Stanley Regnard

THEME BREAKER

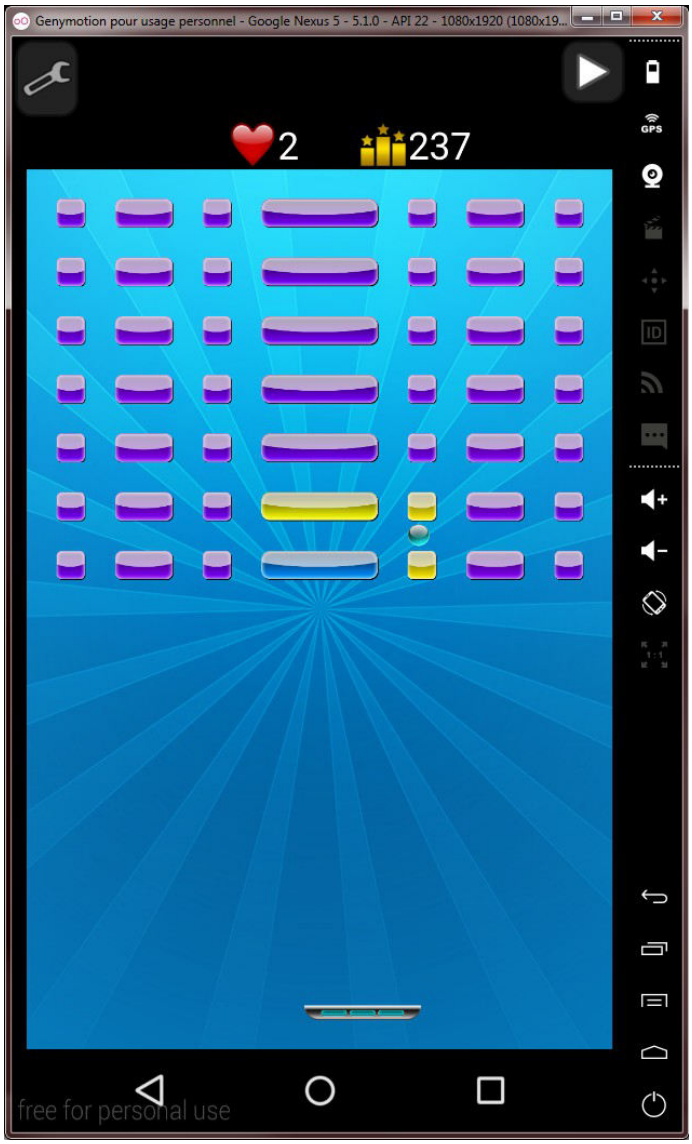
Guide utilisateur

08/06/2015

Stanley Regnard

« Theme Breaker », est une application développée sous Android, elle est inspirée du très célèbre type de jeu vidéo « Casse-briques ».

Le but du jeu étant simple, vous avez une raquette, une balle, et des briques : vous devez envoyer la balle sur les briques à l'aide de la raquette et de rebonds afin de casser ceux-ci, d'où l'appellation « casse-briques » ; en plus de cela vous devez faire attention à ne pas laisser tomber la balle tout en bas, ou bien vous perdrez une vie ; et si vous n'avez plus de vie, vous perdez la partie. La partie est gagnée lorsque toutes les briques sont détruites.



