Projet – River City Ransom^{TM} !

Binh-Minh Bui-Xuan

Mars-Mai 2014

Une fois n'est pas coutume, le projet portera sur un différent jeu que celui apparu lors de l'examen réparti 1.

Introduction

Commercialisé en parallèle avec la série à succès des `Street Fighter`^{TM} de Capcom, le très vénérable (et ôoooh combien vénéré !) jeu de `River City Ransom` est développé par Technos en 1989 sous la forme d'un *beat'em up* aux accents de jeu de rôle. Ce subtil mélange a su conquérir des millions de jeunes coeurs (parfois même sans bouton) en offrant une évolution des personnages du jeu en terme de coups de poing, du *body-building*, et d'illuminations merveilleuses dans la sagesse de baston comme par exemple la maîtrise des subtilités nommées *Acro Circus* et *Dragon Feet*, ou encore l'acquisition du très cher (et parfois mortel) tonneau de poubelle métallique.

Deux jeunes gens, aux doux noms de Ryan et d'Alex, tentent le tout pour le tout et s'en vont dans la rue affronter de terribles gangsters aux noms exotiques comme par exemple les *Generic Dudes*, les *Home Boys*, et les *Squids*. Le duo a pour seul but apparent d'investir les lieux du *River City Highschool*, où le très très tchrés méchant Slick tient en otage Cyndi, jusqu'à nouvel ordre amoureuse de Ryan. Il n'est pas tellement précisé si l'audacieux méfait de Slick a pour but (probable) de corrompre la jeune personne avec de l'argent et du pouvoir que ce méchant a acquis au fil de longues, laborieuses années... Cependant, l'histoire raconte que tout ça, tout ça, et toussa se passe dans le riche décor de *River City*, rempli de tuyaux métalliques, chaînes de vélo – ou moto, caissons de légume – ou déchet, etc, etc, etc. Tout ceci semble très efficace pour tapper sur des gens (mais pas tellement pour autre chose, apparemment).

Cahier total des charges (partiel + projet)

Nous souhaitons formaliser, à l'aide du langage de spécification de services, les notions suivantes :

- les **personnages jouables** Ryan et Alex que l'on peut déplacer dans les quatre directions (gauche, droite, haut, et bas), faire sauter sur place ou dans l'un des quatre directions, faire ramaser un objet ou une personne, faire jeter l'objet ou la personne en question, et faire effectuer une frappe vers l'avant de là où ils se trouvent, éventuellement avec l'objet ou la personne qu'ils portent,
- les **gangsters** aux capacités plus ou moins limitées,
- les **blocs** graphiques qui composent le **terrain** de jeu :
  - les **blocs vides** dans lesquels on se déplace librement,
  - les **blocs-fossé** où le fait de s'y trouver est fatal, surtout pour celui qui s'y trouve,
  - les blocs contenant des **objets** dans lesquels on peut toujours se déplacer librement,
- les **objets** laissés à tout vent ; ils se rangent en deux catégories :
  - les **objets réutilisables** que non seulement tout le monde peut ramasser mais aussi très utiles pour tapper sur les uns les autres,
  - les **objets avec une valeur marchande** que l'on peut mettre dans sa poche, mais aussi les dépenser en magasin en échange de diverses choses merveilleuses.

Enoncé du projet

L'objectif du projet est multiple :

1. donner une spécification complète du modèle de jeu dans le langage de spécification vu en cours,
2. réaliser à partir de la spécification une implémentation contractualisée, service et contrat sous forme de tests embarqués, en utilisant le langage de votre choix comme par exemple le Java vu en TME,
3. définir à partir de la spécification des objectifs de test pour assurer les couvertures logiques, les paires de transitions, et une suite de scénarios utilisateurs,
4. réaliser à partir des objectifs de test une implémentation des cas de tests spécifiés, en Junit par exemple,
5. réaliser deux versions complètes du jeu, dont une totalement "buggée". Ici, peu importe si les deux réalisations sont avec interface graphique ou non. En revanche, elles doivent être en dehors de tout modèle du jeu : pas de spécification, pas de contrats, pas de tests MBT,
6. recueillir les messages d'erreur lors de la confrontation de la version buggée avec les tests spécifiés.

Rendu du projet

Si on décide d'utiliser Java : une archive jar contenant
- un fichier au format PDF comprenant la spécification, les objectifs de tests, ainsi que les messages d'erreur de la version buggée du jeu,
- un sous-répertoire src comprenant le code Java du projet,
- un fichier de construction build.xml avec :
  - une cible compile pour la compilation du projet,
  - une cible run pour lancer le jeu,
  - une cible test pour lancer les tests sans contrat,
  - une cible ctest pour lancer les tests avec contrat.

Sinon, une archive tar contenant le PDF, le src, ainsi qu'un fichier Makefile réalisant les cibles similaires.

