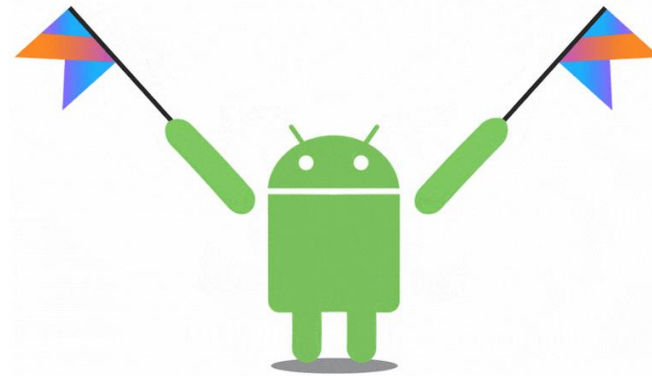


Retour d'expérience sur le développement d'une première application mobile à l'aide de Android Kotlin



Eliott BALDY / Cédric DURAYSSEIX / Léo LACOSTE / Thomas LEIGNAC / Léo VIGUIER



Sommaire

I – Présentation du langage Android Kotlin

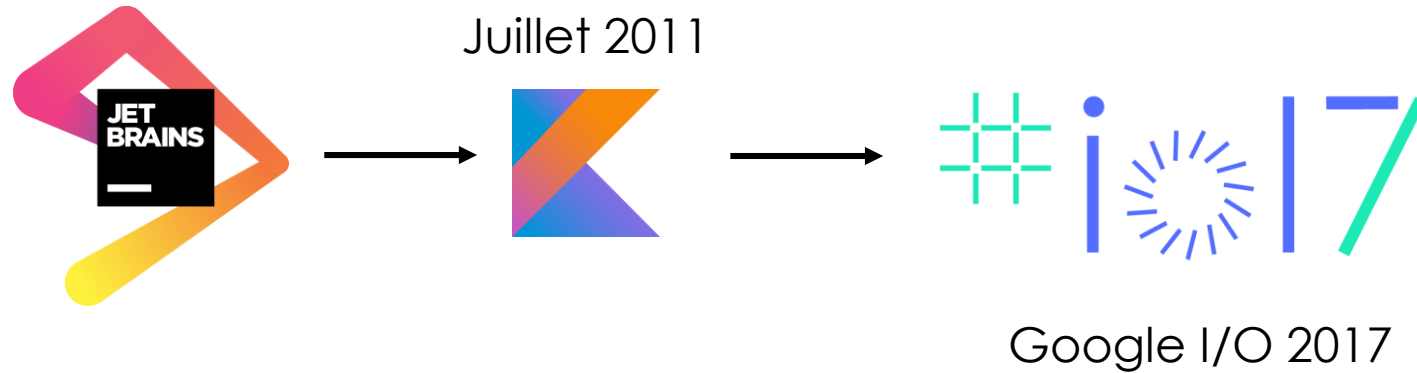
II – Présentation de l'architecture d'une application mobile

III – Présentation de notre projet tuteuré

IV – Les avantages / inconvénients Android Kotlin / Android Java

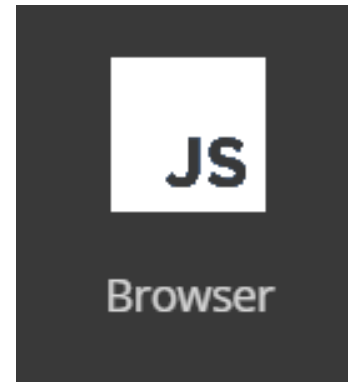
V – Problème rencontrés / les solutions

Présentation du langage Android Kotlin



Applications :





Langage adapté pour ...

ARCHITECTURE D'UNE APPLICATION MOBILE

Les principaux composants d'une application

L'android manifest

Les layouts

Les activités

Les principaux composants d'une application

4

Activity

Intent

View

Service

Content provider

BroadcastReceiver

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="tp.example.com.layout">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Layout"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

