Documentation Développeur

Thomas Bombrun, Dimitri Bouleau, Thomas Coeffic, Caroline Ramond, Théo Van-Lede
INPG - Ensimag

17 Juin 2014

Ce document est un complément de la documentation générée automatiquement, qui justifie l'implémentation et présente la structure des dossiers.

1 Javascript

Ces fichiers sont stockés dans le dossier *js*.

- **constants.js** : Permet de charger le fichier de configuration yaml.

- **cookie.js** : Assure la gestion des cookies (dans le cas d'une utilisation web) ou de la mémoire locale (dans le cas d'une utilisation mobile), afin de conserver l'état d'avancement de l'utilisateur dans les niveaux.

- **create.js** : Permet de créer les différents écrans du jeu : le menu principal, le menu de séléction de niveau, et l'écran correspondant à un début de partie.

- **createLevel.js** : Met à jour l'index des niveaux, fonctionne pour le tutoriel et le jeu classique.

- **eventFunction.js** : Gère la majorité des événements associés aux clicks dans les menus et en jeu, et aussi les actions déclenchées durant une partie (affichage de l'écran de fin, rencontre avec un bloc).

- **help.js** : Gère les écrans d'aide.

- **main.js** : Charge les différentes ressources nécéssaires, contient la définition des variables globales utilisées dans plusieurs fichiers.

- **moveball.js** : Gère les déplacements de la bille (prise en compte des actions de l'utilisateur, appel des fonctions adéquates dans *eventFunction* en cas de collision/chevauchement. Gère aussi les déplacements et repositionnements nécéssaires pour les blocs *turn*.

- **parser.js** : Permet la lecture et le chargement en mémoire d'un fichier de niveau.

- **sound.js** : Gère les animations sonores du jeu.

- **hammer.min.js** : Gère les événements sur mobile. Licence MIT.
  hammer sur Github

- **js-yaml.js** : Parser de fichier yaml. Licence MIT
  js-yaml sur GitHub

- **phaser-arcade-physics.min.js** : Framework permettant la création simple de jeu. Cette version ne contient que la physique Arcade. Licence MIT.
  phaser sur GitHub

2 Ressources

Le dossier *ressources* contient les fichiers png chargées par le jeu, ils sont définis dans *conf.yaml*.
Le dossier *ressource-dev* contient les fichiers gimp correspondant à ces fichiers png, et peuvent être utilisés pour modifier plus rapidement les ressources.
Les sons associés aux actions sont stockés dans *ressources/sounds*, afin d'assurer une compatibilé avec firefox ne pas utiliser de fichier mp3 (ogg semble une bonne alternative).
Les niveaux sont eux dans le dossier levels, ils doivent être de la forme "%nombre.txt" et se suivre. Il est en de même pour les niveaux du tutoriel qui sont dans le dossier tutorial.

3 Justification de l'implémentation

- **Cookie.js** :
  On utilise *constants.USE_CORDOVA* pour savoir si on utilise les cookies stars et levelMax, ou si on utilise la clé stars pour stocker ces données.
  Cela permet d'avoir un code unique pour la plateforme mobile et web.

- **eventFunction.js** :
  Dans toutes les fonctions appelées par overlap (défini dans phaser.physics.arcade), on renforce la condition en forçant les deux sprites à être au moins à moitié chevauchante pour effectuer l'action. Cela permet d'éviter les bugs liés à la bille déclenchant deux overlap simultanés.

- **main.js** :
  On utilise une fonction pour trouver la valeur correspondant à un retour 'OK' après une requête de type *XMLHTTPRequest.open*, cela permet d'avoir un code fonctionnant pour Android (OK = 200, erreur = 0), iOS (OK = 0, erreur = 404) et Desktop (OK = 200,

erreur = 404) fonctionnel, sans utiliser user-agent qui est peu fiable. Cependant cette approche nécéssite de rattraper l'erreur dans le cas où le niveau test n'est pas accessible (et donc renvoie une mauvaise valeur pour OK), pour cela il faut définir une limite maximale pour le nombre de niveau.

- **moveball.js** :
  Ce fichier contient toute la gestion des déplacement de la bille, la gestion du score et la gestion des blocs *turn*.
  Les blocs *turn* sont très sensibles aux lags, il faut donc prévoir beaucoup de rattrapage d'erreurs liées à des mauvaises positions.
  Pour le score il y a simplement *checkMoveGroup* qui vérifie la validité du déplacement demandé et permet donc de créer .