Modalités d'évaluation

La note du projet reposera en priorité sur les critères suivants :
- la qualité de la spécification,
- la qualité des objectifs et des cas de tests,
- l'adéquation entre les spécifications et les contrats impléméntés,
- l'adéquation entre les objectifs de tests et les tests implémentés.

Description des services

On donne ici la liste minimum des spécifications attendues. On peut apporter des observations supplémentaires à un service particulier, ainsi que des services complets en supplément de la liste de base. D'ailleurs, la section suivante décrit quelques idées dans cette perspective.

Attention ! Il est souvent plus intéressant de peaufiner la qualité de la spécification des services de base, plutôt que d'essayer d'avoir des services supplémentaires mal spécifiés !

Personnage
Le service Personnage représente les différents personnages du jeu. Ces passionnés de la bastonnade sont distingués non seulement par leur `nom` et leurs dimensions – plus précisément leur `largeur`, `hauteur`, et `profondeur` – mais aussi par leur `force`, les précieux `points_de_vie` leur restant, et la `somme_d_argent` qu'ils ont amassée, à force de frappe ou à l'huile de coude, peu nous importe. On dispose également d'un observateur donnant leur état de service, c.à.d. si le personnage `est_vaincu` ou si cela se fait encore attendre. Par ailleurs, les personnages peuvent ramasser divers objets au cours du jeu, voire même ramasser un autre personnage (!). Cependant, ils peuvent porter au plus une telle chose à la fois. On dispose pour cela un observateur pour tester si le personnage `est_équipé` de quelque chose, et un observateur pour savoir quelle est `la_chose_équipée`. Ce service offre les opérations de `dépôt/retrait` de points de vie, de `dépôt/retrait_de_l_argent`, et de `ramasser/jeter` une "chose".

Gangster
Le service Gangster supervise les différents types de vilains se joignant joyeusement au jeu de mains et de pieds. Cependant, il est très ressemblant au service Personnage. En réalité, ce service peut très bien être un raffinement du service Personnage, avec des attributs bien choisis lors de sa construction.

Terrain
Le service Terrain représente les nombreuses rues de *River City*, toutes ravagées par la guerre des gangs. Le terrain de jeu est représenté par trois entités, largeur, hauteur, et profondeur, fixées lors de la construction. Le terrain se compose de blocs graphiques, aux coordonnées spécifiques. On pourra modifier un bloc à une coordonnée spécifiée en faisant attention de bien respecter les invariants du jeu.

Le service Bloc permet de modéliser la base commune des objets éparpillés partout dans cette belle ville. Chaque bloc doit permettre notamment des observateurs donnant son type (ex : VIDE, FOSSÉ) et l'éventuel trésor qui y est caché (ex : RIEN, UNDOLLAR, CINQUANTECENTIMES, CHAÎNEDEVÉLO, POUBELLEMÉTALLIQUE, etc). Un bloc de type FOSSÉ représente un fossé mortel : il n'est pas bon de s'y trouver, surtout après le résultat d'un jet malveillant d'un adversaire. Un bloc de type VIDE est vide, on peut y entrer comme dans un moulin (et en sortir !), et il peut y avoir des objets en libre service.

Le service Objet représente les objets utilisables dans le jeu. Ils sont très variés, non seulement par leur nom, mais aussi par l'utilisation que l'on peut en faire. Ainsi, ces objets se rangent en deux catégories principales : les objets équipables et les objets à valeur marchande. Les objets équipables donnent un bonus à la force de frappe de la personne s'équipant de l'objet en question. Les objets à valeur marchande ont, quant à eux, une valeur marchande (!) plus communément appelée de l'argent.

Moteur du jeu
Le service MoteurJeu modélisera le jeu lui-même. Il aura comme principale opération le calcul d'un pas de jeu. Lors de ce calcul, on passera en paramètre une commande éventuelle de Ryan et/ou d'Alex (déplacement, saut, jet, frappe, etc). On mettra à jour les informations concernant ces jeunes gens via un service GestionCombat à spécifier plus tard. Concernant MoteurJeu, on prendra soin de détecter la fin du jeu avec un observateur adéquat : partie gagnée ou perdue, voire nulle, 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. Slick est vaincu alors que Ryan est toujours debout : notre héros a gagné la partie ! 2. Slick et Ryan sont vaincus mais Alex est encore à (a)battre : Ryan sort vainqueur sous les applaudissements et dans les bras de Cyndi même s'il n'avait rien fait ! 3. Ryan et Alex sont tous deux vaincus mais pas Slick : le méchant est victorieux et vit avec sa captive dans la richesse jusqu'à la fin de leurs jours ! 4. par une intervention miraculeuse des esprits supérieurs, ils sont tous les trois KO en même temps (!) la partie est finie, nullement, même s'ils restent beaux dans notre mémoire !

