Rapport Snake

Auteurs:

Bouquoyoue Bilel

Derreumaux Valentin

Tang Jean-Michaël

Les choix d'implémentation effectués

Nous avons choisis d'utiliser le Deque plutôt qu'un ArrayList pour faciliter le stockage de données du serpent. En effet, le système de stack de Deque permet de facilement changer les positions où se trouve notre serpent. Il suffit de push une nouvelle valeur, remplacer les autres par le précédent et supprimer le dernier élément si le serpent n'a pas mangé d'oeuf. De plus, la longueur du stack peut varier, ce qui convient parfaitement au jeu snake, où le serpent change de taille à chaque fois qu'il mange un oeuf.

L'affichage en console est basé sur la logique du jeu qui contient également la partie graphique. Au début, l'affichage en console ne faisait qu'imprimer des coordonnées. Nous nous sommes tourné vers une autre solution après consultation avec le professeur qui voulait une jouabilité sur console. De ce fait, la partie de l'affichage console se base également sur le Deque (cfr ci-dessus). Malheureusement, il faut pouvoir stocker les données du serpent, tout en ayant les murs affichés et avoir les oeufs également. Ce qui nous oblige à travailler sur une matrice à deux dimensions. Pour imprimer cette matrice, il faut faire une double boucle, ce qui ralentit un peu l'affichage.

La connection à la base de donnée est fort simple, elle ne fait que récupérer un Id_player et un Score_player afin de simplement pouvoir créer un tableau de score, pour récupérer les valeurs ou les envoyer dans la base de donnée via des requêtes SQL.

Les difficultés rencontrées

Au début, nous avions quelques difficultées concernant le fonctionnement du push-commit de Github et les packages sur Eclipse. Le problème rencontré sur Github était d'envoyer correctement nos codes dans le repository, il nous fallait plusieurs essais avant de push correctement. Les packages sur Eclipse posaient souvent problème aussi pour lancer notre programme: des bugs de packages sont parfois présentes et nous empêche de lancer le programme car le code ne se trouvait pas dans un package, mais dans un dossier.

Un des problèmes était la compréhension des consignes. Certaines consignes n'étaient pas toujours claires, et il fallait soit remuer nos méninges, soit demander directement au professeur d'expliciter ces consignes.

Le plus gros problème, était l'organisation. Nous ne pouvions en effet pas prendre de l'avance pour respecter les consignes des professeurs. Mais l'apprentissage du MVC et de la partie graphique ont été donnés assez tardivement, ce qui n'est pas optimal pour finir le projet correctement, puisqu'il y a les cours d'un côté et des présentations de l'autre.

Nous avions aussi quelques difficultés à nous organiser entre nous. Un membre du groupe était tombé malade et un autre a eu des problèmes personnels à régler, ce qui réduisait pas mal notre temps de gestion.

La partie de l’affichage console a été un autre de nos problèmes principaux. Nous avions tout d'abord décidé d’afficher en console seulement les coordonnées de l’oeuf, de la tête du serpent, du bonus, le score et si les murs étaient actifs ou pas.

Nous avons appris par la suite, que ce n’était pas suffisant. Dans cet affichage on ne pouvait pas voir le serpent en entier. Nous avions donc du trouver rapidement une solution pour avoir un véritable affichage console sur le peu de temps qu'il restait de la deadline, en sachant qu’il nous restait les bonus à implémenter et toute sorte de petites modifications.

Les pistes d’amélioration éventuelles

Un mode où la vitesse augmente à chaque oeuf mangé serait une bonne idée à implémenter plus tard, ce qui donnerait des petits score mais le jeu serait d’une difficulté variable pendant la partie.

Nous pourrions implémenter des ennemis dans le snake que le joueur devra tuer en crachant du venin dessus par exemple.

Donner un design plus élégant au serpent.

Insérer une musique dans le jeu et dans le menu.

Faire un menu encore plus intuitif et mieux réparti.

Créer différentes maps (Par exemple : une map avec des murs au milieu).

Rajouter différentes musiques.

Pouvoir modifier/ajouter des packs d’arrières plans.

