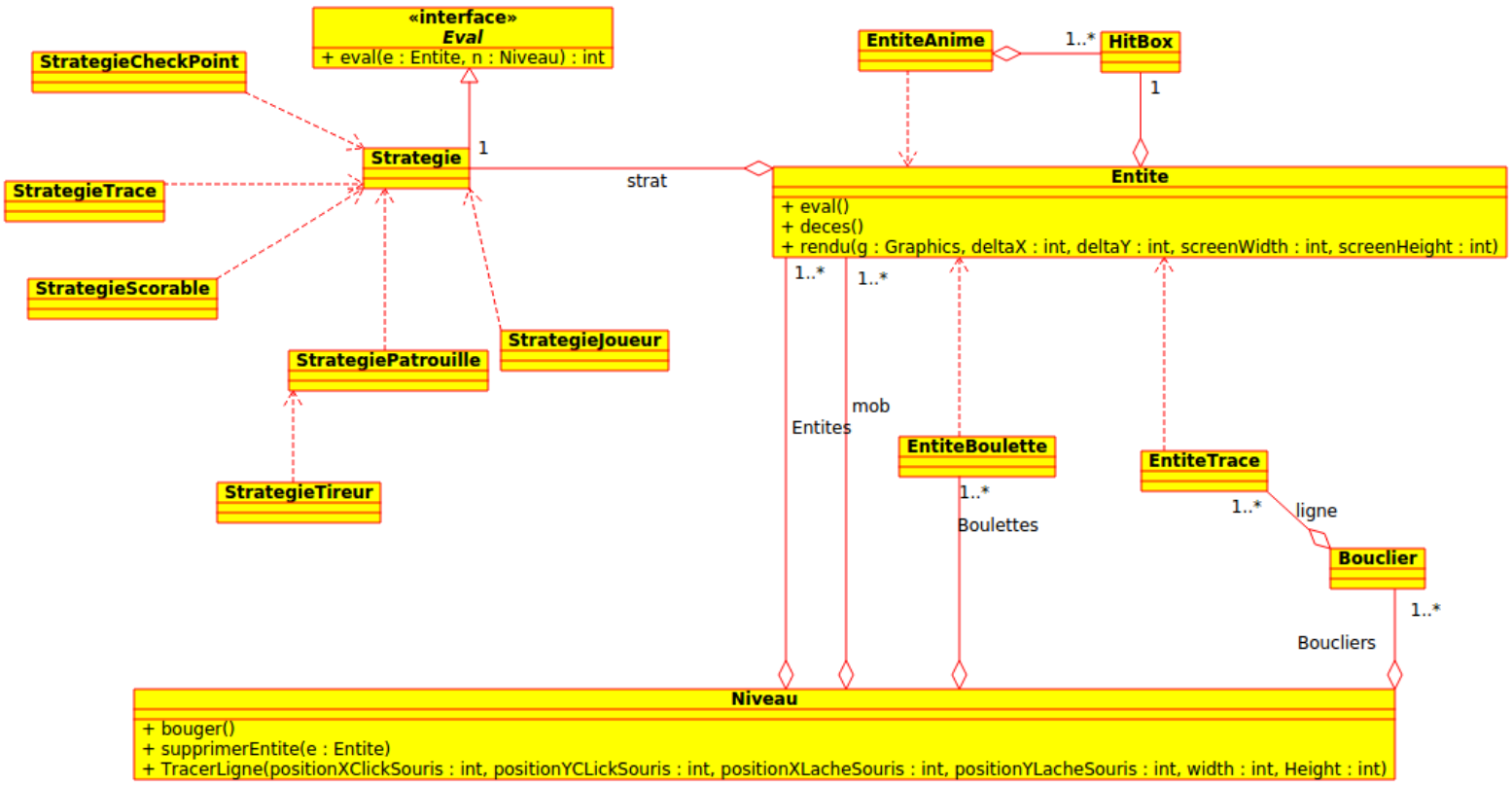
Mini-rapport de Projet de Programmation de 2nd semestre de L3

GRACIANNE Olivier
ROY Bastien
THEVRET Simon

Nous devions faire un jeu ressemblant à un Super Mario. Tous les aspects du sujet ont été respectés. Ce rapport contiendra un résumé de l'architecture du code, de l'organisation du travail au sein de notre groupe et du contenu du jeu lui-même.



