Task 4

Browser Game

Rapport

Jozipovic Thien
02/05/2022

Thien Jozipovic
I6
Sion, le 02.05.2022

1 Table des matières

1. Introduction

Description

Ce labo a pour but de développer une application pour navigateur web. Cette application, sous forme de petit jeu, est couplée avec le travail déjà effectué dans les tâches précédentes. Le jeu développé utilise nos thingies comme entrées-sorties.

Cette application codée en HTML/Javascript tourne en local et, à l'aide d'un websocket, se connecte à notre broker vers lequel sont échangés les messages avec le Raspberry pi. De cette façon, l'application web est notifié lorsqu'un des boutons est appuyé et peut également changer la couleur des leds sur les thingies.

Cahier des charges

Voici les différents points à réaliser :

- Réaliser une application/jeu pour un navigateur web.
- Utiliser les boutons comme entrées.
- Utiliser les leds rgb pour transmettre des indications sur le jeu.
- Réaliser un rapport.

2. Développement

Cette partie traite des différentes étapes du développement de l'application web.

Idée

L'idée de départ consiste à réaliser un petit jeu de type « Mario bros ». C'est-à-dire un jeu se déroulant en 2D dans lequel le joueur se déplace à travers une carte horizontalement. Le premier objectif consiste à traverser le jeu tout en évitant de chuter dans des trous. Pour cela, le joueur peut sauter au-dessus de ces derniers ou bien utiliser des plateformes flottantes. Ensuite si le temps le permet, rajouter des ennemis qui font perdre de la santé au contact pour rendre le jeu plus difficile et intéressant.

Dans le premier cas, un thingy servirait à avancer et l'autre à sauter. Dans le deuxième, le joueur se déplace automatiquement et dans ce cas un thingy servirait à sauter comme précédemment et l'autre à attaquer les ennemis.

Les leds servent à différencier les thingies.

Design

Cette section décrit comment le projet est structuré.

Figure 1 : Lien entre application web et thingies à travers MQTT et websocket (schéma cyberlearn)

Le jeu est connecté depuis le navigateur au broker via MQTT par websocket. Le broker est lui-même lié au Raspberry également avec MQTT pour obtenir les statuts des thingies connectés en bluetooth ou pour contrôler leur led.

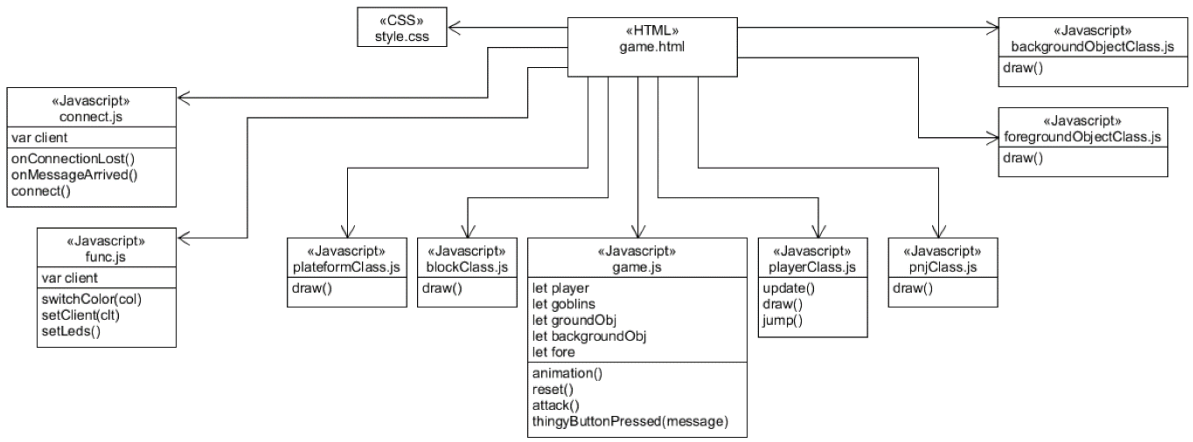

Figure 2 : class diagram

Dans la *figure 2* ci-dessus, nous avons la structure de l'application. Il y a donc une seule page web représentée par game.html avec un fichier css. La majorité du jeu est contenu dans la partie javascript.

