BENNECER Maëlla
MORVAN Alexandre

IUT de Vannes
Département INFORMATIQUE
LP S2IMa

Année 2017-2018

UE3 - Mobile

RAPPORT D'ARGUMENTATION SUR LA CONCEPTION

« FLAPPY FLAG »

Destinataire

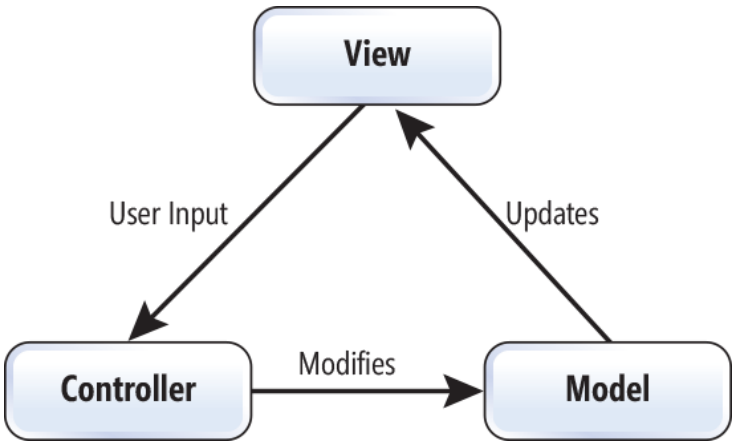
IUT DE VANNES
Mr. LE LAIN Matthieu

I. Introduction

L'application que nous avons choisie de développer se nomme « Flappy flag ». Cette dernière a été créée dans le cadre du développement d'un « serious game » dont le but est d'apprendre quelque-chose à l'utilisateur. Flappy flag est une application qui va permettre d'apprendre au joueur les drapeaux du monde entier.

II. Etude des différents patterns

1) MVC



MVC est l'acronyme de « Modèle, Vue, Contrôleur », c'est le nom donné à une manière d'organiser son code. C'est une façon d'appliquer le principe de séparation des responsabilités, en l'occurrence celles du traitement de l'information et de sa mise en forme.

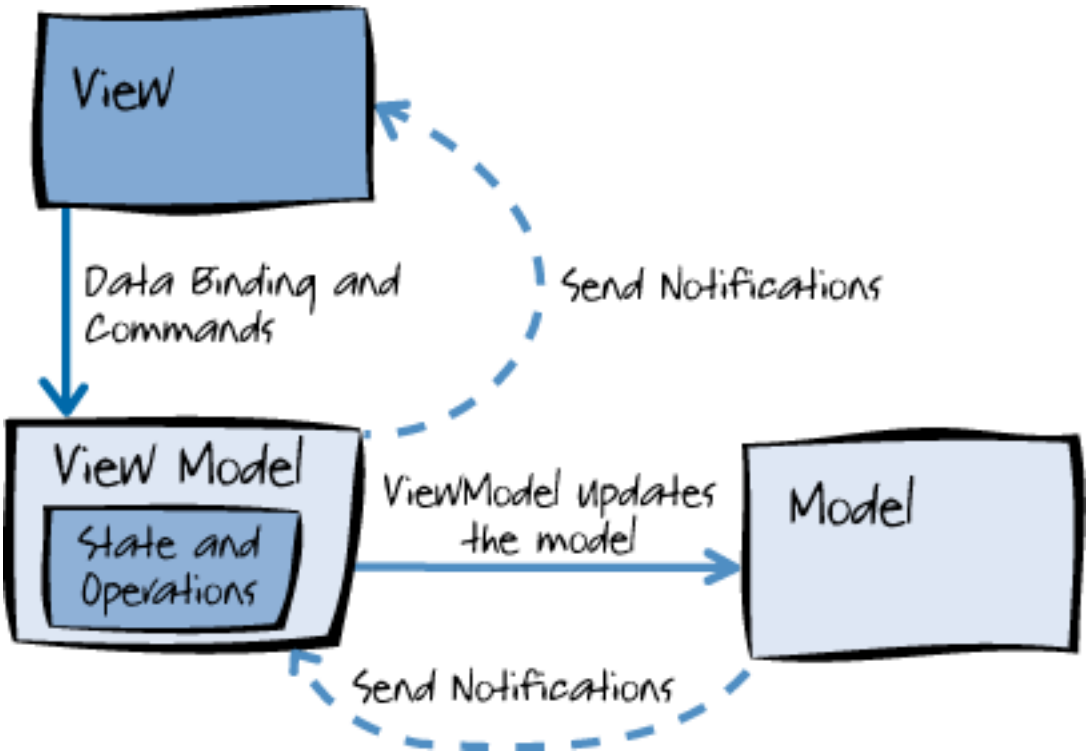
Modèle : C'est là que se trouvent les données. Il s'agit en général d'un ou plusieurs objets Java.