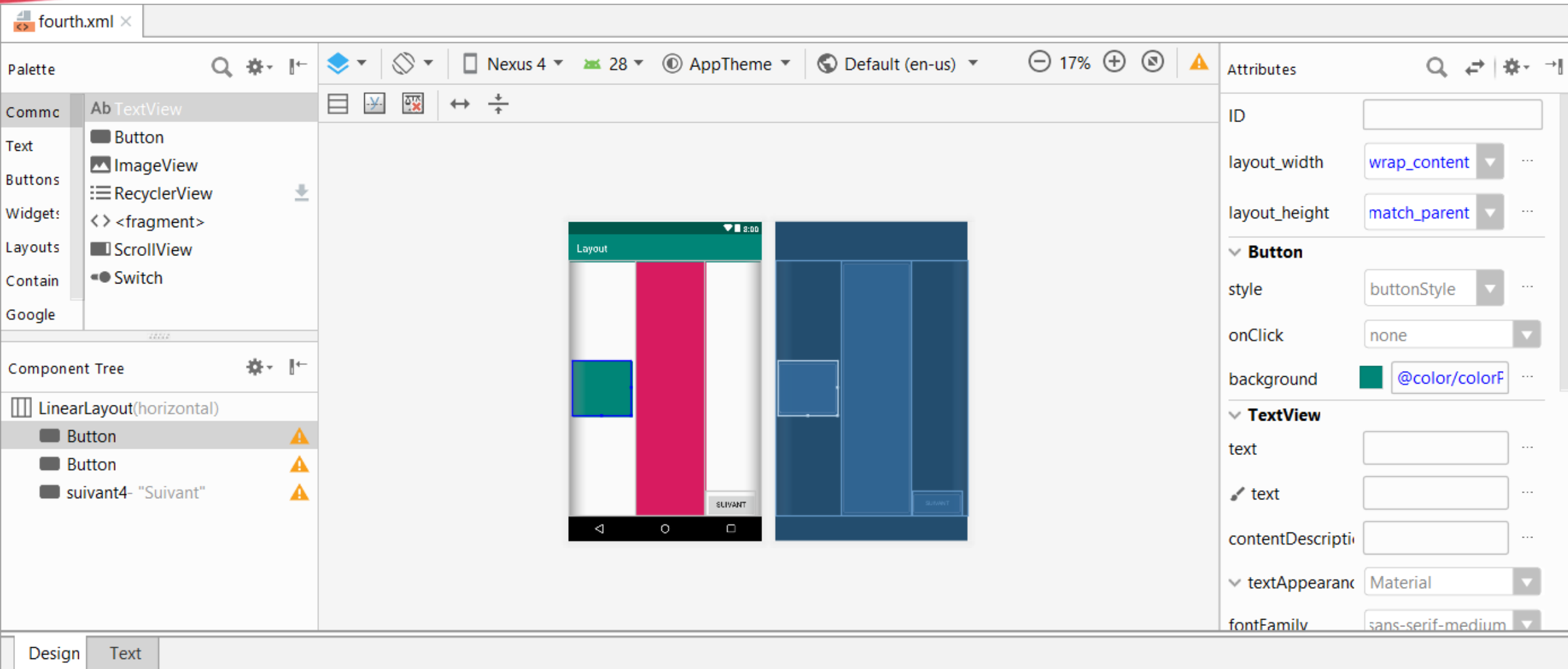
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity"></activity>
        <activity android:name=".ThirdActivity"></activity>
        <activity android:name=".Fourth_activity"></activity>
        <activity android:name=".ScrollActivity"></activity>
        <activity android:name=".listActivity"></activity>
    </application>

</manifest>
```


Les layouts

Créer un layout graphiquement

6




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="200dp"
        android:background="@color/colorPrimary"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="200dp"
        android:layout_weight="0.30" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:background="@color/colorAccent"
        android:layout_weight="0.50" />

    <Button
        android:id="@+id/suivant4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:layout_marginRight="10dp"
        android:layout_weight="0.10"
        android:text="Suivant" />

</LinearLayout>
```