Le jeu

game.js :

Sert à faire tourner le jeu. Crée et initialise les objets nécessaires et contrôle le jeu.

Plateforme, block, player, pnj :

Classes qui représentent les objets utilisés dans le jeu c'est-à-dire le joueur, les ennemis et les objects solides.

backgroundObject, foregroundObject :

Classes pour les objets avec un rôle visuel uniquement.

La communication avec le broker

connect.js et func.js permettent à l'application web d'échanger des messages via MQTT. Connect.js Se établie la connexion avec le broker et indique lorsqu'un message. C'est ici que nous obtenons l'information sur un changement de statut concernant nos thingies.

Func.js est utilisé principalement pour modifier l'état des leds sur les thingies. Pour le jeu, cela permet de différencier les deux boutons.

Implémentation

Cette partie décrit plus en détail le fonctionnement de l'application.

1.1.1 Connexion

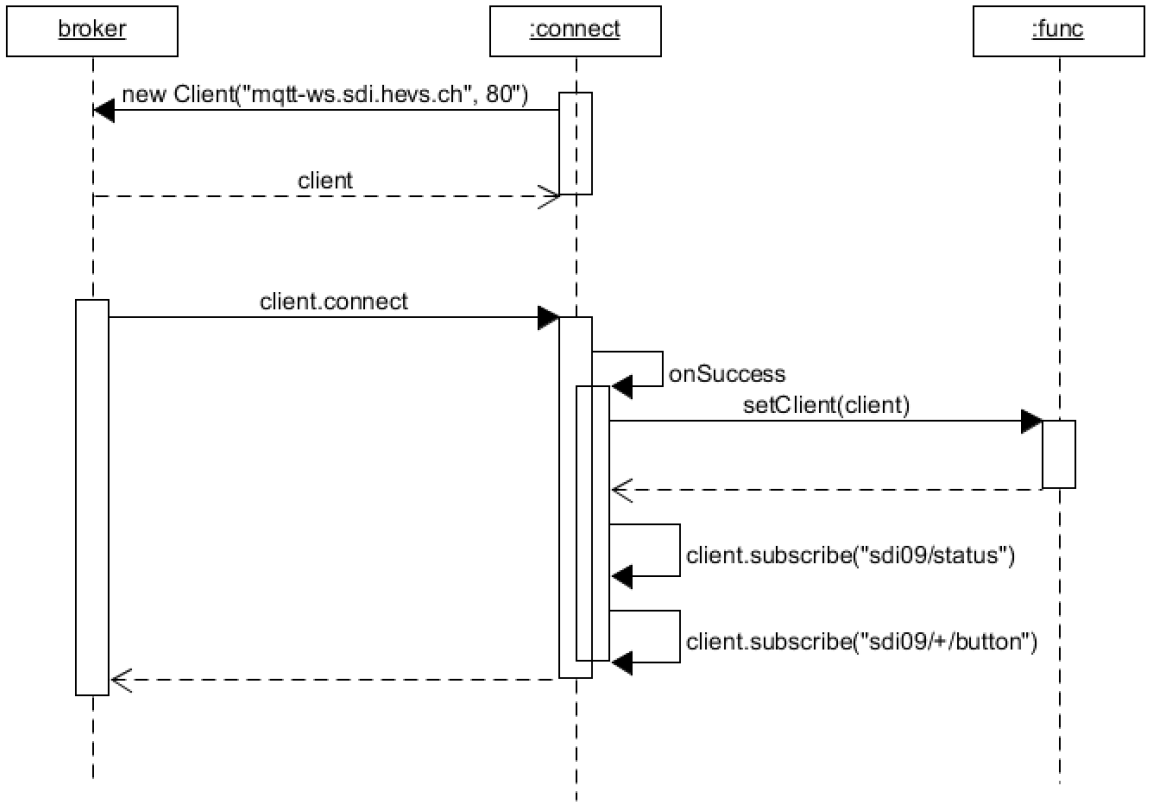

Figure 3 : Connexion avec broker

Le diagramme ci-dessus illustre les étapes qui se produisent lorsqu'une connexion s'établit. Si la connexion se fait avec succès, le client est mis à jour dans la classe « func ». Cela est nécessaire pour ensuite contrôler les leds. Ensuite, les subscribes se font pour qu'on soit notifié des changements sur les thingies.

