COMPTE RENDU PROJET WEB

LOUIS FALGUEROLLE
FLORENT SAUSSAY

- Rédiger un compte rendu, expliquer le choix des technos et APIs
- Expliquer l'architecture du programme
- Les problèmes rencontrés lors du dev
- La division du travail

ARCHITECTURE DU PROGRAMME

LISTE DE FONCTIONS SERVEUR `apps/base/_server/index.js`

- _onIOConnect : Cette fonction sert d'initialisation du serveur, elle appelle la fonction surchargée de la classe ModuleBase. Cette classe initialise le tableau de nourriture disponible sur la carte du jeu. On ne le fait qu'une fois pour ne pas se retrouver avec un tableau surchargé. On appelle aussi toute les fonctions de socket.on qui attendent des messages provenant du client.

- _onPlayerConnectReq : Cette fonction crée une instance de blob, notre classe joueur, à chaque fois qu'un utilisateur se connecte au jeu. On envoie ainsi ces informations au client.

- _onValidate : Cette fonction sert de passage entre l'écran d'accueil et le jeu. Elle est activée uniquement lorsqu'on appuie sur le bouton play et reçoit le pseudo rentré par le joueur dans le formulaire. On vérifie ensuite s'il correspond bien à nos attentes et on renvoie la réponse au serveur.

- _onUpdate : Cette fonction est la fonction de mise à jour du jeu. Elle est activée toutes les 33 millisecondes et reçoit en paramètre les coordonnées de jeu d'un seul client. Lorsqu'on les a reçus, elle calcule aussitôt les collisions entre les autres joueurs et le joueur concerné. On suit ainsi les règles basiques du agar.io en remettant les valeurs du blob qui a perdu la partie à 0. On calcule aussi les collisions entre joueur et la nourriture, on augmente la taille du joueur à chaque fois que l'on touche une nourriture. A la fin de la fonction, on renvoie au client concerné ses données actualisées plus les données de tous les autres joueurs et on envoie à tous les clients les données de nourriture.

- Classe Blob : C'est la classe qui contient les informations du joueur.

- Classe Food : C'est la classe qui permet de créer de la nourriture.

LISTE DE FONCTIONS CLIENT `app/base/js/main.js`

MODELE

- Les fonctions du modèle servent principalement à l'envoi de données au serveur avec la fonction "emit", par exemple "sendUpdatePositionMessage" envoie les données de blob contenues dans le modèle au serveur, ou à l'attente de données avec la fonction "on".

VUE

- showStartWindow : on initialise la page d'accueil.
- setGameStage : On initialise le terrain de jeu avec un canvas de 4000*4000 pixels.
- setScoreTab : On initialise le tableau de score positionné sur la droite de l'écran.
- setBlob : On dessine le blob sur le terrain.
- drawFood : On dessine la nourriture sur le terrain.
- drawOthers : On dessine les autres joueurs présents sur la carte.
- drawGame : C'est la fonction qui permet de dessiner tout le jeu. On efface tout avec clearRect, on sauvegarde l'état initial avec save, on déplace tous les objets avec la fonction translate, c'est la fonction qui permet de garder le joueur au centre du terrain et de déplacer les autres objets à l'inverse du sens de déplacement du joueur concerné. Ensuite, on appelle les 3 dernières fonctions pour dessiner le joueur, la nourriture et les autres joueurs. Pour finir, on calcule des nouvelles coordonnées a notre joueur ce qui va permettre le déplacement, on va empêcher notre joueur de dépasser certaines limites de terrain, et on met en place la dernière sauvegarde du canvas que l'on a sauvegardée.
- loopGame : C'est la fonction qui appelle setInterval pour mettre en place une boucle de jeu. On rappelle donc la fonction DrawGame, pour dessiner le jeu, la fonction sendUpdatePositionMessage, pour actualiser le jeu, et updateScore, dont nous aller parler de suite.
- updateScore : C'est la fonction qui permet l'affichage du score, on copie le tableau de joueur "blobs" dans un autre tableau que l'on va trier du plus grand au plus petit, on vide notre tableau puis on reecrit par-dessus.
- mouseUpdate : c'est un événement qui permet de récupérer les coordonnées de souris.

CONTROLLER

- ioStartGame : On lance les fonctions d'initialisations du jeu dont les fonctions d'attente de donnée serveur (searchDataUpdate) et la boucle de jeu (loopGame).
- valCon : Fonction qui reçoit un message du serveur concernant les pseudo des nouveaux joueurs entrant. Si le message est positif, on lance le jeu sinon, on relance le menu principal.
- updateWorldData : On met à jour les données de jeu que l'on reçoit depuis le serveur,on met à jour les données de tous les joueurs et notre score. Si notre joueur est mort, on actualise les données une dernière fois avant de fermer le jeu.

TECHNOLOGIES UTILISEES

Nous n'avons utilisé rien d'autre à part le template fournis et du Javascript. Nous aurions pu utiliser le template P5 mais nous n'avions pas connaissance de son existence, et lorsque nous l'avons découvert, il était bien trop tard pour changer de cap.
Nous avons partagé notre code avec GitHub et communiquer avec Discord.

PROBLEME RENCONTRÉS

En premier point, nous avons rencontré une certaine difficulté avec la technique de programmation modèle vue contrôleur, en effet, elle est assez déstabilisante au premier abord et comprendre ou doivent être positionnés les éléments de programmation est difficile. Aussi, nous avons pris du temps avant de bien comprendre tout le fonctionnement du template fournis.

En second point, nous avons rencontrés des difficultés sur la méthode de déplacement, nécessitant une mise à jour constante et des calculs bien maîtrisés, nous avons pris beaucoup de temps pour arriver à un stade où le joueur reste toujours au milieu et où les objets bougent en fonction des mouvements de la souris. Nous avons dû réfléchir à un système de mise à jour serveur qui marchait sans envoyer trop d'informations et sans ralentir tout le programme.

En troisième point, nous allons énumérer quelques fonctionnalités que l’on n’a pas pu ajouter faute de temps :
- La vitesse du joueur n'est pas altérée par sa taille comme dans le jeu originel.
- L’écran du joueur ne s'adapte pas à la taille du joueur, nous avons essayé plusieurs fois de rajouter cette fonctionnalité mais nous avons préféré nous concentrer sur l'essentiel.
- Le tableau de joueur ne s'adapte pas au nombre de joueur comme dans le jeu originel et ne s'adapte pas à la taille de l’écran.

DIVISION DU TRAVAIL

Nous nous sommes beaucoup entretenus via des logiciels de communication vocale pour pouvoir suivre l'avancement du travail. Nous programmions a 2, sur 1 seul écran que nous partagions via ce même logiciel. Cela rendait la programmation plus simple avec deux cerveaux au lieu d'un.