Shafie Ichem
Baduel Thomas

Manuel développeur :

I – Définition du projet

L'objectif de ce projet à été d'implémenter un jeu du snake, de type curvytron, avec une interface graphique basique fourni (Zen5), en utilisant les différentes structures et implémentations vues en cours. Il s'agit d'un jeu jouable en solo (extensible multijoueur) où il faut contrôler un serpent qui grandi sans se déplacer (c-a-d que la tête avance, mais pas la queue), qui ne peut que changer de direction vers la gauche ou la droite.
Nous avons cependant ajouté, sous la forme d'un bonus en jeu, la possibilité de passer d'un serpent de type curvytron à un serpent de type Snake classique.

II – Présentation du projet

Ce projet, codé en Java, mets en œuvre ce qu'on a appris en cours.
La Javadoc de ce projet est disponible dans le dossier doc. Elle contient toutes les informations sur l'implémentation des fonctions et des classes ainsi que leur utilisation.
Au besoin, celle-ci pourra être ré-généré grâce au code source.

Présentation des classes

Le projet est divisé en plusieurs classes, toutes contenues dans le package *fr.umlv.curvy*

- Une interface AllSnake, qui regroupe les deux types de snakes que l'on peut avoir dans le jeu.
Les classes Snake et ClassicSnakes implémentent cette interface :
Snake , qui est la version curvytron, implémente les déplacements, la détection de collision ainsi que l'application d'un bonus rencontré par le snake.
ClassicSnake implémente la version classique du snake, ses déplacements, ...

- Une classe Window, qui contient notamment toutes les informations sur la fenêtre de jeu, la taille des bordures autour du plateau, ..

- Une classe Bonus, qui implémente les bonus.
Elle est complétée par les classes BonusAction et BonusGenerator :
-BonusAction implémente les actions en jeu des bonus,
-BonusGenerator implémente leur insertion en jeu.

Shafie Ichem
Baduel Thomas

- Une classe EventManager, qui implémente tous les événements avec l'utilisateur, notamment la récupération du clavier et l'action des touches en jeu.

- Une classe Map, qui regroupe toutes les information du plateau de jeu, le plateau en lui même (le tableau à deux dimensions), et implémente donc les actions d'ajouts et de modification de valeur dans celui-ci.

-Une classe Case, qui représente une case du tableau implémenté dans Map.

- Une classe Menu, qui implémente l'affichage du menu de l'application.

- Une classe Draw, qui s'occupe de l'affichage de tous les éléments
(le background, les shapes pour le snake, et les icônes des bonus).

- Une classe Main, qui est la classe exécutant l'application :
Elle contient notamment la méthode *public static void main().* De plus, elle contient aussi les méthodes permettant l'exécution du jeu en mode solo ou multijoueur.

Choix de l'implémentation

Les différents choix de structures dans ce projet ont été fait de façon a rendre le programme fluide :

La structure de tableau à deux dimension pour stocker l'état du plateau. Elle permet notamment de connaître la position de la tête, sa direction, et n'avancer que celle ci, en ajoutant les nouvelles coordonnées dans le tableau.

Afin de connaître l'état du jeu, le plateau est stocké dans un tableau à deux dimension (de la taille du plateau de jeu, c-a-d de la zone jouable).
Une valeur à 0 signifie que l'emplacement est vide, sinon l'id du snake ou du bonus y est stocké.
Cette implémentation permet notamment d'avoir un parcours constant afin de connaître les positions qui seront accessible par le snake.

Cette façon d'implémenter permet d'optimiser le temps de parcours du serpent, car pour vérifier les collisions contre un autre objet, nous n'avons qu'a regarder si les cases devant le snake sont libres ou occupées.

Difficultés rencontrées

Dans ce projet, la principale difficulté a été de prendre en main l'interface graphique fourni et d’implémenter un jeu autour de celle-ci.

Tout d'abord, nous n'avons pas réussi à rendre ce jeu aussi parfaitement jouable (niveau gameplay)

au niveau du contrôle du serpent : la gestion des événements dans la librairie Zen5 nous a bridé.

Par la suite, afin de pouvoir gérer les collisions (c-a-d la rencontre du serpent avec un autre objet (ou lui même par exemple)), nous devions sauvegarder les coordonnées de chaque partie du serpent (son corps).
Hors stocker chaque valeurs avec une position dans une liste chaînée, et parcourir tout le serpent à chaque fois pour vérifier les collisions, n'était pas efficace. Nous avons donc implémenté le tableau à deux dimension qui ne sauvegardera que le passage de la tête du serpent. Le serpent sera donc comme « imprimé » dans le tableau.

Le seul inconvénient de cette implémentation est que lorsqu'un serpent rentre en collision avec un bonus, le bonus s'applique au serpent et disparaît. Un morceau du serpent peut alors être effacer du plateau, car encore une fois on efface le bonus (et sa zone de valeur) directement dans le tableau. Cependant, cela rajoute du gameplay (car on ouvre alors un nouveau chemin de passage à travers le serpent), nous l'avons donc laissé.

L'ajout de bonus en jeu qui affectent le serpent à été la dernière difficulté. Mais nous avons réussi à créer le générateur aléatoire de bonus ainsi que leurs apparition en jeu et leurs effets sur le serpent.

III - Amélioration possible et perspectives

Ce jeu a été développer de façon à pouvoir être implémenter et jouable en multijoueur :
Cependant, la librairie Zen5 et sa gestion d’événements du clavier nous ont empêcher de pouvoir faire un mode multijoueur (en local) optimal :
En effet la libraire n'attends qu'une seule touche à la fois, hors pour pouvoir jouer en multijoueur il faut plusieurs touches disponibles en même temps.

Autres améliorations possibles :

- Dans une nouvelle implémentation, nous pourrions passer les bonus dans une classe abstraite, afin de faciliter l'ajout d'un nouveau bonus en jeu.

- Améliorer la saisie des touches du clavier afin de rendre le mode multijoueur jouable.

- Toutes les folies possibles et imaginables d'un développeur.