait
test12 : la mine n'est pas abandonnee alors qu'elle le devrait
estEntrerMine a rate test4 : L'appartenance devrait etre ORC pour l'hotel de ville J2 ce n'est pas le cas
MJ init,La valeur du pas courant n'est pas initialise correctement
test3 : L'appartenance devrait etre ORC
Probleme Entrer HotelVille. L'orRestant devrait etre egale a l'orRestant a l'initialisation de HotelVille + 1
test13_1 : La partie devrait etre fini au bout de nbPasJeu max mais ce n'est pas le cas
test14 : L'HotelVillel HUMAIN contient 1664 pieces d'or. La partie devrait etre fini mais ce n'est pas le cas
init, Les point de vie du villageois ne s'est pas initialisee correctement
retrait, retrait ne s'est pas correctement executee
setQtor, setQtor ne s'est pas correctement executee
testAccueilPositif, Le accueil de la mine ne s'est pas fait correctement
testRetraitPositif, Le retrait de la mine ne s'est pas fait correctement
testInitPositif, La hauteur de la mines ne s'est pas initialisee correctement
testEstAbandonnePositif, estAbandonne de la mine n'est pas effectif alors qu'il le devrait
testEstLamineePositif, La mine n'est pas Laminee alors qu'elle le devrait
testSetAppartenancePositif, L'appartenance de la mine n'est pas la bonne
testSetAbandonCompteurPositif, le setAbandonCompteur de la mine ne s'est pas fait correctement

5 Conclusion

Avec plus de temps, nous aurions aimé implémenté de nouvelles extentions comme la commande groupé qui n'est en soit qu'une liste d'instruction sur un liste de villageois (les villageois sélectionnés), Un héros avec une gestion de l'expérience et d'amélioration selon sont niveau ou encore de nouvelles unités.

Ce projet a été très formateur et nous a permis d'avoir un aperçu de ce qu'étais que la programmation par contrats et un moteur de jeu.
Le fait d'être en binôme a été un avantage considérable, on a pu se séparer

les tâches, l'un plus testeur et l'autre plus développeur.
On a pensé qu'un développement en largeur avec plusieurs itérations, pour ce genre de projet, se serait avéré efficace s'il avait duré sur une plus grande période.
A travers ce projet, nous avons découvert l'importance des tests et du pattern Require/Provide qui facilite grandement l'implémentation.
Une petites critique serait par contre le fait que ce pattern est un peu dirigiste malgré le fait qu'elle offre un gain de temps considérable derrière.
On peut aussi noter que les services implémentés sont réutilisables, on pourra donc peut être s'en servir dans un futur projet !