L'architecture du jeu s'organise autour de classes ayant une grande importance, à savoir Stratégie, Entite et Niveau. Le jeu est dans une JFrame, contenant un JPanel qui lui-même contient une instance de Niveau. Cette instance contient, dit grossièrement, tout le jeu. Ses attributs « sont » le contenu du jeu :

- mob est la liste qui contient les éléments du jeu avec lesquels le joueur peut interagir, mis à part les boulettes et les boucliers ;
- entite est le tableau contenant le décor du jeu ;
- les autres attributs ont un nom révélateur de leur contenance et de leur rôle.

Les différentes méthodes de cette classe se chargent du fonctionnement du jeu. Là encore, le nom de chacune est explicite sur son rôle : bouger() fait se déplacer chaque élément qui doit se déplacer selon son mouvement, etc.

Entite, ainsi que les classes qui en héritent, sont le « corps » de chaque éléments que l'on voit dans le jeu : chaque bloc de décor est une Entite, le joueur est une Entite, les boulettes sont des EntiteBoulettes…

Chaque Entite a une stratégie qui définit son comportement. Au cours du développement, chaque fois que nous avions besoin d'une spécialisation de corps ou de comportement pour un nouvel élément, nous avons créé une nouvelle entité ou une nouvelle stratégie. De fait, nous avons a chaque fois réfléchi à si nous pouvions ré-utiliser une spécialisation déjà existante ou pas, ce qui justifie l'existence d'une EntiteBoulette, mais pas celle d'une EntiteMob.

Strategie est la classe qui centralise les caractéristiques du comportement d'une Entite. C'est sa Stratégie qui définit le mouvement, le fait qu'elle soit sensible à la gravité ou non, la fréquence de tir, etc, d'une Entite.

D'autres classes peuvent être ici mentionnées, comme Sprite, qui s'occupe de contenir l'image qui représente une Entité, Physique, qui comme son nom le laisse deviner contient tous les calculs nécessaires à la physique du jeu, et Créateur, qui est la classe qui lit les fichiers texte à partir desquels sont créés les niveaux.

Concernant le jeu lui-même, un résumé du contenu de celui-ci sera plus parlant que de longues explications. Prenez en compte le fait que la totalité du design du jeu a été réalisé par les membres du groupes avant de lancer le programme pour la première fois.

Le joueur incarne un personnage habillé de vert. Il évolue dans un labyrinthe vertical ou horizontal, selon la construction du niveau. Chaque niveau est chargé depuis un fichier texte en passant par la moulinette de la classe Créateur, ce qui permet une grande souplesse d'édition de niveau. Le joueur doit se servir de ses capacités pour atteindre le drapeau de fin de niveau, qui ressemble toutefois plus à un lampadaire qu'à un drapeau. Il peut sauter et créer des boucliers. La difficulté vient des monstres qui peuplent le niveau. Certains d'entre eux ne font que s'y déplacer, d'autres tirent également des boulettes. Tous les monstres sont mortels au contact. Les boulettes le sont aussi, mais elles peuvent être renvoyées avec les boucliers que trace le joueur avec la souris. Elles deviennent alors mortelles pour les monstres également. Il existe trois types de boulettes : des bleus, des rouges et des violettes ; par la même, il existe deux types de bouclier : des bleus et des rouges. Conséquemment, les boucliers bleus renvoient les boulettes bleus et de même avec le rouge. Les boulettes violettes ne peuvent être renvoyées et doivent être esquivées. En traçant des boucliers vers le bas, on trace un bouclier d'un type, en traçant vers le haut, on trace un autre type de bouclier. Le traçage des boucliers est limité par une barre d'énergie. Il existe également des checkpoints et des pieux dans les niveaux.

mois	janvier				
semaine	1	2	3	4	
prévisionnelle		apprentissage		input, affichage, physique	
Oliver		apprentissage			
Bastien		apprentissage			
Simon		apprentissage			
mois	février				
semaine	1	2	3	4	
prévisionnelle	input, affichage, physique		hitBox, bouclier, musique		
Oliver	gestion des inputs			création des boucliers et boulette	
Bastien	gestion de l'affichage static		ajout de la musique		
Simon	gestion de la physique			création des stratégie	
mois	mars				
semaine	1	2	3	4	5
prévisionnelle	rebond, animation, lecture des fichiers				
Oliver	création des boucliers et boulette	gestion des rebond sur les boucliers			gestion des collisions entre entité
Bastien	création d'un menu		ajout des animations		sélecteur de niveau
Simon	création des sous hitBoxs		gestion de la lecture des fichiers		

Nous avons essayé d'organiser efficacement notre travail en planifiant au mieux les tâches que nous devions réaliser. Toutefois, notre manque d'expérience a quelque peu faussé cette planification. Cela dit, nous avons quand même réussi à tenir des délais corrects pour chaque tâche. S'en suit le diagramme de Gant correspondant.