1.1.2 Fonctionnement du jeu

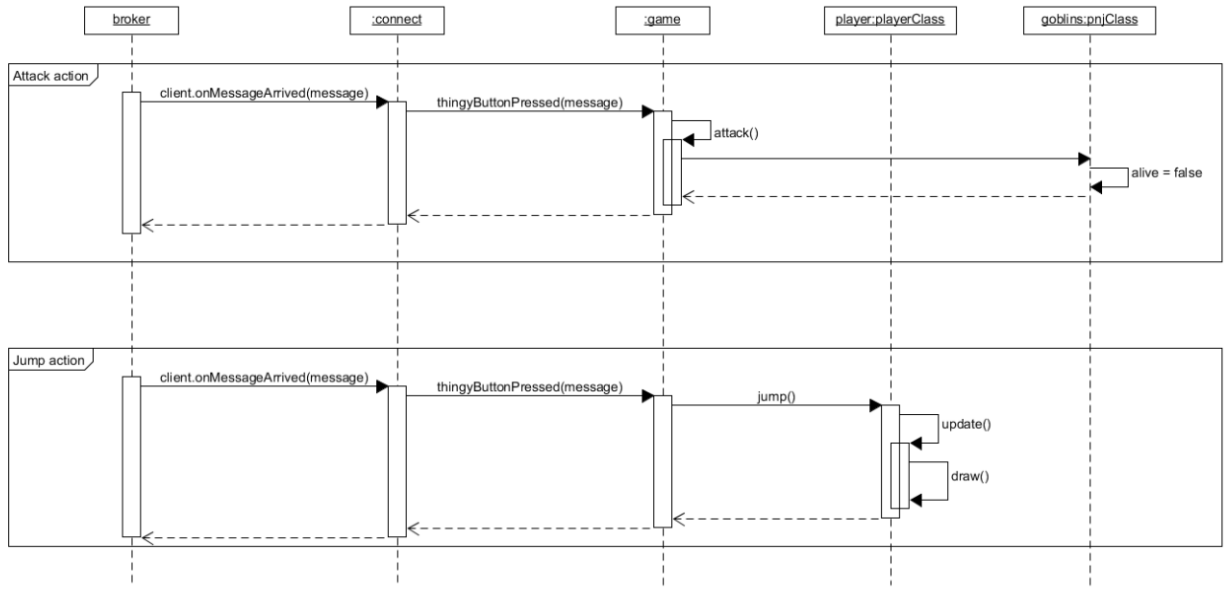

Figure 4 : Utilisation des thingies

Boutons

Le diagramme de séquence de la figure 4 illustre ce qu'il se passe lorsque quelqu'un appuie sur un des boutons. L'un d'entre eux sert à sauter dans le jeu et l'autre à attaquer. Cette information nous parvient par onMessageArrived à chaque fois que le bouton est appuyé. L'information est ensuite transférée à game.js par la méthode « thingyButtonPressed ».

Pour l'attaque, la méthode « attack() » est appelée dans game.js. Elle va comparer la position du joueur avec celle des ennemis. Si un goblin se trouve à proximité, son attribut « alive » passe à false. Cela permet de savoir lesquels sont encore en jeu et doivent encore être pris en compte.

En ce qui concerne un saut déclenché par le deuxième thingy, nous appelons cette fois la méthode « jump » de la classe « playerClass ». La vitesse verticale de joueur sera modifiée et nous refaisons un update qui met à jour l'état du joueur avant de le redessiner.

Contrôle du jeu

Dans le fichier game.js se trouve la méthode « animation ». C'est ici que ce fait la majeure partie du jeu. On y appelle « draw » pour tous les objets à afficher, contrôle les mouvements, applique les collisions et détermine quand la partie est gagnée ou perdue.

La méthode « reset » remet le jeu à l'état initial lorsque le joueur tombe dans un trou ou meurt au contact d'un goblin.

