Task 4

Browser Game

Christophe Grobéty, I6

Rapport de projet Browser Game

1) Introduction

Durant ce laboratoire, nous étions libres de créer un jeux sur html via JavaSckript en utilisant les diverses sources mise à disposition sur internet et en permettant au joueur d'utiliser les Thingies en Bluetooth par les messages envoyés sur le mqtt.

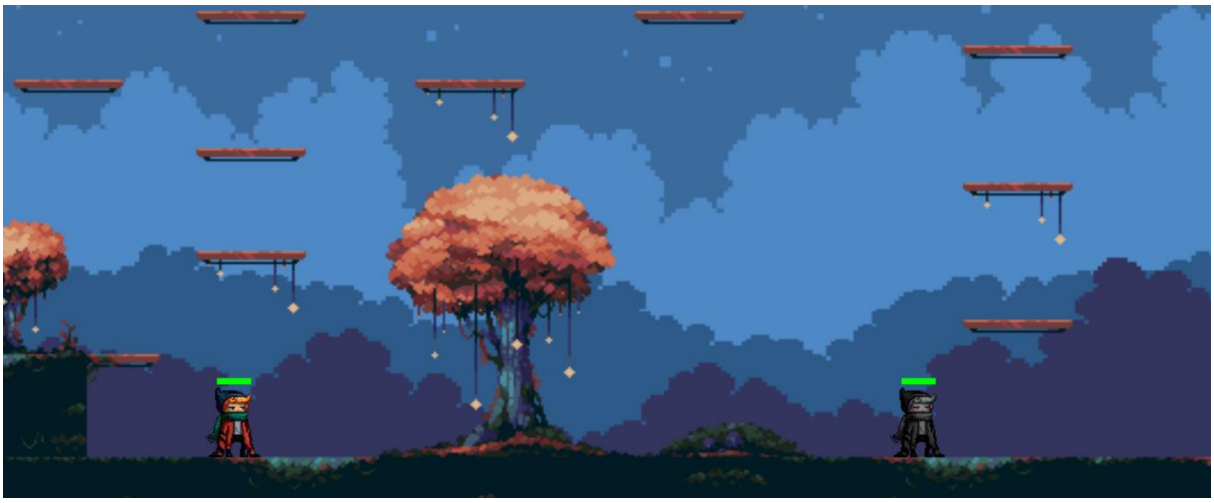

Figure 1 : Image final du jeux

Pour ce projet j'ai donc décidé de faire un ''Smash bros'', c'est un jeux en vue 2D où 2 joueurs s'affrontent grâce à des attaques à distance ou rapproché, pouvant sauter sur plusieurs plateformes jusqu'à ce que l'un des 2 n'aient plus de vie.

Pour rendre ce jeux jouable je me suis donc fixer plusieurs objectifs primaires à faire et des améliorations possible si le temps me le permettais :

Primaires :

- Faire une physique de jeux pour les déplacements et la gravité avec un double saut pour les joueurs
- Détections des collisions entre le joueur et les plateformes
- Créations d'attaques et gestions de ses collisions avec le joueur adverse
- Implémentations des bruitages et de la musique pour rendre le jeux ''vivant''
- Maniement du joueur 1 avec les thingies
- Un game-restart quand le jeux se termine avec l'affichage du joueur gagnant

Optionnels :

- Implémentations de plusieurs attaques et personnages disponibles
- Possibilité de jouer avec 4 thingies différents
- Menu de sélections des personnages et réglages des touches/thingies
- Implémentations d'une I.A
- Reconnections des thingies quand la connexion est perdue

