École Centrale Lyon

UE ELC
Rapport

Jeu Domino

Élèves :
Thomas Hébrard
Edouard Lajouanie

Enseignant :
Daniel Muller
René Chalon

Table des matières

1 Présentation du Jeu

Les dominos sont un jeu de société d'origine chinoise, utilisant 28 pièces (dans le cas d'un jeu « double-six »). On y joue généralement à deux, trois ou quatre personnes. Comme avec les cartes, il existe de nombreuses variantes de jeu. La version du jeu considérée sera le jeu "avec talon".

1.1 Règles

Les règles sont vastes et simples :
- Étaler les dominos sur la table, points cachés. Distribuer le même nombre de dominos à chaque joueur, en laissant un talon de quelques pièces ;
- le premier joueur pose son plus fort domino, son voisin pose à l'une des extrémités un domino dont l'une des parties a le même nombre de points ;
- chaque joueur joue à son tour et l'on constitue ainsi une chaîne ;
- lorsqu'un joueur n'a pas de domino qui convient, il pioche une pièce du talon et passe son tour ;
- le vainqueur est celui qui place le dernier de tous ses dominos.

2 Les techs et stacks utilisés

Pour développer ce programme, nous avons utilisé plusieurs technologies. Premièrement, l'éditeur de texte **Visual Studio Code** a été utilisé, relié à **Github**, ce qui nous a permis de travailler avec github facilement, sans utiliser le terminal (sauf pour lancer le serveur).

Grâce à Node.js, nous avons bénéficié d'une librairie très dense. Nous avons aussi utilisé le gestionnaire de paquet de **Node.js *npm* (Node Package Manager)**. La librairie ***socket.io*** issue de la bibliothèque de Node.js utile les applications Web en temps réel, ce qui nous permet de gérer les différents utilisateurs.
Le code a été testé sur le navigateur **Brave Brower**, qui utilise Google Chrome.

3 Présentation de la structure du programme

(*Le code a été commenté en détail pour une meilleure compréhension*)

Le programme se structure essentiellement en 2 partie : une partie html et une partie Javascript, réparties sur 2 fichier : app.js et index.html. Le fichier app.js gère la partie **serveur**, et fait appel à index.html qui gère la partie **client**. La partie html comporte un script javascript pour faire appel au serveur.

3.1 app.js : le serveur

Le serveur, codé en javascript est constitué de 3 parties :

- Une première partie qui créé les classes (POO) utiles pour le jeu : une classe joueur et une classe Domino (cf. Annxes)

— Une deuxième partie au sein de laquelle sont créées de multiples fonction.
— et une dernière partie qui gère le client grâce aux écoutes des sockets, ou à leurs émissions.

Le serveur est lancé en local, sur le port 8000 pour l'indique la ligne de code suivante :

```
1 serv.listen(8000);
```

3.2 index.html : Le client

Ce fichier est séparé en 2 parties : Une partie codée en *html* et une autre en *javascript* au sein d'une balise *<script>*. Dans la partie html est déclaré toutes les div utiles qui seront exploitées avec le script javascript. Dans la partie javascript, il y a trois parties distinctes.

— Une partie qui gère l'inscription des joueurs,
— Une partie qui gère la messagerie instantanée,
— Une partie qui gère le jeu, et l'affichage des dominos

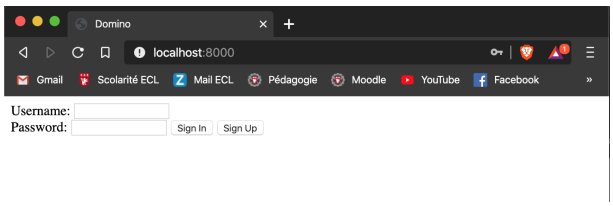
4 Déroulé du jeu

Pour lancer le serveur, il faut lancer grâce à node le fichier qui gère le serveur, donc app.js. Avec le terminal au fichier du code et entre Jeu Dominonode app.js.

