RAPPORT DE Projet NodeJS: *Programmer un Jeu Multijoueur en Temps Réel avec NodeJs*

Le but du projet consiste à utiliser NodeJs pour créer jeu multijoueur client/serveur en temps réel. Dans ce cas il s'agit du jeu 'agar.io'. Dans ce document nous allons parcourir les deux fichiers principaux du projet, à savoir client.js et server.js, afin d'expliquer les différentes fonctions qui y sont implémentées.

I. Fichier server.js

Comme son nom l'indique, ce fichier contient le code qui sera exécuté du côté du serveur. Le serveur aura principalement pour tâches de :

- Gérer la connexion des joueurs : création d'un client, suppression d'un client….
- Gérer le déplacement et la vitesse des joueurs en fonction des coordonnées de la souris et du rayon du joueur
- Générer les 'foods' (nourritures) à des positions aléatoires mais couvrant toute la superficie du jeu
- Gérer les collisions entre joueurs (joueur-joueur) et entre un joueur et un food.

1. Fonction ***generateFoods***

Cette fonction permet de générer des 'foods'. Les foods sont de petits cercles de différentes couleurs réparties un peu partout sur l'aire de jeu. Pour cela nous avons utilisé la fonction Math.random() qui génère une valeur aléatoire entre 0 et 1 que nous multiplions par la longueur puis la largeur de l'aire de jeu pour trouver les coordonnées du food. La quantité de foods présente dépend de la superficie de l'aire de jeu, lorsqu'un joueur mange un food un autre est régénéré automatiquement.

2. Fonction ***sizeCell*** et ***speed***

La fonction sizeCell permet de gérer la grosseur d'un joueur en fonction de la taille du food ou du joueur que ce dernier mange. Pour ce fait, nous avons utilisé une fonction logarithmique afin que la proportion d'agrandissement diminue au fur et à mesure que la taille du joueur augmente. Le même principe est réutilisé au niveau de la fonction speed qui permet de calculer la vitesse du jour, plus le joueur grossit moins il est rapide.

3. Fonction ***detectCollisions***

Comme son nom l'indique cette fonction est utilisée pour gérer les collisions entre joueurs et entre joueur et food. En ce qui concerne la collision entre joueur et food, grâce à la propriété de Pythagore l'on calcule la distance entre le food et le joueur. Si cette distance est inférieure au rayon du joueur alors ce dernier a mangé le food.

Le même principe est utilisé pour les collisions entre joueurs mais en plus on vérifie que le joueur soit au moins plus grand que celui qu'il veut manger.

En dehors des fonctions présentées ici, nous avons d'autres fonctions pour la gestion des connexions, pour l'envoie des variables vers les clients, pour lire et actualiser les positions des clients.

II. Fichier *: client.js*

Ce fichier contient le code qui sera exécuté par chacun des joueurs. On y retrouve principalement la fonction ***DrawPlayer***, ***DrawFood*** et ***DrawCircle*** qui permettent d'afficher les objets dans le canvas du client.

1. Fonction ***DrawCircle***

La fonction ***DrawCircle*** est très importante car elle nous permet de faire les contrôles avant de décider si on doit afficher un objet ou ne pas l'afficher. Comme expliquer dans la vidéo, le canvas du client n'est qu'une petite partie de l'aire de jeu. Il est donc important de s'assurer que les coordonnées de ce que l'on affiche (joueur ou food) sont bien comprises dans l'aire du canvas du client. Dans un premier temps on calcul la distance entre l'objet et notre joueur, ensuite on vérifie que cette distance est inférieure aux limites imposées par les dimensions du canvas, enfin on recalcule les coordonnées en fonction du canvas du client puis on dessine l'objet.

2. Fonction DrawFood et DrawPlayer

Dans ces deux fonctions, l'on appelle la fonction DrawCircle sur chacun des players et sur chacun des foods