2) Design&implémentation du jeux

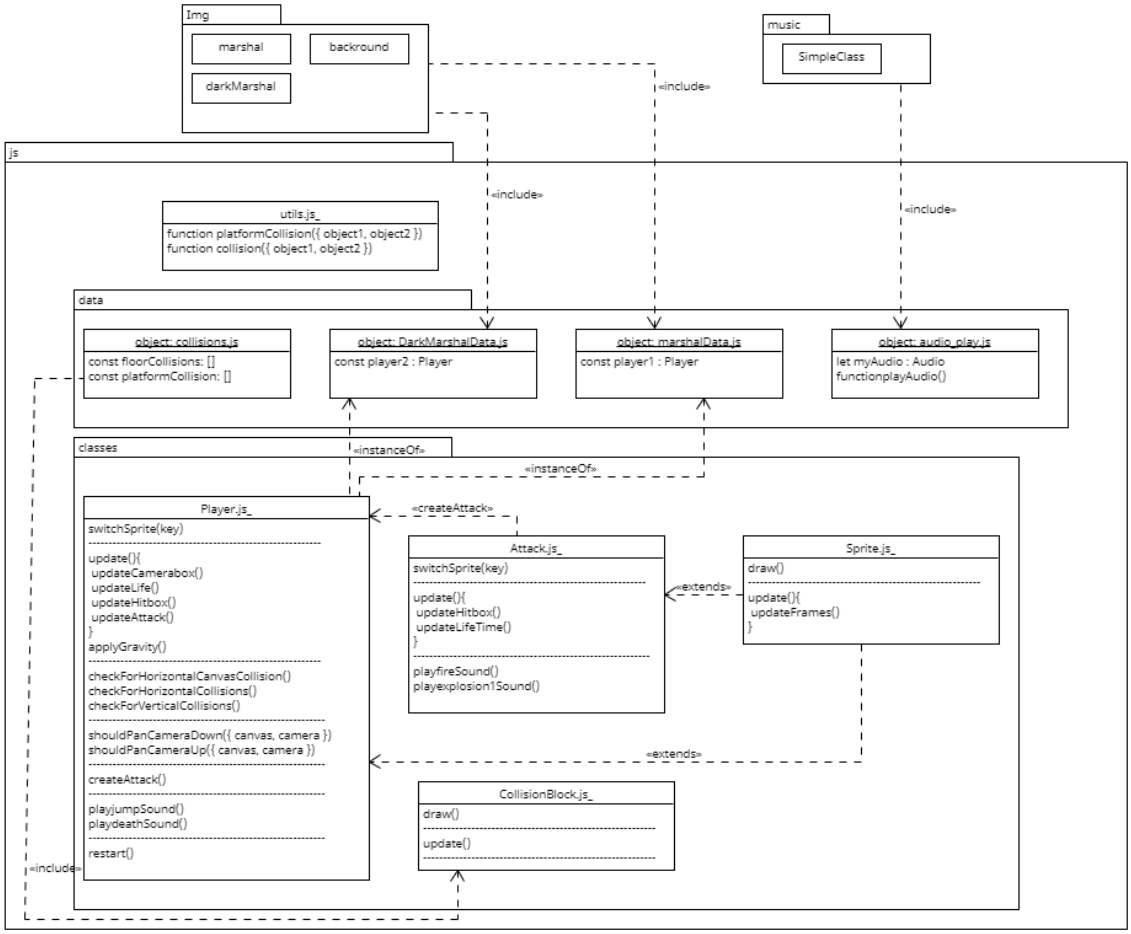

Figure 2 : diagramme UML du jeux

Fonctionnement globale :

Le code entier est utilisé dans la classe index.js qui est lié au fichier index.html, c'est dans cette classe que les différents objets sont appelés et initialisés ainsi que la connexion aux thingies.

La function animate() permet de rafraichir l'affiche de la page et d'appeler les différentes function update() des objets créés afin de permettre les déplacements de la caméra, des joueurs et de leurs attaques. Par soucis de temps, c'est aussi dans cette classe que sont géré les collisions entres les attaques et les joueurs ainsi la gestion de leurs déplacements, animations et vie en fonctions des touches appuyés ou des collisions d'attaques détectés.