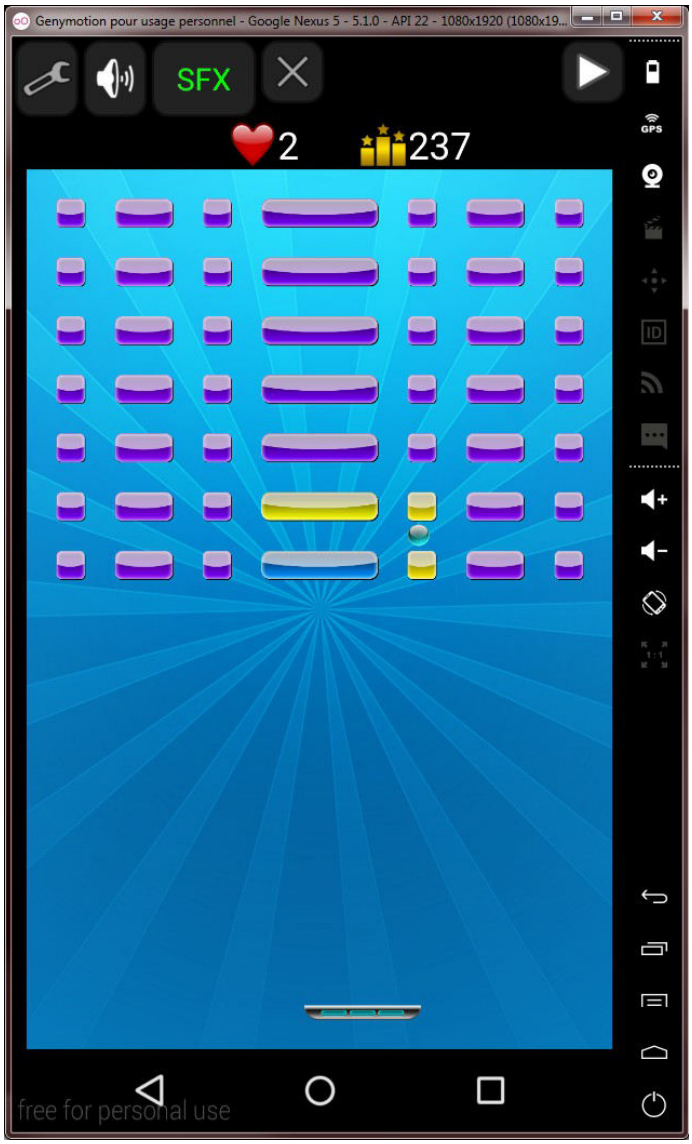
Voici une capture d'écran du jeu « Theme Breaker », on peut constater que les règles du jeu sont rester aux bases, une raquette, une balle, des briques, un compteur de vie ; et en ajout, un compteur de score, qui pour l'instant n'a pas d'autre fonctionnalité que d'afficher un score final à la fin de la partie, peut-être dans le futur il sera possible de conserver les meilleurs scores afin de tenter de les surpasser, voire les comparer avec d'autres utilisateurs. À savoir qu'il y a plusieurs niveaux disponibles avec un système de progression.

08/06/2015

Si vous avez suivi, ou que vous connaissiez déjà les règles, alors vous savez que vous devez utiliser la raquette pour envoyer la balle un peu partout, la question qui se pose est donc, comment la bouger ? Tout simplement, il suffit de déposer le doigt sur la partie de l'écran réservée au jeu, puis le glisser vers la droite, ou la gauche, et la raquette effectuera la même glissade !

Qu'est-ce que la partie de l'écran réservée au jeu, et à quoi sert l'autre partie ? La partie de l'écran réservée au jeu, est facilement reconnaissable car elle est représentée par le fond d'écran du niveau ; ou bien un autre moyen de savoir où elle se trouve, toute la partie de l'écran qui se trouve en dessous de la vie et du score est réservée au jeu.

Concernant l'autre partie, ça se divise en deux sous-parties, une sous-partie pour les options, et l'autre pour les informations sur le jeu qu'on a déjà vu, la vie et le score. Parmi les options, qui se trouvent donc en haut de l'écran, on trouve un bouton en force de clé à molette, qui permet d'activer ou de désactiver l'affichage des autres options :



Comme vous pouvez le voir, trois nouveaux boutons s'affichent, le premier en forme d'enceinte permet d'activer/désactiver la musique, si activé l'icône sera blanche, si désactivé elle

sera grise ; le deuxième bouton sur lequel est écrit « SFX » permet d'activer/désactiver les SFX (sound effects), si activé le texte sera vert, si désactivé il sera rouge ; le troisième bouton en forme de croix permet de quitter la partie pour retourner au menu de lancement de partie, le bouton BACK de votre téléphone peut être utilisé pour la même fonction.

Enfin le dernier bouton à l'extrême droite, permet de lancer, mettre en pause, et reprendre la partie, l'icône changeant selon le statut du jeu.

Maintenant que nous avons vu la partie la plus intéressante de l'application, voyons ce qu'il reste, **les menus** : l'application comporte au total 4 menus que vous aurez l'occasion d'utiliser.

Le premier menu étant bien sur le **menu principal**, présenté lors du lancement de l'application :

(Il est fort possible que le design des menus change)

Comme vous pouvez le constater, le menu principal vous donne deux possibilités, toucher le bouton « Jouer » pour vous retrouvez dans le menu de lancement de jeu, ou bien le bouton « Options » pour accéder au menu options.

