Baba Is You : Notes du développement

Table des matières :

Diagne Ben: ***L3 Info – Groupe 3 – Université Gustave Eiffel***
Kamdom Omairt: ***L3 Info – Groupe 3 – Université Gustave Eiffel***

Partie 1 : Présentation

Le projet entier est stocké dans un package nommé fr.umlv.BabaIsYou. Ce dernier est divisé en sous-packages et chacun d'entre eux contiennent toutes les classes composant le projet. Les sous-packages sont les suivantes :

Arena: Le package qui contient les données du plateau et qui fait la mise en place des règles du jeu.

Database: Package contenant toutes les classes permettant la création plateau de jeu possible. C'est là que sont regroupés les classes définissant les mots (noms, opérateurs et propriétés) et les décors. C'est aussi ici qu'est stocké la base de données des coordonnées qui placent les mots et décors.

Enums : Package listant les classes énumumérées.

Main : Package qui contient la classe Main qui fait toute la mise en place graphique du plateau de jeu.

Partie 2 : Description des classes

Arena

- *PlayGround*: La base de données du plateau. Gère les évènements tels que la création du niveau, le mouvement du joueur, le poussement des décors/mots, la création et vérification des règles de jeu établis et l'état du jeu (victoire ou défaite).
Ses champs sont :

- playGround : Un LinkedHashmap ayant comme clé les mots du jeu et comme valeur une arrayList de coordonnées qui est en fait la liste des positions où le mot peut être dans le jeu.
- decors: Même idée que playGround, mais les clés de mots sont remplacés par des clés de décors.

- Width et Height: La longueur et largeur du niveau.
- YouWon et YouLose : Valeurs booléennes désignant respectivement si le joueur a gagné ou perdu.

Database

- *Word :* La classe représentant les mots. Elle n'est composé que d'un nom qui est de typé énuméré (dont sa classe est contenu dans le package enums) et une méthode de dessin. Elle est cependant le parent des autres classes suivantes :
  - Noun : La classe représentant les noms.
  - Operator: La classe représentant les opérateurs.
  - Property: La classe représentant les propriétés.

Chacune de ces sous-classes contiennent leur propre fonction de dessin qui vont interagir sur le nom de la sous-classe. On a décidé de les couper en sous-classes pour que la méthode d'application des règles (écrite dans la classe Playground) puisse identifier le mot parcouru dans la liste des mots et ainsi pouvoir former une phrase mot->opérateur->propriété définissant une règle.

- *Décor :* La classe représentant les décors. Elle est composé de :
  - representation : Un champs de type énuméré qui définit le type de décor que représente le décor.
  - name_prov : Champs ayant la même fonctionnalité que representation, mais va être utilisé pour l'affichage du décor dans le cas où celle-ci se transforme en un autre décor à cause d'une règle de jeu appliquée.
  - Les 8 variables booléennes passable, pushable, victory, isYou, sink, defeat, hot et melt définissant les propriétés de jeu que le décor possède. Elles sont définis à true si elles possèdent la propriété, false sinon.

À l'instar de la classe Word, elle possède une méthode de dessin, mais possède aussi une méthode same_rules qui va prendre le décor en paramètre et appliquer les mêmes règles de ce décor au décor appelant cette méthode. Cela est utilisé dans le cas où un décor se transforme en

un autre décor et doit alors partager les mêmes règles que le décor sur quoi il s'est transformé.

-*Coordinates* : La classe représentant les coordonnées où peuvent être localisées les classes du même package définies précédemment. Elle possède l'abscisse et l'ordonnée en champs. Elle possède aussi une méthode de déplacement qui modifiera les valeurs des coordonnées selon la direction pris en paramètre et une méthode qui renvoie les coordonnées voisines possibles.

Enums

- *Directions :* La classe listant les 4 directions possibles pour le déplacement du joueur : Up, Down, Left et Right. Elle est utilisé par les autres classes de l'Arena (pour la direction vers où le joueur peut pousser les décors) et du Database (pour la modification des coordonnées à chaque fois que le joueur se déplace).

- *Name:* La classe listant toutes les représentations possibles du jeu. Elle est utilisé par les autres classes de l'Arena pour la création du niveau et du Database pour la représentation des décors et des mots.