Les layouts

Créer un layout par le code

Les activités

```
package tp.example.com.layout;

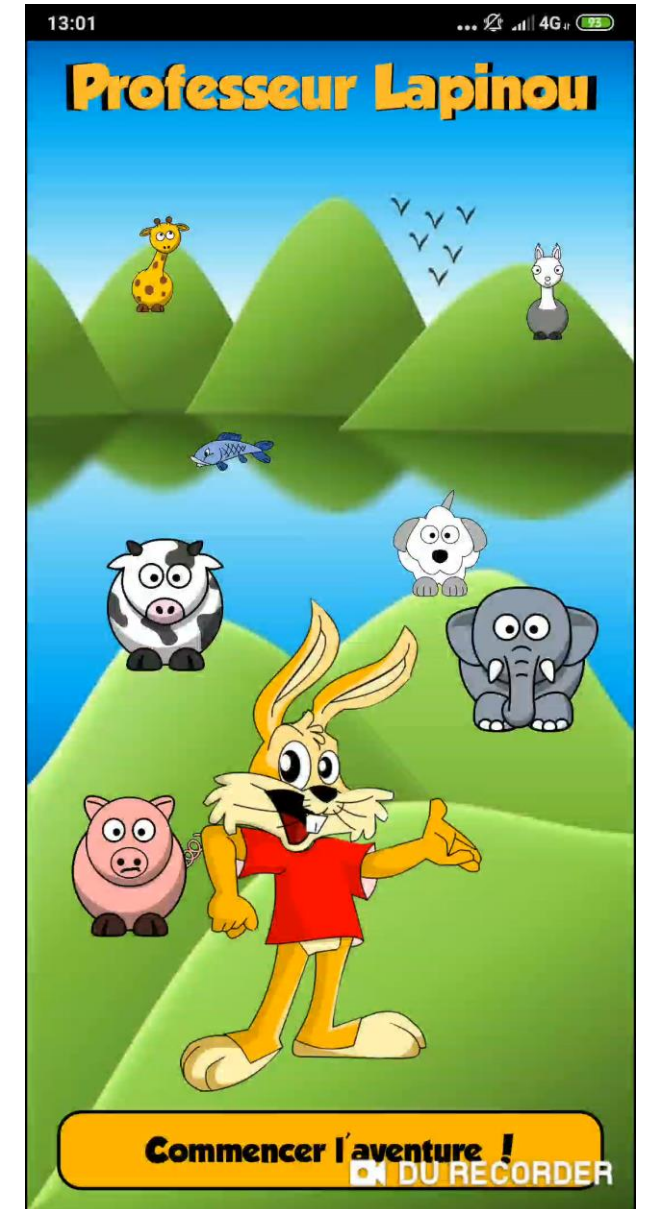
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;

public class Fourth_activity extends AppCompatActivity {
    private Button suivant;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fourth);
        suivant = findViewById(R.id.suivant4);
        suivant.setOnClickListener((view) → {
            Intent intent=new Intent( packageContext: Fourth_activity.this,ScrollActivity.class);
            startActivity(intent);
        });
    }
}
```

Présentation de notre projet tuteuré



Professeur Lapinou



	Android Kotlin	Xamarin	Unity	Unreal Engine
Langage	Android Kotlin	C#	C#	C++
Multi-plateforme	Non sauf avec un Framework	Oui à 96%	Oui	Oui
Popularité	Utilisé par des applications jeunes comme (Pinterest Uber...)	Utilisé par des grandes compagnies comme (UPS)	Alimente le plus grand nombre de jeux sur l'App Store et il est aussi le plus populaire de ce comparatif	Pratiquement aussi populaire que <u>Unity</u> (2 ^{ème})
Difficulté	Facile à utiliser et accès facile à la documentation + des tutoriels vidéo ou pas (<u>OpenClassroom</u> ...)	Uniquement la documentation fournie par Microsoft	Interface vraiment accessible à tout le monde + beaucoup de tutoriels vidéo	Difficile au début pour l'interface mais Visual Scripting aide (permet de à une personne sans arrière-plan de codage d'ajouter des comportements complexes)
Prix	Gratuit	Payant avec toutes les fonctionnalités, mais gratuit pour les étudiants et les projets open sources	Gratuit pour une utilisation personnelle	Gratuit pour une utilisation personnelle
Performances	Application « natif » donc plus rapide et avec son système intelligent (qui utilise des primitifs au lieu des types)	Plus lent sur les applications lourdes en calcul (de la faite de toutes ses librairies)	Rapide et puissant avec performances graphiques correct 2D et 3D	Plus rapide, Plus puissant, meilleures performances graphique (cependant fichier lourd)
Avantages	L'expérience des jeux natifs est transparente, car ils utilisent toute la puissance des GPU intégrés	Multi plate-forme qui permet d'atteindre un équilibre entre budget et performances	Unity 2D est beaucoup plus riche et intuitif (pour des jeux mobiles en 2D) + jeux plus légers	La plus puissante et la plus personnalisable
Désavantages	Pas de multi plate-forme	Peu de documentions	Gratuit uniquement pour une utilisation personnelle	Gratuit uniquement pour une utilisation personnelle

Pourquoi Android Kotlin ?