Stanley Regnard

Intéressons-nous maintenant au **menu options** :

Comme vous pouvez le voir, vous pouvez cocher/décocher la musique et les SFX, afin de les activer/désactiver tout simplement.
Les boutons sont assez explicites sur ce qu'ils font, ils sont surtout là pour aider aux phases de test, « Débloquer tous les niveaux » permet donc d'accéder à tous les niveaux en forçant la progression à son maximum, et quant au bouton « Remettre à zéro la progression » il permet de redescendre la progression à son minimum.

Tout changement d'état ou modification de la progression est immédiatement pris en compte et enregistré en persistance (les données seront restaurées si l'application est fermée et relancée).

08/06/2015

Intéressons-nous maintenant au **menu de lancement de jeu** :

Ce menu offre deux possibilités et donnent deux informations.

La première possibilité étant d'accéder au menu de paramètres de jeu, la deuxième étant de lancer la partie et d'accéder à l'écran de jeu (ce qu'on a vu en premier lieu).

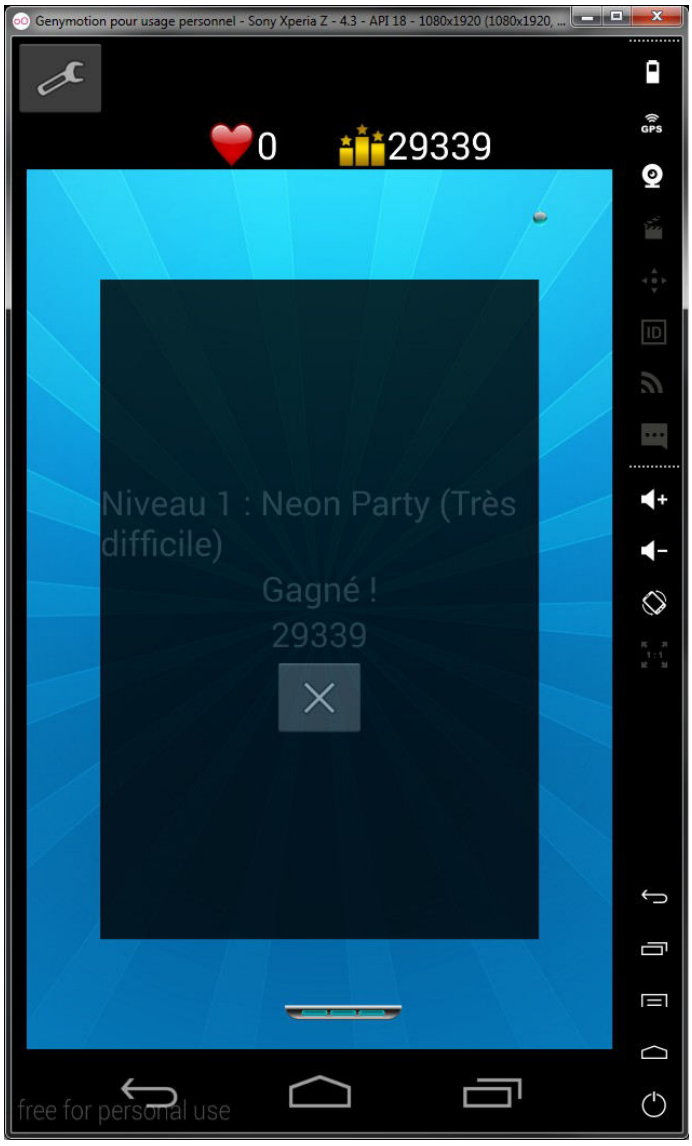
Les informations données sont simplement : le niveau préparée à être lancée, et le niveau de difficulté choisie.

Intéressons-nous maintenant au **menu de paramètres de jeu** :

Menu très simple, composé de deux menus déroulants : le premier permettant de sélectionner un autre niveau, parmi ceux disponibles selon votre progression ; le second permettant de choisir la difficulté à laquelle vous souhaitez jouer, les difficultés étant : Très facile, Facile, Normal, Difficile, Très difficile, Dément. Toutes débloquées par défaut.

Stanley Regnard

Je pense que tout est dit ! Vous devriez désormais tout savoir sur ce que propose l'application et profitez pleinement du jeu, bien sûr, c'est à vous de découvrir les subtilités du gameplay !



