Rapport07

Harkiolakis Laurent c Bardout Rudy

October 13, 2015

1 Présentation de la composante Moteur physique

Nous travaillons sur la composante Moteur physique, c'est-à-dire la représentation des différents objets qui interagissent dans le jeu. Notre rôle sera de gérer le mouvement des objets et leurs intéractions les uns avec les autres, il faudra pour cela détecter leur position et introduire la collision, mais également gérer le terrain ainsi que l'état des objets (principalement, pour déterminer si le joueur est vivant ou mort). Enfin, nous pourrons si les objectifs sont atteints dans le temps imparti réfléchir à intégrer un système de bonus et d'events au jeu.

2 Description des intéractions avec les autres composantes

2.1 Avec la composante Moteur graphique

La composante Moteur graphique a besoin de notre composante pour récupérer les objets (coordonnées et état), leur donner la forme qu'ils auront dans le jeu et les afficher à l'écran.

2.2 Avec la composante Interface utilisateur

Nous avons aussi besoin de la composante Interface utilisateur : c'est grâce à elle que nous allons récupérer les commandes des utilisateurs nécessaires au déplacement. De plus, elle a besoin de nous pour mettre à jour les objets constituant le jeu, et nous devons également récupérer les données des joueurs créés et gérer par cette interface afin de les implémenter dans le système de score.

2.3 Avec la composante Moteur réseau

Enfin, nous comptons sur la composante Moteur réseau pour gérer la possibilité de jouer à plusieurs, à laquelle nous appliquerons donc les fonctionnalités prévues (collisions avec les autres joueurs, positions des autres joueurs, et, si possible, affectations de bonus/malus à soi-même ou aux autres joueurs).

3 Formalisation des intéractions

3.1 Classes de données

Player() : écrite par IU
Game
{
Player[] Participants, Field Terrain
}: écrite par MP
Field
{
Case caseMap[L][l]
} : écrite par MP
Case
{
int coord Player app
} : écrite par MP
Snake
{
int IDplayer Case head(int coord) Case direction(int angle))
} : écrite par MP, lue par MG,IU

3.2 Méthodes à implémenter

-Spawn() : Fait apparaître les serpents à distance suffisante des bords. Invoquée par IU
-Move() : Reçoit les commandes via l'IU et les exécute. Invoquée par IU
-Collision() : Tue le joueur s'il touche un bord de la map ou un autre serpent. Invoquée par IU
-newField() : Reset la map. Invoquée par MG (reset de la map)
-findSnake() : Localise les autres serpents sur le terrain. Invoquée par MR
-getPositionX() : Récupère la coordonnée en X du joueur
-getPositionY() : Récupère la coordonnée en X du joueur
-Optionnel : affectXX (XX nom du bonus/malus) : Modifie le fonctionnement du jeu. Invoquée par MG,MR

3.3 Méthodes à invoquer

-drawObject() : Dessine un objet. Écrite par MG
-displayField() : Affiche le terrain. Écrite par MG
-displaySnake() : Affiche le serpent. Écrite par MG
-getCommande() : Récupère les commandes rentrées par le joueur. Écrite par IU
-displayResult() : Affiche le résultat de la partie. Écrite par IU
-displayStats() : Affiche les statistiques des joueurs. Écrite par IU