






## PREUVE MAITRISE DES COMPETENCES

### Documentation : (dossier documentation)

- 1- Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert. 
- 2- Je sais faire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application. 
- 3- Je sais concevoir un diagramme UML intégrant des notions de qualité et correspondant exactement à l'application que j'ai à développer. 
- 4- Je sais décrire un diagramme UML en mettant en valeur et en justifier les éléments essentiels 

### Code :

- 1- Je sais utiliser les Intent pour faire communiquer deux activités. 

*Dans MainActivity*

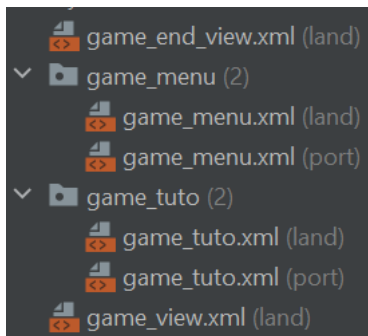
```
public void VueOption(View view){  
    Intent intent = new Intent( packageContext: this, OptionActivity.class);  
    startActivity(intent);  
}
```

- 2- Je sais développer en utilisant le SDK le plus bas possible. 

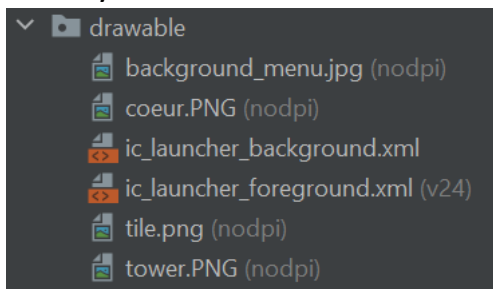
```
defaultConfig {  
    applicationId "com.androidtowerdefense"  
    minSdk 16
```

Le plus petit SDK est utilisé pour toucher 100% des appareils

3- Je sais distinguer mes ressources en utilisant les qualifier



Nous utilisons les qualifieurs concernant l'orientation du téléphone pour nos layouts.



Nous utilisons également le qualifier 'nodpi' car il s'agit de ressources indépendantes de la densité. Le système ne met pas à l'échelle les ressources marquées avec ce qualificatif, quelle que soit la densité de l'écran actuel.

4- Je sais faire des vues xml en utilisant layouts et composants adéquats



Nous avons principalement utilisé le ConstraintLayout, LinearLayout et TabLayout. Nous avons également utilisé les composants adéquats. Pour plus de détails voire le code de l'application

5- Je sais coder proprement mes activités, en m'assurant qu'elles ne font que relayer les évènements



Nos activités ne font que relayer les évènements. Elles n'exercent pas de responsabilités supplémentaires.

6- Je sais coder une application en ayant un véritable métier



Notre métier est bien conceptualisé en ayant chaque classe avec une seule responsabilité

7- Je sais parfaitement séparer vue et modèle



Notre modèle et notre vue sont séparé. Notre modèle n'a pas besoin de connaître la vue.

8- Je maîtrise le cycle de vie de mon application



Nous maîtrisons le cycle de vie de notre application Nous n'avons eu besoin d'utiliser que onCreate et onSaveInstanceState (onSaveInstanceState se trouve dans MainActivity).

9- Je sais utiliser le findViewById à bon escient



Nous n'utilisons le findViewById que quand nous avons besoin de récupérer des éléments de notre vue. De plus, si nous avons besoin de réutiliser plus tard un élément récupéré via findViewById, nous stockons la référence dans une variable.

10- Je sais gérer la persistance légère de mon application



La persistance légère est gérée automatiquement grâce aux id données dans les fichiers xml. De plus, nous utilisons onSaveInstanceState.

11- Je sais gérer la persistance profonde de mon application



Pour faire de la persistance profonde, nous utilisons les préférences. La responsabilité de sauvegarde ainsi que de chargement des données est délégué au RankingManager. Pour ce faire nous sauvegardons les scores de notre tableau des scores dans des JSONObject pour ensuite placer ces JSONObject dans un JSONArray.

## 12- Je sais afficher une collection de données

Nous utilisons une RecyclerView pour afficher les scores et nous passons à l'Adapter une List<ScoreRanking>. Nous affichons donc une collection de ScoreRanking.

## 13- Je sais coder mon propre adaptateur

*package modelandroid/view*

```
public class MyAdapter extends RecyclerView.Adapter
```

## 14- Je maîtrise l'usage des fragments

Premièrement, nous avons utilisé les fragments avec notre RecyclerView via un ItemClickListener. Vous pouvez afficher le détail d'un score du tableau des score en cliquant sur celui-ci. Le fragment se nomme RankingDetailFragment

Deuxièmement, notre menu de jeu possède un TabLayout. Pour le changement d'onglet nous avons utilisé un FragmentContainerView, nous permettant de changer le Fragment affiché en fonction de l'onglet sélectionné. MenuFragment correspond à l'affichage de la vue du menu et RankingFragment correspond à l'affichage de la vue du tableau des scores.

## 15- Je maîtrise l'utilisation de Git

Nous avons utilisé le gitlab de l'UCA pour versionner notre projet Android. Nous avons constamment poussé nos travaux sur gitlab tout en corrigeant d'éventuels conflits.