08/06/2015

Stanley REGNARD

THEME BREAKER

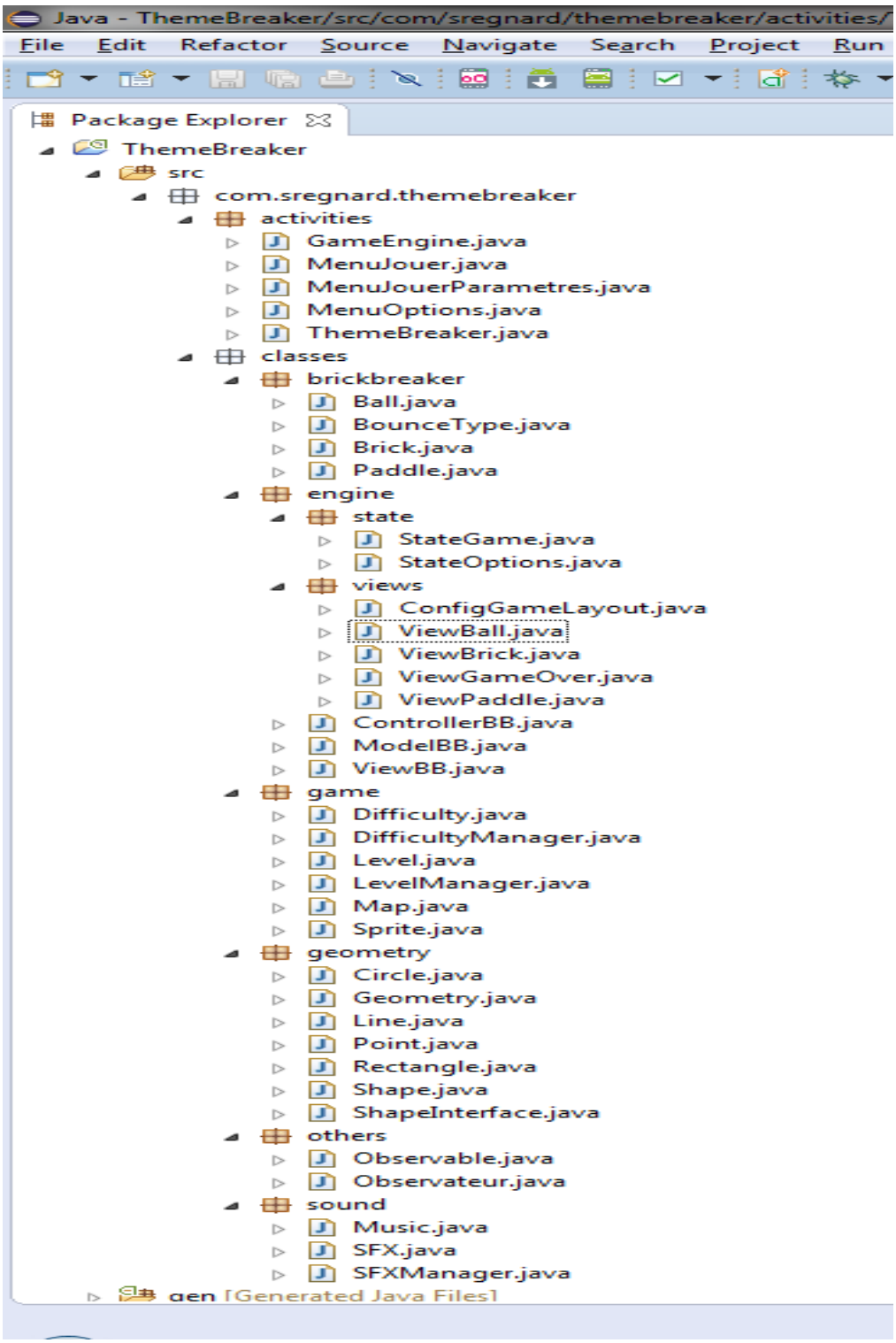
Guide développeur

08/06/2015

Stanley REGNARD

Si ce n'est pas déjà fait, je vous conseille de d'abord lire le guide utilisateur, qui apporte un point de vue plus globale à l'application, et qui par conséquent peut aider à la compréhension lors de mes explications dans ce guide développeur.

