Documentation

I) Organisation du dépôt Git

Le dossier *assets* contient toutes les images et musiques utilisées par l'application, ainsi que tous les niveaux.

Les sources se trouvent dans le dossier *requirejs/app*.

Le reste constitue les librairies utilisées avec les fichiers de configuration utilisées pour les différentes plateformes.

II) Graphisme

Les images utilisées se trouvent dans le dossier *assets*. Ces dernières peuvent être modifiées sans aucun problème à condition que les dimensions soient les mêmes (et le nom de fichier soit identique également).

Nous avons utilisé deux types d'images : soit une unique image par fichier, soit plusieurs images dans un même fichier pour faciliter les animations (*spritesheet*). Par exemple, pour le miroir, chaque image est dans un fichier séparé ; mais pour le joueur, elles sont toutes stockées dans un même fichier (pour faciliter les animations).

Pour modifier les graphismes, le plus simple est de conserver leurs dimensions, et pour les spritesheet le nombre d'images et leur ordre (par exemple, l'ordre des couleurs pour les photons).

Néanmoins, si les dimensions d'une image venaient à être modifiées :

- Pour les images simples, il n'y a pas de modification spéciale à effectuer.
- Pour les spritesheet, il faut modifier dans le code source la taille unitaire de chaque image composant le spritesheet. Pour cela, il faut modifier dans le fichier source concerné (par exemple button.js pour le bouton) la largeur et la hauteur dans la fonction de chargement de le spritesheet. Pour vous faciliter la tâche, des constantes ont été définies en tête de chaque fichier concerné. Il vous suffit de modifier ces constantes pour prendre en compte les modifications des dimensions.
- Dans les deux cas, il faudra probablement adapter tous les niveaux (fichiers JSON).

III) Musique

Concernant les musiques utilisées, il est très facile de modifier les musiques utilisées pour le menu principal (MainMenu.mp3/ogg) et la phase de jeu (musicTheme.mp3/ogg), ainsi que le bruitage utilisé pour le tir de photons (photonFire.mp3/ogg). Pour cela, il suffit de remplacer le fichier en conservant le nom. Il est important de conserver les deux formats pour assurer la compatibilité avec le maximum de navigateurs / mobiles.

Pour les musiques utilisées dans l'introduction, les modifier impliquerait une modification assez conséquente du code dans les fichiers sources (comme leur enchainement a été calibré manuellement). Si elles devaient être modifiées, nous conseillons de garder au minimum la même durée des animations.

IV) Création de nouveaux niveaux

Afin de faciliter la création d'autres niveaux en plus de ceux déjà fournis, un fichier Exemple avec commentaire est fourni. Nous allons tout de même ici en reprendre les points principaux :

Chaque niveau est généré à partir d'un fichier texte au format JSON. Ces fichiers sont des fichiers de donnée externe dont le format est propre au Javascript. Pour la syntaxe de ce type de fichier, nous vous proposons de vous reporter au fichier exemple fourni. De même pour pouvoir vérifier facilement la validité syntaxique, le lien ci-présent vous renverra directement vers un site permettant de vérifier en ligne le formatage de votre fichier : http://jsonlint.com/

Les caractéristiques du niveau : Chaque fichier JSON correspondant à un niveau contient un certain nombre d'informations concernant les caractéristiques ainsi qu'une liste des objets présents et de leurs caractéristiques. Parmi celles-ci, on demande les limites du niveau (dimensions en pixels), le nom du fichier image qui servira d'arrière-plan pour le niveau et les coordonnées de la position initiale du personnage dans le niveau.

Suite à ça, il faut définir la liste des objets du niveau, voici un récapitulatif des différents objets :

- Plateformes : l'élément de base pour l'environnement du niveau, celles-ci peuvent être normales ou colorées, les plateformes colorées sont celles qui permettront au joueur de changer de couleur.
- Plateformes en mouvement : les plateformes en mouvements ont les mêmes caractéristiques mais sont déclarées séparément puisqu'on ne les définies pas à l'aide d'une position mais d'une liste de positions indiquant leur trajet.
- Les filtres : ces objets sont décrits simplement par leur position et leur couleur ainsi que quelques paramètres optionnels (taille, orientation).
- Les miroirs : de même ils sont simplement définis par leur position et leur orientation. On peut cependant décider de créer des miroirs mobiles auquel cas il faudra définir des limites à leur mouvement.
- Les switchs et boutons sont des objets qui interagissent avec les autres, ils sont donc définis par leur position, leur couleur, mais aussi par une action et ses arguments. Chaque action et ses arguments sont décrits dans le fichier actions.txt (situé dans le même dossier que les JSON). Pour définir la cible de leur action, il existe un identifiant pour chaque type d'objet ainsi qu'un identifiant optionnel qui peut être définit pour

chaque objet on identifie donc la cible de l'action par un code objet et id propre à l'objet.

- Les ennemis : il existe deux types d'ennemis : les volant et les normaux. Pour les créer, il suffit de définir une position initiale, une vitesse et des limites à leur mouvement, ainsi qu'une précision du type d'ennemi désiré.
- Les murs : il s'agit de plateformes qui peuvent être détruites à l'aide de photons. On les définies par leur position, taille et l'énergie minimale pour les détruire.
- Les fins de niveaux : il suffit de définir leur position
- Les étoiles : il suffit de définir leur position et valeur

V) Structure du code

Le jeu est divisé en plusieurs états, chaque état étant défini dans un fichier du sous-dossier *states* :

- L'état *boot* démarre l'application
- L'état *prelude* joue l'animation présentant le scénario du jeu
- L'état *preload* charge les différentes images/musiques utilisées
- L'état *mainMenu* correspond à l'écran d'accueil
- L'état *chooseLevel* correspond à l'écran de sélection de niveaux
- L'état *game* correspond au jeu. C'est l'état principal
- L'état *finishLevel* correspond à la réussite d'un niveau.
- L'état *dead* correspond au Game Over.
- L'état *restart* correspond au rechargement du niveau actuel.

L'état principal est l'état *game*. Il comporte deux fonctions principales :

- *create* : crée le niveau en le chargeant depuis le fichier JSON
- *update* : met à jour le jeu à intervalles réguliers

L'état game permet de superviser les différents éléments du jeu

- Le joueur (player.js) et les photons (photon.js)
- Les éléments des niveaux (plateformes, miroirs, ennemis…), dont la gestion a été centralisée par le module objectManager
- Les autres modules gérant le score, la pause, etc.

De manière globale, chaque module possède une fonction *create* appelée lors de la création des niveaux et une fonction *update* pour gérer les déplacements, événements (action de l'utilisateur), collisions, etc…

L'ajout de nouveaux objets peut se faire simplement en créant un fichier implémentant ces fonctions et en l'ajoutant à l'objectManager sur le même modèle que les autres.