

# Rapport Android

## Exercice

T\_6\_21

Réalisé par :

Asmae NEDDAY

Encadré par :




Mme .ABIK Mounia



## Remerciements

Mes remerciements sincères vont à notre encadrante Mme ABIK Mounia pour ses conseils et le temps qu'elle nous a consacré, ses directives précieuses, et la qualité de son suivi et de son appui tout au long du processus.

## Table des matières

-  Vision globale sur l'application
-  Vision technique sur l'application
-  Calcul des statistiques

## Vision globale sur l'application

L'exercice T\_6\_21 consiste à donner à l'enfant une image qu'elle doit préciser sa catégorie, par exemple si on lui présente une banane, puis un panier des légumes et un panier des fruits, l'enfant doit choisir le panier des fruits puisque la banane est une sorte de fruits.

Mon application se compose de quatre rubriques principales :

- Rubrique apprentissage
- Rubrique jeu
- Rubrique langue
- Rubrique paramètres de voix



### Rubrique apprentissage

Dans cette rubrique, on présente à l'enfant un menu de 6 sujets : les sujets des fruits, des légumes, des animaux, des poissons, des fournitures scolaires et des vêtements.

Ici l'enfant va apprendre ces éléments, avec un simple clic sur l'élément, il peut entendre le son qui indique de quoi s'agit l'élément.



### Rubrique jeu

Dans cette partie, on va voir à quel point l'enfant a saisi l'apprentissage.

On lui propose un menu de 3 sujets :

Le sujet numéro 1 : légumes & fruits

Le sujet numéro 2 : fournitures scolaires & vêtements

Le sujet numéro 3 : animaux & poissons

Dans chaque sujet, on présente à l'enfant un objet et on lui demande de faire glisser l'objet dans la catégorie qui convient

Dans le premier sujet, on lui présente l'image soit d'un fruit soit d'un légume et il doit faire glisser l'objet soit vers le panier des fruits soit vers le panier des légumes.

Dans le deuxième sujet, on lui présente l'image soit d'un type de vêtements soit d'un type de fournitures scolaires et il doit faire glisser l'objet soit vers le cartable soit vers l'armoire

Dans le troisième sujet, on lui présente l'image soit d'un poisson soit d'un animal et il doit faire glisser l'objet soit vers l'aquarium soit vers le zoo.



**\*\*** Dans chaque niveau, j'ai mis la consigne qui montre comment se fait le jeu. L'icône de la consigne est sous forme d'un petit garçon avec un point d'interrogation. L'enfant peut à chaque fois réentendre la consigne.



Pour chaque tentative de l'enfant, un audio va se lancer, un audio de type excellent si la tentative est correcte, et un audio de type encouragement si la tentative est fausse.

Si l'enfant s'attarde pour choisir la réponse, on lui envoie une animation sous forme d'un enfant avec un audio qui lui dit de continuer et qu'il est très proche du succès, c'est un audio de type bien.

### Rubrique langue

Dans cette rubrique, on propose à l'enfant un menu qui présente quatre langues : la langue française, la langue arabe, la langue anglaise et Darija .



### Rubrique paramètres de voix

Dans cette rubrique, on donne au tuteur la possibilité d'ajouter des audio pour les utiliser dans le jeu, il y'a aussi l'option d'importer un dossier des audio ou l'exporter pour le partager avec d'autres.

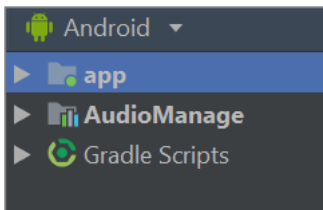


Sur chaque écran de l'application on fournit à l'enfant le bouton de retour, et on lui fournit aussi le bouton pour mettre la musique de background soit en mode 'on' soit en mode 'off'.

