UNIVERSITÉ D'AVIGNON
ET DES PAYS DE VAUCLUSE

C E N T R E
D'ENSEIGNEMENT
ET DE RECHERCHE
EN INFORMATIQUE

L3 informatique
Ingénierie Logicielle
UE Projet de programmation

>>>

Rapport de projet

Rémi Drissi & Valentin Girardon

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 Description du projet

La composante **Moteur Graphique** doit récupérer toutes les autres infor- mations, valeurs et variables calculées par les autres composantes afin de pouvoir afficher le jeu en continu, en fonction des déplacements de chaque joueur et des résultats de chaque partie. Les informations liées à chaque joueur lors d'une partie seront communiquées à notre composante pour que nous puissions afficher le déroulement du jeu depuis le lancement de la partie, avec un design visuellement confortable pour l'utilisateur. Moteur Physique : Nous sommes liés avec cette composante car, lorsqu'ils auront calculé tous les algorithmes de déplacement, de collision etc... nous devrons alors afficher les déplacement / arrêter d'afficher un ou plusieurs dé- placements s'il y a collision entre les joueurs ou avec les limites du cadre de jeu. L'affichage du jeu dépendra donc grandement du travail réalisé par les autres composantes, car il nous sera impossible d'afficher des déplacements corrects si l'algorithme de collision n'est pas correctement codé, et il en est de même pour les déplacements, les calculs de score, etc. L'illusion de mouvement des points sera créée grâce à la boucle du jeu, une nouvelle position affichée toutes les X milisecondes. Interface Utilisateur : Lors d'une collision, l'affichage des points gagnés par le joueur se fera au niveau de l'interface utilisateur. Il nous faudra donc recevoir de la composante interface utilisateur les données adéquates pour afficher un score correct. Au cours de la partie, si un utilisateur entre en collision avec un objet, on affichera les points qui iront s'ajouter à son score total, ces points qui dépendent évidemment du temps resté en jeu (S'il est le 1er joueur éliminé, il ne gagnera pas de point, tandis que s'il termine 'dernier joueur survivant' sur le plateau de jeu, il obtiendra des points équivalents au (nombre de joueurs - 1). L'affichage des pseudonymes, (s'il y en a) des scores et des trajets parcourus par chaque joueur se fera selon un code couleur. Assignée à chacun d'entre eux au début de la partie, chaque joueur se verra attribué d'une couleur aléa- toire parmi une palette des couleurs distinctes les unes des autres, pour éviter toute confusion.

2 Implémentation

Il sera necessaire de faire implémenter des fonctions dans d'autres composantes pour pouvoir afficher correctement le jeu. Nous aurons besoin d'avoir :

- Les classes **Board** et **Snake** : le Board, plateau de jeu, sera pourvu d'un attribut **size**, d'objets de type Snake, et éventuellement des objets de type **Bonus/Malus** (objets ronds qui déclencheront des évenements lorsqu'un snake entrera en collision avec).
- Des fonctions **GET** dans les classes citées ci-dessus, telles que :

  - **GET-ID** pour avoir l'identifiant de chaque joueur

  - **GET-COLOR** à évaluer avec la composante **Interface Utilisateur** pour connaître la couleur associée à chaque joueur et donc à chaque snake

  - **GET-POS** (ou autre nom défini dans la classe) : Nous avons choisi de représenter les positions (X et Y) occupées par les snakes grâce à une Hashmap. Il s'agit d'une matrice creuse, qui definira la couleur de chaque pixel du plateau. Cette matrice se vera modifiée à chaque tour de boucle, pour chaque mouvement de snake.

  - **GET-SCORES***(joueur J)* La composante interface utilisateur nous à laissé le loisir de nous charger de l'affichage des scores. En effet, cette tâche étant l'une des plus liées avec l'affichage du plateau, nous voulions également nous charger de cette partie car nous trouvions la notre un peu pauvre en travail.

- Des classes **JFrame** et **JPanel** avec *Swing* pour l'affichage des objets graphiques.
  On pourra eventuellement se servir (par exemple) d'une fonction **DRAW (ID, GET-POS-X, GET-POS-Y)**