Des eventListeners sont utilisés pour détecter les actions sur les touches du clavier.

Problèmes

Il y a plusieurs problèmes qui ont été rencontrés. Le premier et le plus gênant est celui du retard observé avec les boutons. En effet, lorsqu'un bouton des thingies est appuyé, il y a un petit délai qui se produit avant que l'action ne prenne effet. Sur le sdi mqqt Tester, on s'aperçoit que la notification est instantanée. Le retard se produit donc du côté de l'application. En retirant les « console.log() », j'ai remarqué une petite amélioration. Cependant, l'effet se fait toujours sentir et le gameplay n'est pas aussi optimal qu'avec les touches du clavier.

Ensuite, j'ai eu des problèmes d'animations. Les images n'ayant pas toutes le même format, il a fallu ajuster les valeurs des dimensions et du cropping à chaque fois. Cela rajoute beaucoup de lignes de code en plus et réduit un peu la lisibilité.

Finalement, lorsque j'ai voulu séparer le travail en plusieurs fichiers, pour mieux structurer les choses, j'ai dû prendre un peu plus de temps que prévu pour importer tous les scripts dans le bon ordre et séparer le contenu correctement pour que tout puisse fonctionner à nouveau en utilisant les variables/méthodes d'un script à l'autre.

Instructions du jeu

Voici les instructions pour jouer au jeu :

1. Mettre en marche les thingies, Rapserry … etc
2. Lancez le « game.html » pour parvenir sur la page du jeu.

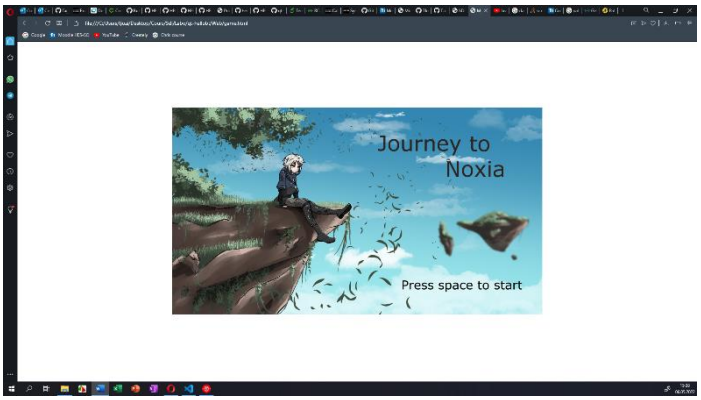

Figure 5 : Start screen

3. Appuyez sur la touche « espace » pour lancer la partie.

Figure 6 : Game started

4. **Jouer avec les thingies** : Le thingy bleu permet de sauter. Le rouge d'attaquer
   Jouer avec le clavier : **a** et **d** pour se déplacer latéralement, **espace** pour sauter et **c** pour attaquer.

5. Atteignez l'autre côté de la carte en évitant les trous et les goblins.

3. Conclusion

La dernière tâche de ce super lab nous a permis de développer une application web dans laquelle nous avons pu en apprendre plus sur la programmation en html et javascript. Cela oblige à s'adapter à ces nouveautés tout en améliorant les connaissances obtenues en classe. L'application développée était un petit jeu dans le navigateur dans lequel le joueur se déplace sur un plan en 2D. Certains problèmes ont été rencontrés comme mentionné plus tôt mais globalement le jeu est pratiquement terminé. Il est jouable avec les thingies ou le clavier du début à la fin.

Certains points n'ont pas pu être atteint par manque de temps. Il n'y a pas de message félicitant le joueur lorsqu'il gagne, les goblins disparaissent tout simplement lorsqu'on les tape (il aurait été bien qu'il y est un petit effet), et malheureusement il n'y pas le moindre effet sonore ou musique pour rendre le jeu plus intéressant. Tous ces ajouts auraient pu apporter un plus au résultat final.

Néanmoins je suis content du résultat et toutes les tâches, du début jusqu'au jeu final, fonctionnent bien dans l'ensemble. Le petit retard entre le bouton et l'action rajoute du challenge.