UNIVERSITÉ D'AVIGNON
ET DES PAYS DE VAUCLUSE

CENTRE
D'ENSEIGNEMENT
ET DE RECHERCHE
EN INFORMATIQUE

Licence Informatique
Parcours Ingénierie Logicielle
UE Projet de Programmation

>>>

Rapport d'Introduction au projet "Curve Fever"

Alexandre Latif & Quentin Castillo

13 octobre 2015

CERI - LIA
339 chemin des Meinajariès
BP 1228
84911 AVIGNON Cedex 9
France

Tél. +33 (0)4 90 84 35 00
Fax +33 (0)4 90 84 35 01

http://ceri.univ-avignon.fr

Table des matières

1 Moteur physique

Le rôle de ce moteur est de gérer l'ensemble des calculs/actions lors d'une partie de Curve Fever. Nous nous chargerons donc de coder les informations que le moteur graphique et l'interface utilisateur afficheront. C'est notre rôle d'effectuer les déplacements, gérer les collisions, calculer lorsqu'un round est terminé, etc, ...

Notre interaction principale sera avec le moteur graphique, étant donné que ce que nous coderons sera utilisé pour permettre la création d'objet «visible» par l'utilisateur. Nous aurons également un lien avec les deux autres composantes. Nous envoyons les informations du score à l'interface utilisateur et il nous envoie le lancement d'une partie. Quant à la composante réseau, nous gérons les différents joueurs qu'il aura mit en relation.

2 Interactions

2.1 Réseau

Le lien que le moteur physique aura avec la composante réseau sera probablement inexsistant car l'échange de données concernant les joueurs qui rejoindront la partie sera géré par l'interface utilisateur, le moteur physique étant chargé de coder les informations du jeu, nous n'avons pas de lien réel avec cette composante.

2.2 Moteur Graphique

Concernant le moteur graphique, nous aurons forcément des relations notables, étant donné que ce dernier est censé afficher en terme graphique ce que nous aurons créé en terme codage. Nous aurons donc besoin d'envoyer toutes sortes de données nécessaires au moteur graphique pour que ce dernier puisse afficher ce dont il a besoin (plateau de jeu, snakes, ...)

2.3 Interface Utilisateur

Nous aurons également un lien avec l'interface utilisateur étant donné que ca sera dans un premier temps à lui de nous fornir des informations, comme le nombre de joueurs, les touches associées (pour aller à gauche/droite), la taille du plateau (à voir avec les autres groupes). Puis pendant la partie, nous aurons besoin d'envoyer le score à l'interface utilisateur pour qu'il puisse l'afficher et retenir ces données pour les statistiques personnelles de chaque joueur.

3 Formalisation des interactions

3.1 Classe Board

La classe Board sera celle qui représentera l'espace de jeu, avec donc les Snakes à l'intérieur, puis les bonus/malus plus tard.

Nous aurons besoin dans cette classe de deux entiers représentant la longueur et la largeur. (ces données seront fournies par l'interface utilisateur. Ces données pourront dépendre du nombre de joueurs ou tout simplement d'un choix arbitraire).

Il nous faudra ensuite un tableau à deux dimensions pour correspondre à l'ensemble de données du tableau (que l'on fournira à la composante graphique).

3.2 Classe Snake

La classe snake sera celle qui répresentera le tracé des joueurs, avec pour but pour chacun d'entre eux, la survie le plus longtemps possible.

Il nous faudra dans cette classe beaucoup de données : boolean alive (pour vérifier s'il est toujours en jeu), int[][] position (vérification de position), String couleur. On pourra également implémenté toutes les données correspondant au bonus/malus qu'on pourra rajouter (vitesse, invisibilité, taille, etc, ...).

Certaines de ces données seront fournies à la composante graphique.

3.3 Méthodes à invoquer

int[] giveSize() ; Qui permet de récuperer la taille en longueur et largueur du tableau. Avec tab[0] qui sera la longueur et tab[1] la largeur.

Player[] givePlayers() : Qui permet de récupérer le nombre de joueurs et leurs données (touches clavier et ID plus particulierement pour ensuite renvoyé leurs points).

3.4 Méthodes à implémentées

int[] giveSize() ; permet d'envoyer au moteur graphique la taille du tableau qu'il devra afficher.

String[] giveSnakesInfo() : permet d'envoyer au moteur graphique les données concernant les snakes (couleur, position, etc, ...)

void SnakeDied() : permettra à l'interface utilisateur d'incrémenter le score des joueurs encore vivant.

4 Conclusion

Pour conclure, nous avons principalement des liens avec le moteur graphique et l'interface utilisateur, avec pour le premier, des données à envoyer et pour le dernier, un échange de données selon le stade de la partie. Il faudra ainsi voir avec eux le nom des classes, les variables à créer (s'ils en ont besoint pour une quelconque raison) ainsi que les méthodes.