Vue : C'est de la présentation, c'est comment on veut que la donnée soit présentée à l'utilisateur.

Contrôleur : Il permet de faire le lien entre la vue et le modèle lorsqu'une action utilisateur est intervenue sur la vue. C'est cet objet qui aura pour rôle de contrôler les données.

2) MVVM

MVVM est un design pattern ou patron de conception, très souvent utilisé ces derniers temps par des bibliothèques Javascript et Java. A l'origine, MVVM aurait été introduit par Microsoft.



Model (Modèle en français) : le modèle contient les données. Généralement, ces données proviennent d'une base de données ou d'un service externe comme un API.

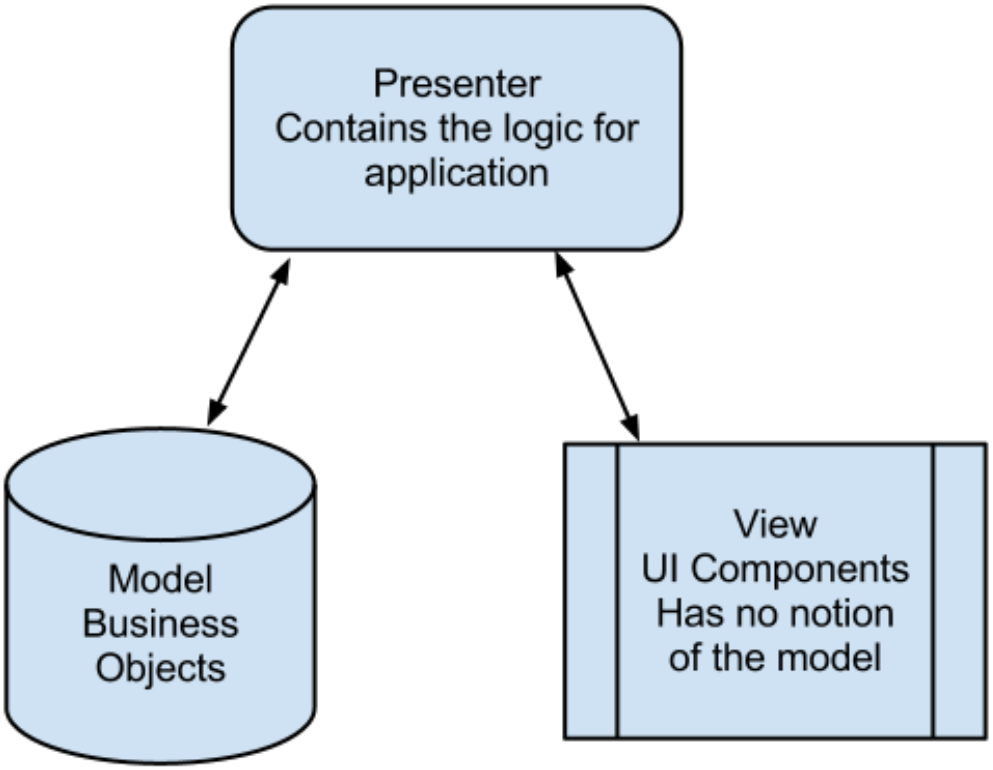
View (Vue en français) : la vue correspond à ce qui est affiché (la page web dans notre cas). La vue contient les différents composants graphiques (boutons, liens, listes) ainsi que le texte.

ViewModel (Vue-Modèle en français) : ce composant fait le lien entre le modèle et la vue. Il s'occupe de gérer les liaisons de données et les éventuelles conversions. C'est ici qu'intervient le binding.

L'idée à retenir avec MVVM est simple : la vue ne doit jamais traiter de données. Elle s'occupe uniquement de les afficher. Le View-Model aura en charge les conversions et les accès au modèle de données.

3) MVP & VP

Le modèle-vue-présentation (en abrégé MVP, de l'anglais model-view-presenter) est un patron d'architecture, considéré comme un dérivé du patron d'architecture modèle-vue-contrôleur. Il garde les mêmes principes que MVC sauf qu'il élimine l'interaction entre la vue et le modèle parce qu'elle sera effectuée par le biais de la présentation, qui organise les données à afficher dans la vue.



VP est une version simplifiée du pattern MVP, il ne possède juste pas de partie modèle.

III. Notre choix : une version simplifiée du pattern MVP

Nous avons choisi le pattern VP car nous n'avons pas besoin de model. Nous avions imaginé d'utiliser un fichier JSON pour stocker les différents drapeaux, donc nous ne stockons pas les données dans une base de données.

Le fichier JSON est structuré de la sorte : nous avons un tableau nommé "question" contenant des drapeaux nommés "id". Ces derniers sont associés à 3 réponses correspondant aux choix possibles pour chaque question, ainsi qu'une entrée appelée "response" qui indique quelle réponse est la bonne.