FICHIER CLASSES :

Player() :

C'est dans cette classe que l'on s'occupe de concevoir le personnage en lui donnant les informations principales dans son constructeur (vie, largeur, gif d'animations, plateformes de collisions, etc…). Ses fonctions sont multiples, les plus importantes étant la mise à jour du déplacement et des images à utiliser en fonctions des informations reçus, la création et la destruction des attaques lancées ainsi que l'utilisations des sons pour rendre le tout plus vivants.

Le but de cette classe était vraiment d'en faire une classe complétement globale afin de pouvoir créer différents personnages sans avoir besoin de concevoir à chaque fois une classe différentes pour cela, malheureusement par manque de temps et de connaissance du javaSkript je n'ai pas pu rendre la function de créations d'attaque complétement globale du à l'utilisations de tableau d'attaques qui demandais d'utiliser des ''new()'' ce qui m'empêchais de pouvoir lui donner des constantes à la place.

Attack() :

Comme pour la classe player, celle-ci permet de servir de constructeur global pour me permettre de pouvoir créer différents types d'attaques sans avoir pour autant à en créer à chaque fois un spécifique. La plus part de ses functions sont comme celles du Player(), les attaques ont donc un lifeTime pour permettre au joueur en les lançant de savoir quand il faut donc les supprimer afin d'éviter de surcharger le programme.

Sprite() :

Cette classe est celle dont héritent les deux dernières classes présentées, c'est celle-ci qui permet l'affichage et le découpage des images données en gif animé. Elle crée donc des box avec les paramètres de longueur et de largeur des images données et simplifie le changement d'animations en fonction des actions utilisées par le joueur.

CollisionBlock() :

Ici l'objectif et de concevoir les ''blocs'' en fonction des données présentent dans les datas afin de générer les informations nécessaires pour la classe Player(), afin qu'il puisse détecter quand il touche le sol ou une plateformes sur le background, pour stopper sa chute (fonctions de détections des collisions implémenté dans la classes utils.js).

FICHIER DATA :

Ce fichier réuni les différentes constantes et fonctions utilisées dans l'index afin de faciliter la création de nouveaux personnages, d'audio ou de nouvelles map

FICHIER Img & music

C'est dans ces fichiers que se trouvent les différentes images et musiques utilisées pour les constantes du jeu.

But :

Le fonctionnement désiré du jeux était de pouvoir créer plusieurs classes globales pour simplifier le rajout de plusieurs map, musiques, personnages et d'attaques propres à chacun via l'utilisations de constantes créées dans le fichier data et appelé dans l'index.

3) Problèmes

Comme pour chaque programme à concevoir, celui-ci à aussi fait face à une multitude de problèmes/ bugs à comprendre et à résoudre ou en essayant de changer de méthode de conceptions.

Personnellement, le plus gros problème auquel j'ai dû faire face durant ce projet était la gestion du temps et l'abandon de certaines fonctionnalités optionnelles qui étaient trop complexe à mettre en place dans un espace de temps réduit (exemple : l'I.A, un menu de sélection, trouver d'autres personnages, etc…).

J'ai aussi dû faire face à la complexité de l'appelle des services des thingies et de leurs caractéristiques, comme pour l'utilisation du gyroscope ou de l'accéléromètre. De base je voulais rendre le jeux jouable avec 1 thingy :

Le service d'euler pour utiliser les rotations afin de déplacer le personnage, l'utilisation du bouton pour permettre les sauts et enfin l'accéléromètre pour utiliser l'attaque (un peu comme l'utilisation faite sur les natels pour les applications de lancer de dés par exemple, qui détectent les mouvement brusques), malheureusement je n'ai pas réussi à utiliser au mieux les valeurs envoyées par les thyngies sur les accélérations, ce qui fait que seul un joueur peut utiliser les thyngies durant le jeux.

Il y a aussi le fait que les thyngies se déconnectent souvent après une certaine période aléatoire. Je voulais créer une fonction permettant de pouvoir les reconnecter, mais je n'ai pas eut le temps de l'implémenter et de la tester.