Un système de pièce qui permettrait d’acheter des objets tels que des skins de serpent. Exemple : Un ver de terre, différentes espèces de serpent, un dragon chinois, une corde etc…

Dans le cas où il y ait un ajout d'ennemis, une boutique vendrait des armes, des armures, des potions

Conclusion individuelle de chaque membre du groupe

Pour conclure la fin du rapport, nous allons donner notre avis personnel.

- **Bouquoyoue Bilel :**

Ce projet a été une excellente façon de bien nous faire rentrer dans l'apprentissage JAVA. J'ai aimé que malgré les consignes, on était assez libre de faire "ce que l'on voulait". On était pas du tout sûrs de pouvoir programmer entièrement un snake. Mais on s'est vite mis à travailler dessus. La version du snake de base à été finie en plus ou moins 3 semaines et demi. Le reste à surtout été des ajustements, des rajouts, beaucoup de debuggage etc…

Ensuite, j'ai été content d'avoir eu un groupe comme celui-ci. Ils étaient à l'écoute, comprenaient vite ce que je leur disait malgré des problèmes personnels d'un des membres et la maladie de l'autre.

Concernant le twist, on avait plein d'idées mais pas nécessairement le temps ou la niveau de codage. On a tout de même décidé de faire un snake avec un maximum de choix possibles, tels que la vitesse de jeu (3 choix), l'activation des murs, la couleur du serpent (3 choix).

Ce projet m'a réellement fait aimer JAVA, d'ailleurs certains étudiants de première, ce sont intéressé à ce que je faisais et ont même commencé à apprendre le JAVA. Pour rejoindre mes camarades, les consignes m'ont moi aussi, perturbé. Par exemple, le fait de commencer par l'interface console. Pour nous, le snake était impensable en console… Maintenant que le projet est terminé, on se rend compte qu'on aurait dû bel et bien commencer par là.
Pour terminer, ce projet est une très bonne approche de JAVA, et j'en garderai un bon souvenir.

- **Derreumaux Valentin :**

La création d'un projet de petite envergure avec un groupe de 3 personnes est une excellente idée pour nous apprendre les méthodes de programmation et de travail actuel, cela nous force à communiquer, à nous adapter, à réagir rapidement et à s'entraider pour obtenir un résultat dit "fonctionnel".
Cela m'a permis d'apprendre plus et de comprendre (ou du moins essayer) d'autres méthodes de travail, d'autres logiques de programmation. Il s'agit en soi d'une mise en condition complète et d'une immersion obligatoire dans du code afin de comprendre pourquoi et comment les autres membres du groupe on pris telle ou telle décision.
J'ai eu beaucoup de mal à suivre le groupe à cause de mon manque de logique de programmation mais j'ai eu la chance d'avoir un groupe qui m'expliquait ce que je ne comprenais pas, j'espère avoir pu en faire de même avec mes parties de code.
De plus, cela permet de souder les groupes de travail.
Cependant les consignes du projet étaient pour moi parfois fort nébuleuses, et ne

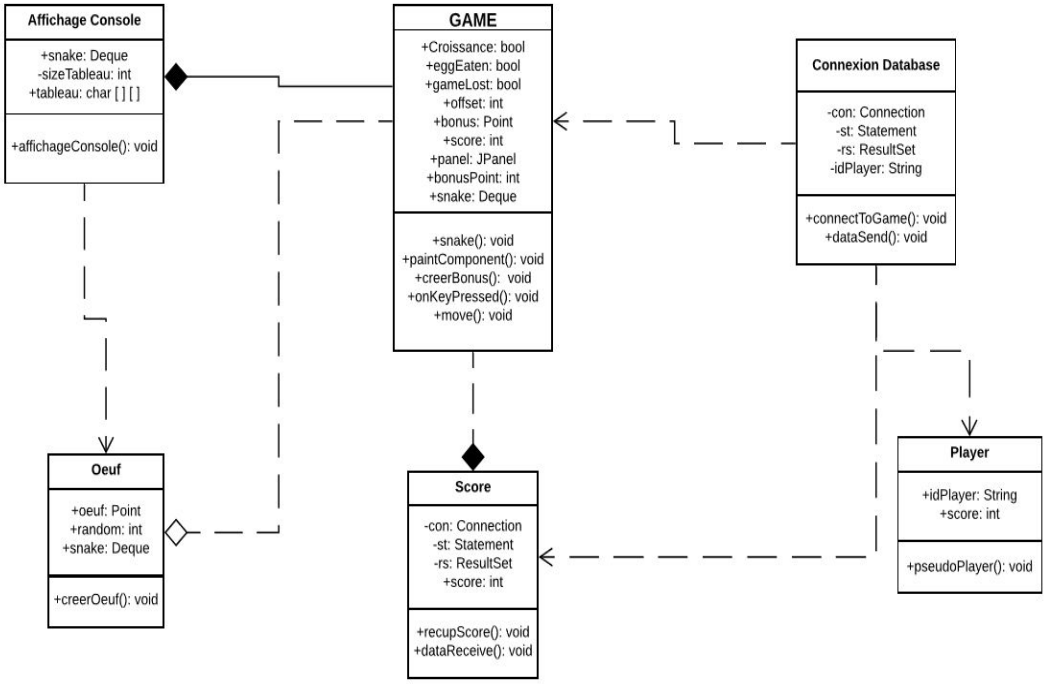
permettaient pas un suivi correct du projet, certaines méthodes sont données fort tard par rapport au planning des remises de projet. Cela étant dit cela nous force à avoir des deadlines comme dans le monde du travail. Le travail demandé pour le projet demande une quantité non négligeable d'investissement, parfois au détriment d'autres cours et le suivi de la matière n'est pas toujours complet vis-à-vis de cela, nous obligeant à chercher, se renseigner sur d'autre plateforme ce qui permet de remplir notre base de donnée de connaissance.

