Rapport de projet

Sommaire

I. Principe du jeu	1
II. Déroulement du jeu	2
1.Version interface	2
2 .Version sur terminal	3
III. Aspects techniques	3
1.UML	4
2.Classes et fonctions	4
3.Problèmes rencontrés	4
4.Bogues	5
5.Améliorations possibles	
IV Conclusion	

<u>I. Principe du jeu</u>

Le jeu que nous avons créé, basé sur Pet Rescue Saga, consiste à sauver un certain nombre d'animaux coincés par un plateau de blocs et d'obtenir un certain de nombre de points accumulés au fur et à mesure que le joueur détruit des pairs de blocs. Le plateau de blocs est créé de manière aléatoire sans prédisposition à une victoire ou à une défaite.

Pour se débloquer d'une impasse, il est possible d'utiliser certains outils à disposition qui sont le «Block Buster», permettant de détruire un bloc choisi par le joueur, le «Color Pop», qui détruit les blocs d'une même couleur au choix et enfin le «Column Blaster», qui servira à détruire tous les blocs d'une colonne au choix.

Le jeu comprend plusieurs niveaux qui se distinguent par le nombre d'animaux à sauver ainsi que les points à obtenir afin de passer au niveau suivant.

Les points sont obtenus par rapport aux destructions de blocs et permet d'obtenir un certain nombre d'étoile variant de un à trois.

Pour débloquer un niveau, il suffit pour cela d'avoir joué le niveau précédent et de l'avoir réussi. De plus, le jeu ne met à jour le nombre de vos étoiles pour un niveau donné uniquement s'il est supérieur au nombre d'étoiles obtenus auparavant.

Le joueur ne peut pas lancer infiniment une partie. Si le joueur perd une partie, il perd une de ses vies. Il les regagne petit à petit au bout de cinq minutes.

Si le joueur réussit une partie, il reçoit un certain nombre de pièces, de lingots et de boosters (Column Blaster, Color Pop et Block Buster).

Le joueur peut quitter à tout moment une partie au risque de perdre une vie.

II. Déroulement du jeu

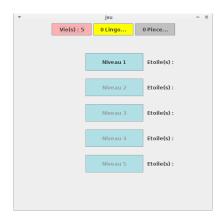
1. Version interface

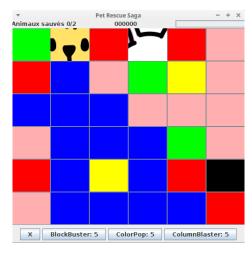
Lorsque l'on ouvre le jeu, on tombe sur la page d'accueil nous demandant si nous voulons jouer.



En cliquant sur le bouton jouer, une fenêtre avec tous les niveaux et les informations du joueurs apparaît.

Au départ, nous n'avons accès qu'au premier niveau, et en cliquant sur le niveau 1, un plateau s'affiche avec différents blocs et des animaux. On a également des informations sur le nombre de boosters que nous possédons.



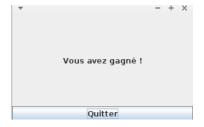


Un plateau arrangé de manière aléatoire apparaît. Il suffit de cliquer sur un bloc de couleur pour que tous ses blocs de même couleur au voisinage direct s'effondrent, si le coup est bien valide. Les blocs détruits sont représentés en boutons de couleur blanche.

Après chaque coup sans booster, le nombre d'animaux sauvés, les points et une barre montrant le nombre d'étoiles obtenus se mettent à jour.

Après un coup avec un booster, le nombre du booster utilisé diminue.

Une fenêtre message s'affiche que ce soit lorsque l'on gagne une partie, perd une partie ou quitte une partie.







2 .Version sur terminal

Le joueur doit entré un pseudonyme et entrer par clavier s'il veut jouer ou non. On lui liste ses informations telles que le nombre de vies, de lingots et de pièces qu'il possède puis les niveaux qui lui sont accessibles. En affichant les niveaux disponibles, le terminal indique également le nombre d'animaux à sauver et les points minimum à obtenir afin de gagner la partie qui va être lancée.

On nous demande aussi le niveau que l'on souhaite joué. Au départ, on ne peut jouer qu'au premier niveau.

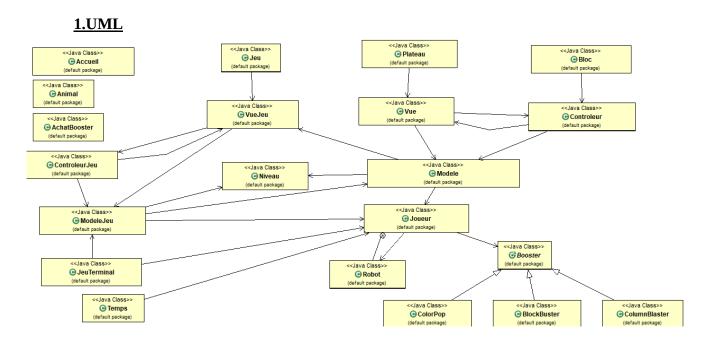
Un plateau arrangé de manière aléatoire s'affiche avec des lettres, qui indique la place d'un animal, et des chiffres. Les chiffres, ici, font office de blocs, il faut donc détruire des pairs de chiffres voisins.