Structuration de l'application – Pattern MVC



1. La vue notifie le contrôleur

2. Le contrôleur met à jour le modèle

3. La vue est mise à jour par le contrôleur

```
hideAnimalsHard.findElement ( numberOfSelectedAnswer: 3)  
infos.setText ("Essaye encore !")
```

```
public fun findElement(numberOfSelectedAnswer: Int):Int{  
    var result=-1  
  
    if (isResultNotPickedInGoodOrder(numberOfSelectedAnswer)) {  
        result = 0  
    }else{  
        updateBoardOfAnswerOrder(numberOfSelectedAnswer)  
        result = 1  
    }  
  
    return result  
}
```

Pourquoi le Pattern MVC ?

12

```
import com.project.wukay.wukayapp.metier.HideAnimals
import org.junit.Assert
import org.junit.Test
```



```
class HideAnimalsTest {

    @Test
    fun return_true_is_win_if_we_find_3_animals() {
        var gameHideAnimals = HideAnimals()
        gameHideAnimals.findElement( numberOfSelectedAnswer: 0)
        gameHideAnimals.findElement( numberOfSelectedAnswer: 1)
        gameHideAnimals.findElement( numberOfSelectedAnswer: 2)
        gameHideAnimals.isWin()
        Assert.assertTrue(gameHideAnimals.isWin())
    }
}
```

Avantages et Inconvénients de Kotlin



- Un langage plus concis
- Un langage plus flexible
- Un langage plus riche
- L'interopérabilité (totale) avec Java
- Disparition des « Null Pointer Exception »



- Langage jeune donc peu de documentation, librairies,...

SYNTAXES ET CONCEPTS DE BASE



VAL et VAR



Les classes



Une nouvelle structure de
contrôle « when »

VAL

Java

```
final String text = "Hello world !";
```

Kotlin

```
val text = "Hello world !"
```

```
val text :String = "Hello world !"
```

VAR

Java

```
String text = "Hello world !";
```

Kotlin

```
var text = "Hello world !"
```

```
var text :String = "Hello world !"
```

```
public class Contact {  
    private String prenom;  
    private String nom;  
    private String email;  
  
    Contact(String nom, String prenom, String email) {  
        this.nom = nom;  
        this.prenom = prenom;  
        this.email = email;  
    }  
  
    String getNom() {  
        return this.nom;  
    }  
    String getPrenom() {  
        return this.prenom;  
    }  
    String getEmail() {  
        return this.email;  
    }  
    void setEmail(String email) {  
        this.email = email;  
    }  
}
```

```
class Contact(var prenom: String, var nom: String, var email: String)
```

● ● ● Et non !

Utilisation des classes

Java

```
Contact contact = new Contact("John", "Doe", "mail@ippon.fr");  
contact.setEmail("nouveau_mail@ippon.fr");  
String myEmail = contact.getEmail();
```

Kotlin

```
var contact = Contact("John", "Doe", "mail@ippon.fr")  
contact.email = "nouveau_mail@ippon.fr"  
var myEmail = contact.email
```

La structure de contrôle « when »

```
var loopLevel = prefs!!.actualLevel
when (loopLevel % 8) {
    1 -> {
        lapinouSkin.x = widthScreen*30/100F
        lapinouSkin.y = -heightScreen*15/100F
        levelCounter.text = "Niveau : " + loopLevel
    }
    2 -> {
        lapinouSkin.x = -widthScreen*16/100F
        lapinouSkin.y = -heightScreen*33/100F
        levelCounter.text = "Niveau : " + loopLevel
    }
    3 -> {
        lapinouSkin.x = widthScreen*32/100F
        lapinouSkin.y = -heightScreen*45/100F
        levelCounter.text = "Niveau : " + loopLevel
    }
}
```


Le Null Safety en Kotlin

Java

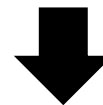
```
private Contact contact = new Contact( nom: "Jean", prenom: "Robert", email: null);
```

Kotlin

```
class Contact(var prenom: String, var nom: String, var email: String)
```

```
 var contact = Contact( prenom: "Jean", nom: "Robert", email: null)
```

Null can not be a value of a non-null type String



```
class Contact(var prenom:String, var nom: String, var email: String?)
```

```
var contact = Contact( prenom: "Jean", nom: "Robert", email: null)
```