4.1 Lancement de l'application

Une fois le serveur lancé, il aura fait appel à index.html, et en ouvrant *localhost :8000* sur le navigateur, le menu suivant se présente :

FIGURE 1 – Page d'accueil

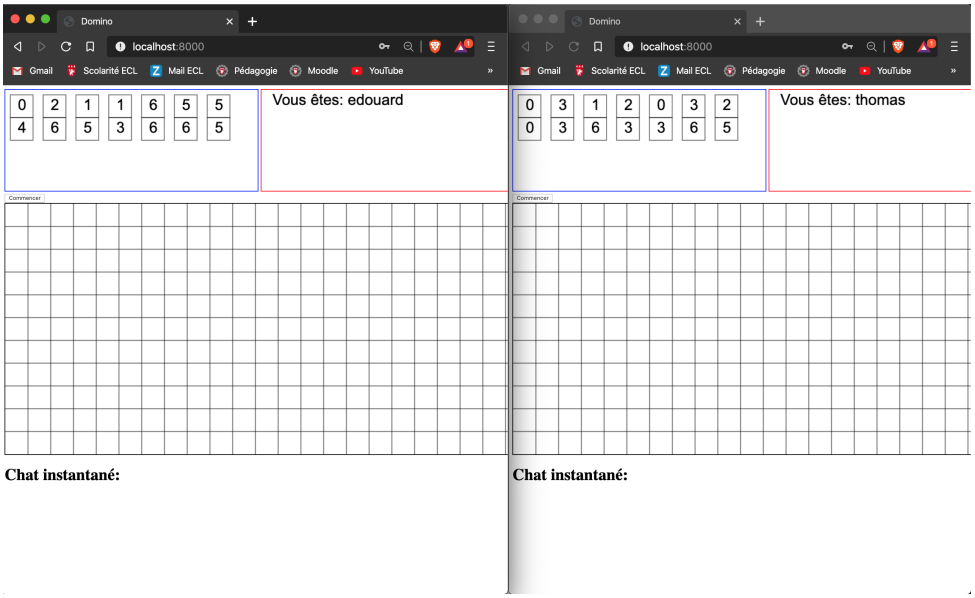


Le joueur peut soit se connecter si il a déjà créé un compte, où si il est déjà enregsitré dans la base de donnée (il y a déjà un compte *thomas* et *edouard*), soit se créer un compte avec SignUp.

Le nombre de joueur a été limité à 2 pour plus de facilité. Si 2 joueurs différents se connectes, le jeu pourra commencer. Si un joueur est déjà connecté et qu'un autre essaye de se connecter avec ce compte, il ne pourra pas. Si les joueurs se trompent de mot de passe ou de username, ils devront recommencer.

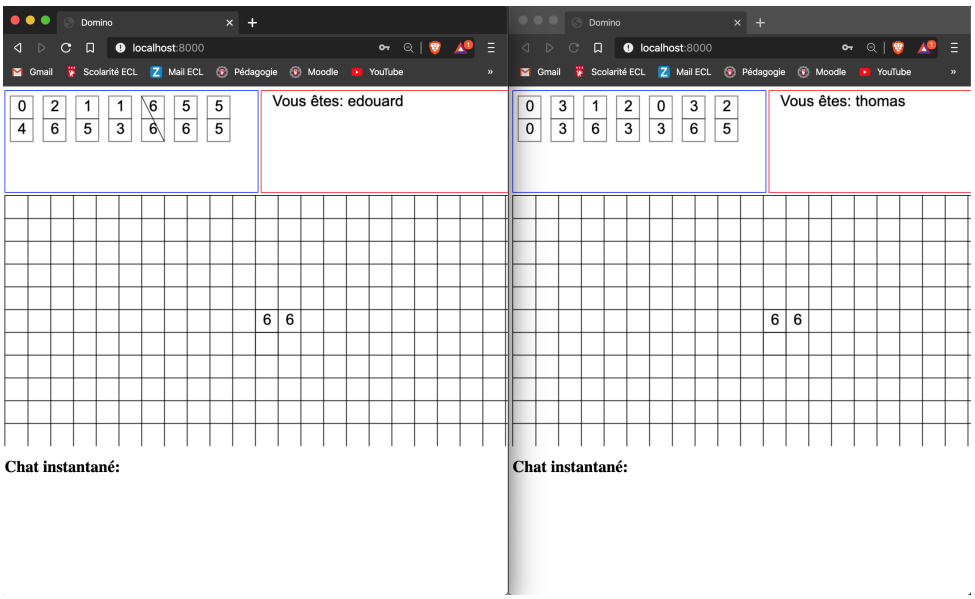
4.2 Lancement du jeu

Une fois un joueur connecté (ici, pour l'exemple, *thomas* et *edouard* se sont connectés, leurs informations sont enregistrées de base dans la base de données dans app.js), il arrivera sur cette page.



On peut observer que chaque joueur est connecté sur un compte différent, et qu'ils possèdent chacun un deck de domino de départ. Ils sont uniques et randomisés.

On observe le terrain de jeu, ainsi que le chat instantané. Le bouton *commencer* va permettre de lancer le jeu et d'afficher le premier domino. Le premier domino est choisi par le serveur selon une règle simple : C'est le joueur possédant le plus grand double qui commence, avec justement ce double. Si aucun joueur ne possède de doubles, il faut piocher dans les dominos restant jusqu'à l'obtention d'un double.



