Rapport Dévelopeur

Baba is You

BOUKA TCHAMBA Kemet
MARIÉ Brayan
Licence 3 - Informatique
Année 2020/2021

Table des matières

I. Introduction Baba is You

Ce projet, consiste à créer le jeu Baba is You. Il s'agit d'un jeu vidéo de puzzle de type casse-tête au tour à tour créé par le développeur finlandais indépendant *Arvi Teikari*. Dans ce jeu, il est question de créer des propriétés sur certains Blocs pour finir les niveaux.

II. Conception du Projet

1. Partie 1 : Architecture du projet

La partie 1, consiste en une compréhension globale du sujet. Après avoir lu l'entièreté du sujet, un découpage se profile.

Pour commencer, nous avons remarqué trois types d'objets différents dans le jeu. En effet, il y a les **Name**, les **Operator** et les **Properties**. Nous avons décidé d'en faire des classes. Puis nous avons aussi fait une classe **Obj** qui va représenter les éléments de jeu. Toutes ces classes sont des sous-types d'une interface **Block**. Ces classes sont très similaires dans leurs conceptions. En effet elles possèdent toutes les trois :

- Un name, soit NameEnum, soit PropEnum, soit « Is »
- Une coordonnée, X
- Une coordonnée, Y

Ensuite pour stocker tous ces Block, nous avons choisi de créer une classe **Board**, qui est composée de :

- Un board, une ArrayList<Block> pour stocker tous les éléments présents dans le niveau
- Un entier X, pour représenter la taille
- Un entier Y, pour représenter la taille

Une fois que les éléments sont créés et qu'ils sont stockés quelque part, il faut pouvoir récupérer les différentes propriétés présentes dans le plateau. Pour cela nous avons créé une classe **Parser**, celle-ci est composée de deux

HashMap. L'une pour récupérer les propriétés de type « Name Is Properties » et l'autre pour « Name is Name ».

Structures des HashMap :

- HashMap<PropEnum, ArrayList<Block>>
- HashMap<NameEnum, ArrayList<Block>>

Pour stocker les propriétés dans nos HashMap, on fait appel à la méthode « *RecupProp* » dans la classe Board. Cette méthode va regarder la disposition de tous les Blocks présents dans le tableau pour ensuite pouvoir les stocker dans les HashMap de la classe Parser.

Pour la gestion des événements, le classe **Engine** s'occupe de cela. En effet c'est dans cette classe que le jeu va se dérouler. La classe va analyser les actions que le joueur va faire et agir en conséquence. Les mouvements, la défaite, la victoire, tout cela est géré par cette classe.

Enfin pour les gestions des arguments, une classe a été faite à cet effet. C'est la classe **Agrs**, elle va permettre au joueur de choisir différentes façons de jouer. De ce fait on peut choisir d'ajouter des propriétés ou bien de jouer un seul niveau.

2. Partie 2 : Affichage Graphique

Pour la partie graphique nous avons pris la liberté de ne pas faire de classe qui gère tout l'affichage. En effet, dans le projet les méthodes d'affichage sont présentes dans les classes qui concernent les éléments à afficher à l'écran. Par exemple, dans la classe Block on peut retrouver les différentes méthodes qui permettent d'afficher un bloc ou bien de supprimer un bloc a l'écran. Pour ce qui est de la classe Board, elle possède aussi des fonctions d'affichage, par exemple une fonction qui va afficher tout le tableau à l'écran.

Les éléments qui sont présents à l'écran sont des images présentes dans le répertoire « IMG ». Ces images sont chargées grâce à des méthodes prévues à cet effet.

Lorsqu'un élément bouge, on n'actualise pas tout le plateau. En effet on va modifier à l'écran seulement les endroits où il y a eu du mouvement, pour cela on va se servir de fonctions de suppression ou bien d'actualisation d'un bloc ou d'une zone donnée. Cela va permettre de ne pas avoir un effet de flash qui peut être dérangeant pour l'utilisateur. Malgré cela il peut y avoir certains bugs graphiques notamment lorsqu'il y a beaucoup d'éléments « You ».

3. Partie 3 : Chargement des niveaux

Pour le chargement des niveaux, c'est dans la classe **File** que cela ce passe. En effet, les niveaux sont présents dans le répertoire « lvl », ils sont écrits d'une certaines façon à ce que le programme puisse retranscrire parfaitement la disposition des blocs présents dans le fichier. Pour cela nos fichiers de niveaux sont écrit de la façon suivante.

Taille x du plateau

Taille y du plateau

Type du bloc [x,y]

…

Cela va permettre d'avoir toutes les informations nécessaires à l'initialisation de notre plateau de jeu. Le programme va regarder les types de blocs et les créer pour les ajouter dans le plateau.

III. Retour de la soutenance β

Apres la soutenance β avec Mr.JUGÉ. Il nous a été dit qu'il y avait trop de duplication de code, notamment pour la fonction de mouvement. En effet les fonctions bougegauche, bougedroite, bougehaut et bougerbas ne sont maintenant qu'une seul fonction « Move ». En plus de cela, la plupart des fonctions avec de la duplication ont été modifiées.

De plus nous avons des fonctions qui paraissaient trop longues. Nous avons essayé de factoriser au maximum nos fonctions pour qu'elle ne dépasse pas vingt lignes.

Nous avons suivi les conseils de Mr.JUGÉ à propos de l'utilisation de la HashMap qui doit reconnaitre les Name qui nous posaient problème avant la soutenance.

Nous avons aussi ajouté la java doc ainsi que de la sécurisation de code avec des changements de visibilité et des analyses d'arguments.

IV. Anecdotes

1. Niveau supplémentaire

Pour le niveau supplémentaire nous avons créé un niveau où pour le réussir, il faut détruire des murs pour atteindre un flan qui win. Pour cela nous avons implémenté dans notre jeu un nouveau nom et une nouvelle propriété.

En effet, nous avons ajouté :

- Arouf, qui est un personnage connu sur internet
- Gang, qui va permettre de détruire les murs

2. Problèmes rencontrés

Il y a eu certaines difficultés lors de la conception de ce projet:

- Compréhension et Maitrise de zen5
- Récupération des propriétés et des noms avec l'utilisation de Hashmap
- Lecture de fichier
- Difficultés à la visualisation du projet après lecture du sujet

V. Conclusion

Ainsi ce projet s'avéra être très formateur.

En effet, il était nécessaire de bien réfléchir à une architecture pouvant convenir et essayer de la rendre la plus efficace possible.

Ce projet nous a aussi permis de gagner de l'expérience en modularisation de projet et dans notre approche (l'implémentation peut différer fortement d'une personne à l'autre).

Enfin, un projet avec peu d'indications nous a permis de retrouver et améliorer notre autonomie et notre logique de programmation.