Un autre problème a été l'utilisation du javaScript, deux en particulier ont été complexe à gérer :

- La compilation des fichiers, car dans le fichier html la position des classes à une importance ce qui fait qui si une classe en utilise une autre, mais que celle-ci est positionnée en-dessous d'elle, alors elle ne pourra pas la détecter et l'utiliser. Cela a rajouté une difficulté supplémentaire vis-à-vis de l'implémentation des différentes classes
- La création des objets d'attaques : Comme décrit plus haut, le but premier étaient de rendre les classes globales, le plus indépendantes possibles Cependant pour rendre la création et la destructions d'objets faisables, il fallait que je crée à chaque fois de nouveaux objets à placer dans un tableau pour permettre leur destruction sans effacer les valeurs de base. Mais je n'ai pas réussi à faire en sorte que ces nouveaux objets puissent en copier un déjà existant qui aurait été donné au constructeur de player en paramètre, notamment à cause du fait que celle-ci possède un super constructeur qui rend la copie complexe.

Pour la plupart des autres problèmes ils s'agissaient de simples bugs fixés ou alors d'implémentations de certaines fonctionnalités qui étaient trop gourmande en matière de temps.

Instructions pour lancer le jeux et l'utiliser :

Jouer avec les thyngies :

Pour jouer avec les thyngies, il vous faut d'abord commencer par lancer le programme dans le fichier HelloBT en allumant les 2 différents thyngies. Une fois les que les lumières bleues arrêtent de clignoter, cela signifie que les appareils sont connectés au mqtt et peuvent envoyer les différents messages liés à leurs services.

ATTENTION : si vous souhaitez changer de thyngies, il vous faudra modifier le fichier json: ''sdi-config.json'' dans le dossier HelloBT ainsi que les valeurs dans le dossier testGame pour les variables buttonOne, buttonTwo, eulerOne et eulerTwo et modifier le userName et le password dans la partie client.connect(ils sont situés en haut du fichier index.js) .

Les déplacements horizontaux du personnage se font en penchant le thyingy D5 sur les côtés quand le témoin lumineux et placer devant vous en haut à gauche.

Le bouton permettant les sauts se situe aussi sur le D5, quant à celui pour l'attaque il se situe sur le FC.

Jouer avec le clavier :

Le joueur 1 se déplace avec les touches ''A'' et ''D'', saute avec ''W'' et tire avec ''S''.

Le joueur 2 se déplace avec les touches ''J'' et ''L'', saute avec ''I'' et tire avec ''K''.

But du jeux :

C'est donc un jeux de combat, le but est de faire diminuer les point de vie de l'adversaire, jusqu'à ce que sa barre de vie soit complétement rouge. Il est donc recommandé de bouger en permanence afin d'éviter de se faire toucher, en utilisant les plateformes atteignable par un saut ou un double saut.

Pour pouvoir tirer vous êtes obligé de ne plus utilisé les touches de déplacement horizontale, si vous êtes en l'air et que vous appuyez sur la touche de tire durant un déplacement horizontale, vous serez alors dans l'état ''dash'' ce qui augmente votre vitesse tant que vous ne touchez pas le sol.

Un fois un des 2 joueurs morts le vainqueur est affiché sur l'écran et vous pouvez recommencer en appuyant sur la touche ''P''. (la touche ''P'' permet aussi de lancer la musique)

Conclusion :

Ce projet était très intéressant à programmer et à designer d'une manière générale et m'a permis de mieux comprendre comment utiliser javaScript, HTML et comment débuguer ces programmes en ligne.

Le projet de jeux que j'ai choisi était peut-être un peu trop complexe vis-à-vis de mes attentes finales ainsi que du temps à disposions afin de pouvoir implémenter l'intégralité des fonctions que je voulais ainsi que de l'utilisation des thyngies.