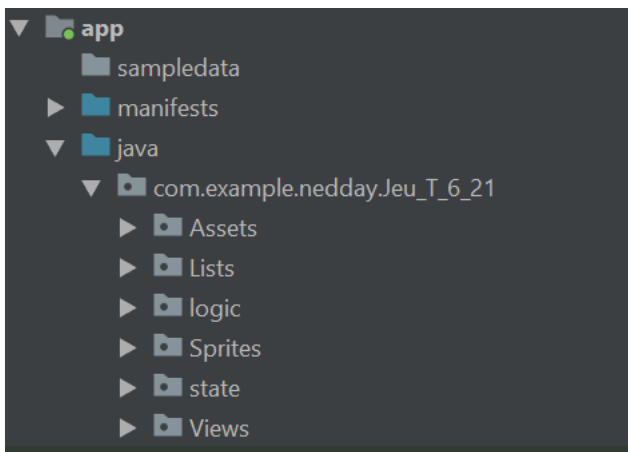


## Vision technique sur l'application

Le projet se compose du module 'app' plus le module AudioManage



Pour la structure du code java, je l'ai divisé en 6 packages



Le package Assets contient les objets de type image et audio qu'on va les utiliser.

Le package Lists contient les différentes classes des listes des objets : liste des fruits par exemple

Le package logic contient la classe où est codé la logique du jeu

Le package Sprites contient des types d'objets avec des caractéristiques, par exemple un sprite qu'on peut le draguer, on met **setStatic(false);**

Le package state contient les différentes activités et les android game de mon application

Le package Views contient les différents Screens

## ❖ Détails

La première activité qui se lance est `LoadingActivity`, c'est une simple image qui va apparaître quelques secondes avant le lancement du premier menu

Cette activité fait appel à `First`, elle est de type `android game`, j'ai utilisé cette classe pour initialiser plusieurs assets .

Puis la classe `First` fait appel à l'activité `PopupActivity`, cette activité présente le premier menu qui apparaît à l'enfant , dans cette activité , j'ai codé aussi la demande de permission de localisation qui va me permettre d'avoir la longitude et la latitude de l'emplacement de l'enfant , j'ai mis les variables longitude et latitude en tant que statique pour les stocker après dans une autre classe.

Selon le choix de l'enfant, c'est-à-dire l'id du bouton cliqué, on va faire un traitement. Si le choix était pour l'apprentissage, on va démarrer une intent vers l'activité `Menu_learn`, et pour chaque bouton cliqué, on a une variable qui s'appelle `niveau` où on stocke la valeur, puis ensuite dans la même fonction du listener, on fait appel à la classe « `MyGame_Activity` » qui est de type `android game` .

Dans la fonction `public Screen getInitScreen()` de la classe `MyGame_Activity` , on fait un switch selon la valeur de la variable `niveau` et on va retourner un screen.

On a pour cette partie, six screens.

```
switch (PopupActivity.niveau) {
    case 1:
        screen = new Learn_Legumes_Screen( game: this, logic, game_activity: this);
        break;
    case 2:screen = new Learn_Fruits_Screen( game: this, logic, game_activity: this);
        break;
    case 3:screen = new Learn_animals_Screen( game: this, logic, game_activity: this);
        break;
    case 4:screen = new Learn_Fish_Screen( game: this, logic, game_activity: this);
        break;
    case 5:screen=new Learn_Fournitures_Screen( game: this, logic, game_activity: this);
        break;
    case 6:screen = new Learn_Clothes_Screen( game: this, logic, game_activity: this);
        break;
```

Puis dans chaque screen, on a la fonction **private void putPlaceHolders()** où on va mettre nos sprites dans l'interface , et on va les manipuler par la fonction **void handleDragging(int x, int y, int pointer)**, à chaque fois qu'on drague un élément cette fonction s'exécute, c'est pourquoi on va faire une condition concernant le type du sprite .

```
if (s==sprite1 && !(manager.isMusicActive()))  
{Voice.im3_ar_3.play( volume: 1);}  
  
if (s==sprite2 && !(manager.isMusicActive()))  
{Voice.im3_ar_4.play( volume: 1);}  
if (s==sprite3 && !(manager.isMusicActive()))  
{Voice.im3_ar_5.play( volume: 1);}  
if (s==sprite4 && !(manager.isMusicActive()))  
{Voice.im3_ar_6.play( volume: 1);}
```

**\*\*Note :** puisque à chaque fois, l'enfant touche un sprite, il doit savoir de quoi il s'agit avec un audio .Si l'enfant clique sur un autre sprite , on désactive l'action d'envoyer un audio en utilisant la condition **if !(manager.isMusicActive())** , c'est-à-dire si il n' y a pas un audio qui est déjà en cours , on peut lancer un autre audio .

