Projet programmation 2
Sujet – 3ème partie

Vincent Lafeychine – Stefan Schwoon

1. Jeu en réseau

Pour cette dernière partie, il est attendu que votre projet se scinde en deux programmes :

- un programme client qui s'occupe de la partie visuelle du jeu, et ;
- un programme serveur qui s'occupe de la logique du jeu, et étant en capacité de gérer plusieurs clients.

Pour communiquer sur le réseau, un *snippet* vous a été présenté qui utilise les notions du TP réseau du cours d'Architecture et Système. Vous êtes libre de modifier ces *snippets*.

Pour envoyer des messages à travers le réseau, vous allez vous utiliser Protocol Buffers à l'aide de ScalaPB. Vous trouverez le protocole décrit en Section 3, intégrant les interactions demandées lors du dernier rendu.

Le serveur dispose de deux modes :

- un mode « salle d'attente », pour permettre à vos clients de rejoindre et d'être synchronisés, et ;
- un mode « jeu », qui représente une journée de jeu.

Le serveur doit accepter plusieurs clients, où les clients sont en coopération avec des ressources communes. En soit, chaque client représente un cuisinier durant une même journée dans un même restaurant.

Le serveur agissant comme une autorité, les clients doivent attendre la réponse du serveur avant de montrer le résultat des actions du joueur. Par exemple, lorsqu'un joueur prend un ingrédient mais qu'une latence réseau fait que le stock d'ingrédient du serveur est vide, le client peut se voir refuser la requête par le serveur.

Il n'est pas demandé d'implémenter des *rollbacks* ou toute autre forme de re-synchronisation. Il suffit d'attendre la réponse positive du serveur ou l'événement équivalent, pour pouvoir afficher le résultat au joueur.

Dans le protocole, les identifiants représentent :

- un identifiant incrémental dans le cas de `dishId`, ou ;
- un nombre entre 1 et 6 dans le cas de `seatId`, ou ;
- la i$^{\text{ème}}$ ligne des tableaux présents dans le sujet de la phase 2 (et 0 pour un plat douteux).

2. Modalités de rendu

Le projet sera à rendre le 26 avril à 23h59. Votre projet devra contenir un fichier `DEBRIEF.md` de *quelques* lignes qui contiendra vos retours d'expérience, notamment si vous avez rencontrés des difficultés.

Une soutenance est prévue le 12 mai de 9h à 12h, au cours de laquelle vous présenterez votre jeu et de votre code, où une discussion plénière aura lieu pour discuter de votre implémentation.

3. Protocole

Dans ce protocole, trois types d'échanges sont possibles :

- une requête (`Request`) est le seul échange possible du client vers le serveur ;
- une réponse (`Response`) est le résultat envoyée par le serveur vers le client de la requête ;
- un événement (`Event`) est une notification envoyée par le serveur vers les clients du mode correspondant.

3.1. Requête

Une requête est l'un des éléments suivants, lorsque le serveur est en mode « salle d'attente » :

`JoinRequest` Rejoindre la salle d'attente pour accéder à la prochaine partie de jeu.

- `string protocolVersion`: version du protocole, actuellement `0.1.0` ;
- `string playerName`: nom du joueur.

`LeaveRequest` Quitter la salle d'attente, interrompant la connexion avec le serveur.

`DayStartRequest` Fermer la salle d'attente et démarrer la partie, changeant le serveur en mode « jeu ».

Une requête est l'un des éléments suivants, lorsque le serveur est en mode « jeu » :

`LeaveRequest` Quitter la partie de jeu en cours, interrompant la connexion avec le serveur.

`IngredientTakeRequest` Prendre un ingrédient du stock commun, pour le tapis en bambou du joueur.

- `int32 ingredientId`: Identifiant de l'ingrédient.

`IngredientPutBackRequest` Prendre un ingrédient du tapis en bambou du joueur, pour le stock commun.

- `int32 ingredientId`: Identifiant de l'ingrédient.

`DishMakeRequest` Réalisation de la recette avec les ingrédients présents sur le tapis en bambou du joueur.

`DishJunkRemoveRequest` Enlever une assiette contenant un plat douteux.

- `int32 dishId`: Identifiant de l'assiette.

`DeliveryCallRequest` Appeler une livraison pour réapprovisionner le stock en ingrédients.

