Projet – Bomberman^{TM} !

Binh-Minh Bui-Xuan

Mars-Mai 2013

Durée : 2 heures

Document autorisé : Notes de cours

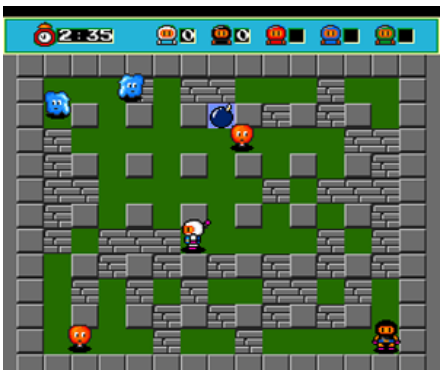
Introduction

Ce sujet de projet fait partie d'un devoir global (partiel+projet) consistant à formaliser, à l'aide du langage de spécification de services vu en cours, un modèle légèrement simplifié du mode multi-joueur du jeu `Bomberman`^{TM} version `DOS`, développé par Hudson Soft en 1992 sous le nom de `Dyna Blaster`.

Dans cet épisode de la série, le héros, l'adorable et certainement très adoré *Cheerful White* dans sa quête pour délivrer la fille du Dr. Mimori, son inventeur, décide d'attaquer à coups de dynamite l'étage le plus haut d'un bâtiment très "*hightech*". Le kidnappeur, riche propriétaire desdits locaux *hightech*, n'est rien d'autre que le magnifique autre invention du même Dr. Mimori, le fameux *Cool Black*. Cet étage du bâtiment est également peuplé de misérables vilains fauchés, qui ne s'intéressent qu'à l'argent et la capture du héros ou du kidnappeur, peu leur importe : ce sera pour eux une promotion chez le riche victorieux, ou l'argent du riche capturé le cas échéant.

Cahier total des charges (partiel + projet)

- le **héros** (bonnet blanc) que l'on peut soit déplacer dans les quatre directions (gauche/droite et haut/bas) soit faire fabriquer une bombe à l'endroit où il se trouve,
- le **kidnappeur** (bonnet noir) que l'on peut faire faire les mêmes opérations,
- les **vilains** (bleus et oranges) explosibles à la dynamite,
- les **blocs** graphiques qui composent le **terrain** de jeu :
  - les **blocs vides** (fond vert) dans lesquels le héros, le kidnappeur, et les vilains peuvent se déplacer librement,
  - les **murs en brique** (gris non-uni) explosibles à la dynamite : le mur en question devient alors soit un bloc vide, soit un objet consommable appelé *power-up*,
  - les **murs métalliques** (gris uni) infranchissables, immobiles et indestructibles,
  - les **power-ups** que le héros et le kidnappeur peuvent utiliser pour diverses conséquences merveilleuses,
- les **bombes** que le héros et le kidnappeur peuvent créer rien qu'avec leurs mains pleines de doigts(!).



Enoncé du projet

L'objectif du projet est multiple :

1. donner une spécification complète du modèle de jeu dans le langage de spécification semi-formel 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 cas de tests spécifiés.

Rendu du projet

Si on décide d'utiliser Java : une archive jar contenant
– un fichier au format PDF comprennant la spécification, les objectifs de tests, ainsi que les messages d'erreur de la version buggée du jeu,
– un sous-répertoire src/
– 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 sous-répertoire 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 implémentés,
– l'adéquation entre les objectifs de tests et les tests implémentés.

Description des services

On donne ici la liste minimum des services attendus. La section suivante décrit des services supplémentaires qu'on pourrait penser à spécifier.

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

Terrain de jeu

Le service Terrain représente l'étage le plus haut du complexe *hightech* que possède maintenant *Cool Black*. Ce niveau richement décoré est composé d'un ensemble de blocs disposés selon un rectangle. Chaque coordonnée (i, j) dans ce rectangle correspond à un bloc. Sur la figure du jeu ci-dessus, le bloc aux coordonnées $(1, 1)$ est le mur métalique en haut à gauche, juste au dessus d'un vilain fantôme bleu en déplacement de $(2, 3)$ vers $(2, 2)$. Le bloc en dessous, en $(2, 4)$, est un mur en brique et le bloc en $(3, 3)$ à droite du fantôme est un mur métalique. L'adorable héros est caché en $(7, 8)$ entre deux murs métaliques, $(7, 7)$ et $(7, 9)$, et un mur en brique en $(6, 8)$. Il peut par exemple y larguer une bombe et aller se cacher, jouer de la flûte, ou faire une partie de solitaire en $(8, 7)$ avant que sa bombe explose. Le kidnappeur, pour l'instant, semble content de son petit couloir en forme de L, et reste tranquille à la case $(14, 12)$.