De plus, pour être informé de nos différents commits sur gitlab, nous avons utilisé une intégration 'webhook'


### Active integrations

Integration	Description	Last updated
✓ <a href="#">Discord Notifications</a>	Send notifications about project events to a Discord channel.	1 month ago

Nous l'avons intégré à un salon discord

POSTE SUR #GITLAB-HISTORY

**history-AndroidTowerDefense**  
Créée le 3 févr. 2022 par Mage-Ox#3942



**NOM**  
history-AndroidTowerDefense


**SALON**  
#gitlab-history


[Supprimer](#)[Copier l'URL du webhook](#)[Suppression de webhook](#)


# gitlab-history


changes/  
37fc79ca: Explication diagramme de classes (en cours) + corrections diagramme de classe... - ugo vignon  
23/03/2022

25 mars 2022





**history-AndroidTowerDefense** BOT Hier à 16:50

**ugvignon**  
ugvignon pushed to branch [main](#) of [ugvignon / AndroidTowerDefense](#) ([Compare changes](#))  
[480a68e8](#): légères modifs - ugo vignon  
Hier à 16:50

**history-AndroidTowerDefense** BOT Hier à 23:13

**ugvignon**  
ugvignon pushed to branch [main](#) of [ugvignon / AndroidTowerDefense](#) ([Compare changes](#))  
[78d37025](#): explication DDclasses terminée et corrections du DDclasses et du code - ugvignon  
Hier à 23:13

+ Envoyer un message dans #gitlab-history

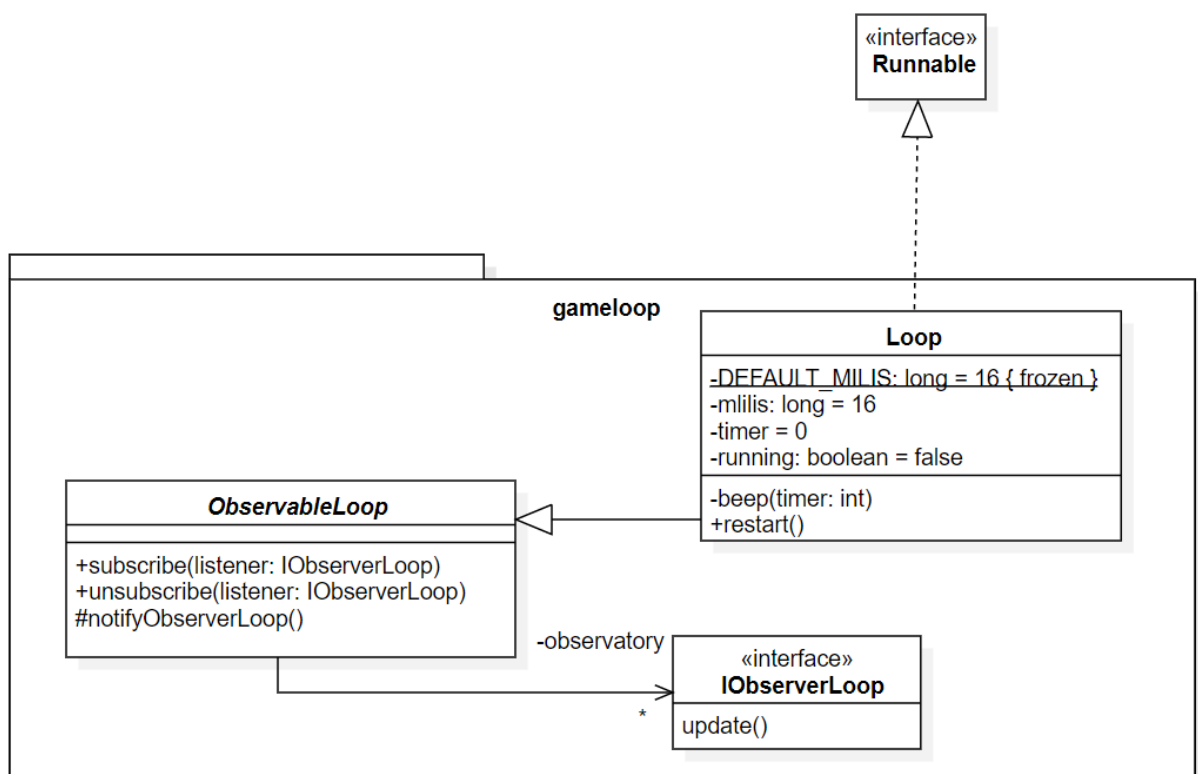
   

### Application :

1- Je sais développer une application sans utiliser de librairies externes. 🍴

2- Je sais développer une application publiable sur le store. 🍴

3- Je sais développer un jeu intégrant une boucle de jeu threadée observable. 🍴



Pour démarrer la boucle de jeu :

*model/gamelogic/GameManager*

```
public void start() {
    loop.setRunning(true);
    Thread boucleThread = new Thread(loop);
    boucleThread.start();
}
```

Pour mettre en pause la boucle de jeu :

*model/gamelogic/gameloop*

```
@Override
public void run() {
    while(running) {
        try {
            sleep(millis);
            timer++;
            beep(timer);
            if(!running){
                synchronized (this) {
                    wait();
                }
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Pour relancer la boucle de jeu après la pause :

*model/gamelogic/gameloop*

```
public void restart(){
    synchronized (this) {
        notify();
    }
}
```

4- Je sais développer un jeu graphique sans utiliser de SurfaceView.



Nous n'avons pas utilisé de SurfaceView