l'école d'ingénierie informatique

EPSI BORDEAUX - I1 EISI

Module Atelier - Architecture applicative (9/9)

Code Module	Durée	Titre Diplôme	Bloc de Compétences	Promotion	Auteur
ARCE842 - DEVE702	20h	EISI / RNCP 35584	Concevoir & Développer des solutions applicatives métiers	2023/2024	Julien COURAUD

15. Tips: Dévéloppement du mode "Vs BOT"

Le mode de jeu "Vs BOT" est prévu dans l'architecture applicative.

Nous avons déjà préparé le workflow utilisateur côté Front-end (bouton du menu vers un écran: 'vs-bot-game.screen.js'. On imagine déjà facilement récupérer les composants React du mode de jeu en ligne précédemment développé car ceux-ci sont indépendants et ne reçoivent que des informations issues du traitement serveur afin d'uniquement les afficher.

De la même manière, côté WebSocket Server, beaucoup de méthodes du `GameService` peuvent être réutilisées, notamment toutes les actions sur le `gameState` lors d'une action utilisateur.

La grande différence résidera dans la gestion du timer, et du tour par tour. Vous l'aurez compris, l'essentiel des nouveaux développements auront lieu au niveau du 'index.js'.

Méthodologie suggérée

Afin de ne pas vous perdre dans les étapes à réaliser, voici une suggestion de méthodologie afin d'avancer sereinement dans le développement du nouveau mode de jeu.

1. Création et déroulé d'une partie

- Lorsque l'utilisateur clique sur le bouton "Vs BOT" du menu d'accueil, la partie commence directement (pas besoin de le mettre dans une file d'attente).
- Le joueur possède un timer mais celui du BOT n'est pas forcément nécéssaire, cependant c'est important de montrer au joueur les différents lancers de dés, le choix de la combinaison et la case cliquée par le BOT. Vous pouvez utiliser le `setTimeout()` ou le `setInterval()` configuré à quelques secondes.
- L'utilisateur interragit avec l'écran de jeu de même manière que lors d'une partie en ligne.

2. Moteur de jeu

- Vous pouvez, à priori, continuer à utiliser la variable globale `games[]` pour stocker la partie "Vs BOT" en cours, auquel cas, vous aurez besoin d'un flag dans le `gameState` pour différencier les types de parties. Vous pouvez également créer une nouvelle variable `botGames[]` par exemple, dans laquelle vous ajouterez les parties "Vs BOT" en cours.

- Nous aurons donc à créer une nouvelle méthode `createBotGame()`, dans la même structure que notre précédent `createGame()`.

- Nous utiliserons les méthodes existantes du `gameService` pour initialiser et manipuler l'objet de jeu `gameState` et également celles qui envoient des objets à l'interface utilisateur.

- Les nouvelles méthodes qui doivent apparaître dans ce service seront liées aux choix du BOT suites aux différents lancers, au verrouillage de certains dés, aux combinaisons et cases disponibles. Vous n'êtes pas obligé dès le début de vos développement d'implémenter entièrement les algorithmes de choix faits par le BOT. Vous pouvez, pourquoi pas, partir sur des choix 'random', afin de préparer seulement la structure de votre code dans un premier temps.

- L'essentiel des nouveaux développements interviendront donc au sein du 'index.js':

    - Dans votre méthode de création de partie.
    - Dans les 'listeners' WebSocket qui adapteront donc la manipulation de l'objet `gameState` en fonction du type de partie.
    - Le Web SocketServer pourra utiliser les mêmes clés WebSocket pour emettre des informations à la vue.

- Une fois votre moteur de jeu finalisé pour ce nouveau type de partie, vous vous pencherez sur l'affinage de vos méthodes de choix faits par le BOT afin que l'expérience de partie 'Vs BOT' propose un certain challenge.

3. Définir une stratégie de jeu pour le BOT

Il n'est pas chose facile de transcrire en instruction algorithmique des choix stratégiques liées à une partie de Yam Master.

Globalement, lorsque vous faites une partie, vous pensez à des stratégies dans votre tête pour optimiser vos chances de victoires:

- Au premier lancer, quels dés dois-je verrouiller afin de m'assurer de réaliser une combinaison ?

- Lorsque des combinaisons sont disponibles, lesquelles sont intéressantes pour moi en fonction de l'état de la grille de jeu ?

- Quelle case de la grille est-elle plus intéressante pour moi ?

- etc etc …

L'enjeu est donc de définir des grandes règles stratégiques (à vous de voir mais on peut imaginer deux modes offensif/defensif ou jouer pour une ligne complète ou pour gagner aux scores, etc..) qui seront ensuite affinées par des choix 'heuristiques d'évaluation'.

Pour un jeu de Yam Master, voici quelques idées d'heuristiques d'évaluation que vous pourriez utiliser pour évaluer la qualité d'une position de jeu :

- Score potentiel : évaluez le score potentiel que le joueur pourrait obtenir en plaçant ses pions dans une certaine configuration. Prenez en compte les différentes combinaisons possibles et leur valeur en points.

- Contrôle du territoire : considérez la manière dont la position des pions affecte le contrôle du territoire sur la grille. Par exemple, avoir des pions dispersés sur la grille peut empêcher l'adversaire de former une ligne complète.

- Blocage des lignes adverses : cherchez à bloquer les lignes potentielles de l'adversaire en plaçant vos pions de manière stratégique. Cela peut impliquer de placer des pions dans des positions qui rendent difficile pour l'adversaire de compléter une ligne.

- Flexibilité future : évaluez la flexibilité des placements de pions en fonction des futurs lancers de dés. Par exemple, préférez les positions qui permettent de s'adapter à différentes combinaisons disponibles sur la grille.
- Distance par rapport à la victoire : évaluez la proximité de chaque joueur par rapport à la victoire en considérant le nombre de pions nécessaires pour compléter une ligne et la position des pions déjà placés.
- Évaluation dynamique : ajustez vos évaluations en fonction de l'évolution du jeu, en prenant en compte les actions de l'adversaire et en adaptant votre stratégie en conséquence.