Je vais donc vous présenter l'application du côté du développement, la première chose que j'ai faite, et donc à faire ici, est de s'intéresser à l'organisation générale du projet, puisqu'une simple image peut parfois valoir mille mots, voici un aperçu de mon projet :



08/06/2015

Stanley REGNARD

Tout d'abord, comme vous pouvez le voir, j'ai tenté de développer mon application en évitant au maximum d'utiliser la langue française (mise à part les commentaires), j'ai la fâcheuse tendance à coder en « franglais ». **Toutes les classes et variables ont donc été nommées en anglais.**

Autre précision, j'écris ce guide développeur car je crains de ne pas avoir le temps de le faire plus tard, donc entre le moment où j'écris et celui où je présenterai mon projet **il est possible que le code voire son organisation change.** Bien sûr dans le but de l'améliorer.

Retour au vif du sujet, tous les packages sont ici déroulés pour voir toutes les classes. Comme on peut le voir, la première étape de mon organisation était de clairement séparer les activités des « simples » classes, je tenais à séparer le « visible » de « l'invisible ».

Commençons par les activités, si vous avez lu le guide utilisateur vous savez alors que l'application comporte 4 menus en plus du jeu en lui-même. On peut constater que le package « activities » comporte 5 classes, chacune étant une activité :

(Voir le guide utilisateur pour des aperçus en image de chacun)

GameEngine.java ; Moteur du jeu, je n'ai pas trouvé de nom plus approprié pour l'activité comportant le jeu

MenuPlay.java : Correspond au menu de lancement de partie

MenuPlayParameters : Correspond au menu de paramètres de jeu

MenuSettings : Correspond au menu options

ThemeBreaker.java : Correspond au menu principal

08/06/2015

Le menu principal en lui-même n'a rien de bien intéressant, mais la classe qui le contient **ThemeBreaker.java** si. Si cette classe porte le nom de l'application c'est parce que elle en est la « classe principale », c'est elle qui contient les variables globales à l'application, qui sont les suivantes :

```
private final String DATA_FILENAME = "BrickBreaker_Data";
private SharedPreferences data;
private static Editor editor;

public static int LEVEL; // Le dernier niveau sélectionné
public static int LEVEL_PROGRESSION; // Le dernier niveau débloqué
public static int DIFFICULTY; // La dernière difficulté sélectionnée
public static boolean SFX; // Les SFX sont-ils activés ?
public static boolean MUSIC; // La musique est-elle activée ?

public static Music backgroundMusic;
```

Les trois premières lignes concernent directement le fichier de sauvegarde (BrickBreaker_Data), qui n'est lu (data) et édité (editor) que dans (ou à travers) cette classe.

Les cinq lignes suivantes représentent des variables utilisées à travers toute l'application, leurs valeurs sont récupérées dans le fichier de sauvegarde et y sont sauvegardées à chaque modification.

Enfin la dernière ligne permet d'avoir une musique en fond commune aux menus.

C'est aussi ici que sont initialisés les modèles, les niveaux, et les difficultés :

```
// Initialise les niveaux & les niveaux de difficultés

Level.init();
Difficulty.init();
```

Rien de plus à dire sur cette classe.

Le menu options, ou **MenuSettings.java**, est concrètement une interface permettant de modifier 3 des 5 valeurs sauvegardées sur le fichier, c'est-à-dire l'activation ou non de la musique, de même pour les SFX, et le niveau de progression. Elle lit et modifie ces valeurs à travers ThemeBreaker.java

Le menu de lancement de partie, **MenuPlay.java**, en plus d'apporter un simple menu avec deux boutons, se contente de lire et afficher le niveau et la difficulté actuels à l'aide de ThemeBreaker.java

Le menu de paramètres de jeu, **MenuPlayParameters.java**, permet de modifier 2 des 5 valeurs restantes, à travers ThemeBreaker.java il lit et modifie le niveau sélectionné et la difficulté sélectionnée.

Enfin viens **GameEngine.java**, qui si ce n'est la tête elle représente le cœur de l'application, c'est autour de cette activité que toutes les classes tournent, voyons d'abord en détail le package « classes ».

