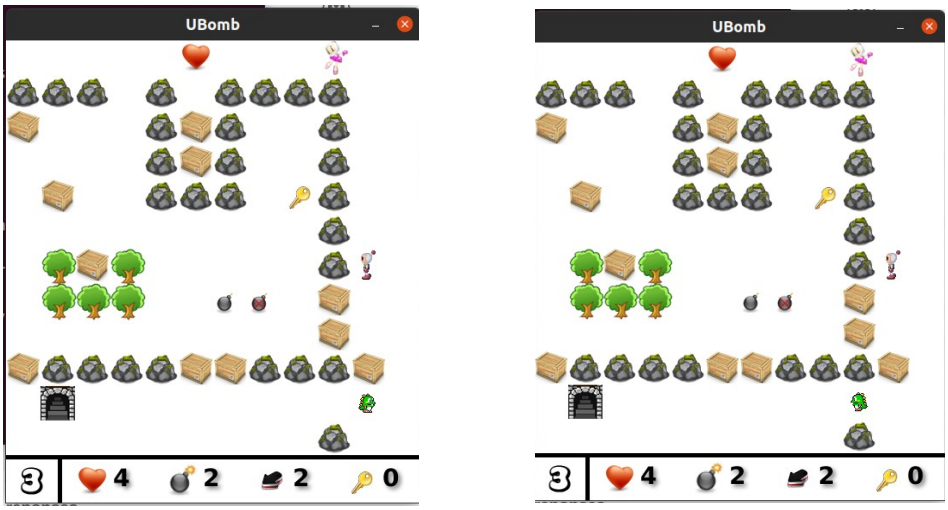
Rapport Projet POO 2020

Ronan Thoraval / Erell Gachon (MI501A1)

Tout d'abord, nous n'avons pas rencontré de problème particulier concernant les règles imposées par le jeu : toutes les options doivent marcher correctement.

Concernant la catégorie *« Pour aller plus loin... »*, plus le niveau est élevé, plus les monstres se déplacent rapidement. Cette modification de vitesse ne marche pas seulement pour un jeu à trois niveaux, elle dépend du nombre de niveaux et peut être adaptée à différents jeux : plus il y a de niveaux, moins la différence de vitesse sera importante entre deux niveaux consécutifs.
Les monstres sont également de plus en plus intelligents lorsque les niveaux augmentent. Au tout début, ils se déplacent aléatoirement. Petit à petit, ils se déplacent de moins en moins aléatoirement, et de plus en plus vers le joueur. Au dernier niveau, ils ne se déplacent que vers le joueur. Par exemple, si le joueur se trouve au Nord-Ouest par rapport au monstre, ce dernier prendra une direction aléatoire entre le Nord et l'Ouest.
Cette intelligence comporte néanmoins des limites. Tout d'abord, le monstre n'a pas l'idée de contourner les obstacles : si il est séparé du joueur par un ou plusieurs objets, et qu'il ne peut pas aller dans sa direction, il choisira une direction aléatoire. Cela implique qu'il existe donc certains cas de figures où il répétera indéfiniment une séquence de mouvements. Voici un exemple :



Le monstre varie entre ces deux positions.

Nous avons réalisé certains choix là où nous avions des libertés.
Une bombe explose en deux temps. Tout d'abord, en son centre, là où elle a été posée puis elle se propage suivant sa portée avec une seconde d'intervalle. C'est aussi valable pour les dommages causés.
Les bombes ont été implémentées en deux classes : la classe Bombe et la classe Explosion. Tant que l'objet n'a pas explosé, c'est une bombe, puis le propagation de celle-ci est faite avec des explosions, en fonction de la portée. Nous trouvons que cette méthode est plus simple car nous ne voulions pas considérer les explosions propagées comme des bombes.
Enfin, un monstre ne bouge plus quand le joueur n'est plus dans son niveau.

Concernant la programmation en elle-même, il nous a fallu un certain temps d'adaptation pour comprendre vraiment l'intérêt de la programmation orientée objet. Nous avons par exemple au début fait une grand partie de notre code en utilisant des *instanceof*, au lieu de créer des méthodes

directement dans les différentes classes. Mais une fois ce changement appliqué, nous nous sommes vite rendus compte des avantages que cette méthode engendre, comme une factorisation énorme du code (nous sommes passés d'une trentaine de lignes de code à une demi-dizaine dans le code de *update* de *player*), mais aussi une meilleure lecture et une compartimentation du code.

Nous avons encore quelques hésitations sur ce principe de codage. Par exemple dans le code de *canOpenDoor* dans *Player*, qui dit si la porte peut être ouverte, nous ne savons pas si c'est vraiment nécessaire/judicieux de changer *instanceof Door* pour le remplacer par une méthode implémentée dans *Decor* et spécialisée dans *Door*. Faire cela rajouterait des lignes de code, et nous obligerait à tester si la case n'est pas vide avant d'appeler la fonction, ce qui au final rajouterait du code. Nous avons donc fait le choix de laisser *instanceof*.

En espérant que notre code soit agréable à lire, et le jeu agréable à jouer,

Ronan Thoraval et Erell Gachon
MI 501 A1.