Java

```
private Button suivant;  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    suivant = findViewById(R.id.suivant);  
    suivant.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View view) {  
            Intent intent=new Intent( packageContext: MainActivity.this,SecondActivity.class);  
            startActivity(intent);  
        }  
    });  
}
```

Kotlin

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    suivant.setOnClickListener { it: View!  
        val next = Intent( packageContext: this@MainActivity, SecondActivity::class.java)  
        startActivity(next)  
    }  
}
```




Benoît Prioux

@binout

Suivre



Converted a side project from [#java](#) to [#kotlin](#), results : 3336 loc -> 1772 loc = 46% less loc 🎉 😎

Exemple de code en Kotlin

Problèmes rencontrés

Drag & Drop



Problèmes rencontrés

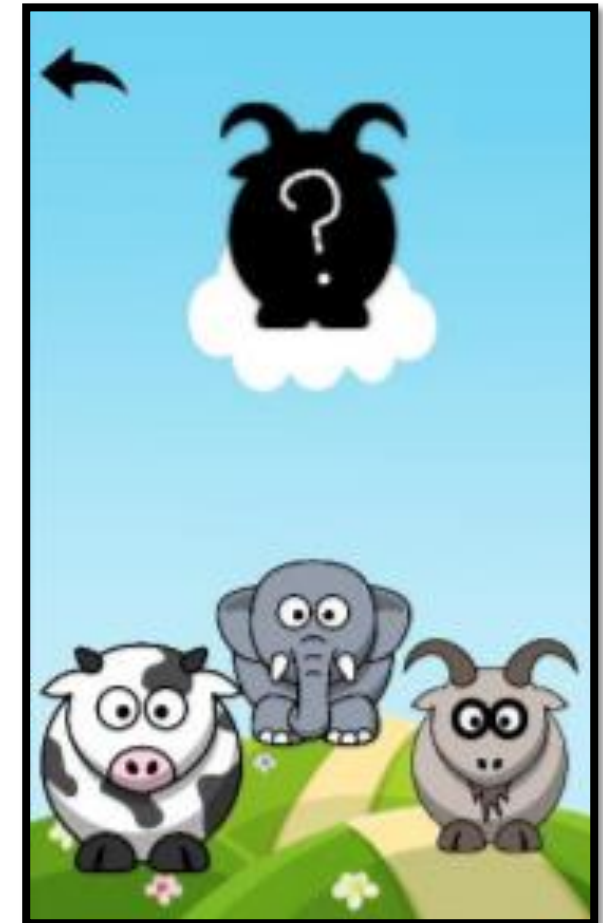
Drag & Drop

```
if(m.x >= animalAnwser1.x && m.x < animalAnwser1.x + animalAnwser1.width && m.y >= animalAnwser1.y && m.y < animalAnwser1.y + animalAnwser1.height ) {  
    animalAnwser1.x = m.x - animalAnwser1.width / 2  
    animalAnwser1.y = m.y - animalAnwser1.height / 2  
}  
else{  
    if(m.x >= animalAnwser2.x && m.x < animalAnwser2.x + animalAnwser2.width && m.y >= animalAnwser2.y && m.y < animalAnwser2.y + animalAnwser2.height) {  
        animalAnwser2.x = m.x - animalAnwser2.width/2  
        animalAnwser2.y = m.y - animalAnwser2.height/2  
    }  
    else{  
        if(m.x >= animalAnwser3.x && m.x < animalAnwser3.x + animalAnwser3.width && m.y >= animalAnwser3.y && m.y < animalAnwser3.y + animalAnwser1.height) {  
            animalAnwser3.x = m.x - animalAnwser3.width/2  
            animalAnwser3.y = m.y - animalAnwser3.height/2  
        }  
    }  
}
```

Problèmes rencontrés

Drag & Drop

```
if(isFingerOnPicture(motionEvent,animalAnwser1)) {  
    makeThePictureFollowTheFinger(motionEvent,animalAnwser1)  
}  
else{  
    if(isFingerOnPicture(motionEvent,animalAnwser2)) {  
        makeThePictureFollowTheFinger(motionEvent,animalAnwser2)  
    }  
    else{  
        if(isFingerOnPicture(motionEvent,animalAnwser3)) {  
            makeThePictureFollowTheFinger(motionEvent,animalAnwser3)  
        }  
    }  
}
```



Problèmes rencontrés

Drag & Drop

