Projet CPS (MI047) - River City Ransom

Kevin Coquart Quentin Bunel

14 avril 2014

Introduction

L'objectif du projet est de réaliser la spécification du jeu River City Ransom, puis d'implémenter celle-ci (décorateurs, contracts, tests) selon la méthodologie du cours. Nous avons découpé le travail en 11 services.

Les spécifications ont été redigées dans le format org. C'est un format de l'éditeur Emacs qui permet de générer facilement des fichiers html ou pdf (voir plus bas).

Solutions

Un des premiers problèmes qui s'est présenté à nous est qu'il fallait différencier les objets équipables des objets marchands. En effet, les objets marchands ont un prix et peuvent se vendre, alors que les objets équipables doivent contenir un bonus, c'est-à-dire un nombre de points de vie à retirer au personnage cible d'un éventuel jet. Notre service Objet déclare uniquement un observateur nom() de type String et possède deux sous-services : ObjetEquipable et ObjetMarchand.

Un autre facteur vient cependant modifier la solution : un personnage peut être porté par un autre ! Personnage raffinerait donc ObjetEquipable ? et donc Objet ? La solution que nous avons choisi a été la création d'un service Chose qui décrit ce qui est portable et le bonus apporté. Personnage et ObjetEquipable raffinent tous les deux Chose. Ainsi, quand on a besoin de manipuler des objets, comme dans les blocs du terrain, on est sûr de ne pas confondre avec les personnages.

Pour avoir accès aux personnages, dans GestionCombat, nous avons fait le choix d'avoir une HashMap dont les clés sont les noms des personnages, et les valeurs les instances de Personnage (ou Gangster). On peut par exemple appeler mPerso.get("Alex") pour récupérer Alex. De plus, la méthode gerer() prend en paramètre une HashMap<String, COMMANDE> qui associe à chaque personnage la commande qu'il doit réaliser dans le pas de jeu.

Annexes

Vous trouverez ci-joint :

- Dans /spec, les specifications au format org et le Makefile pour générer les formats html et pdf (voir README.txt)

– Le code des services, décorateurs, contracts, tests et implémentation dans /src

Conclusion