Projet de Java : Baba Is You
Licence d'informatique
–2020 - 2021–

Table des matières

Johnson VILAYVANH
Groupe 2

ARCHITECTURE

LES FONCTIONNALITÉS MANQUANTES

Avant débuter ce rapport, je tenais à préciser qu'une partie des fonctionnalités demandées ne sont pas intégrés au projet. Une partie y sera dédiée à la fin du rapport, mais si durant le rapport il semble manquer des fonctionnalités qui devraient se trouver dans la partie correspondante, c'est sans doute car elle n'a pas été implantée.

WORD

L'intégralité des mots du projet sera considéré comme étant un « Word ». De ce fait, les classes des noms (Noun), opérateur (Operator) et propriétés (Properties) vont toutes extend un Word. C
ela permettra ainsi de tous les manipuler de la même façon : pour les bouger et les dessiner sur le plateau.
Évidemment, puisqu'elles possèdent toutes de même des fonctionnalités différentes (un nom n'est pas une proriété), elles seront toutes complétées dans chacun de leur classe respective.
Un word possède donc un WordName qui nomme l'objet (Baba, You, Is), qui sera lui-même extends par chacun des objets (NounName, OperatorName, PropertyName), des coordonnées, et un BufferedImage wordImg, correspondant à l'image qui est affichée sur la fenêtre du jeu pour représenter un mot.

NOUN

Noun

En plus des fonctionnalités d'un Word, le nom va posséder des méthodes liées aux propriétés auxquelles il peut être associer. On peut ainsi savoir quelle propriété est associé à un nom, et agir en fonction de celles-ci.

NounList

Pour stocker l'intégralité des noms, on utilisera des nounlist.
On y retrouvera les méthodes pour ajouter un nom, de les récupérer et retirer selon les coordonnées données, les dessiner, et les remplacer (dans le cas où on a une règle de la forme « NOM IS NOM ».

NounMap

Enfin NounMap, Map qui permet de stocker des NounList lié à un NounName.
On a donc une unique classe dans laquelle on peut stocker l'intégralité des noms d'un niveau.
On peut savoir quel nom est associé à You, les dessier, remplacer un nom par un autre : en résumé, pas besoin de chercher quel nom existe sur un niveau, tout se trouve au même endroit, dans NounMap.

OPERATEUR

Etant donné qu'on possède seulement l'opérateur Is, on y retrouvera moins de méthodes par rapport aux noms et propriétés. Comme les propriétés, la seule « règle » d'un opérateur, c'est qu'il peut être bougé !

On retrouvera tout de même les méthodes qui permettent de déterminer si un opérateur forment une règle dans le niveau, en cherchant dans la Map où se trouvent l'intégralité des propriétés, qu'on expique dans la partie suivante.

PROPRIETES

Propriete

La propriété en elle-même non plus possède aucune méthode en plus d'un mot : on sait seulement qu'elle peut être bougé. Cependant, pour les stocker, on utilisera une map.

ProprieteMap

Ici, on retrouvera de la même façon que NounMap, toutes les propriétés d'un niveau rassembler au même endroit. On y retrouve également les opérateurs, et les « mots » qui sont des propriétés, ceux qui permettent de créer une phrase.

On pourra ainsi facilement savoir quelles propriétés sont formées puisqu'elles se trouvent toutes au même endroit. Elle permet également de nous donner quelle mot est actuellement « You ».

BOARD

Pour rassembler l'ensemble de ces objets, il y a également la classe Board : on y trouve ainsi une NounMap, une PropertyMap, et une WordList représentant les propriétés.

NIVEAUX

Chaque niveau est représenté par 1 fichier dans le dossier correspondant.

Au début de chaque fichier, on crée l'ensemble des mots avec des boucles « for » lorsqu'il est nécessaire.

J'ai fait le choix de ne pas créer de méthode pour créer les mots, les « for » sont ici plus pratiques de mon point de vue.

AJOUTS : VIRUS ET POISON

Comme indiqué dans le manuel utilisateur, j'ai crée le nom virus et sa propriété poison.

Son implémentation était simple : si le joueur va sur un mot associé à poison, il lui reste 10 mouvements : on crée donc un compteur dans Board dès lors qu'il a marché dessus, qui va se décrémenter jusqu'à atteindre 0.

FONCTIONNALITÉS

CHANGEMENTS DEPUIS LA SOUTENANCE

Les méthodes sont moins longues comme il m'avait été dit dans la soutenance, et on ne retrouve plus de code « inutile », comme le contrôle d'un get sur une map « != null ».
J'ai également représenté les mots par les sprites correspondants dans le jeu.

FONCTIONNALITÉS MANQUANTES

Par manque de temps (ça peut se voir pas ce rapport très court qui est si mal rédigé fait dans les dernières minutes...), de nombreuses fonctionnalités ne sont pas implémentées : pas de ligne de commande disponible donc pas de lancement de niveau à la chaîne, et pas d'execute mot1 mot2 mot3.

Aucune amélioration proposée n'a non plus été ajoutée, et enfin, pas de build.xml et de jar exécutable : tout doit être exécuté et compilé « à la main ».

Malheureusement, c'est le temps qui m'a manqué pour ce projet : on peut tout de même voir que le jeu fonctionne (j'espère ?!), et je pense qu'il s'agissait tout de même de la partie la plus dure du projet. Le reste des fonctionnalités auraient donc pu se faire si j'avais prévu plus de temps pour le projet !