Cette « deuxième partie » du projet est plus grande que la première, ainsi que plus complexe. Elle est-elle même divisée en plusieurs packages, afin de bien distinguer qui fait quoi, voyons les différents packages :

• « brickbreaker » : à l'intérieur de ce package j'ai placé ce que je pense être des éléments propres à un jeu du style brickbreaker, c'est pourquoi on y trouve les classes suivantes :

- Ball.java : contient les informations nécessaires à la balle (extends Circle)
- Brick.java : contient les informations nécessaires à une brique (extends Rectangle)
- Paddle.java : contient les informations nécessaires à la raquette (extends Rectangle)
- BounceType.java : contient les différents types de rebond possible

• « game » : dans celui-ci se trouvent les éléments que l'on retrouverait dans la plupart des jeux :

- Difficulty.java : définition d'une difficulté
- DifficultyManager.java : classe « static » qui a pour but de gérer toutes les difficultés existantes
- Level.java : définition d'un niveau
- LevelManager.java : même principe que DifficultyManager mais pour les niveaux
- Map.java : C'est ici que sont définies les différentes cartes
- Sprite.java : C'est ici que sont définies les différents modèles

• « geometry » : toutes les classes concernant la géométrie :

- Circle.java : définition d'un cercle (extends Shape)
- Line.java : définition d'une droite
- Point.java : définition d'un point, je sais que Java propose une classe Point de base mais j'ai préféré écrire la mienne
- Rectangle.java : définition d'un rectangle (extends Shape)
- Shape.java : définition d'une forme (implements ShapeInterface)
- ShapeInterface.java : définit les méthodes que doit implémenter une forme
- Geometry.java : contient les méthodes impliquant au deux formes et autres calculs géométriques

• « others » : contient les classes Observable et Observateur, je sais que Java en propose de base mais j'ai pris l'habitude d'écrire les miennes depuis longtemps :

- Observable.java : implémenté par les classes devant être surveillée par une autre
- Observer.java : implémenté par les classes devant surveiller une autre

• « sound » : contient les classes manipulant le son du téléphone :

- Music.java : C'est ici que sont définies les différentes musiques et leurs méthodes
- SFX.java : C'est ici que sont définis les SFX et les méthodes permettant de les jouer
- SFXManager.java : sert de manager, c'est elle qui regarde si le volume des SFX doit être muté ou pas ; et de OnCompletionListener, c'est elle qui gère un SFX lorsqu'il s'est « terminé »

• « engine » : Oui je l'ai gardé pour la fin ! Je l'ai nommé « engine » tout simplement car il représente le cœur du moteur de jeu, voyons son contenu :

- ModelTB.java : implements Observer, Observable
- ViewTB.java : implements Observer
- ControllerTB.java : implements Observer, Observable
- « state »
- « views »

On peut voir que je me base donc sur un MVC, Modèle-Vue-Contrôleur, déclaré comme tel dans GameEngine.java :

```
Sprite.init(this);
new SFXManager(this);

model = new ModelTB(this);
controller = new ControllerTB(model, this);
view = new ViewTB(model, controller, this);
```

À savoir que, le modèle et le contrôleur observent les classes comprises dans le package « state » ; quant à la vue elle observe à la fois le modèle et le contrôleur. On est donc dans un MVC « push », puisque la vue va réagir aux changements d'état du modèle et du contrôleur.

Le modèle comporte bien sûr toute la « logique » du jeu, le contrôleur gère les interactions utilisateurs, et c'est dans la vue qu'est géré l'affichage.

Le package « state » comprends deux classes, StateGame.java et StateOptions.java, le premier de gérer l'état du jeu, savoir s'il est en pause, en cours, etc.. ; le deuxième permet de gérer l'affichage du « sous-menu » lorsqu'on est en jeu, voir le guide utilisateur pour savoir de quoi il s'agit.

Le package « views » contient les vues inclues dans la vue principale :

- ConfigGameLayout.java : n'est pas une vue mais effectue les calculs permettant d'adapter la taille du layout contenant le jeu à la taille de l'écran
- ViewBall.java : gère l'affichage de la balle
- ViewBrick.java : gère l'affichage d'une brique
- ViewPaddle.java : gère l'affichage de la raquette
- ViewGameOver.java : gère l'affichage de l'écran de fin de partie

Stanley REGNARD

Je pense avoir dit tout ce qui me semblait important, il y a des points qui mériteraient sûrement d'être approfondis, comme la gestion des collisions, gestion des sons, gestion des modèles, le déplacement de la raquette etc.. Mais je ne sais pas si ça a sa place ici.

08/06/2015