Rapport Web Avancé

2019-2020
Semestre 5

Guillaume MAGNIADAS
Etienne PENAULT

Compte Rendu

Nous avions pour projet de réaliser un jeu comme Tetris mais avec une composante coopérative, c'est à dire de scinder la partie de Tetris en X (Égale au nombre de joueurs) tableaux fusionnés où les pièces de Tetris (Tétromino) ne seraient contrôlables à l'intérieur de ces derniers seulement par le joueur N associé.

Pour ce faire, nous avions besoin d'un serveur capable d'échanger en temps réel des informations. Nous avons donc utilisé ***Node.js*** pour le serveur couplé avec la librairie ***socket.io*** qui permet d'ouvrir et de gérer des web sockets très facilement.

Concernant le client, nous n'utilisons que du javascript classique, avec la partie cliente de ***socket.io***.

Nous sommes assez contents du résultat obtenu, nous avons un jeu jouable et stable, compatible aussi bien sur ordinateur que sur smartphone.

Par-contre les contrôles sur smartphone seraient à travailler pour qu'ils soient plus fluides et plus intuitifs car pour le moment ils ne sont pas forcément très simple d'utilisation.

Il y a quelques features dont nous sommes assez déçus de ne pas avoir intégré par manque de temps :

- De meilleurs contrôles pour les utilisateurs tactiles.
- Une petite campagne, c'est à dire avoir quelques niveaux contre des boss qui seraient des IA capables d'attaquer les équipes de joueurs en envoyant des blocs pleins sur le terrain des joueurs, cela aurait pu renforcer l'aspect coopération.
- Un système de ping, pour avertir les joueurs que l'on aimerait avoir la pièce par exemple, cela aurait été plus simple que devoir écrire dans un chat pour avertir le joueur alors que le jeu est en temps réel.
- Une réserve pour stocker une pièce d'avance par joueur.
- Une meilleure gestion des écrans des joueurs, au début nous voulions faire une interface qui se limite à maximum 3 parties du tableau, la nôtre au centre, et entouré de nos deux voisins de partie de chaque côté, et de ce fait rendre « invisible » pour nous les tableau des autres joueurs, cela aurait pu rendre la partie plus mystérieuse mais aussi beaucoup plus compliquée et un système comme cela n'aurait pas été jouable sans les ping pour avertir du besoin d'une pièce.

Architecture du Programme

Aussi bien le client que le serveur est enfaite une classe application, qui est le coeur du programme, cette classe fait ensuite appel à d'autres classes suivant l'architecture ***MVC***.

Client :

Le client possède donc la classe Base depuis laquelle il initialise d'abord la classe MVC qui commencera avec le **model**, la ***vue*** et le **contrôleur** de l'écran de base du jeu, celui de connexion aux salons de jeu.

Ce premier écran gère la connexion des joueurs aux salons de jeu, on peut y rentrer son pseudo et le numéro du salon que l'on veut rejoindre. Il est aussi affiché en bas les ***Hall Of Fame***, qui est un tableau qui enregistre les 5 meilleurs scores sur le jeu. (Ce tableau est enfaite un fichier ***json*** stocké par le serveur et envoyé au client quand celui ci finit d'initialiser sa première classe MVC).

Quand il essaie de se connecter, si le salon qu'il essaie de rejoindre n'existe pas encore ou n'est pas encore en partie, il le rejoint et le client décharge le MVC de l'écran de connexion et charge à la place le MVC de jeu, il charge également le MVC du chat.
Si le salon qu'il essaie de rejoindre est déjà en partie, il ne se passera rien.

Le MVC de jeu possède un **canvas** principal qui affichera le terrain de jeu mais aussi un plus petit **canvas** qui affichera la pièce suivante. Sur ce MVC les entrées utilisateurs aussi bien tactiles que celles du clavier seront écoutés pour pouvoir gérer les déplacements de la pièce qui est entrain de tomber, le **contrôleur** s'occupe d'envoyer toutes ces informations au client.