Gestion du combat
Le service GestionCombat supervise un terrain de jeu sur lequel se trouvent les deux dénommés "Ryan" et "Alex", ainsi qu'un certain nombre de gangsters aux noms poétiques, dont un s'appellant "Slick". La composition du terrain comprend un certain nombre de blocs-fossé, de blocs vides, et de blocs contenant des objets précieux, initialisés de manière parfaitement aléatoire. Ryan et Alex sont initialement positionnés sur le terrain à quelques pixels de la bordure gauche du terrain, mais pas dans un fossé. Slick est initialement positionné à quelques pixels de la bordure droite du terrain, sur un bloc vide. Les positions des autres vilains sont tirées au hasard sur des blocs vides restant. Les attributs de tous ces jeunes gens, ainsi que les dimensions du terrain, sont initialisés par des chiffres conventionnels comme par exemple 16, 51, 1001, etc.

La seule opération de ce service est la gestion du combat suite à des commandes joueur : on mettra à jour les positions des personnages, gangsters, et objets divers suivant ces commandes, ainsi que les chocs éventuels résultants de ces actions intrépides. On supposera que :

- le fait de frapper gèle le personnage qui frappe ; le fait d'être touché par une frappe gèle celui touché par la frappe pendant quelques pas de jeu fixé à l'avance (ex : 3 pas de jeu) ; le fait d'être ramassé gèle le personnage qui est ramassé tant qu'il reste dans cet état ; il n'y a pas d'autre façon de geler un personnage (cependant, on peut être gelé plusieurs fois de suite) ;
- un personnage gelé ne peut entreprendre d'action pendant le pas de jeu suivant ce gel ;
- un personnage est touché s'il y a collision entre le personnage et un adversaire au moment précédant le pas de jeu et que l'adversaire réussit à entreprendre l'action de frappe pendant ce pas de jeu ;
- il y a collision entre personnages si et seulement s'ils sont suffisamment proches (attention : nous nous plaçons en 3D à la différence du 1D apparu lors de l'examen 1) ;
- un personnage touché est renvoyé d'un certain nombre de pixels en arrière, dans la limite des pixels disponibles ; un personnage touché perd l'objet (ou la personne) éventuel dont il est équipé :

l'objet (ou la personne) tombe alors par terre ; un personnage touché perd un nombre de points de vie égal à la force de son adversaire, plus le bonus de force provenant d'un objet dans le cas où l'adversaire est équipé d'un objet (tel que la poubelle métallique) ;
— un gangster vaincu disparaît du terrain de jeu comme par miracle, il sera remplacé par un objet d'une certaine valeur marchande fixée à l'avance.

Suppléments

Les questions ci-dessous sont données à titre de points bonus pour le projet (l'attribution des points est variable pour chaque supplément). Les questions peuvent être traitées en partie, et dans n'importe quel ordre. Spécifier, ou apporter les modifications nécessaires à votre spécification afin de permettre les implémentations suivants.

— Les *sauts* et les *jets* : prendre en compte les situations relatives à l'acrobatie des protagonistes, à savoir que
  — le fait d'être en état de vol, resp. de saut, gèle le personnage se trouvant dans cet état ;
  — le fait de jeter un objet, ou de jeter un personnage, place ce dernier en état de vol plannant pendant quelque pas de jeu fixé à l'avance (ex : 7 pas de jeu) ;
  — un objet ou un personnage en état de vol se déplace horizontallement de quelque pixels dans la direction du vol ;
  — le fait de sauter place le personnage en état de saut pendant quelque pas de jeu fixé à l'avance (ex : 5 pas de jeu) ;
  — un personnage en état de saut se déplace comme se doit un saut (ex : une parabole) ;
  — un personnage est touché s'il y a collision entre le personnage et un objet volant, ou un personnage volant, au moment précédant le pas de jeu ;
  — il y a collision entre personnage et objet si et seulement s'ils sont suffisamment proches.
— Les Magasins : des havres de paix où il est bonne manière d'être gelé, en ce qui concerne la gestion de combat. On peut y dépenser son argent en échange de service merveilleux comme de la cuisine *fastfood* ou du sauna revitalisant (ex : ajout de points de vie).
— La technique du AcroCircus : l'apprentissage de cette technique, via la lecture d'un livre très coûteux acheté en magasin (ou ramassé dans un rarissime bloc vide), permet de se dégeler pendant un saut, mais uniquement pour frapper.
— La technique du DragonFeet : l'apprentissage de cette technique, via la lecture d'un livre approprié, permet de frapper trois fois plus vite. Attention : le pas de jeu n'est pas divisible. Afin de prendre en compte cette technique, il serait plus simple, au contraire, de ralentir des frappes de base sans le DragonFeet.