Houla project.

Rapport d'avancement

1. Le projet

1. Présentation générale

Membres du projet :

- Lucas Clementeï
- Jérémy Telliez
- Grégoire Thibaud
- Stéphanie Toitot

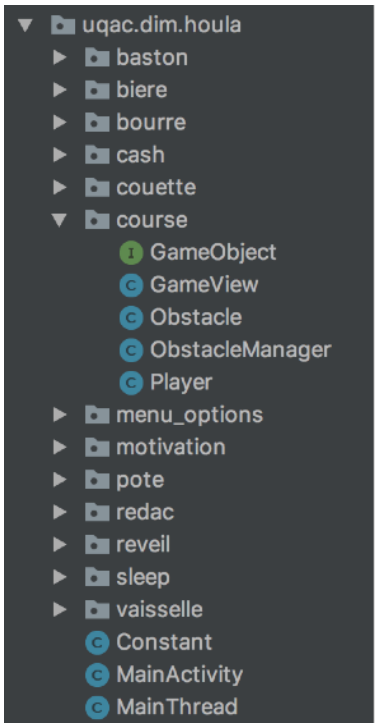
Houla est une application Android basée sur le concept des Warioware Game. Le joueur partage les galères du quotidien d'un étudiant en jouant à des mini-jeux très rapides. L'application comporte une dizaine de mini-jeux dont l'objectif est d'interagir avec le téléphone dans un laps de temps court, faisant appel à des compétences d'agilité et de réflexe. Le temps de chaque jeu baisse au fur et à mesure de l'avancée dans le jeu (10 secondes au départ). L'objectif est d'arriver à la fin du jeu en réussissant le maximum de mini-jeux.

Vous pouvez retrouver la liste de tous les mini-jeux sur dans le pdf « Présentation du Projet ».

2. Les solutions techniques retenues

Notre projet consistant en un enchainement de plusieurs mini-jeux proposés à la suite à l'utilisateur, nous avons décidé de créer une arborescence prévoyant un package par mini-jeu. Chaque package contient une activité ainsi que les classes nécessaires au fonctionnement du jeu.

De plus, nous avons en dehors de tous les packages une classe utilitaire Constant qui contient diverses informations générales telles que la taille de l'écran de l'appareil etc.



Constant.java

```java
package uqac.dim.houla;

import java.util.Random;

/**
 * Created by Quelqu'un on 17/03/2018.
 */

public class Constant {
    public static int SCREEN_WIDTH;
    public static int SCREEN_HEIGHT;

    public static int randomInt(int borneInferieure, int borneSuperieure)
    {
        Random r = new Random();
        int res = r.nextInt( bound: borneSuperieure - borneInferieure) + borneInferieure;
        return res;
    }
}
```

Pour construire la partie graphique des mini-jeux, nous avons utilisé deux approches différentes. La première consiste à utiliser les fichiers xml et les modifier dynamiquement, comme nous l'avons fait en cours. C'est la méthode la plus simple et la plus rapide à mettre en œuvre. Cependant, elle présente quelques limites pour des applications très graphiques (telles que le jeu « Course » par exemple). Pour cette raison, nous avons décidé d'utiliser des canevas (vous pouvez le voir sur la

capture d'écran présente sur la page suivante) qui sont plus compliqués à mettre en oeuvre, mais plus complets (et tout de même plus simple que de l'OpenGL pur).