```
private fun makeThePictureFollowTheFinger(motionEvent: MotionEvent, idImageView: ImageView) {  
    idImageView.x = motionEvent.x - idImageView.width / 2  
    idImageView.y = motionEvent.y - idImageView.height / 2  
}  
  
private fun isFingerOnPicture(motionEvent: MotionEvent, idImageView: ImageView): Boolean {  
    return motionEvent.x >= idImageView.x && motionEvent.x < idImageView.x + idImageView.width &&  
        motionEvent.y >= idImageView.y && motionEvent.y < idImageView.y + idImageView.height  
}
```


Problèmes rencontrés

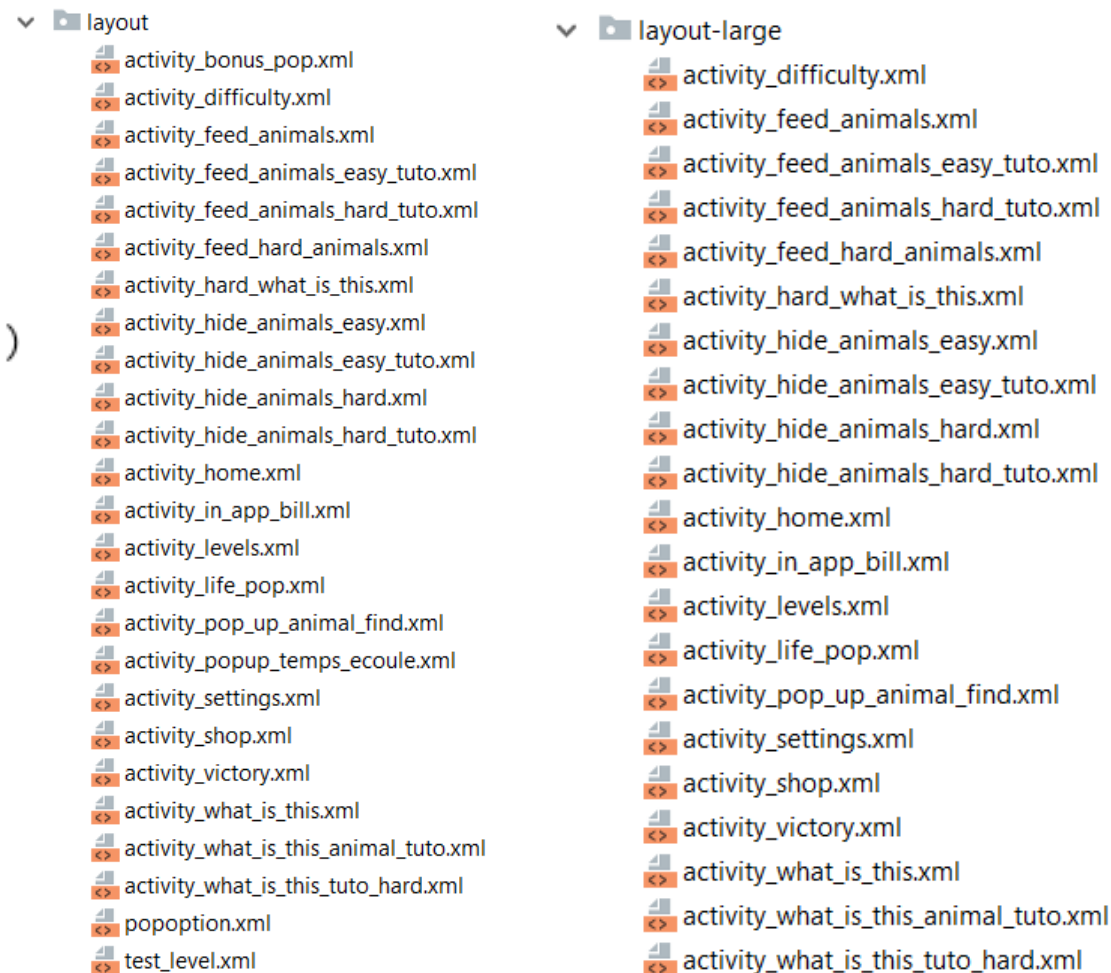
Problème d'affichage



Problèmes rencontrés

Problème d'affichage

```
val displayMetrics = DisplayMetrics()  
windowManager.defaultDisplay.getMetrics(displayMetrics)  
val heightScreen : Int = displayMetrics.heightPixels  
val widthScreen : Int = displayMetrics.widthPixels
```





CONCLUSION