Main

Elle est composé d'une unique classe Main :
C'est une classe qui va gérer les évènements de l'utilisateur et utiliser toute la base de données pour afficher le jeu graphiquement. Pour cela, la classe va s'aider d'une boucle dont son nombre de boucle va dépendre du nombre de niveaux donnés dans la ligne de commande (--levels). Cette boucle contiendra aussi une autre boucle imbriqué infini qui ne peut être cassée que si le joueur quitte, gagne ou perd le jeu. Dans cette boucle, on parcourera la map des mots/décors et appeler pour chaque entité leur méthode de dessin pour qu'ils puissent être dessiné en jeu. On affichera aussi l'écran de victoire/défaite dans le cas où le joueur gagne ou perd. Enfin, on gère les évènements de l'utilisateur comme les déplacements du joueur avec les touches directionnelles.

Partie 3 : Création des niveaux

Les niveaux sont crées à l'aide de fichiers texte qui vont contenir la structure des niveaux. Pour integrer le contenu de ces fichiers textes dans notre base de données, on initialise l'objet Playground, on utilise un Reader pour lire le fichier ligne par ligne et pour chaque ligne, on ajoute les valeurs et types marqués dans le fichier sur nos 2 maps playGround et decors. La structure du fichier texte est la suivante :

(entête de fichier) Longueur Largeur
X Y Type Name
…
…
…
…
X Y Type Name

Dont :
Longueur désigne la taille du niveau en longueur.
Largeur désigne la taille du niveau en largeur.
X est l'abscisse de l'objet en entier.
Y est l'ordonnée de l'objet en entier.
Type est le type que l'entité peut être : N pour Nom, O pour Opérateur, P pour Propriété, D pour Décor.
Name est la représentation que l'entité peut être. Dans ce cas là, il faut se référer au type énuméré Name.

Partie 4 : Organisation du projet

Pendant le développement du projet, nous avons fait beaucoup de changement au niveau de l'architecture, surtout suite à la soutenance bêta.

Changement général effectués

-Au lieu d'avoir des coordonnées, on avait des cardinaux North, South, West et East pour permettre aux entités de se repérer par rapport aux autres entités dans le jeu. On s'est rendu compte que cela rendait le dessin graphique de ces entités difficile puisque on ne pouvait pas vraiment trouver un moyen de dessiner les entités à une coordonnée précise. On est donc parti sur des coordonnées.

-On allait au départ ajouter une interface fonctionnelle Bloc pour représenter les blocs (Word et Décor) en général, mais plus on avançait, plus Décor commençait à se distinguer de Word. On a donc virée cette idée.

Changement effectués suite à la soutenance

- Nous sommes au départ partis sur une représentation de la liste des mots et décors de la classe Playground avec des tableaux à 2 dimensions et les coordonnées étaient définis non pas dans les valeurs du tableau, mais au niveau des champs des classes Word et Décor. Nous nous sommes rendus assez tard (après la soutenance) que cela était pas la bonne idée puisque à cause de cela, on ne pouvait pas avoir plusieurs décors sur une seule place car on ne pouvait pas avoir deux objets sur une même indice dans le tableau. Pour contourner ce problème, il fallait ajouter une troisième dimension au tableau, ce qui compliquait encore plus les déplacements des entités dans le tableau. Nous avons donc décidé de faire une refonte du projet en remplacant le tableau par une Map de clé/valeur Entité/Coordonnées, qui nous a alors permis d'avoir plusieurs décors sur une même place.

- On avait de base une arrayList définie dans la classe Décor pour représenter les propriétés qu'elle contenait, et cette arrayList était composée de Strings qui énoncaient le nom de la propriété. Après que notre jury de soutenance nous a fait la remarque qu'il n'était pas bon de représenter les objets avec des Strings, nous avons supprimée cette idée

et remplacée l'arrayList par des variables booléennes au niveau de la classe Décor tandis que pour les strings représentant les propriétés, nous les avons remplacé par un type énuméré Name qu'on définit dans un package à type énuméré, et ce type sera utilisé pour la création des objets Word et Décor.

- Nos méthodes de dessin étaient très grosses puisqu'on faisait beaucoup de conditions if sur la méthode pour vérifier chaque nom de décor possible et, dans le cas où c'est vrai, aller chercher l'image dans un chemin fixe. Pour cela, nous avons enlevé les conditions if et faite une concatenation de strings pour inclure le champs name (pour les word) ou representation (pour les décors) dans le chemin pour éviter la redondance.