`Aperçu du fichier questionsFr.json

Ce fichier est appelé dans la classe main uniquement, c'est là que le JSON est chargé. À partir des données chargées, nous pouvons alors afficher les questions avec les drapeaux correspondants.

Nous avons donc un dossier contenant les classes java, un dossier contenant les layout (les vues en xml) et des dossiers contenant les drawables, images, etc.

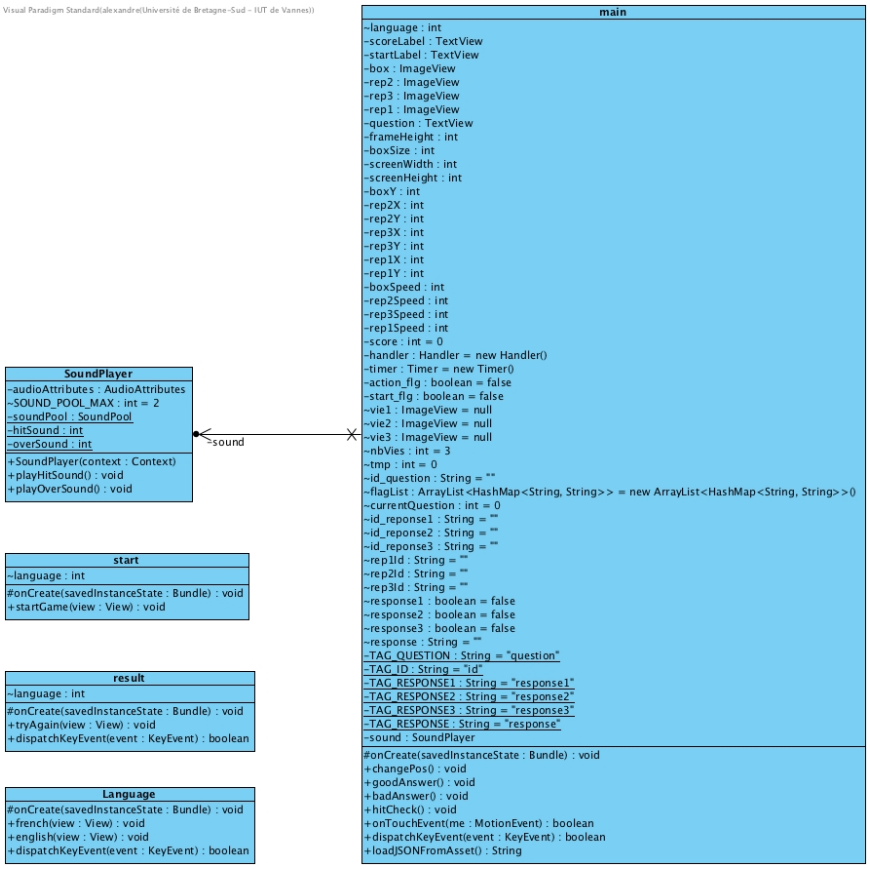


Diagramme de classe

IV. Problèmes rencontrés

Il n'était pas évident de mettre en place le système de mouvement que vous pourrez observer dans la vidéo. En effet, il fallait imaginer quel système nous allions devoir mettre en place pour les mouvements des drapeaux, ainsi que les mouvements de notre personnage. Une autre partie non évidente était la « hitbox », c'est à dire la détection de percussion entre le personnage et le drapeau. Il a fallu imaginer et créer plusieurs conditions pour accepter ou non le point selon le drapeau touché. Il fallait pouvoir le percuter par la gauche mais aussi par le haut et le bas. Ces problèmes sont résolus.

V. Evolutions possibles

L'application est entièrement fonctionnelle, nous n'avons pas relevé de bug quelconque. Par contre, nous pouvons améliorer la partie graphique, au niveau des couleurs et des layouts en général, la disposition des éléments, peut-être ajouter des paramètres modifiables par l'utilisateur (par exemple choix des couleurs, du personnage, etc.). Enfin, nous avons utilisé un json pour le choix des drapeaux. Il est donc possible de faire évoluer notre quiz en plusieurs quiz différents. C'est à dire qu'on pourrait créer un nouveau JSON (prenons l'exemple de résolutions de calcul mental sous le même principe que les drapeaux). Ainsi, après le choix de la langue dans le jeu, nous pourrons choisir quel quiz nous voulons faire. Notre application a donc un certain potentiel.

VI. Conclusion

Nous avons réussi à développer un « serious game », qui nous permet d'apprendre les drapeaux du monde, et qui est en plus ludique. Nous voulions nous démarquer en ne proposant pas qu'un quiz simple. Nous comptons modifier l'application pour ensuite la publier sur le play store.