ÉCOLE
CENTRALELYON

S8 ELC-D3 - Applications Web

Application Web : jeu d'Othello

Auteurs :
Risa KAMBE
Emma DELAROZIERE

Enseignant :
M. Daniel MULLER

Version du
31 mars 2021

Table des matières

Introduction

On s'intéressera dans ce présent rapport à la programmation d'une application web d'Othello. Othello est un jeu de stratégie à deux joueurs, aux règles facilement assimilables mais tout de même assez complexes pour être un sujet de programmation intéressant.C'est ainsi l'opportunité de tester sur un projet plus conséquent Node js et des modules tels qu'express et socket.io.

Express est un module permettant de créer une application web. Socket.io est destiné à la création de websockets assurant l'interaction en continu entre le serveur et chaque client. Il nous permettra ici de mettre en place un système de salons pour que plusieurs parties puissent avoir lieu simultanément sans interférer les unes avec les autres.

1 Cahier des charges

— On souhaite organiser une partie de jeu othello entre deux clients. Il faut donc pour cela utiliser des websockets pour assurer la communication entre le serveur et les deux clients. Pour que le serveur puisse reconnaître ses joueurs, les sockets des clients ne doivent jamais être déconnectés du serveur. Une fois connecté, le client ne doit pouvoir interagir avec le serveur qu'à l'aide des commandes socket.io, au risque sinon de déconnecter le websocket.
— Pour que le joueur puisse facilement prendre en main son système, il faut une interface graphique claire.
— Si le joueur ne connaît pas les règles, il peut accéder immédiatement à une source expliquant ces règles
— Pour que plusieurs parties puissent être gérées simultanément par le serveur, on doit mettre un place un système de salons de jeu. Les joueurs peuvent choisir quel salon rejoindre à l'aide d'un identifiant de salon. La partie ne doit être lancée qu'à l'arrivée du second joueur dans le salon.
— On souhaite intégrer un chat une fois la partie lancée. Le client doit pouvoir correctement distinguer qui est l'expéditeur d'un message donné. Seules deux personnes dans un même salon devraient avoir le droit de communiquer.
— À des fins de débug et pour vérifier le bon déroulement d'une partie, il faut pouvoir surveiller du côté serveur les échanges de données. On s'attachera donc à afficher autant que possible et de façon la plus intelligible qu'il soit les informations circulant entre le serveur et les clients dans la console.

1.1 Description du code

1.2 Principe de fonctionnement général

— On met d'abord notre identifiant sur la page de login.
— Ensuite, on entre l'identifiant de salon. Si le salon n'existe pas, il est alors automatiquement créé avec l'identifiant donné.
— S'il y a pas d'autre joueur, on attend jusqu'à ce que l'autre personne entre dans la salle. Si quelqu'un entre dans la salle, le message apparaît sur la page de la première personne.

— On commence le jeu.

L'event listener pour rejoindre le salon étant le même pour les deux joueurs, pour savoir si le deuxième joueur est connecté, nous avons utilisé un système "ping-pong". Lors de la connexion du deuxième joueur, on envoie automatiquement aux personnes présentes dans le salon un message de bienvenue, qui n'est reçu par un client que si un joueur est déjà connecté au salon. De la même façon, on peut grâce à cela envoyer directement un événement dans le salon, et si un joueur est déjà présent, le client déjà connecté renvoie un pong au serveur indiquant que la partie peut commencer.

1.3 Les problèmes rencontrés

Nous avons rencontré trois problèmes majeurs que nous n'avons pas réussi à résoudre.

— Tout d'abord la mécanique du contrôle des tours n'est pas fonctionnelle. Nous souhaitions passer dans le client une variable globale indiquant la couleur permise et la comparer avec celle du joueur, et couper la réponse de l'événement mouseclick du plateau si ces deux variables n'étaient pas égales afin d'empêcher un joueur de jouer quand ce n'est pas son tour. La variable indiquant quelle couleur joue un client est générée par défaut comme égale à -1 (la variable associée au plateau de jeu est initialisée à 1), et doit être modifiée pour un des deux joueurs au début de la partie. Cependant, en testant en conditions réelles, aucun des deux joueurs ne semble pouvoir jouer au premier tour avec cette méthode. Si on initialise la couleur utilisable par les joueurs et la couleur autorisée par le plateau à la même valeur, on constate que les joueurs peuvent poser des pions. Il s'agit donc d'un problème de callback : le joueur dont la couleur doit être mise à jour n'a pas cette modification en temps voulu, ce qui fait que les joueurs ont les mêmes permissions au premier tour. Ceci est problématique car on aurait besoin d'éditer la couleur du joueur au cours de la partie, notamment pour passer le tour. Ceci nous empêche aussi de tester les fonctions liées à la victoire.

— Malgré l'envoi de coordonnées de jeu à l'autre joueur et l'utilisation de la même méthode pour poser un pion, les clients n'ont pas le même résultat en fin de tour. La source de ce bug reste indéterminée.

— Malgré la limite en hauteur imposée à la division du chat avec une option pour pouvoir faire défiler la division, le chat ne semble pas limité en taille quand on poste de nombreux messages.

Conclusion

Nous avons pu ainsi obtenir le squelette d'un jeu othello en ligne. En dépit des problèmes techniques rencontrés lors du développement de l'application, ce projet a amélioré notre compréhension du fonctionnement de socket.io et de l'utilisation de l'objet canvas.