Au fur et à mesure, le terminal demande si l'on veut utiliser un booster puis après avoir répondu à cette question, nous demande les coordonnées d'un bloc. Si le coup est valide, le jeu cherchera luimême tous les blocs voisins et les éliminera. Les blocs éliminés seront numérotés -1.

```
Donner un numéro de ligne :
Donner un numéro de colonne :
           2
                   3
                                            6
   2
                   -1
                                    2
                                            2
                   3
   0
           3
                           L
                                                   2
                                                   3
   2
           3
                           0
                                            0
                   0
   0
                                    2
                                            2
   0
           2
                   0
                           3
                                    2
   3
           0
   points
                       ColorPop: 5.
                                       ColumnBlaster(s)
```

A la fin d'une partie, le jeu nous demande si l'on veut continuer à jouer. Si la réponse est oui, le jeu continue comme auparavant, sinon, le jeu termine.

III. Aspects techniques



2. Classes et fonctions

Nous avons créé plusieurs classes telles que la classe Joueur, sa classe interne Robot, une classe Animal, Bloc, Booster qui possèdent des sous-classes comme BlockBuster, ColorPop et ColumnBlaster, Niveau, Temps.

Pour les classes comme Plateau et Jeu, nous avons décidé de les diviser en trois comme vu en cours, c'est-à-dire avec Modele et ModeleJeu, Controleur et ControleurJeu et enfin Vue et VueJeu respectivement.

La classe JeuTerminal enclenche le jeu sur le terminal. Il utilise des fonctions se trouvant dans la classe Modele également utilisée par le Jeu version interface graphique.

La plupart des attributs sont **private** et le reste est **protected**.

Pour le plateau, nous avons pris la décision de créer un tableau bi-dimensionnel d'Object afin de pouvoir placer à la fois des blocs et des animaux. Nous avons rajouté des lignes supplémentaires et des colonnes supplémentaires afin de faciliter la recherche de voisins sans erreur. Dans les lignes et colonnes supplémentaires, il n'y a que des blocs de numéro -1.

Une grande part des fonctions est pour le bon fonctionnement des coups de blocs. On a implémenté des fonctions comme voisins qui d'abord identifie les blocs identiques qui se trouvent juste à côté puis la fonction garde en souvenir les coordonnées de ses blocs voisins et cherchent les blocs voisins de ses mêmes blocs, jusqu'à trouver l'intégralité des blocs voisins.

On a également des méthodes qui permettent d'ajuster le plateau, de sauver les animaux s'ils se retrouvent en dernière ligne et de décaler les colonnes au cas où une colonne au milieu du plateau se trouve vide. Nos méthodes ne suppriment pas les blocs mais les fait tomber.

On a une méthode permettant de déterminer si le joueur atteint une victoire et une autre méthode déterminante s'il est possible de continuer le jeu en cherchant s'il y a au moins une paire de blocs et un nombre de boosters non nul.

On a, pour finir, implémenter des méthodes pour mettre à jour plusieurs informations essentielles telles que le nombre d'animaux sauvés, le nombre de points obtenus etc.

3.Problèmes rencontrés

Pour créer le jeu sur le terminal, à certaines occasions la lecture du scanner n'était pas possible. Le scanner s'arrêtait tout seul, par exemple au moment où l'on veut continuellement demander si le joueur veut jouer après la fin d'une partie.

Nous avons réglé ce problème en appelant un nouveau sc.nextLine() (sc: nom du Scanner). De même pour la question «Utiliser un BlockBuster ?» posait souci tandis que pour les autres demandes et même pour les demandes d'utilisation des autres boosters aucun souci n'était présent. Pour remédier à ce problème, la solution citée auparavant a été de nouveau utilisée.

Nous avons tenté d'implémenter un robot qui exécute des coups aléatoires. Cependant, les fonctions qui permettent de faire un coup renvoie des erreurs et nous n'avons pas pu les corriger.

Nous avons également essayé de faire en sorte que le joueur puisse acheter des booster selon le nombre de pièces et de lingots qu'il possède. Mais nous n'avons pas pu relier les classes entre elles. Dans la classe Joueur.java des méthodes permettent d'acheter des booster, la classe AchatBooster.java correspond à la fenêtre et un bouton Achat de Booster est présent dans la classe Jeu.java.

4.Bogues

A certains moments, l'utilisation de booster ne fonctionne pas correctement sur la version Interface Graphique. La source du problème n'a pas pu être identifiée. Cependant, les premières utilisations des boosters sur cette version-même ne pose pas de souci.

5. Améliorations possibles

Pour améliorer ce projet, il aurait été possible de coder le jeu en sorte qu'il garde en mémoire les joueurs avec leur pseudonyme ainsi que les résultats de leurs parties passées.

De plus, fusionner certaines méthodes utilisées à la fois dans la version Interface Graphique et la version sur terminal aurait permis une meilleure approche du projet.

Nous aurions pu également implémenter d'autres fonctionnalités supplémentaires telles que les grillages sur les blocs ou des cages autour des animaux.

IV. Conclusion

Ce projet a été infiniment instructif que ce soit au niveau du langage Java, de l'organisation d'un code sans méthodes faites et données au préalable ou encore au niveau de l'organisation d'un projet en équipe.

Ce projet, pédagogique et ludique, nous a donné l'opportunité d'être responsable d'un projet du début jusqu'à la fin.

Naïma AMMICHE, Tharsiya VIGNESWARAN, binôme 43.