Pour le cas où l'enfant a choisi la partie jeu, on va entrer dans Menu\_jouer activité, c'est la même logique avec l'autre cas, et après on a aura l'orientation vers les screens, soit MyScreen ou MyScreen2 ou MyScreen3 ,selon la valeur d'une variable statique. Dans les screens de cette partie, j'ai utilisé la fonction **public void render(float deltaTime)** qui se répète tant qu'on est dans le même screen .J'ai mis dans le corps de cette fonction l'appel à la fonction job() où j'ai défini le traitement à faire.

Quand notre sprite va être contenu dans l'un des objets, on va voir si c'est correct ou pas et on va lancer un audio en fonction du résultat.

```
if (panier_legumes.contain(sprite.getX(), sprite.getY())) {  
    switch (this.logic.check()) {  
        case "legume":  
  
            sprite.setWidth((int) (game.getScreenWidth() / 6));  
            sprite.setHeight((int) (game.getScreenHeight() / 7));  
            sprite.setPosition((int) (game.getScreenWidth() / 1.3), (int) (game.getScreenHeight() / 1.3));  
            sprite.setDragged(false);  
            sprite.setStatic(true);  
            game.getCurrentScreen().  
                addSprite(new Sprite(Correct_faux.correct, (int) (game.getScreenWidth() / 1.74),  
win_score++;
```

## Calcul des statistiques

Quand l'enfant clique sur un bouton du Menu\_jouer , j'ai une fonction **Level()** où je remplit les champs : id de l'application, id de l'exercice, longitude, latitude, la date d'entrée et l'id de niveau puisque c'est toujours 1 du fait que j'ai un seul niveau de difficulté.

```
//j'initialise le GameStat()
```

```
gameStat = new GameStat()
```

```
//stocker date d'entree au level
```

```
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd ' ' HH:mm:ss");
```

```
String currentDate = sdf.format(new Date());
```

```
gameStat.setCreated_at(currentDate);
```

```
//stocker l'id de l'application , de l'exercice et du level
```

```
gameStat.setApp_id("2019_3_3_4");
```

```
gameStat.setExercise_id("T_6_21");
```

```
gameStat.setLevel_id("1");
```

Pour les autres paramètres :

- Le temps min :

J'initialise dans l'activité Menu\_jouer une liste où je vais stocker le temps de chaque tentative réussie.

```
array_L = new ArrayList<Integer>();
```

Dans chaque screen , je calcule la date d'entrée et dans la fonction job() qui teste si

l'enfant a réussi, je calcule la date actuelle, et je fais la différence et je la stocke dans la liste .

```
Date win=new Date();
```

```
// je divise par 1000 pour convertir de millisecons vers secondes
```

```
int between = (int) ((win.getTime() - entree.getTime())/1000);
```

```
array_L.add(between);
```

Quand l'enfant clique sur le bouton du retour. Au niveau de la fonction

**public void handleDragging(int x, int y, int pointer)** ,où je teste le clic sur le sprite de retour , je vais calculer le max de la liste et l'affecter au champs Min\_time\_succeed .

```
Object min = Collections.min(array_L);
```

```
gameStat.setMin_time_succeed_sec(min.toString());
```

- Le temps moyen:

Quand l'enfant clique sur le bouton du retour. Au niveau de la fonction

**public void handleDragging(int x, int y, int pointer)** ,où je teste le clic sur le sprite de retour , je vais calculer le moyenne de la liste et l'affecter au champs Avg\_time\_succeed\_sec.

```
int somme=0;
```

```
float moy;
```

```
for (Object obj:array_L)
```

```
{
```



```
somme=somme+(int)obj;  
}  
moy=(float) somme/array_L.size();  
gameStat.setAvg_time_succeed_sec(String.valueOf(moy));
```

- Nombre d'opération non réussis & nombre d'opérations réussis :

Dans la fonction **job ()** de chaque screen, après chaque réussite, je fais win++  
Et après chaque échec lose++, avec lose et win sont des variables statiques que je les initialise à zéro à chaque entrée au Menu\_jouer.

- Localisation :

Je la traite dans l'activité PopupActivity , je demande la permission d'accéder à la localisation et je stocke la longitude et la latitude.

- ✓ Dans chaque screen, dans la fonction **public void handleDragging(int x, int y, int pointer)** , si la condition qui teste si on a touché le sprite du retour est vérifiée ,je stocke mes données dans le serveur  
Avec la fonction :

```
FoxyAuth.storeGameStat(activity_pop,gameStat);
```