La réalisation de ce projet a été très enrichissante. Si à son début nous n'étions pas spécialement convaincus par le fait de devoir intégralement s'auto-former, cela s'est révélé très satisfaisant. En effet, les premières difficultés étaient certes difficiles à surmonter, mais nous avons directement appliqué tout ce que nous avons appris. Pour la partie graphique, nous avons littéralement vu ce que étions en train d'apprendre et les progrès que nous faisions, ce qui a été assez gratifiant. Par ailleurs nous nous sommes un peu plus familiarisé avec les éléments inhérents à un projet de groupe, comme répartir le travail de manière efficace, gérer le versionnage, communiquer entre les membres du groupe, etc. Par ailleurs le fait que ce soit un jeu l'a rendu beaucoup plus intéressant pour les amateurs de jeu vidéo que nous sommes. En effet, nous avons découvert le code et le travail résidant derrière un jeu.

Si nous avons passé quelques temps au début du projet à regarder dans toutes les directions du travail à réaliser, nous avons rapidement séparé les tâches pour que chacun traite d'une partie importante du jeu.

Olivier s'est occupé des boucliers, des boulettes, du comportement en réaction aux collisions, des rebonds des boulettes et des inputs liés aux commandes. Une grande partie de son travail a donc reposé sur le choix des structures de données utilisées pour stocker les éléments concernés par sa partie. Par exemple, la liste contenant les boulettes. Son choix s'est porté sur une LinkedList car on ne pas peut savoir combien le niveau contiendra de boulette à un instant donné. La taille de cette structure devait donc être dynamique. Par ailleurs, à chaque tour de thread, on doit faire se déplacer chaque élément de la liste, potentiellement la supprimer et ajouter un nouveau. Il a donc beaucoup travaillé avec des ListIterators et dû pensé le code de manière à éviter les ConcurrentModificationException.

Bastien s’est chargé de l’affichage, des threads, de l’animation, du son et de la gestion des menus. Une partie importante de son travail a été de comprendre l’utilisation des layouts Managers pour la gestion des menus. Il s’est par exemple servit d’un card layout pour pouvoir avoir plusieurs JPanel dans la même frame, en évitant de devoir s’occuper en direct du positionnement des panels. Il est ainsi possible de sélectionner des niveaux, de jouer et de revenir à la sélection des niveaux. Concernant l’animation, on utilise un tableau de Sprite pour contenir ceux utilisés pour l’animation. On les rend ainsi faciles d’accès tout au long de l’animation. En outre, les images ne sont chargées qu’une fois grâce au SpriteStocker, se servant d’une image fonctionnant comme un tableau, au sens où l’image est découpée en carré de 25x25 et que le SpiteStocker s’occupe de récupérer la l’image contenue dans le carré aux coordonnées spécifiées.

Simon a fait la physique, les hitboxs et la lecture de fichier. De fait son travail a servi de base à une grande partie de celui des autres membres du groupes. Par exemple, une fois la physique définie et de fait la détection des collisions, Olivier n’a eu besoin que de spécifier le comportement de réaction à celle-ci de chaque Strategie. Pendant la réalisation de la physique, il a essayé plusieurs stratégies de calculs, en commençant par essayer de traiter des collisions uniquement avec un tableau, à la manière du stockage du décor. La version adoptée fonctionne de la sorte : à chaque itération du jeu, on déplace les Entite. Pour leur mouvement, on trace un rectangle comprenant l’entité et son mouvement, et on se sert de ce rectangle pour détecter les Entite avec lesquelles l’Entite en mouvement va entrer en contatct. On peut savoir si le mouvement est possible et on l’effectue pixel par pixel. On peut déterminer ainsi si on arrive en collision avec une des entités détectées plus haut. Cette méthode peut paraître lourde, mais elle est acceptable si on considère que tous les mouvements sont inférieurs à 20 pixels.