La taille du terrain sera fixée lors de la construction, mais toujours des chiffres *impairs* (13 lignes et 15 colonnes dans le jeu original). On pourra modifier un bloc à une coordonnée spécifiée en faisant attention de bien respecter les invariants du jeu. Cependant, il existe une règle. La disposition des murs métalliques est une question de tradition, et elle doit toujours être de la façon suivante : 1. les blocs entourant le terrain de jeu sont des murs métalliques, 2. si i et j sont tous les deux impairs alors (i, j) est un mur métallique, 3. il n'y a pas de murs métalliques autres que ceux-ci.

Le service Bloc permet de modéliser la base commune des objets propres au bâtiment du beau kidnappeur. Chaque bloc doit permettre notamment des observateurs donnant son type (VIDE, MURBRIQUE, MURMETAL) et l'éventuel trésor *power-up* qui y est caché (RIEN, BOMBUP, FIREUP, WALLPASS, BOMBPASS, et FIRESUIT). Un bloc de type MURMETAL représente un mur métallique : un pilier indestructible, immobile, infranchissable, et ne contenant pas de *power-up*. Un bloc de type MURBRIQUE représente un mur en brique : un tas de brique destructible, immobile, parfois franchissable, et repère à *power-ups* aux valeurs inestimables. Un bloc de type VIDE est vide, on peut y entrer comme dans un moulin (et en sortir !), et il peut y avoir des *power-ups* laissés à tout vent.

L'utilisation des *power-ups* par le héros ou le kidnappeur leur donne des pouvoirs suivants :
- BOMBUP : par défaut, le héros/kidnappeur ne peut déposer qu'une bombe à la fois ; ce *power-up* augmente d'une unité le nombre de bombes que peut poser simultanément le bomberman.
- FIREUP : par défaut, le héros/kidnappeur a une force vitale de 3 ; ce *power-up* augmente cette force de 2 unités, pour un maximum de 11.
- WALLPASS : par défaut, le héros/kidnappeur ne peut passer par des murs ; ce *power-up* leur permet de "passer au dessus" des murs en brique. Attention : même si le héros/kidnappeur passe au dessus d'un mur en brique contenant un *power-up*, il ne peut l'utiliser. La seule façon d'utiliser un *power-up* est de le ramasser lors d'un passage dans un bloc vide.
- BOMBPASS : par défaut, le héros/kidnappeur ne peut passer par des bombes (sauf s'il vient de créer la bombe) ; ce *power-up* leur permet de passer par des bombes.
- FIRESUIT : le héros/kidnappeur est à l'abri des explosions pendant 100 pas de jeu !

Bombe

Le service Bombe permet la gestion des bombes présentes pendant le combat. Les bombes sont des objets spécifiques aux inventions hautement technologiques que sont des *Bombermen*, dont *Cheerful White* et *Cool Black* sont des prototypes. Selon la force vitale de chacun, la bombe ainsi larguée aura une amplitude spécifique (entre 3 et 11 généralement). Sauf si on décide de donner les spécifications concernant le *power-up* appelé DETONATOR (voir section suivante sur les suppléments), toute bombe est à retardement et explose après un certain nombre donné de pas de jeu : compte à rebours initialisé par une valeur fixée à l'avance (10 par exemple).

Vilain

Le service Vilain supervise les différents types de vilains se joignant joyeusement à la partie de pétard. Ce service doit permettre notamment des observateurs donnant son type et l'éventuel pouvoir surnaturel qu'il possède. Les deux types de vilains par défaut sont le BALLONORANGE aux capacités limitées et le terrifiant FANTOMEBLEU capable de traverser les murs en brique (en revanche, l'explosion d'un mur en brique entraine la destruction du vilain s'y trouvant). Tout vilain est hyperactif et avide : il se déplace à tout moment et, qu'il attrape le héros ou le kidnappeur, la partie est finie pour ce dernier.