Le premier domino est placé au centre du terrain.

4.3 Chat instantané

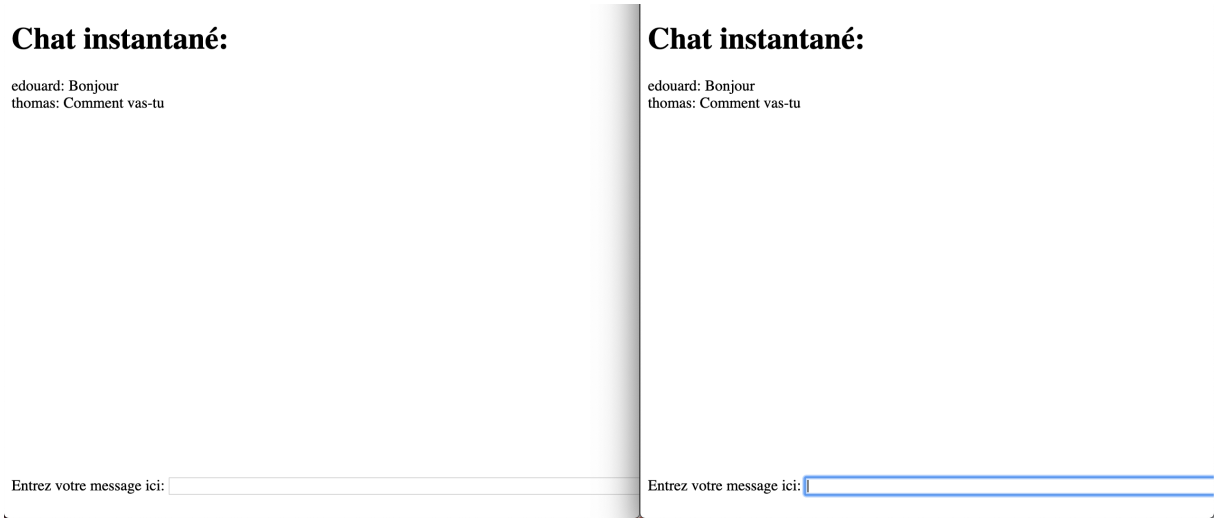
Concernant le chat instantané, il repose sur un principe assez simple. Une *div* et un formulaire ont été créé dans la partie html :

```
<div id="chat-text" style="width:1000px;height:500px;overflow-y:scroll">
            <div>
                <h1>
                    Chat instantan :
                </h1>
            </div>
        </div>
        <!-- Le forumulaire pour faire int ragir les joueurs avec le chat-->
        <form id="chat-form">
            <label for="chat-input">Entrez votre message ici:</label>
            <input id="chat-input" type="text" style="width:500px"></input>
        </form>
```

Pour modifier le texte du chat, dès qu'un joueur entrera du texte, toute la *div* sera modifiée, et on accumulera du texte.

```
socket.on('addToChat', function(data){
        chatText.innerHTML += '<div>' + data + '</div>';
    });
```

De plus, chaque joueur sera identifié sur le chat. Le chat se met à jour instantanément.



5 Difficultés rencontrées

5.1 Difficulté principale

La grande difficulté de ce projet résidait dans les règles du jeu. En effet, contrairement à un jeu d'échec qui est très binaire (chacun joue à son tour), dans le jeu des dominos, il y a plusieurs interruptions, un système de pioche, de rajoute de domino dans le deck, de passage de tour etc. Cette désynchronisation est compliquée à mettre en place.

5.2 Mise à jour différenciée

Une des difficultés résidait dans le fait que sur la page de chaque client, il y a des parties personnelles (le deck, la partie information en rouge) et des parties publiques, qui doivent être mise à jour en même temps sur tous les clients, comme le terrain de jeu et le chat instantané. Cette spécificité peut-être gérée avec les émissions et réception des sockets, mais aussi avec la fonctionnalité *socket.broadcast.emit*.

6 Pistes d'amélioration

Une amélioration du code possible aurait été de densifier les actions de la classe Domino. En effet, beaucoup d'actions sont exécutées sur les dominos, il aurait été peut-être plus agréable de tout centraliser en méthode au sein de la classe Domino. Par manque de temps, nous n'avons pas pu finir le projet et donc il était compliqué de voir toute l'action d'un domino.

Il aurait aussi été intéressant de compléter la feuille de style *.css* pour ajouter de l'esthétique.