Dans notre exemple, nous utilisons le jeu « Course ».
Le noyau de cette approche est la classe MainThread qui va créer le thread et appeler les fonctions draw et update de la gameView passée en paramètre lors de sa création.
La fonction draw gère l'affichage graphique des éléments et la fonction update les calculs logiques.
Il y a donc un principe de GameObject pour ces mini-jeu, c'est-à-dire que chaque élément hérite de la classe GameObject ce qui les oblige à redéfinir les méthodes draw et update.
La fonction gameView est personnalisée en fonction des mini-jeu mais va toujours avoir pour but de récupérer les entrées comme les appuis ou les glissements des doigts de l'utilisateur sur l'écran. Elle sert également à délimiter le début et la fin du jeu.

```
@Override
public void run() {
    long startTime;
    long timeMillis = 1000/MAX_FPS;
    long waitTime;
    int frameCount = 0;
    long totalTime = 0;
    long targetTime = 1000/MAX_FPS;

    //Game Loop
    while(running) {
        //Peut aussi utiliser milliTime mais moins precis et moins couteux en ressources
        startTime = System.nanoTime();
        canvas = null;

        try {
            canvas = this.surfaceHolder.lockCanvas();

            synchronized (surfaceHolder) {
                //Ces deux lignes sont le coeur de la boucle de jeu
                //effectue les calculs logiques
                this.gameView.update();
                //Dessine l'ecran
                this.gameView.draw(canvas);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (canvas != null) {
                try {
                    surfaceHolder.unlockCanvasAndPost(canvas);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        timeMillis = System.nanoTime() - startTime/1000000;
        waitTime = targetTime - timeMillis;
```

Obstacle est la classe régissant les obstacles (carrés noirs) qui vont être des images choisies au hasard (bananes, poubelles, personnes agées) parmi les images du projet.
La classe player gère le joueur.
ObstacleManager sert de contrôleur de jeu, c'est elle qui détecte les collisions, qui génère les obstacles et les fait avancer.

2. Avancée

1. Répartition du travail

Nous avions prévus à l'origine du projet de suivre les étapes suivantes :

- Imagination des mini-jeux
- Mise en place d'une charte graphique et musicale, création des éléments graphiques à intégrer ensuite dans les activités
- Création de maquette pour chaque activité
- Création de l'UML
- Création de l'arborescence (classes / packages)
- Répartition des tâches en fonction de l'UML
- Codage des classes
- Tests unitaires
- Tests de système (test de l'ensemble de l'application sur différents devices)
- Création d'un exécutable

Nous nous trouvons actuellement dans la phase de codage des classes. Pour chacun puisse toucher aux différentes facettes de la conception des mini-jeux, nous avons décidé de nous répartir le travail de manière équitable : sur les 12 mini-jeux prévus, chacun doit en réaliser 3. Cette répartition du travail n'est certes pas la plus efficace, mais nous permet à chacun d'aborder l'intégralité du processus de conception, ce qui a un intérêt pédagogique intéressant.

Notre objectif est de finir de coder les 3 mini-jeux respectifs avant lundi 16 avril afin d'avoir une semaine pour finir les ajustements et effectuer les tests. En parallèle, nous continuons à développer la structure générale du projet (communication entre les mini-jeux, scores etc).

2. État actuel et prévisions

Pour l'instant, la structure générale du projet ainsi que deux mini-jeux sont finis. Voici les étapes suivantes pour Jérémy :

- Il est prévu de rajouter une brève explication du mini-jeu en image au début de celui-ci.
- L'ajout d'une barre décomptant le temps restant en plus du nombre.
- Ajout d'images pour rendre le tout plus agréable visuellement (c'est-à-dire remplacement des carrés)
- Début du prochain mini-jeu
- Le système de score sera rajouté quand on aura assez de mini-jeux pour faire une vrai partie (environ 4-5).

Pour Grégoire :

- Ajout d'images pour rendre le mini-jeu réveil plus agréable visuellement
- Adaptation responsive
- Début du mini-jeu suivant

Pour Lucas :

- Intégration du gestionnaire de mouvements
- Interface graphique
- Début du mini-jeu suivant

Pour Stéphanie :

- Résolution de bugs d'Android Studio et git avec le reste de l'équipe
- Avancer dans le mini-jeu et commencer le suivant

Nous pensons être de manière générale plutôt bien dans notre timing, maintenant que les bases ont été mises en place, nous allons pouvoir avancer plus vite sur la conception des mini-jeux.