Moteur du jeu

Le service MoteurJeu modélisera le jeu lui-même. Ce service devrait permettre des observateurs donnant le terrain de jeu, la liste des bombes présentes, la liste des vilains en vie, les coordonnées du héros/kidnappeur, ainsi que leur état de santé. La force vitale par défaut des deux antagonistes est de 3 unités, leurs positions sont par défaut sur des blocs vides diamètralement opposés, le nombre maximum de bombes que chacun peut déposer à tout moment est initialisé à 1, et les deux bombermen sont par défaut en parfaite santé. Les positions des vilains sont par défaut tirées au hasard, mais uniquement sur des blocs vides. Le service MoteurJeu aura comme principale opération le calcul d'un pas de jeu. Lors de ce calcul, on pourra passer en paramètre un déplacement éventuel du héros/kidnappeur dans une direction gauche, droite, haut ou bas, ou encore l'action consistant à fabriquer une bombe là où il se trouve. On mettra à jour le terrain et la liste des bombes, ainsi que les positions des héros/kidnappeur/vilains. Puis, on doit aussi superviser les explosions éventuelles des bombes présentes dans la liste. On suppose que l'explosion d'une bombe détruit simultanément tout ce qui se trouve dans la même ligne ou le même colonne de celle-ci, sauf ceux qui se trouvent strictement plus loin que l'amplitude de la bombe, ou qui se trouvent strictement derrière un mur métallique, ou un mur en brique, ou un bloc vide contenant un *power-up*. L'explosion d'un mur en brique remplace celui-ci par un bloc vide, ainsi révèle l'éventuel trésor qui y est caché. L'explosion d'un bloc vide contenant un trésor détruit le trésor, malheureusement. 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, voir 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. le kidnappeur est explosé en mille et un morceaux : le héros a gagné la partie !
2. le héros se trouve carbonisé : la partie est perdue pour lui, même s'il reste beau dans notre mémoire !
3. le kidnappeur se fait assassiner par des vilains : le héros sort vainqueur sous les applaudissements même s'il n'avait rien fait !
4. le héros se fait capturer par des vilains : le kidnappeur est victorieux et vit avec sa captive dans la richesse jusqu'à la fin de leurs jours !
5. les deux antagonistes survivent même après que le nombre maximal de pas de jeu est atteint : la partie est nulle, et les spectateurs se contenteront d'attendre le prochain épisode !

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.

- Le vilain ChienRouge : il se déplace deux fois plus vite que la normale. Attention : il peut attraper le héros/kidnappeur, par le bonnet, même si litéralement le ChienRouge "passe au dessus" de ce dernier lors d'un déplacement fulgurant.
- L'explosion d'un bloc vide contenant un *power-up* non seulement détruit ce dernier, mais de plus le *power-up* en question se divise en 4 vilains ChienRouge en s'explosant !
- Le *power-up* SpeedUp : le héros/kidnappeur se déplace deux fois plus vite (culmullatif). A l'instar du ChienRouge, le héros/kidnappeur hyperactivé peut rammaser les *power-ups* même s'il "passe au dessus" de ceux-ci. Bien entendu, il ne peut pas voler par dessus d'un mur métallique, ni un mur en brique ou une bombe sans l'utilisation du *power-up* approprié. Le fait de passer au dessus d'un vilain est fatal, pour celui qui est au dessus du vilain.
- Le *power-up* Detonator : permet au héros/kidnappeur, lors d'un calcul d'un pas de jeu, de choisir l'action qui consiste à faire exploser la bombe la plus anciennement posée. Bien entendu, seules les bombes posées après l'utilisation du Detonator auront cet effet.
- Le *power-up* BombKick : permet au héros/kidnappeur, lors d'un calcul d'un pas de jeu, de choisir l'action qui consiste à donner un coup de pied (gauche/droite/haut/bas) qui déplace l'éventuelle bombe présente dans le bloc en question de 3 blocs maximum : la présence d'un mur (brique ou métallique) bloque la progression de la bombe en mouvement.
- Le *power-up* Push : permet au héros/kidnappeur, lors d'un calcul d'un pas de jeu, de choisir l'action qui consiste à traîtrement pousser (gauche/droite/haut/bas) l'éventuel autre antagoniste se trouvant dans le bloc en question de 1 bloc maximum, mais uniquement si le bloc d'arrivé est un bloc vide. Ce mouvement involontaire de la victime doit être traîté avant la gestion des explosions des bombes dans le calcul du pas de jeu (pour être intéressant, d'un point de vue du traître...).