- **Tang Jean-Michaël :**

Faire un projet de ce type est une bonne idée pour se former à collaborer plus tard avec de futurs collègues et à savoir faire des recherches pour comprendre certains bouts de programme.
C'est aussi un bon entraînement nous forçant à devenir curieux pour certaines choses, par exemple, je ne connaissais pas le Deque, ce qui m'a obligé à aller chercher des informations là-dessus pour le comprendre.
Un projet de groupe permet aussi de travailler un bout de code ou une algorithmique programmée par quelqu'un d'autre et savoir continuer ce travail par la suite. Ceci peut nous apprendre à voir les choses différemment tout en improvisant et en s'améliorant au fur et à mesure du temps passé à programmer ensemble.

En revanche, les détails donnés pour réaliser le projet nous sont donnés trop tardivement, ce qui nous empêche de perfectionner le code du projet.
Par exemple: MVC aurait dû être mentionné dès le départ du projet, car on a dû chipoter tout le code pour avoir plus ou moins ce qu'il faut, ce qui est une grosse perte de temps.
Pouvoir faire la partie graphique dans les deux dernières semaines n'arrange pas non plus les choses, car il faut se renseigner en plus des cours pour avoir une interface graphique potable.

DIAGRAMME UML



CAHIER DES CHARGES

Idée du projet informatique:

Création d'un jeu snake contenant plusieurs difficultés de jeu, des twists et un peu de personnalisation

Contexte du projet :

Créer un jeu snake afin de pouvoir l'introduire dans plusieurs ordinateurs de bureau pour que les employés puissent en profiter durant leurs pauses tout en les mettant en compétition via un tableau de score.

Objectifs du projet :

Divertir les utilisateurs qui jouent au jeu et se démarquer des autres jeux de snake qui sont nombreux au vu de l'ancienneté et de la popularité de ce jeu.

Contraintes techniques :

- Avoir deux interfaces (graphique et console).
- Avoir une communication réseau.
- Contenir une structure de données du framework Java Collection.
- Gérer la vitesse du serpent.
- Gérer les déplacements du serpent de manière cohérente.
- Avoir un Gameplay intuitif.
- Ajouter des bonus car le jeu est trop classique.
- Développer nos connaissances en Java.

Description des besoins fonctionnels :

En tant que joueur, je recherche un jeu fonctionnel et intuitif. On souhaite faire en sorte qu'un joueur, expérimenté comme débutant, puisse effectuer une partie agréable. Le jeu aura besoin de contrôles intuitif et de mouvements cohérents.
Il est aussi nécessaire d'avoir des difficultés correctement gérées afin que le jeu en mode normal soit jouable par un joueur lambda.
Les objectifs doivent être précis et facilement compréhensible afin de ne pas être perdu dans le jeu.
Il faut des bonus permettant de varier le jeu et de mettre à défis le joueur.