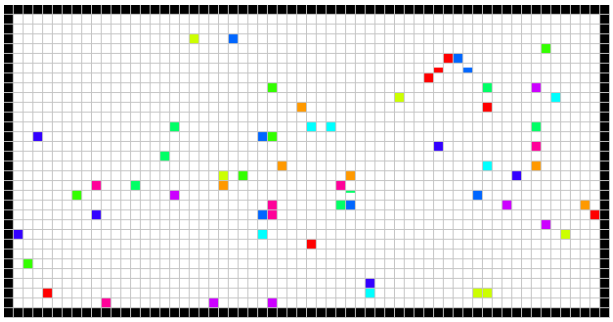
TP2 : un casse-brique en Java RMI

PAR : Programmation d'Applications Réparties

Troisième année



★ **Exercice 1. Prise en main.**

Vous trouverez dans le dépot (et à l'adresse `http://www.loria.fr/~quinson/teach-PAR.html`) un fichier nommé `CasseBrique.java`. Il s'agit d'une version préliminaire d'un petit jeu de casse-brique. L'objectif est de modifier ce programme de façon à expliciter le modèle MVC (model-view-controller) et faire communiquer les différentes parties par RMI.

▷ **Question 1.** Listez les différentes classes de ce projet, et indiquez leur objectif.

▷ **Question 2.** Faites un diagramme UML de l'application existante.

★ **Exercice 2. Passage à RMI.**

▷ **Question 1.** Modifiez l'application pour que les communications entre les composants se fasse par le biais de Java RMI. Vous serez amenés entre autres à expliciter les interfaces des différents composants et placer chaque classe dans un fichier séparé. Il sera également nécessaire de mettre en œuvre le concept de callback (pour que le ground puisse informer le viewer des changements).

▷ **Question 2.** Ajoutez la possibilité d'avoir plusieurs viewers du même ground.

★ **Exercice 3. Ajout d'une raquette.**

Modifiez l'application pour ajouter une raquette. Une façon de procéder est de :

- ajouter une nouvelle constante à la classe `Ground` pour indiquer la présence de la raquette ;
- ajouter un champ à la classe `Ground` pour indiquer la position courante de la raquette, ainsi qu'une constante pour indiquer sa largeur (de valeur 5) ;
- ajouter une méthode `moveRaquette` au `Controller` pour demander à déplacer la raquette. On peut s'inspirer de ce qui est fait pour déplacer les balles.
- définir une classe `RaquetteSouris` étendant `MouseMotionAdapter` pour recevoir les commandes de l'utilisateur avant de les transmettre au `Controller`. Il suffira alors d'appeler `addMouseMotionListener()` avec une nouvelle `RaquetteSouris` pour voir la raquette bouger avec la souris. On n'oubliera pas de faire des appels aux méthodes `paint()` appropriées pour que les déplacements de raquette soient effectivement affichés.

★ **Exercice 4. Sortie de balle.**

Une balle qui sort du jeu en passant à côté de la raquette doit terminer : Le processus associé doit finir sur requète du `Controller`. Quand il n'y a plus de balle en jeu, le `Controller` doit arrêter la partie.

★ **Exercice 5. Casser des briques.**

Rajoutez des cases briques qui se désintègrent quand elles sont frappées par une balle, et un décompte du nombre de briques cassées.

★ **Exercice 6. Multi-joueurs.**

▷ **Question 1.** Faites en sortes d'avoir deux raquettes sur le terrain : l'une en haut et l'autre en bas. Par défaut, les joueurs collaborent à l'objectif commun : détruire les briques placées au centre. Ils perdent tous les deux quand toutes les balles sont sorties.

▷ **Question 2.** Une variante consiste à faire deux types de balles. Celles du premier type sont celles du joueur du haut. Elles rebondissent sur le bord inférieur comme s'il s'agissait d'un mur, mais sont perdues quand elles tapent sur le bord supérieur à coté de la raquette. Les briques cassées ne rapportent de points qu'au propriétaire de la balle en question.

▷ **Question 3.** Toute autre extension est la bienvenue.