- `int32 ingredientId`: Identifiant de l'ingrédient.

`SeatCleanRequest` Nettoyer un emplacement après que le client soit parti.

- `int32 seatId`: Identifiant du siège.

3.2. Réponse

Une réponse est l'un de deux cas suivants :

- si la requête est acceptée, la réponse est un **`SuccessResponse`** ne contenant aucun champ.
- si la requête est refusée, la réponse est un **`ErrorResponse`** contenant la raison du refus.

Lorsqu'une requête est acceptée, le client est assuré de recevoir un événement correspondant à sa requête.

Les raisons de refus sont les suivants :

`UNSUPPORTED_VERSION` Le client et le serveur n'ont pas une version compatible.

`UNSUPPORTED_REQUEST` Le serveur n'est pas en capacité de traiter la requête du client.

`INVALID_MODE` La requête n'est pas réalisable dans le mode actuel du serveur.

`INVALID_REQUEST` La requête est incohérente avec l'état du jeu.

`UNKNOWN_ID` La requête contient un identifiant inconnu.

`PLAYER_NAME_TAKEN` Un autre joueur possède déjà ce nom.

3.3. Événements

Un événement est l'un des éléments suivants, lorsque le serveur est en mode « salle d'attente » :

`DayWaitingEvent` Événement *unicast* indiquant les informations de la prochaine partie de jeu.
- `int32 day`: Numéro de la journée.
- `int32 duration`: Durée de la journée, en secondes.
- `int32 balanceGoal`: Objectif du compte à la fin de la journée.
- `int32 balanceCurrent`: Montant du compte en début de journée.
- `repeated int32 recipes`: Liste des recettes disponibles pour cette journée.

`PlayerJoinedEvent` Un joueur a rejoint la salle d'attente.

`PlayerLeavedEvent` Un joueur a quitté la salle d'attente.

`DayStartedEvent` La salle d'attente ferme et la partie démarre, le serveur est en mode « jeu ».

Un événement est l'un des éléments suivants, lorsque le serveur est en mode « jeu » :

`DayEndedEvent` La partie se termine et la salle d'attente ouvre, le serveur est en mode « salle d'attente ».
Il est attendu que les clients souhaitant continuer le jeu effectuent de nouveau une requête **`JoinRequest`**.

`CustomerEnteringEvent` Un client entre dans le restaurant.
- `int32 seatId`: Identifiant du siège sélectionné par le client.
- `int32 recipeId`: Identifiant de la recette sélectionné par le client parmis les recettes disponibles.

`CustomerEatingEvent` Le client prend une assiette correspondant à la recette souhaitée.
- `int32 seatId`: Identifiant du siège du client.
- `int32 dishId`: Identifiant de l'assiette sélectionnée par le client.

`CustomerLeavingEvent` Le client quitte le restaurant, après avoir fini de manger ou par impatience.
- `int32 seatId`: Identifiant du siège du client.
- `int32 amountPaid`: Argent récolté.

`CustomerPatienceEvent` Le client a perdu 1/5 de sa patience totale.
- `int32 seatId`: Identifiant du siège du client.

`IngredientTakenEvent` Un joueur a pris un ingrédient du stock commun, pour son tapis en bambou.
- `int32 ingredientId`: Identifiant de l'ingrédient.

`IngredientPutBackEvent` Un joueur a pris un ingrédient de son tapis en bambou, pour le stock commun.
- `int32 ingredientId`: Identifiant de l'ingrédient.

`DishMadeEvent` Un joueur a réalisé une recette, et l'assiette est déposée sur le tapis en bambou.
- `int32 dishId`: Identifiant de la nouvelle assiette.
- `int32 recipeId`: Identifiant de la recette réalisée.

`DishJunkRemovedEvent` Un joueur a enlevé une assiette contenant un plat douteux.
- `int32 dishId`: Identifiant de la nouvelle assiette.

`DeliveryCalledEvent` Un joueur a appelé une livraison pour réapprovisionner le stock en ingrédients.
- `int32 recipeId`: Identifiant de l'ingrédient commandé.

`DeliveryDoneEvent` La livraison est terminée.

`SeatCleanedEvent` Un joueur a nettoyé un emplacement après qu'un client soit parti.
- `int32 seatId`: Identifiant du siège.

Lorsqu'un événement est à l'origine d'un joueur, l'événement comporte également le nom du joueur.