Il y a aussi un chat qui est ouvert à l'entrée dans le son qui relie les joueurs d'une même partie. (Note : il est caché de base mais peut s'ouvrir en appuyant sur la touche C ou sur le bouton « Chat » dans l'angle haut gauche). Ce dernier est composé d'un tableau qui affiche tous les nouveaux messages, d'un champ de texte pour écrire un nouveau message et d'un bouton pour envoyer le texte. A noter qu'en appuyant sur la touche entrée, il est possible de mettre le focus sur le champ de texte, et appuyez dessus envoie le message et fait perdre le focus au champ de texte.

Une partie se lance en appuyant sur le bouton au centre **Start**.
Quand une partie se termine, une petite animation s'exécute et à la fin de celle-ci il est possible de relancer une partie en re-cliquant sur le bouton **Start**.

Serveur :

Le serveur possède donc lui aussi la classe **Base**.
Cette dernière possède toutes les méthodes qui sont en faite appelées quand les sockets des clients le demande, dont celle qui permet au client d'être relié à une room, une room est enfaite un **GameMVC** du serveur, qui est une classe qui suit l'architecture du modèle **MVC**.

C'est dans cette dernière qu'une partie se déroule entièrement.
Le serveur relie les joueurs dans ces rooms et commence la partie quand un des joueurs décide qu'il peut jouer. A partir de là, le serveur lance une boucle de jeu qui va faire descendre le Tétromino principal et qui à chaque mouvement enverra le tableau de jeu aux joueurs connectés à cette room. Il gère toute la physique du jeu, le timing entre chaque descente de la pièce, et les autorisations de déplacement d'une pièce par un joueur. Il gère aussi les messages du chat, pour le chat il ne sert que de relais, il ne fait aucune vérification sur le contenu de ces derniers. Lorsqu'une fin de partie est détectée, il arrête sa boucle de jeu et prévient les clients, vérifie si leur score mérite d'être inscrit dans le fichier **hallOfFame.json** puis attendent qu'un joueur demande de recommencer la partie. A chaque fois qu'un joueur demande à se connecter à une room, le serveur vérifie si elle n'est pas déjà prise ou en cours. Si elle n'existe pas, le serveur la créer et connecte le client dessus. Si un client quitte une room en cours de partie, la partie se termine. Si tous les clients quittent une room, la room est détruite.

Problèmes rencontrés

Guillaume a rencontré un problème avec la fonction de rotation, des fois même lors de collision le tetromino se déplaçait à l'intérieur de bloques, nous avions laissé ça de côté et quelque jours avant de rendre le projet on s'est rendu compte que la copie des positions qu'il faisait avant le test de collision n'était pas vraiment une copie, Il a suffit de rajouter un **.slice()** à la copie de tableau pour régler le problème.

Etienne a rencontré quelques soucis sur le display en jeu, plus particulièrement sur l'affichage de manière responsive. Effectivement, cela relevait beaucoup de mathématiques ainsi que de test afin de s'assurer que l'affichage du jeu suivait peu importe les dimensions de l'écran. Il a fallu beaucoup tâtonner avant d'arriver à un affichage que nous jugeons satisfaisant, tout dépendant aussi du nombre de joueurs étant variable.

Information utiles sur le jeu

Le jeu est jouable sur ordinateur comme sur mobile, mais est tout de même plus agréable/facile à jouer sur ordinateur (qui possède des inputs exclusifs comme le Fast Share).

Voici les touches sur ***ordinateur***:

- ***R***: Lancer ou relancer la partie (en début ou en fin de partie)
- ***Z***: Fast Fall (Descente instantanée)
- ***S***: Descente accélérée
- ***Q***: Déplacement de la pièce à gauche
- ***D***: Déplacement de la pièce à droite
- ***A***: Fast Share à gauche (téléporte la pièce dans le camp du joueur à gauche)
- ***E***: Fast Share à droite (téléporte la pièce dans le camp du joueur à droite)
- ***C***: Ouverture du chat
- ***ENTRER***: Focus sur le champ texte du chat
- ***ENTRER*** (Quand le champ texte possède le focus): envoie du message et perte du focus sur le champ texte

Voici la gestuelle sur ***mobile***:

- ***Swip Up***: Fast Fall (Descente instantanée)
- ***Swip Down***: Descente accélérée
- ***Swip Left***: Déplacement de la pièce à gauche
- ***Swip Right***: Déplacement de la pièce à droite

De plus, nous pouvons noter que le jeu lance une musique réalisée par Etienne lui même lorsque nous somme en jeu.